HUMAN AWARE NAVIGATION OF A MOBILE ROBOT IN CROWDED DYNAMIC ENVIRONMENTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

AKIF HACINECIPOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
MECHANICAL ENGINEERING

SEPTEMBER 2019

Approval of the thesis:

HUMAN AWARE NAVIGATION OF A MOBILE ROBOT IN CROWDED DYNAMIC ENVIRONMENTS

submitted by **AKIF HACINECIPOĞLU** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Mehmet Ali Sahir Arıkan
Head of Department, **Mechanical Engineering** _____

Prof. Dr. Erhan İlhan Konukseven
Supervisor, **Mechanical Engineering, METU** _____

Assist. Prof. Dr. Ahmet Buğra Koku
Co-supervisor, **Mechanical Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Yavuz Samim Ünlüsoy
Mechanical Engineering, METU _____

Prof. Dr. Erhan İlhan Konukseven
Mechanical Engineering, METU _____

Assist. Prof. Dr. Kutluk Bilge Arıkan
Mechanical Engineering, TED University _____

Assist. Prof. Dr. Ali Emre Turgut
Mechanical Engineering, METU _____

Assist. Prof. Dr. Andaç Töre Şamiloğlu
Mechanical Engineering, Başkent University _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Akif Hacınecipoğlu

Signature :

ABSTRACT

HUMAN AWARE NAVIGATION OF A MOBILE ROBOT IN CROWDED DYNAMIC ENVIRONMENTS

Hacınecipoğlu, Akif
Ph.D., Department of Mechanical Engineering
Supervisor: Prof. Dr. Erhan İlhan Konukseven
Co-Supervisor : Assist. Prof. Dr. Ahmet Buğra Koku

September 2019, 144 pages

As mobile robots start operating in dynamic environments crowded with humans, human-aware and human-like navigation is required to make these robots navigate safely, efficiently and in socially compliant manner. People can navigate in an interactive and cooperative fashion so that, they are able to find their path to a destination even if there is no clear path leading to it. This is clearly a dexterity of humans. But the mobile robots which have to navigate in such environments lack this feature. Even perfect trajectory prediction of people is not sufficient if crowd density is above a certain level. Interactive and cooperative navigation ability of humans should be incorporated into navigation algorithms. Therefore, the scope of this study is to develop a navigation method that can be implemented in mobile robots which will make them able to navigate in crowded dynamic environments without freezing and/or frustration. For this purpose, pose-invariant and real-time people detection and tracking methods are developed initially. Then, an interactive and cooperative trajectory prediction algorithm is introduced. A mobile robot is regarded just as another agent, like other humans in the scene, so that, predicted trajectory for the robot itself becomes

the planned path which results in human-like and human-aware navigation. All the developed components are tested and validated separately, and finally, together on a mobile robot. Results of the real world experiments showed that the developed method can effectively make a mobile robot navigate in human-aware fashion in dynamic environments crowded with humans.

Keywords: Human navigation, Trajectory prediction, Mobile robots, Path planning, Machine vision, Autonomous navigation, Human awareness, Social robots

ÖZ

MOBİL ROBOTLARIN KALABALIK VE DİNAMİK ORTAMLARDA İNSAN FARKINDALIKLI NAVİGASYONU

Hacınecipoğlu, Akif
Doktora, Makina Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Erhan İlhan Konukseven
Ortak Tez Yöneticisi : Dr. Öğr. Üyesi. Ahmet Buğra Koku

Eylül 2019 , 144 sayfa

Mobil robotlar insanların bulunduğu dinamik ortamlarda yer almaya başladıkça, bu ortamlarda güvenli, verimli ve sosyal beklentilere uygun hareket edebilmek için insan-farkındalıklı ve insansı özel navigasyon yöntemlerine ihtiyaç duymaktadırlar. İnsanlar, hedeflerine giden açık bir yol olmasa bile sahip oldukları etkileşimli ve işbirlikçi navigasyon becerileri sayesinde hedeflerine ulaşabilmektedirler. Bu, insanların sahip olduğu bir yetenek olmakla birlikte, benzer ortamlarda hareket etmesi gereken robotlar bu özelliğe sahip değildir. Etraftaki insanların hareketlerinin mükemmel bir şekilde tahmin edilmesi bile, insan kalabalığı belli bir seviyenin üzerindeyse, navigasyon için yeterli değildir. İnsanların etkileşimli ve işbirlikçi hareket kabiliyetlerinin navigasyon algoritmalarına eklenmesi gerekmektedir. Bu çalışmanın kapsamı mobil robotlara uygulandığında onların karışıklık yaşamadan kalabalık ortamlarda hareket etmelerini sağlayacak bir navigasyon yönteminin geliştirilmesidir. Bu amaçla öncelikle insanların duruşundan bağımsız ve gerçek zamanlı çalışan insan tespit ve takip sistemleri geliştirilmiştir. Daha sonra ise etkileşimli ve işbirlikçi bir şekilde çalışan bir navigasyon yöntemi geliştirilmiştir. Bu yöntemde mobil robot da ortamdaki diğer

insandan birisi gibi modellenmiş, bu sayede robotun kendisi için tahmin ettiği yol, planlanmış yol olarak kabul edilebilmiştir. Sonuç olarak insansı ve insan-farkındalıklı navigasyon yöntemi elde edilmiştir. Geliştirilen bütün bileşenler önce ayrı ayrı, daha sonra ise bir bütün olarak bir mobil robot üzerinde test edilmiş ve doğrulanmıştır. Gerçek dünya deneyleri neticesinde, geliştirlen yöntemin bir mobil robotun kalabalık ve dinamik ortamlarda insan farkındalıklı olarak hareket edebilmesini sağladığı gösterilmiştir.

Anahtar Kelimeler: İnsan navigasyonu, Hareket tahmini, Mobil robotlar, Yol planlaması, Makine görmesi, Otonom navigasyon, İnsan farkındalığı, Sosyal robotlar

To my lovely wife Fatmanur and my dear son Ömer Tuna...

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my advisors Prof. Dr. E. İlhan Konukseven and Asst. Prof. Dr. A. Buğra Koku for their invaluable guidance, advice, patience and support that made this Ph.D. study possible. I would like to thank Prof. Dr. Y. Samim Ünlüsoy and Assist. Prof. Dr. Kutluk Bilge Arıkan for their guidance and advice as members of thesis progress committee.

I am also grateful to my dear friends and colleagues Hasan Ölmez, Kamil Özden, Burak Şamil Özden and Göker Ertunç for their great friendship, moral support and encouragement. I would also thank great people at Milvus Robotics for their help and support.

I would like to express my gratitude to my lovely parents, Mehmet and Ayşe, and sisters, Hatice, Elif, Gülsüm and Fatma Sena for their love, trust and support throughout my life. Besides, I would like to express my love to my aunt Havva and all other great members of my family. Also, I would like to thank all great people who became a part of my life.

My final but most meaningful words are for my beloved wife Fatmanur and my dear son Ömer Tuna. Through my hardest times, they were always with me with their invaluable love, patience and faith. I will always owe a debt of gratitude to them for their support, understanding and the wonderful times we had.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

1D	1 Dimensional
2D	2 Dimensional
3D	3 Dimensional
AFH	Adjacent Features Histogram
CoT	Cost of Transport
CPU	Central Processing Unit
DET	Detection Error Tradeoff
EER	Equal Error Rate
ESF	Ensemble of Shape Functions
FPPF	False Positives Per Frame
FRR	False Rejection Rate
GAC	Gaussian Agent Cost
GP	Gaussian Process
GPU	Graphics Processing Unit
HH	Human to Human
HOD	Histogram of Oriented Depths
HOG	Histogram of Oriented Gradients
HOHD	Histogram of Height Difference
HR-C	Human to Robot - Cooperative
HRI	Human-Robot Interaction
HR-R	Human to Robot - Reactive
ID	Identity
IGP	Interacting Gaussian Processes
IRL	Inverse Reinforcement Learning

JHCH	Joint Histogram of Color and Height
KTP	Kinect Tracking Precision
LfD	Learning from Demonstrations
LSN	Local Surface Normals
LSTM	Long-Short Term Memory
MTV	Mean Trajectory Valley
NN	Nearest Neighbor
PCA	Principal Component Analysis
PCL	Point Cloud Library
PD	People Detector
PR	Precision and Recall
PT	People Tracker
RANSAC	Random Sample Consensus
RBF	Radial Basis Function
RGB-D	Red Green Blue - Depth
ROI	Region of Interest
ROS	Robot Operating System
RRT	Rapidly-exploring Random Tree
SLAM	Simultaneous Localization and Mapping
STHOG	Spatio-Temporal Histogram of Oriented Gradient
SVM	Support Vector Machine
TP	Trajectory Predictor
VFH	Viewpoint Feature Histogram
VOC	Visual Object Classes

CHAPTER 1

INTRODUCTION

1.1 Mobile Robots

About a half-century ago, automated machines started to take place in industry. Their tasks were very well-defined, therefore, they were confined to move in a certain environment and do necessary movements at desired time intervals. Production and assembly lines were ideal for this innovation because movements like welding, picking and placing objects were not subject to change during operations. Because of these reasons, there was no need to human interference and intelligence applications under normal working conditions.

The need for robotic applications has grown throughout past years. People started to include robotic agents outside of the assembly lines for hard and/or time consuming applications. As a result, mobile robotics has emerged. Starting from early 90s up to recent years, robotic applications moved out from their predefined paths and structured environments to dynamic environments. These new environments have unpredictable moving objects around, and traversable areas are not well defined. Therefore, a robotic agent should sense an object and take precautions according to its state.

1.2 Social Robots

As mobile robotics developed, robots started to carry out their tasks in environments which are also populated with humans. In this new world, like humans do, these agents had to interact with people. This interaction can be the source of control, movement and action. A robot can get control commands from a human or can inter-

act with him to fulfill its mission. Even this interaction can be a part of a socialization progress. Then the question is; how should a robot and a human interact with each other? How can these two populations live together, sharing same environments?

One of the early attempts to make mobile robots operate in environments crowded with people is the Rhino robot, which is deployed in the Deutsches Museum Bonn for six days in 1997 [1]. Its purpose was to guide visitors in the museum while employing probabilistic reliable navigation in dynamic environments. Rhino project provided an extensive insight to researchers on the need for safe and reliable navigation and human-robot interaction. After a short period of time from the deployment of Rhino, another museum tour-guide robot, Minerva is developed and deployed in Smithsonian's National Museum of American History for two weeks [2]. With gathered experiences from Rhino project, the focus for Minerva was more on human-robot interaction research. Authors shared their experiences with these projects and pointed out that there is a considerable need for research on mobile robots sharing the same environment with people [3]. After positive impression of these two projects, researchers from different parts of the world focused on development of service and social robots to make them operate among humans. Robots designed to work in environments like museums, shopping malls, offices and homes are developed in following years [4–7].

Social robotics topic is a multi-disciplinary field that includes control engineering, cognitive science, mechanical engineering, computer science, even sociology and human psychology. Since these robots started to share environments with humans, they have to incorporate human psychology, movement behaviors and society rules in their algorithms. Without this blend, it would be difficult to integrate robots into our daily life.

1.3 Problem and the Motivation

While sharing the same environment, robots should not disrupt human beings. However, with this constraint, still they have to carry out their missions efficiently and effectively. In the navigation aspect, a mobile robot navigation should be safe, reli-

able, human-aware and human-like while avoiding the uncanny valley problem [8]. As noted in related chapters, without behaving similarly to humans in navigation, it may become impossible for a robot to move in dynamic environments crowded with people. For this purpose, we have to understand how people live, how we share information and how we behave while navigating around other humans and objects. Then, we need to blend this behavior into navigation algorithms of mobile robots operating in crowded dynamic environments.

The motivation for this study originated from the need for a safe and cooperative navigation method with high human-likeness and awareness. We believe that, as previous studies in this topic did, this effort will pave the road for the future of mobile robots helping to overcome difficult and/or dangerous tasks or chores for humans.

1.4 Methodology

The main focus of this study is to develop a navigation method for mobile robots which will operate in dynamic environments crowded with people. As mentioned earlier, developed navigation algorithm should be human-aware and human-like, i.e., interactive and cooperative. Therefore, detecting people around a mobile robot is an essential part of a successful navigation algorithm. We start our research with development of a fast running, yet, a robust detection method. Without detecting people reliably and fast enough to be employed on a moving robot, prediction of trajectories of people would be impossible.

Detected people should be tracked by a tracking algorithm to keep history of their trajectories. Past trajectory information becomes crucial for predicting next steps of people to act accordingly. Tracking algorithm also should be fast and computationally efficient since it will run on a moving robot.

Finally comes the navigation algorithm which is the most essential part of this study. Using tracking information of people, especially in close vicinity of the robot, navigation algorithm predicts future trajectories of all agents. During this prediction, core research is on interaction and cooperation modeling of humans. We believed that, if we can model this navigation behavior of humans, we can achieve a human-like

navigation method that will be well-accepted by people.

Each developed component is implemented, tested and verified individually. Then, all components are brought together and as aimed, deployed on a mobile robot. Extensive real-world experiments are carried out with humans, and it is shown that components developed during this study are successfully functioning together to achieve human-aware navigation for a mobile robot.

1.5 Contributions and Novelties

In this study, we have different essential components like people detection, tracking, trajectory prediction and path planning. Each of these parts has its own research topics and novelties. To summarize these novelties by topics, our primary contributions to the literature can be listed as follows:

People detection:

- Using an efficient scene simplification technique,
- Developing a point cloud slicing method to gather human body parts in arbitrary poses,
- Iteratively using Principal Component Analysis (PCA) to extract pose invariant head region of human body,
- Introducing the Adjacent Features Histogram (AFH) 3D shape descriptor,
- Implementing the algorithm in low resource intensive and real-time manner using depth information only.

People tracking:

- Implementing an efficient track management, and accurate high speed tracking.

Trajectory prediction and path planning:

- Developing a combined cost-based interaction and cooperation model,

- Introducing the online future conditioning and sampling of Gaussian processes on prediction space,
- Using a step-wise prediction model for each agent on a scene,
- Blending penalized velocity sampling due to human metabolic energy costs based on speed changes,
- Implementing the algorithm in low resource intensive and real-time manner to be used on a mobile robot.

Other:

- Being the first study to provide a complete navigation solution for a human-aware mobile robot,
- Developing all components in a way to make them run relying only on a CPU (i.e., computer with no GPU hardware).

1.6 The Outline of the Thesis

After the introduction to the topic and to the study in the recent chapter, we provide details of our people detection method in Chapter 2. Related work in the literature about people detection, details of our pipeline and experiments are presented. Then, we move on to people tracking application in Chapter 3. Again, literature review on tracking methods, our implementation and experiments conducted on a dataset are given. Chapter 4 is dedicated to our main research on this study which is the trajectory prediction and path planning. We state the problem and provide the literature review. Then, we explain the details of the proposed method. Results of the experiments on a widely used public dataset are also given in this chapter. After these results, Chapter 4 is finished. In Chapter 5, we have comprehensive real-world experiments which include all components developed during this study. We provide the results again in Chapter 5, then, we conclude the whole study in Chapter 6.

CHAPTER 2

PEOPLE DETECTION

2.1 Introduction

Robots are becoming an integral part of our daily life. Social, advertisement, security, search and rescue robots are examples of robots that share same environments with people. The key point for operating in these environments is the ability to detect human-beings around. Robots have to distinguish people from objects and behave accordingly. People detection is a complicated process especially in dynamic and cluttered environments. Although people share some properties in common, each human can move, pose and behave differently in an environment. Changing environmental conditions (ambient light, occlusions, etc.) are also other factors making robust people detection difficult.

For decades, researchers developed different methods to solve this problem. With help of developing computation and sensor technologies, more satisfying results are obtained in recent years. Especially having affordable and simple sensors like Microsoft Kinect, Asus Xtion and Intel Realsense made researchers use RGB-D sensors in their studies. However, recent human detection methods are far from being applicable to social mobile robots sharing same environment with people. Due to human nature, people can be present in different poses (standing, bent, laying, sitting, etc.) and can behave different from each other. Recent human detectors generally assume that people are standing upright or walking to be detected. However, in dynamic environments where people act naturally, detection of arbitrary poses is a challenge. In addition, since people detection systems evolve from stationary systems to mobile robots, a good people detection method should be computationally effective and run

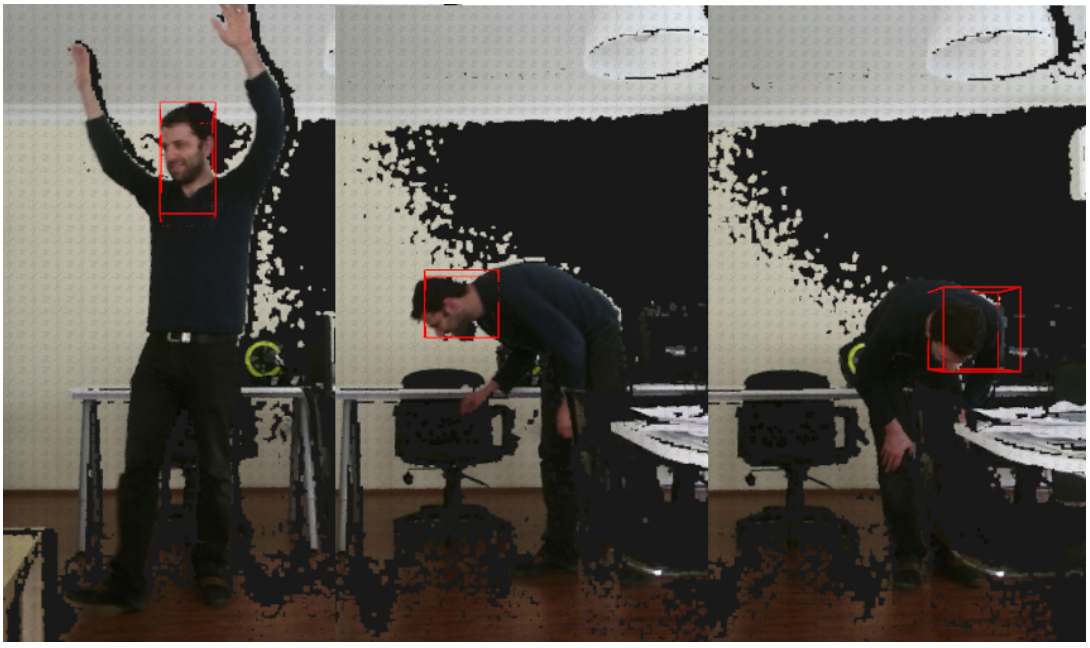


Figure 2.1: Detection results of proposed method in sample challenging arbitrary poses.

in real-time. Therefore, we aimed to develop a novel people detection method which can run on a computationally limited mobile robot effectively, detecting people in arbitrary poses and under changing light conditions.

This chapter is organized as follows: the first section is the introduction to the topic. The second section includes related work in the literature. Developed method is described in third part. Results are given and this study is concluded in last two sections.

2.2 Related Work

People detection has been an active research area in the last two decades resulting in various methods. Especially for people detection in outdoor scenes like pedestrian detection, most widely used method is Histogram of Oriented Gradients (HOG) in 2D images [9]. Dalal and Triggs defined a fixed size window for detection and subdivided it into grid of cells. Orientations of gradients in each cell are computed and a 1D histogram is constructed. This histogram is used for training a linear SVM. In classification stage, fixed-size windows are used in different scales over the image.

The method performs better in upright and non-occluded poses. However, it fails to classify in changing light conditions and body postures. Since there is no depth information, small image patches resulting in similar HOG descriptors may lead to false positives. To describe the pedestrian appearance and motion features, Hua et al. use the spatio-temporal histogram of oriented gradient (STHOG) [10]. They used this descriptor for pedestrian detection at large distance interval on a normally driven vehicle with monocular camera. In the last decade, researchers have incorporated depth information besides color for human detection. Ess et al. used stereo camera pair to detect and track people [11]. They propose a tracking-by-detection method which provides satisfactory results even in challenging scenes. However, their approach is highly resource intensive where each frame is processed in about 30 seconds. This is far from being real-time to be applicable.

With evolving technology in sensor hardware, incorporation of depth information into people detection methods became affordable. Even though we had stereo vision cameras before, RGB-D sensors like Microsoft Kinect, Intel Realsense or Asus Xtion made depth perception available besides color information in a simpler and affordable way. These sensors perform best when there is no direct sunlight in the environment due to infrared light interference, therefore, they are used in indoor people detection applications especially. In a more recent study, Spinello et al. used RGB-D data along HOG descriptors [12]. The method is similar to HOG, but authors incorporated depth information also and developed a method called Histogram of Oriented Depths (HOD). They combined HOG and HOD descriptors to achieve best performance in different distance ranges. However, their method densely scans each frame for people and relies on GPU implementation for a real time performance. They do not provide a test result for articulated human bodies or other arbitrary poses. Still, due to nature of the descriptors they use, it can be concluded that their method will perform best for people standing upright with the help of GPU implementation which may be a limitation for computational resources of mobile robots. HOD is also used in another study [13]. Choi et al. segment the depth image into an initial number of regions and then eliminate regions which are not consistent with some heuristics like width and height. Remaining regions are classified using HOD descriptors with SVMs.

Top-down/bottom-up segmentation is used along Local Surface Normals (LSN) de-

scriptor by Hegger et al. [14]. They split the depth cloud into smaller clusters using a layered subdivision and a top-down/bottom up segmentation. Descriptor consists of local surface normals and additional statistical features. Their method runs at 5Hz which is a low frame rate for a mobile robot moving in a dynamic environment populated with people. Pesenti et al. study people detection for mobile robots where the RGB-D sensor is located at knee level [15]. Hence, people detection is performed using lower part of human body, namely legs only. Since legs are not very distinctive features of a human body, they use a large training set, i.e. 26000 instance training samples are acquired in 15 different real-world environments. Two new features for people detection are introduced by Liu et al. with RGB-D sensors [16]. Histogram of Height Difference (HOHD) and Joint Histogram of Color and Height (JHCH) are used to classify clusters as human or not. Before classification, authors gather human body plausible positions using a height map. Local height maxima are assumed as head crowns. However, this assumption yields false results when there is a person with his hand raised over his head. Therefore, this method may not be applicable for detecting people in arbitrary poses. In a more recent study, Zhang et al. employed a similar method and generate depth contours, detect candidate head locations and then use a deep network to train and classify also using RGB information [17]. However, they are assuming that maxima in depth contours will represent head tops. This assumption fails when one raises a hand or bends, etc. Therefore, it is not applicable to arbitrary poses.

Tseng et al. used top-view depth cameras to detect people [18]. To have an occlusion-free view, they position depth cameras overhead and they detect heads using hemiellipsoidal head model. They have satisfactory results for a stationary surveillance system positioned on a level above people but it is not applicable to mobile robots. Also it may fail in other poses than upright standing pose due to its dependency of hemiellipsoidal head shape. A similar setup is used in [19]. Dan et al. use a human model with head and shoulders shape to detect humans in top-down depth views. Another method with a top-down camera has been used by Migniot et al. [20] where they filter the depth data and fit two ellipsoids representing head and shoulder part of the body. Using these two ellipsoids, they estimate body and head orientation. Bajracharya et al. [21] filter depth data geometrically and apply a linear classifier to each prefiltered

region's point clouds. They use features like variances of point clouds and eigenvalues computed from the scatter matrix of point clouds and constraints like width, height, depth, and volume.

A similar study by Munaro and Menegatti uses RGB-D sensor to detect people and provides satisfactory results in upright human poses [22]. They downsize the point cloud with a voxel grid filter to make it easier to work with and to have constant point density along the depth. The assumption for removing the ground plane is that people walk on a ground plane in all scenes. When clusters are separated with removal of the ground plane, they label remaining clusters using Euclidean distances between points. To avoid over-segmentation problem (clustering parts of a body in different sets), they merge clusters that are very close in ground plane coordinates. To solve under-segmentation problem (clustering more than one human into one set), they implement a head detection method using a height map. Local maxima in height map are regarded as heads of people in the scene and a bounding box is associated with a certain region around these detected head positions as candidate people locations. Finally, corresponding regions in RGB image are used for classification of these patches. HOG detector is used with SVM in 2D image to detect people in these regions of interest (ROI). Although this method detects people in 23 fps with reduced depth resolution of QQVGA (160x120 pixels) instead of VGA resolution available in Kinect, it is apparent that it fails in non-upright postures due to HOG descriptor and head-local maximum matching assumption. If a person raises his hand above his head level, similar to the case in [16], assumption of local maxima as being head positions will not hold.

2.3 The Method

We propose a novel approach that is able to detect people in poses also other than upright position (Fig. 2.1). The method has eight steps shown in Fig 2.2 and implemented using Robot Operating System (ROS) [23] and Point Cloud Library (PCL) [24].

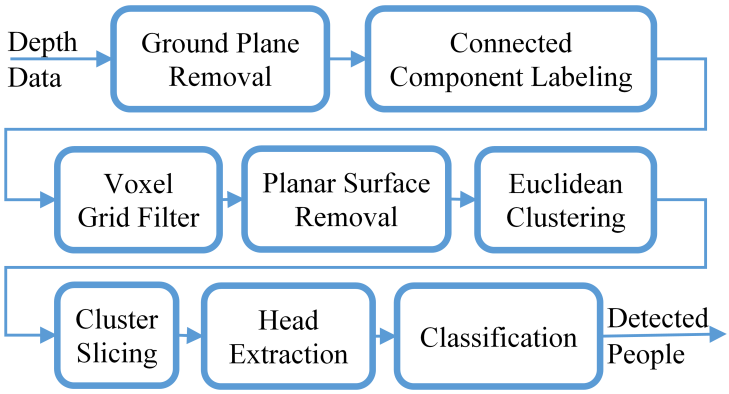


Figure 2.2: People detection pipeline of the proposed method

2.3.1 Ground Plane Removal

Assuming humans are connected via a ground plane, removing it makes people disconnected from each other physically. We select bottom-most three points on the cloud to compute the ground plane coefficients. We use a RANSAC-based least square method [25] and remove all points present within a distance threshold (0.1 m), due to sensor noise and shoes. Since we represent a ground plane with;

$$G(x, y, z) = ax + by + cz + d = 0 \tag{2.1}$$

we estimate coefficients a, b, c, d. Removed points set (G^*) is expressed as follows,

$$G^* = \{v : \frac{|G(v)|}{h} < 0.1, h = \sqrt{a^2 + b^2 + c^2}, v \in V\} \tag{2.2}$$

This process makes processing step faster since considerable amount of points belonging to the ground plane are removed from the scene. In a fixed physical configuration, ground plane coefficients can be input before the operation.

2.3.2 Connected Component Labeling

Depth cloud gathered from a sensor is originally in organized form which means that row-column information is preserved in grid structure (i.e., depth image). This makes neighboring operations (like connected component labeling) more efficient since we do not need to construct a search tree and do resource intensive search operations. Since downsampling (like voxel grid filter) destroys organized structure,

before downsampling we apply connected component labeling. The purpose in this step is to segment the scene regarding the distance between points.

The method used in this stage is borrowed from Trevor et al. [26]. In this method, each point $P(x, y)$ in organized point cloud $\mathbf{P}$ is labeled as $L(x, y)$ and divided into set of segments $\mathbf{B}$. Points are compared with a comparison function C which is denoted in general form as;

$$C\left(P(x_1, y_1), P(x_2, y_2)\right) = \begin{cases} true & if \;\; similar \\ false & otherwise \end{cases} \tag{2.3}$$

In our implementation we have used a comparison function regarding the Euclidean distance (i.e., $L2$ norm) of 0.01 meters between two points. If we label our coordinates in 3D as x, y and z, we can represent our comparison function for two points $P_1 = P(x_1, y_1, z_1)$ and $P_2 = P(x_2, y_2, z_2)$ as follows;

$$\begin{aligned} dist &= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \\ C(P_1, P_2) &= \begin{cases} true & if \;\; dist < 0.01 \\ false & otherwise \end{cases} \end{aligned} \tag{2.4}$$

Note that, this method may result in under-segmentation if two objects are in direct contact. We resolve this problem using our novel cluster slicing and head extraction steps.

2.3.3 Voxel Grid Filtering

The point cloud data from set of segments ($\mathbf{B}$) have high resolution and also density of points decreases with the increasing distance from the sensor. Voxel grid filter creates a 3D voxel grid (V^k) in space over the point cloud. Superscript k corresponds to k^{th} cluster. Points (b_j^k) in each voxel (i.e., a cube with a fixed size) of k^{th} cluster are represented with their centroid (v_i^k). Each cluster $B_k \in \mathbf{B}$ is downsampled with

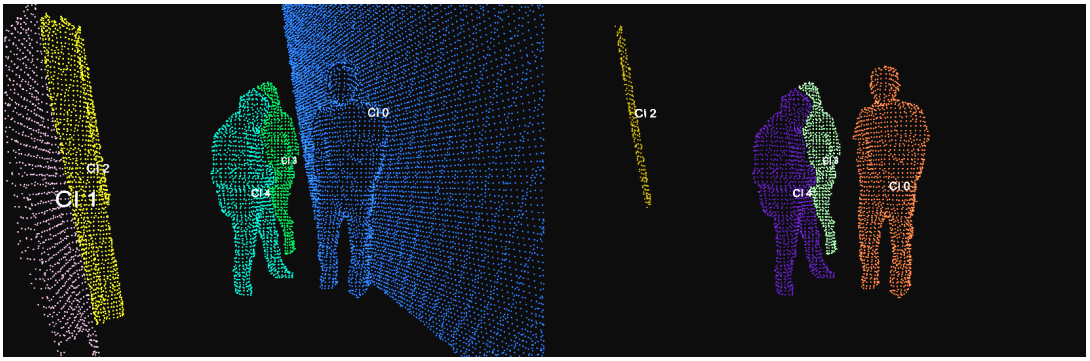


Figure 2.3: Before planar surface removal, one person is segmented with the wall he is in contact with (left, Cluster 0). After planar surface removal, he is extracted from the over-segmented cluster without being merged with the wall (right, Cluster 0).

voxel grid filter. This filter can be expressed as;

$$v_i^k = \frac{1}{N_i^k} \sum_{j=1}^{N_i^k} b_j^k \tag{2.5}$$

where $v_i^k \in V^k$ and $b_j^k \in B_k$. Here, N_i^k represents number of points lying in i^{th} voxel of cluster B_k.

This reduces (downsamples) point cloud to a reasonable number of points which makes processing easier without losing important features. For example, in a scene, we have 160709 points originally, while this is reduced to 19236 with voxel grid filter. Our default voxel size (i.e., edge length of cubic voxel) is 0.03 meters which provides us the detailed enough point cloud for further processing and classification while keeping computational loads low.

2.3.4 Planar Surface Removal

Removing planar surfaces like walls, tables, doors, etc. simplifies the scene and results in faster processing. Also, it helps to separate humans from planar objects (like walls) they are possibly in contact with (Fig. 2.3).

To detect possible plane coefficients, we again use RANSAC-based method for each cluster obtained from the previous step. We have used 2-degrees angular and $2/3 \times$

$voxel\ size$ distance threshold. If F is the plane we fit to cluster B, we define a *planarity ratio*, κ, for each cluster as number of points in fitted plane (N_F) over total number of points in a cluster (N_B), i.e., we define κ and its thresholds as follows;

$$\kappa = \frac{N_F}{N_B} \tag{2.6}$$

$$B \rightarrow \begin{cases} \text{is a pure plane} & if\ \ \kappa \geq 0.9 \\ \text{includes a plane} & if\ \ 0.9 > \kappa \geq 0.5 \\ \text{includes no plane} & if\ \ \kappa < 0.5 \end{cases} \tag{2.7}$$

One example scene for the second case can be seen in Fig 2.3. In this figure, κ for "Cluster 0" is calculated as 0.73. As a result, planar surface (which is a wall the person is in contact with) is removed from "Cluster 0" revealing the human body.

2.3.5 Euclidean Clustering

After removing planar surfaces from clusters, there may be disconnected point cloud blobs in clusters. One example to this scenario can be two people in contact with a wall. When we remove the wall as a planar surface, two different human bodies remain unconnected in the same cluster. Therefore, we apply another clustering algorithm to these clusters. We use Euclidean distance as a metric. We generate a K-D search tree for an efficient loop through every point in point cloud. Then we get a seed point and compare it with its neighbors. If Euclidean distance between these two points is below a certain threshold (twice the voxel side length, 0.06 m, in this case), we assume that these two points belong to the same cluster ($\mathbf{E_i}$) [27].

2.3.6 Cluster Slicing

Extracting head region of human bodies in different postures in cluttered and dynamic environments is important since head provides 1-to-1 matching for a human and it is the most possible non-occluded part of a human body. We propose a method similar

Figure 2.4: Cluster slicing stage with two hands raised (left) and bent (right) cases.

to a sliding window which is slicing a point cloud vertically with consecutive zones intersecting in a pattern, i.e., next slice starts at a fixed distance before the end of the previous one. With this approach, it is guaranteed that at least one of the slices will contain full head region of a person even in non-standard poses (Fig. 2.4). To determine a single slice width, anthropometric data for humans are used [28, 29]. Mean shoulder and elbow-to-elbow breadths for adults are given as approximately 0.45 to 0.50 m. To compensate for additional width from cloths, we select slice width as 0.60 m and intersection width between adjacent slices as the 2/3 of the slice width, i.e., 0.40 m. Each slice ($\mathbf{S_j}$) can be defined with Eqn. 2.8, 2.9 and 2.10. In Eqn. 2.10, E_w is total width of cluster, S_w is width of a single slice (0.6 m) and S_o is overlap amount between slices (0.4 m). Total number of slices necessary becomes $N + 1$.

$$S_s = S_w - S_o \tag{2.8}$$

$$\mathbf{S_j} = \{\mathbf{p} : S_s j < p_x - p_{x,min} < S_s j + S_w, \mathbf{p} \in \mathbf{E_i}\} \tag{2.9}$$

$$j = \{0, ...N\} \quad N = \lceil \frac{E_w - S_w}{S_s} \rceil \tag{2.10}$$

To eliminate the possibility to slice the point cloud with head region divided into two sub-clusters, we use consecutive slicing with intersection. Slicing is carried out vertically, starting from left-most point towards right (on x-axis of Kinect). For example

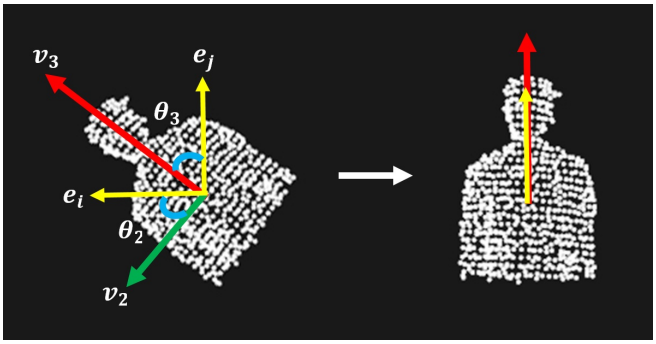


Figure 2.5: Alignment of eigenvectors with the vertical and horizontal axes.

in Fig. 2.4, we have a cluster with width of 0.94 meters including a person. We divide this cluster into 3 sub clusters as: Cluster 1 (0.0 m - 0.60 m), Cluster 2 (0.20 m - 0.80 m), Cluster 3 (0.40 m - 0.94 m). Therefore it is provided that as least one slice (middle one in this case) would contain the whole head portion. This method eliminates the assumption that highest part of the body should be the head [22]. Even if hands are raised above head level (Fig. 2.4), it is obvious that slice in the middle will contain head portion for further processing.

2.3.7 Head Extraction

After having sub clusters which are candidates for possible human detections, we need to extract head and shoulders region to be able to classify it in next step. For this purpose, PCA is used. The assumption is that, the eigenvector corresponding to the largest eigenvalue of the covariance matrix of point cloud is along the line that points from lower part of human body towards the head, i.e., largest variance of a human body is along height. Also, since RGB-D sensor is providing frontal surface points, depth variance should be always smaller than width variance, which means that we can also use the second largest eigenvalue and its corresponding eigenvector for aligning a cluster with camera field of view in the horizontal axis (i.e., x-axis of Kinect). Centroid of the point cloud $\vec{p}$ is calculated first where N is the number of

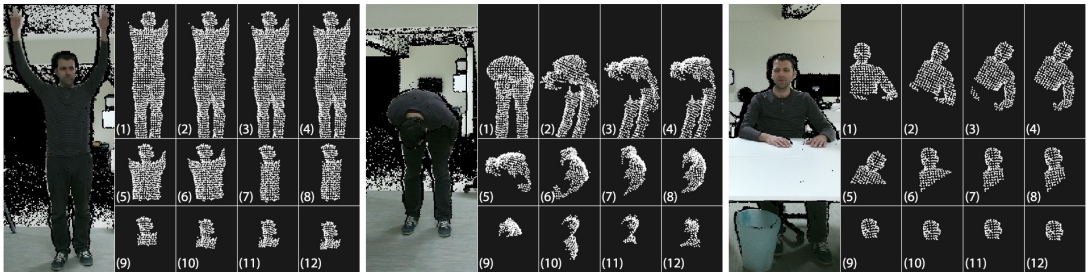


Figure 2.6: Head extraction pipeline for different scenarios; Two hands raised (left), bent (middle) and sitting behind a table (right). Note that in all scenarios we can extract head region at last (12th) step for classification.

points and $\vec{p_i}$ is iterator for all points in the slice.

$$\vec{\bar{p}} = \frac{1}{N}\sum_{i=1}^{N}\vec{p_i} \tag{2.11}$$

Covariance matrix $\mathbf{C}$ of the point cloud is obtained with Eqn. 2.12 for further PCA application. In Eqn. 2.13, $\vec{v_j}$ and λ_j represent eigenvectors and eigenvalues respectively where $j = 1$ for the smallest eigenvalue and $j = 3$ for the largest one.

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\vec{p_i} - \vec{\bar{p}})(\vec{p_i} - \vec{\bar{p}})^T \tag{2.12}$$

$$\mathbf{C}\vec{v_j} = \lambda_j\vec{v_j}, \quad j \in \{1, 2, 3\} \tag{2.13}$$

To be able to detect people in various poses, despite state of art methods in the literature [22], in our approach we do not have the assumption of having people standing upright. People can be in various poses, therefore, we align clusters to upright position (Fig. 2.5) while cropping them to keep including possible head sections. This novel approach enables us to have head sections of people even if they are bending or leaning in any direction since we extract the head section from rest of the cluster with a fixed bounding box in x and y axis (i.e., width and height respectively) of RGB-D sensor. This approach results in pose invariant human detection.

We start our head extraction pipeline with the first alignment (Fig. 2.6, Step 2). Having eigenvalues and corresponding eigenvectors using Eqn. 2.12 and 2.13, we align

the eigenvector, $\vec{v_2}$, corresponding to the second largest eigenvalue, λ_2, with horizontal x-axis of the optical frame. In other words, we rotate the cluster to face its maximum width to the camera view. Details of rotation process is given for alignment with the vertical axis, but procedure is the same if we exchange $\vec{v_3}$ for $\vec{v_2}$ and $\vec{e_j}$ for $\vec{e_i}$. As the second ration, we align the eigenvector, $\vec{v_3}$, corresponding to the largest eigenvalue, λ_3, with the vertical y-axis of the optical frame, i.e., we rotate the cluster around its centroid to align the direction of maximum variance with the vertical axis (Fig. 2.6, Step 3). Rotation axis is $\vec{r}$ and rotation angle is θ ()Eqn. 2.14, 2.15). Here $\vec{e_i}$ and $\vec{e_j}$ are unit vectors along x-axis and y-axis respectively (Fig. 2.5) and $\|\vec{v}\| = 1$.

$$\begin{aligned} \vec{r_2} &= \vec{v_2} \times \vec{e_i} \\ \vec{r_3} &= \vec{v_3} \times \vec{e_j} \end{aligned} \tag{2.14}$$

$$\begin{aligned} \theta_2 &= cos^{-1}(\vec{v_2} \cdot \vec{e_i}) \\ \theta_3 &= cos^{-1}(\vec{v_3} \cdot \vec{e_j}) \end{aligned} \tag{2.15}$$

To rotate the point cloud around its centroid, we first need to translate it to the origin using translation matrix $\mathbf{T}$ in homogenous coordinates which is defined as,

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -\bar{p}_x \\ 0 & 1 & 0 & -\bar{p}_y \\ 0 & 0 & 1 & -\bar{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

This translation is followed by a rotation by an angle of θ around $\vec{r}$, i.e., $\mathbf{R}_{\vec{r}}(\theta)$. Rotation axis $\vec{r}$ is arbitrary, therefore, we normalize $\vec{r}$ and calculate $\mathbf{R}_{\vec{r}}(\theta)$ using Rodrigues' rotation formula.

$$\mathbf{R}_{\vec{r}}(\theta) = \mathbf{I} + \tilde{\vec{r}} sin(\theta) + \tilde{\vec{r}}^2(1 - cos(\theta)) \tag{2.17}$$

where $\tilde{\vec{r}}$ is an antisymmetric matrix of form,

$$\tilde{\tilde{\mathbf{r}}} = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \tag{2.18}$$

Once we rotate and align the point cloud, we translate it back to its original position using translation matrix $\mathbf{T}^{-1}$. If we assume $\mathbf{R}_{\vec{r}}(\theta)$ is expressed in homogenous coordinates, then aligned point cloud $\mathbf{P}^{'}$ is,

$$\mathbf{P}^{'} = \mathbf{T}^{-1} \times \mathbf{R}_{\vec{r_3}}(\theta_3) \times \mathbf{R}_{\vec{r_2}}(\theta_2) \times \mathbf{T} \times \mathbf{P} \tag{2.19}$$

After the first two alignments, we aim to gather upper body part of the person in the slice. We know that for a human, average shoulder breadth is about 0.5 m and upper body height (i.e., stature-crotch height) is about 0.9 m [30]. Therefore, we crop the slice with these dimensions consecutively (Fig. 2.6, Step 4 and 5). Height is cropped starting from the centroid to the top. At this point, if the target slice contains a human, we assume that we have the upper body in the cropped cluster. These cropping operations are followed by alignment again to keep the possible head at the top in upright position (Fig. 2.6, Step 6).

Since we aim to extract head, we need to crop further to average head size while always keeping a possible head at the top with alignment. Average human head width is taken as 0.25 m [30]. When we crop the cluster to width of 0.25 m, we also eliminate shoulders, arms and other parts of body in width (Fig. 2.6, Step 7). Cropped cluster is aligned again since the maximum variance direction can be changed with cropping (Fig. 2.6, Step 8). By cropping the width to 0.25 m, to keep the aspect ratio same with 5/9 we also crop the height to 0.45 m starting from centroid to top and align (Fig. 2.6, Step 9 and 10).

Finally, we crop the cluster to 0.35 m in height referenced from the top most point ($p_{y,max}$) and align the resulting cluster again in horizontal axis (Fig. 2.6, Step 11 and 12). Our purpose at this step is to gather the final portrait pose of the head in 0.25 m x 0.35 m fixed size in x and y axis respectively. These fixed dimensions for a possible head region provide a fixed size test sample for classification process.

2.3.8 Classification

At the last step we make use of machine learning techniques to decide whether a candidate head cluster belongs to a human or not. SVM is used as the classification

method at this stage. Implementation of the method is carried out using LIBSVM, which is an open-source library developed for SVM applications [31]. To train the SVM, we extracted training samples from the processed clouds. In total 2873 samples are gathered from an indoor test environment, and they are labeled by hand as human or not. Training set contains 1571 negative and 1302 positive samples. SVM is trained with these samples using Radial Basis Function (RBF) kernel and probability estimation option. Probability estimation feature of LIBSVM fits a sigmoid function to outputs and maps them to a probability scale [32]. This allows us to grade the output of the classifier based on 0 to 1 scale with probability of 0 being non-human while 1 is certainly a human.

To be able to classify clusters correctly, we need to have a descriptor which can distinguish a head shape from other ordinary objects in the world. There exist different descriptors in the literature. Rusu et al. introduced Viewpoint Feature Histogram (VFH) [33] as a global descriptor for object recognition. They compute the vector between the viewpoint and the centroid of point cloud. For all points in the cloud, the angle between this vector and normal vector of the point is calculated, and a histogram is constructed. Additionally three angles are computed between the normal of the centroid and normal of each point resulting in total of four histograms. The Ensemble of Shape Functions (ESF) is developed by Wohlkinger et al. [34] which is a combination of three different shape functions like distances, angles and area. This descriptor does not rely on surface normals. These descriptors are designed for object recognition which covers generic shapes, not for a specific shape like human head. They try to describe generic 3D shapes as much as possible while a part of them using RGB data [35]. Since we aim to use only depth data to be resistant to light and brightness changes, we did not use descriptors incorporating color data. Majority of the descriptors, which do not use color information, rely on surface normals [33,36,37]. Calculation of these normal vectors for each point in the point cloud adds extra overhead which makes them poor choice for a real-time classifier. Additionally, since we capture scenes only from a single point of view, we do not have all surrounding points of objects. Therefore, normal vectors can be misleading in our case.

Due to the method we use to crop and align candidate head clusters described in previ-

ous sections, the head shape we search is nearly invariant to viewpoint and rotations. Even a bent over person will be aligned with the vertical axis eventually. This eliminates the need for such complicated shape descriptors and enables us to use a simpler descriptor for head shapes. To keep the calculations fast and to describe a head shape sufficiently, we came up with a basic shape descriptor: Adjacent Features Histogram.

2.3.8.1 Adjacent Features Histogram (AFH)

After head extraction operation, we have point clouds, with constant size and fixed viewing angle, ready to be classified. Therefore, instead of working with resource intensive or view dependent descriptors, we introduce a new simple descriptor which divides whole cluster volume into adjacent volumetric histogram bins in 3D. So, we call this descriptor Adjacent Features Histogram.

AFH divides a volume in the radial, angular and depth directions. Number of bins can vary depending on the application. In our case, with 0.03 m voxel size, we found that 0.05 m in radial direction, 30° in angular direction and 0.05 m in depth provide best results. However, for a lower voxel resolution, number of histogram bins can be made smaller to describe shapes better. As descriptor, we set values of bins to number of points they contain.

First, we calculate the 3D centroid of the point cloud. We coincide the center of the descriptor with this centroid. We number the bins starting from zero angle, smallest radius and foremost one as 1 and increase bin number in radial direction first, in angular direction second and in depth third.

For each point we calculate three values $d_{x,y}$, d_z and $\theta_{x,y}$ using the 3D centroid ($\bar{\mathbf{p}}$) of the point cloud as:

$$d_{x,y} = \sqrt{(p_{i,y} - \bar{p}_y)^2 + (p_{i,x} - \bar{p}_x)^2} \tag{2.20}$$

$$d_z = p_{i,z} - p_{z,min} \tag{2.21}$$

$$\theta_{x,y} = atan2(p_{i,y} - \bar{p}_y, p_{i,x} - \bar{p}_x) \tag{2.22}$$

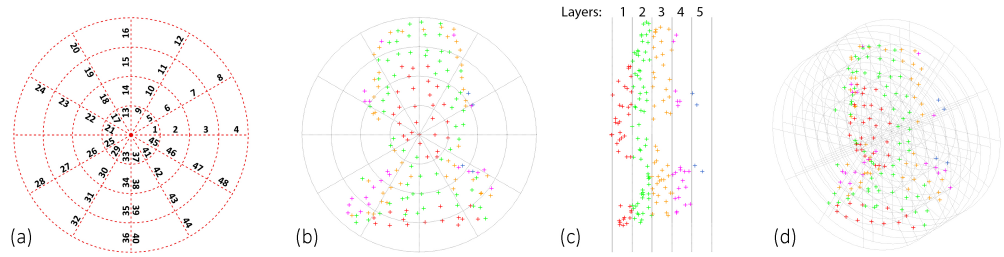


Figure 2.7: AFH Descriptor. (a) Bin numbering for the foremost layer, (b) bins with a sample point cloud belongs to a human, (c) different layers of bins in depth and (d) isometric view of the descriptor.

Corresponding histogram bin index (h_{ind}) is calculated with radial step size ($\Delta d_{x,y}$), number of radial steps (n_r), angular step size ($\Delta\theta_{x,y}$), number of angular steps (n_θ) and depth step size (Δd_z) as shown in Eqn. 2.23 and as a result, value of this bin is incremented by one.

$$h_{ind} = \lceil \frac{d_{x,y}}{\Delta d_{x,y}} \rceil + \lfloor \frac{\theta_{x,y}}{\Delta\theta_{x,y}} \rfloor n_r + \lfloor \frac{d_z}{\Delta d_z} \rfloor n_r n_\theta \tag{2.23}$$

An example of 48 bins for a single layer can be seen in Fig. 2.7. We number the bin with zero angle and smallest radius as one, and adjacent bins in radial direction have consecutive numbers. Then come the next angle and its radial direction. This is repeated for the full layer, then, same numbering scheme is applied for the upcoming layers in depth. If we have five layers in depth, our histogram will have 240 bins in total as seen in Fig. 2.7 (c) and (d).

The resultant clusters from the head extraction stage are classified based on AFH using SVM classifier. If the same head region is present in two different slices of clusters due to intersecting slices, they are both classified as people. However, in such a case we merge two results into one classified as people if they are not apart more than 0.15 meters in 3D coordinates. A decision threshold is defined for probability estimates. By adjusting this threshold, we can change precision and recall values to a reasonable level.

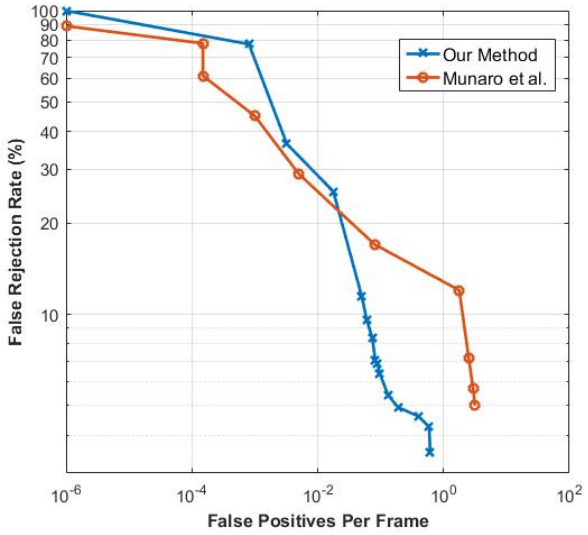


Figure 2.8: Comparison of detection performance of our method with [22].

2.4 Experiments

We evaluated the classification and generalization performance of our method using state of art Kinect Tracking Precision (KTP) dataset from the literature which is distributed with ground truth positions labeled [22]. Due to their method, this dataset contains no scene with arbitrary poses like bent, lean, sit, hands up, etc. Still, we tested with this dataset to be able to compare our results and see generalization performance. Besides KTP, we also implemented our algorithm on a mobile industrial robot to see its real case performance. For 2D object detection, PASCAL VOC challenge [38] defines a minimum of 50% overlap for bounding boxes of detection and ground truth to be regarded as a match. Munaro et al. extends this threshold to 3D case as maximum of 0.3 meters distance in 2D (ground plane) coordinates only. They disregard the height difference in evaluation of results. Instead, we use 0.15 m of 3D Euclidean distance (regarding the height) as our threshold. We consider a result as true positive if ground truth position of head and the detected bounding box centroid are not apart more than 0.15 meters in 3D coordinates. 0.15 m is chosen because it is nearly half-width of a human head [30] so there can only be a single head at this proximity.

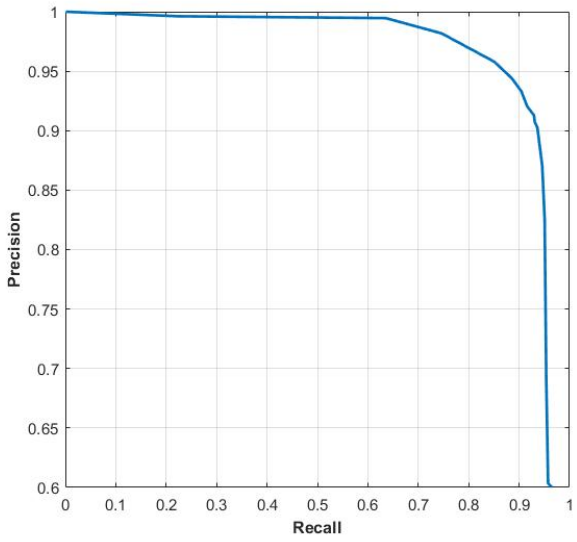


Figure 2.9: Precision and recall curve for our detection method.

2.4.1 Results with KTP Dataset

Besides occlusions and crowded scenes, RGB-D datasets generally lack arbitrary pose like KTP and Freiburg RGB-D People Dataset by [12]. However, it is still a good evaluation set for detection and generalization performance since it is recorded with older Kinect device with various people movements in it.

Results are evaluated using Detection Error Tradeoff (DET) curves [39] which plots the False Rejection Rate (FRR) versus the number of False Positives Per Frame (FPPF) in log-log scale. From Fig. 2.8, it can be seen that our method performs comparable with [22] up to FPPF about 0.015. After this point our method becomes superior and FRR decreases rapidly to 3.5% with 0.6 FPPF, i.e., even before reaching 1 FPPF which is the benchmark point to compare results [39].

2.4.2 Results with Mobile Robot

We also evaluate our results on a mobile robot (SEIT 100) from Milvus Robotics. We mounted Kinect v2 on a configuration shown in Fig. 5.2. It is placed on 1.6 m height (similar to a human) and 10° inclination for better coverage. We captured various poses shown in Fig. 2.1, 2.4 and 2.6. In total, we have recorded 2.7 minutes

Table 2.1: Frame processing frequency with different processors

Processor	**FPS**
Core i3 2.4 GHz	28
Core i7 3.4 GHz	37

long stream in real time and evaluated results. Ground truth is labeled by hand using another stationary Kinect camera. Results are shown with precision and recall (PR) data. Our PR curve (Fig. 2.9) shows that we can obtain high precision even with high recall rate. Equal Error Rate (EER), which is the point where precision equals to recall is 92%. For comparison, with their own dataset, authors achieved EER of 86% with one of their methods which relies only on depth information [12]. We did not compare with results of other proposed methods, because they incorporate 2D detection and color information from RGB-D sensors.

Besides accuracy, with voxel size of 0.03 m, we achieved an average processing rate of 28 frames per second which is considered to be satisfactory for a mobile robot moving in a dynamic cluttered environment. The computer we used in tests has Intel i3 processor with 2.4 GHz clock frequency (Table 2.1).

2.5 Conclusions

In this chapter, we introduced a novel method for detecting people in various poses using only depth data gathered using RGB-D sensors. Since head region of a human body is the most representative part for classification, we extract this part even in challenging scenes using the proposed method. In this process, we also proposed a new feature descriptor for head classification. We evaluated our method on a public dataset (KTP) and on a mobile robot. Robustness of our method results in high performance in both scenarios, i.e., one including upright standard poses and another with non-standard poses.

The method provides high detection rate with a high frame rate relying solely on the CPU of an average laptop computer. This enables the proposed method to be

implemented on mobile robots without requiring high computational resources. Also, since the algorithm uses only depth information, light and contrast changes in indoor environments should not affect classification results directly.

By developing this people detection method, the first stage of the proposed human-aware mobile robot navigation method is completed. Once people can be detected robustly, they can be tracked for further trajectory prediction operations.

CHAPTER 3

PEOPLE TRACKING

3.1 Introduction

This study covers developing navigation methods which can make mobile robots navigate safely and in a socially compliant manner in environments crowded with humans. In order to plan a path and act according to people around, a robot has to detect and track these people reliably. People detection stage is handled with the previously proposed method. To plan a path in crowded environments, robots need to estimate future locations of people around. Therefore, a tracking process is necessary after detection phase. In this chapter, tracking methods in the literature are investigated. Since the experiment results for the proposed detection method is satisfactory and running at high frequency, a reliable tracking method in the literature is chosen and implemented, instead of developing a new one. Efficient track management method is developed to keep track information updated and accurate.

3.2 Related Work

Tracking phase generally takes people locations as input from detection algorithms. Depending on the tracking method, measures for similarity of detected persons in adjacent frames are investigated. If these matches get scores higher than a threshold, they are considered as same persons and tracked accordingly. In the study by Munaro and Menegatti [22], detection modules provide human positions to tracking module and then, the tracking module associates these detections with trackings by maximizing a joint likelihood of probability of color, being human and motion. They use an

online classifier with features composed of color histograms. However this type of methods generally suffer in changing light conditions if they rely heavily on color appearance.

For matching detections to their predicted positions, majority of studies use 3D distance metric [16, 21, 40]. The main reason behind this idea is that, when objects occlude each other while passing, 3D location information provides better handling of trajectories especially in camera's viewpoint. Salas and Tomasi [41] use a likelihood formulation blending candidates' 3D positions with their appearance and motion.

Han et al. do not use 3D distance for tracking [42]. Instead, their method relies on similarities in color and depth variations between frames. Even though Bansal et al. use 3D coordinates in detection stage for ROI selection, they carry out matching in 2D [43]. Also in [44], tracking with Kalman filter is carried out in 2D image plane.

There are studies in the literature suggesting that combining color and depth information results in more consistent matching of detections to tracks. As an example, Luber et al. used color information from RGB image together with Haar-like features in depth and intensity images [45]. Full body color and height histogram is used as an appearance model by Liu et al. [16]. Bhattacharyya similarity measure is computed for matching this appearance model to newly detected candidates.

Combination of overlap of two 3D shapes and Bhattacharyya measure of similarity of their color histograms is used for association in a study by Dan et al. [19]. In [46], authors use a bit more complicated correspondence measure by assuming planarity of standing people. They compute depth-based likelihood of the detection where the mean of normal distribution being the distance to the camera and standard deviation being a heuristically chosen value, inversely proportional with the confidence in depth. These likelihoods are computed for both head and torso, separately. Additionally, color based appearance models are used for head and torso. For the head, Bhattacharyya similarity measure is used while for the torso, fitting an ellipse at the estimated head location on the image, using image gradients.

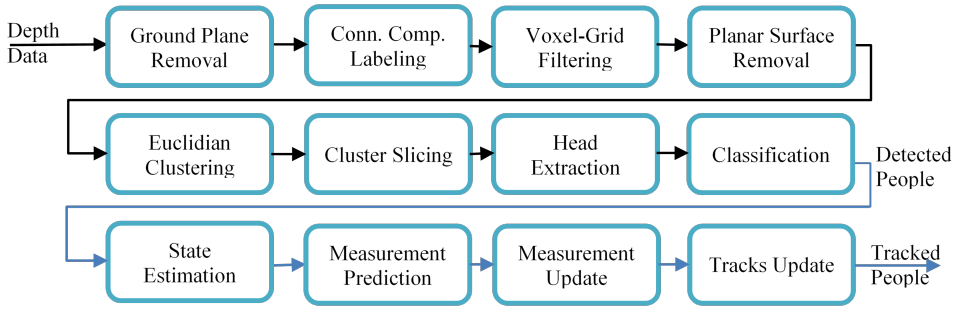


Figure 3.1: Overall pipeline of detection and tracking methods.

3.3 The Method

There are different methods for tracking in the literature. A summary of these methods was provided in the previous section. Most widely used methods rely on (extended) Kalman filter or particle filter. In both of these methods, tracking pipeline is generally composed of prediction, data association and update (or correction) steps. Tracking algorithms gather measurement of human positions from detection algorithms. Any previously detected person is associated with a track according to a threshold for consistency (e.g. number of frames that the person is present). At each loop, state of previously tracked people is predicted with a motion model (kinetic or social). After, these predictions are associated with measurements taken from detection phase regarding a measure (Euclidean distance, Mahalanobis distance, etc.). Using this pairing, state of tracked people is updated as a combination of predicted state and measured state.

For estimation of trajectories of people around the robot, we implemented a tracking method after the detection phase. In the tracking phase, detection results gathered from people detection algorithm are used. We match these detections with previously created tracks with a data association method. If detections are associated with tracks, those tracks are updated. If not, new tracks are created if detections meet certain criteria. The overall pipeline of detection and tracking is given in 3.1.

To maintain positions and velocities of people on the ground plane, we use Kalman filter. For state estimation, we assume that we have a discrete linear dynamic system.

We model the dynamic system as Eqn. 3.1. Here, $\xi(k)$ is the process noise which is modeled as zero-mean normal distribution (Eqn. 3.2).

$$x(k+1) = F(k)x(k) + \xi(k) \tag{3.1}$$

$$\xi(k) \sim N(0, Q(k)) \tag{3.2}$$

Measurement model is given as Eqn. 3.3. Since we have noise in the sensor, we need to add $\epsilon(k)$ as normally distributed measurement noise (Eqn. 3.4).

$$z(k+1) = H(k)x(k) + \epsilon(k) \tag{3.3}$$

$$\epsilon(k) \sim N(0, R(k)) \tag{3.4}$$

The filter consists of four steps; state prediction, measurement prediction, data association and update.

3.3.1 State Prediction

First step of the filter is state prediction. The last state of a track from the previous tracking cycle is updated using state transition matrix $F(k)$. To be able to define a state transition matrix, we need to have a model for human motion. There are different methods to capture and model human motion in tracking. To handle full occlusions, a constant velocity model can be considered, as described in [47]. Data is gathered from RGB-D sensor at a rate of 30 Hz. In such a rate, it is safe to assume human motion locally linear, i.e., constant velocity. For constant velocity model, we define the state vector and state transition matrix as Eqn. 3.5 and 3.5 where dt is the time interval between frames. We define the state only in two ground plane axes which are x and y.

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix} \tag{3.5}$$

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

State prediction is carried out with Eqn. 3.7 and 3.8. At this step, we have a new state which is predicted only using the motion model of the system and noise of this motion model.

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) \tag{3.7}$$

$$\hat{P}(k+1|k) = F(k)\hat{P}(k|k)F^T(k) + Q(k) \tag{3.8}$$

3.3.2 Measurement Prediction

After having the new state predicted using motion model, we predict the measurement using this new state using Eqn. 3.9 and 3.10.

$$\hat{z}(k) = H(k)\hat{x}(k+1|k) \tag{3.9}$$

$$\hat{S}(k) = H(k)\hat{P}(k+1|k)H^T(k) + R(k) \tag{3.10}$$

With RGB-D sensor we measure only positions derived from centroids of detected people. So we define the measurement vector and measurement model as Eqn. 3.11 and 3.12 to include positions in two axis.

$$\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T \tag{3.11}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.12}$$

3.3.3 Data Association

For a robust tracking performance we need to associate existing tracks with detections taken from the people detector. There are various data association methods in the literature such as Nearest Neighbor (NN), Probability Hypothesis Density and Joint Probabilistic Data Association [48]. However in [49], authors state that these methods perform similar in environments with high complexity when tracking from a mobile robot. Therefore, we implement Nearest Neighbor method for data association due to its low computational complexity. We use Hungarion (or Munkres) method [50] with NN to solve the complete linear assignment problem between existing tracks and detections. For the calculation of the cost matrix we use Mahalanobis distance between prediction and detection to include uncertainty.

The difference between measurement z and measurement prediction $\hat{z}$ is called as innovation. To calculate Mahalanobis distance we use innovation (Eqn. 3.13), innovation covariance (Eqn. 3.14) and squared Mahalanobis distance (Eqn. 3.15).

$$v_{ij}(k) = z_i(k) - \hat{z}_j(k) \tag{3.13}$$

$$\hat{S}_{ij}(k) = H(k)\hat{P}_j(k+1|k)H^T(k) + R_i(k) \tag{3.14}$$

$$d_{ij}^2 = v_{ij}(k)^T \hat{S}_{ij}(k)^{-1} v_{ij}(k) \tag{3.15}$$

Pairings supplied by the Hungarian method are accepted if the Mahalanobis distance is smaller than a threshold. We select this threshold to be 0.5 empirically. If the distance is larger, we create a new track for the detection since it does not match with existing tracks.

3.3.4 Update

If data association is successful for a track and detection pair, we update the state of the track in the last step of the filter. Kalman gain is calculated (Eqn. 3.16), state and state covariance matrix are updated (Eqn. 3.17, 3.18) using this Kalman gain.

$$K(k) = \hat{P}(k+1|k)H^T(k)\hat{S}(k)^{-1} \tag{3.16}$$

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k)v(k) \tag{3.17}$$

$$\hat{P}(k+1|k+1) = (I - K(k)H(k))\hat{P}(k+1|k) \tag{3.18}$$

After this update we have a better state estimation incorporating measurement and prediction in a weighted manner.

3.3.5 Track Management

Creation of new tracks, update of existing tracks and removal of old tracks are managed depending on certain criteria. We have four labels assigned to tracks according to their states; appearing, live, disappearing and terminated (Fig. 3.2). Only tracks with the label “live” are accepted as real people for further processing.

When we have an unassociated detection, we create a new track and label it as "appearing". We do not regard this track as "live" track until we get same detection for 2 seconds. If not, we terminate this track. Similarly, if we have an unassociated track, we label it as "disappearing". If it does not get associated for 5 seconds, we terminate the track. If it is associated during this time, it switches to “appearing” state and the same procedure is applied as a new track generation. Since terminated tracks are regarded as tracks which are not recoverable, they are removed from the track list in the next cycle.

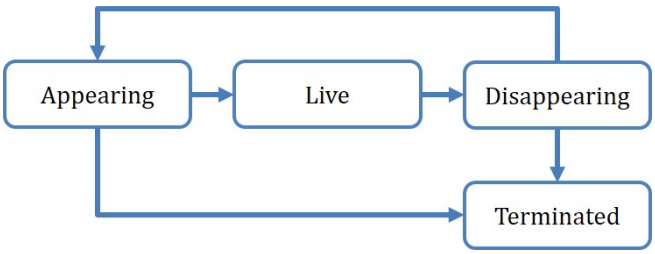


Figure 3.2: Possible routes for track states.

3.4 Experiments

Human-aware path planning depends on correct tracking of people around. Our tracker implementation on top of our detector performs satisfactorily by visual inspection. Yet, we have validated the performance also by comparing the results with ground truth position data provided in KTP dataset.

In scenes with people having abrupt velocity changes, or people leaving the scene and returning back, our tracker may switch track IDs or create new tracks. However, for path planning, trajectories of the closest people to the robot are more important since we do not deal with a recognition problem. Also, the time span needed is generally a few seconds. Therefore, these situations do not cause problems in the general navigation method.

3.4.1 Results Obtained Using KTP Dataset

KTP dataset contains five different scenarios (Fig. 3.3) as,

1. back and forth: a person walking back and forth respect to the camera,
2. random walk: three people walking with random trajectories for about 20 seconds,
3. side by side: two people walking side-by-side with a linear trajectory,
4. running: one person running across the room,
5. group: five people who gather in a group and then leave the room.

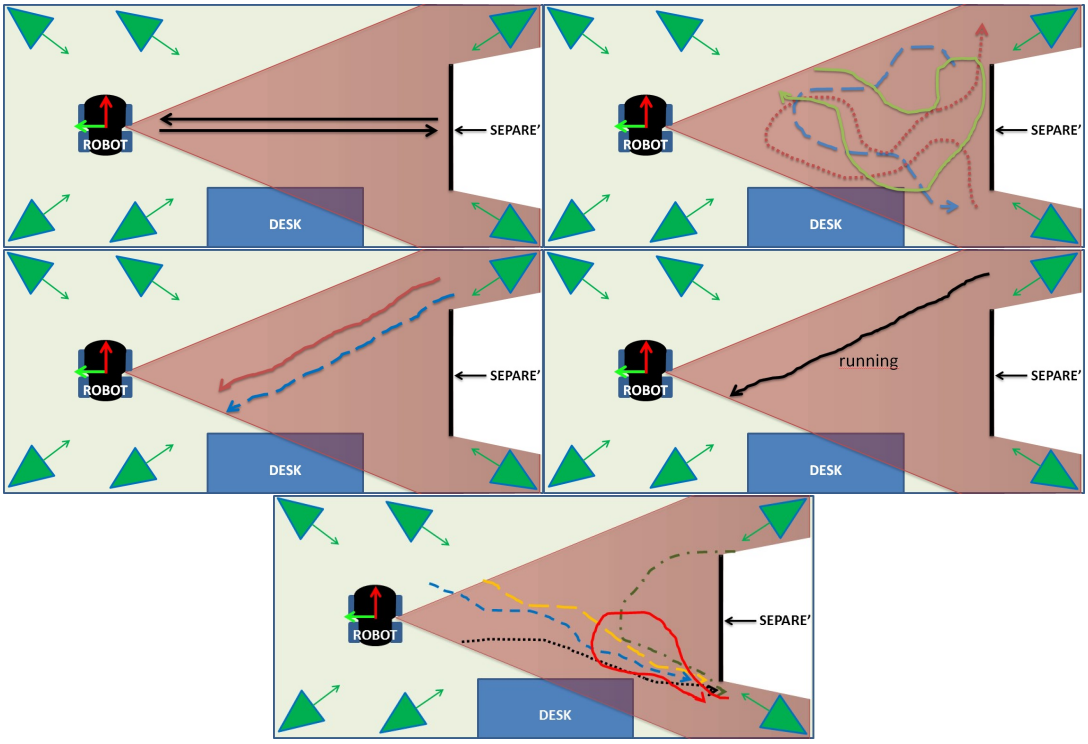


Figure 3.3: Five different scenarios present in KTP dataset [22]

Among these five scenarios we used "random walk" (Fig. 3.4) case since it is the most similar case to our target scenario which we assume there will be multiple people walking around randomly.

This scenario includes three people walking around randomly. Our detection and tracking system is expected to detect all three people in the scene and track them till the end. This scene takes about 25 seconds to complete. There are strong occlusions in some parts of the dataset. When we run our program against this scene, it detects three people and labels them as P0, P1 and P2 (Fig. 3.5). During the scene, it can perfectly track each individual with correct labeling. Also it does not fail to track even in persistent full occlusions.

During the tests, we recorded our tracker output. At each cycle of the algorithm, we recorded the ID of the tracked person with ground plane coordinates. KTP dataset provides ground truth position information of labeled people in the scene. 3D positions of people have been obtained by a motion capture system operating by detecting an infrared marker placed on each person's head in the scene. However, they have re-

Figure 3.4: A screenshot from scenario three, random walk.

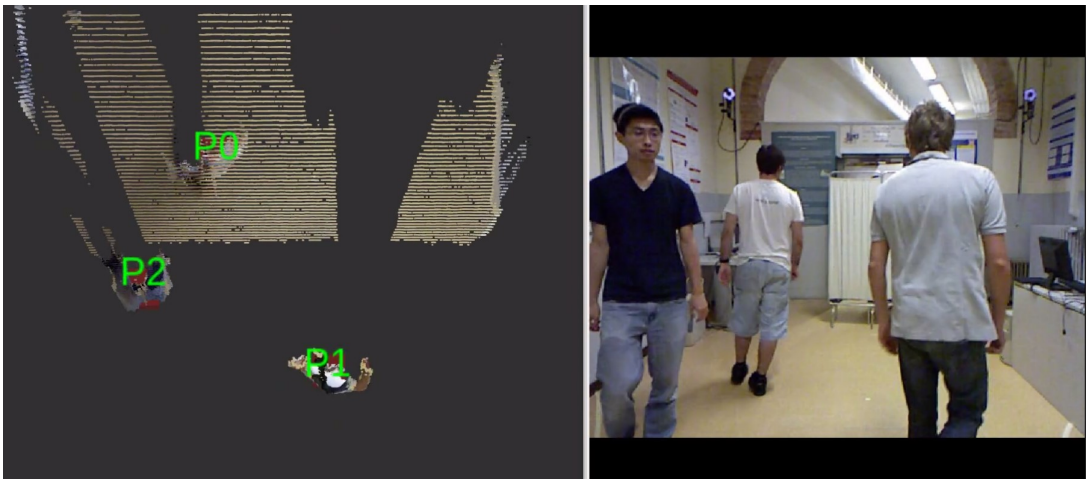


Figure 3.5: Tracker in action. It detects and tracks three people in the scene. Top view and labels are provided on the left.

Table 3.1: Mean and standard deviations of differences between trackings and true positions for three people in Fig. 3.5.

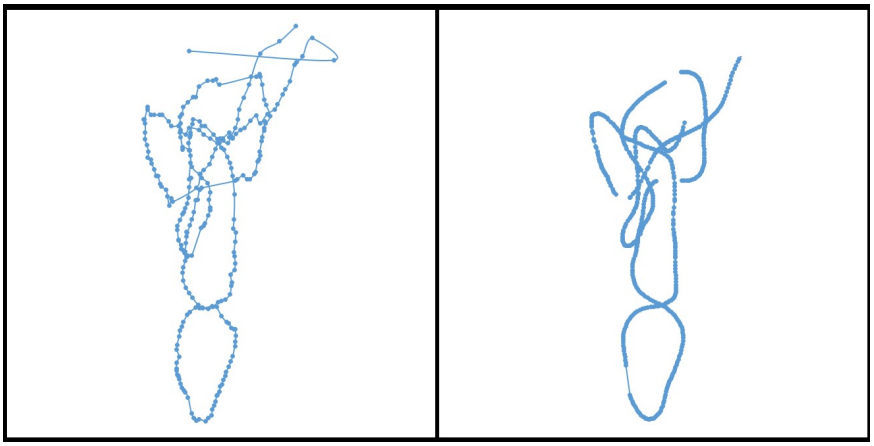
Person	**Mean**	**St. Dev.**
P0	0.192	0.170
P1	0.246	0.167
P2	0.206	0.119

moved ground truth data of people if they are occluded in 2D image. Therefore, we have unconnected paths in ground truth position data. After running a tracking session, we compared our tracker results with ground truth positions (Fig. 3.6a-b-c). As it can be seen in Fig. 3.6 and in Table 3.1, our system tracks three people in random walk scenario satisfactorily. The mean differences between trackings and true positions are at about 0.2 meters. The main reason for these differences is different tracking points of the dataset and our method. KTP dataset records positions by tracking infrared markers centered on human head-tops. However, in our method, due to the nature of the sensor, data from RGB-D sensor provides only frontal surface points. Since we track centroids of point clouds, this half surface shifts the centroid towards the sensor. Density of points is different because the dataset is recorded at 30 Hz while our system with tracking is operating just above 20 Hz due to overhead of the algorithm.

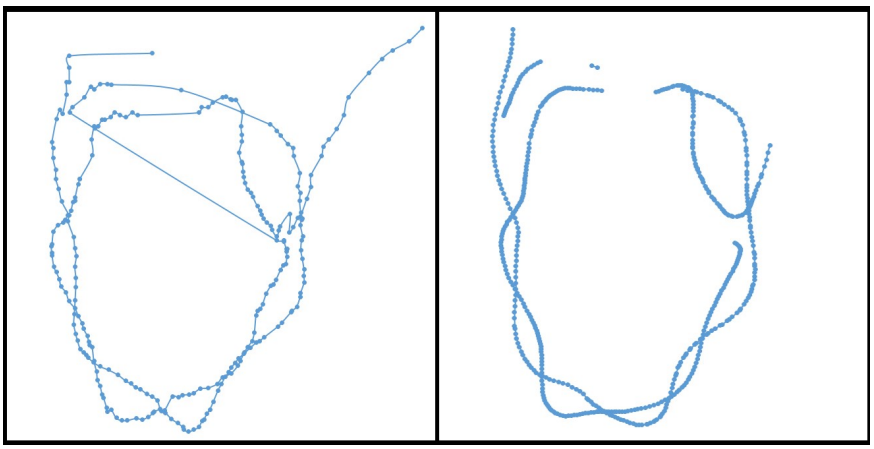
3.5 Conclusions

In this chapter, the people tracking component is implemented. Detected people information from the previously introduced detector is used as the input. Using constant velocity motion model, a Kalman filter is implemented. Data association stage is carried out with using a nearest neighbor matching algorithm coupled with Munkres (or Hungarian) algorithm.

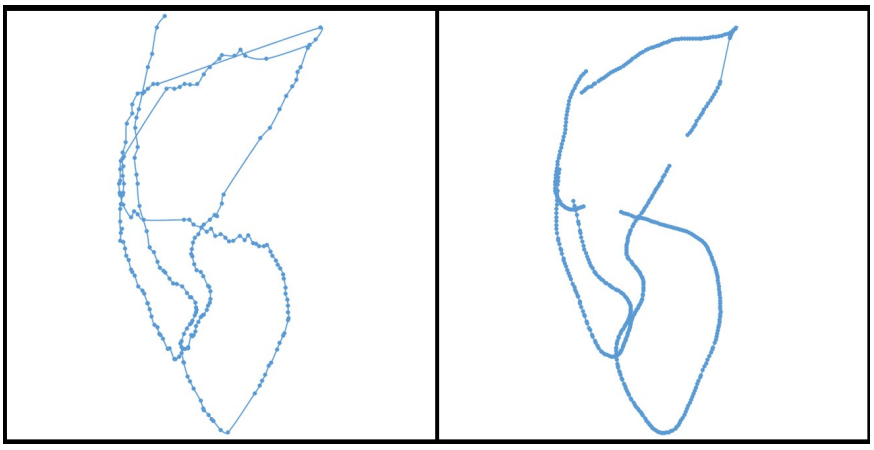
With constant velocity motion model, long occlusions and abrupt direction changes become difficult to track. However, in the context of this study, putting large effort



(a)



(b)



(c)

Figure 3.6: Tracked path (left) and ground truth path (right) of person P0 (a), P1 (b) and P2 (c)

onto solving these problems would be redundant because navigation of a mobile robot is more related with people in close vicinity. In other words, few identity changes or renewals do not pose a problem since what we are trying to solve is not a recognition problem. Accurately tracking people for last few seconds is sufficient for predicting future trajectories because as trajectory information gets old, its influence on the future trajectory also gets lowered.

Testing of the implemented tracking algorithm is again carried out with the previously developed detector on KTP dataset. Tracking information generated by the tracker is recorded and compared with the ground truth positions in the dataset. Results are satisfactory for using the method as a component in development of a human-aware navigation method.

CHAPTER 4

TRAJECTORY PREDICTION AND PATH PLANNING

4.1 Introduction

The main scope of this doctoral study is to develop a human-aware navigation system for mobile robots. The first phase was developing a robust people detection method. The second phase was about tracking the detected people which we have managed by implementing constant velocity motion model with Kalman filter. The third phase of the study is to develop a path planning method regarding the information gathered from the people tracking phase. As the first step in this section, we define and explain the problem, namely "*the freezing robot problem*". Then, we have implemented a classical planning method to demonstrate the problem, see the drawbacks clearly. After providing the related work on the topic, we explain the proposed method and test the method on a public dataset. Results are compared with other state-of-art methods and shown to be superior in terms of different evaluation metrics.

4.2 The Freezing Robot Problem

In dynamic environments, especially crowded with people, it is a challenging task for a robot to navigate to a goal position. As we aimed with this study, a robot should be able to predict the trajectories of other agents at least in close vicinity. When this prediction is carried out up to a certain extent, an uncertainty grows at each step. This *uncertainty explosion* is one of the reasons for preventing robots from generating reasonable paths for navigating among humans (Figure 4.1). This is called as *the freezing robot problem* [51].

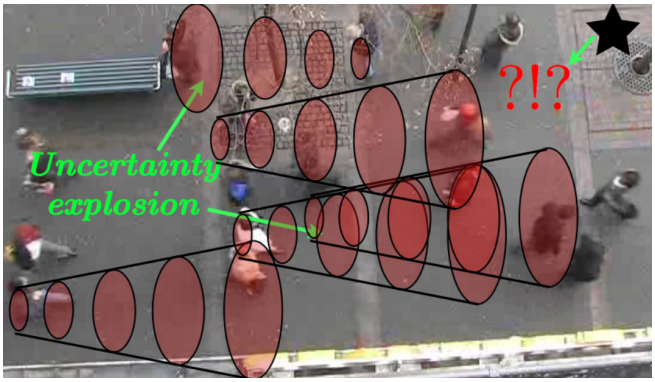


Figure 4.1: Uncertainy explosion causing the robot to freeze [52].

Although uncertainty explosion is the primary reason behind the freezing robot problem, it is shown that even in perfect knowledge of individual trajectories (or with insignificant uncertainty), problem still occurs in certain crowd configurations. The reason behind this is the lack of human cooperation in path planning models. For example, when people walk shoulder to shoulder any independent planner will assume the path is blocked and will try to navigate around the crowd. However, in human navigation, even while walking shoulder to shoulder, humans cooperate and make room for others [53–55]. Lack of this model causes planners to fail or navigate through a sub-optimal path. This behavior is shown in Figure 4.2. If the crowd is dense enough, the freezing robot problem will always occur. Therefore, a human-aware navigation algorithm should incorporate human cooperation while predicting trajectories and planning paths. Usefulness of this cooperation (or joint collision avoidance) is shown in different studies like [56], [57] and [58].

4.3 Problem Demonstration

Our motivation for developing a human-aware navigation method is fed from problems of classical methods which treat humans as one of the ordinary objects in the environment. Without regarding human navigation behavior and motion prediction, these classical methods fail to plan a path to the goal in an environment crowded with humans. To be able to demonstrate these problems of classical methods, we have developed a case with using KTP dataset and a grid based A* planner.

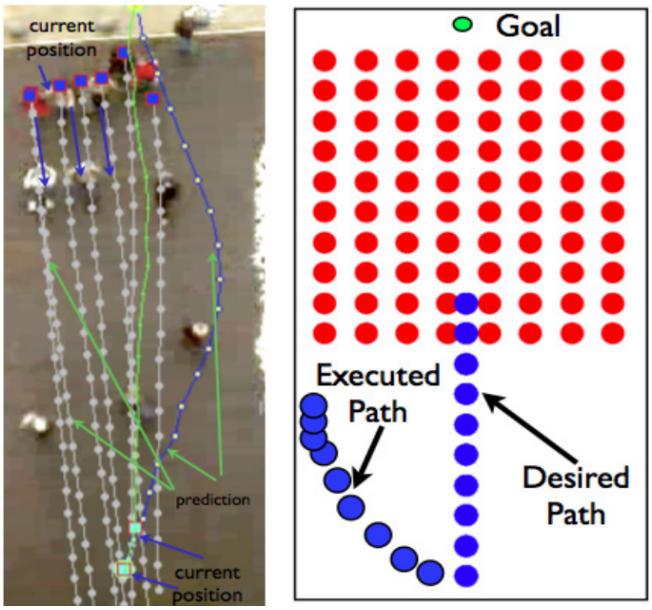
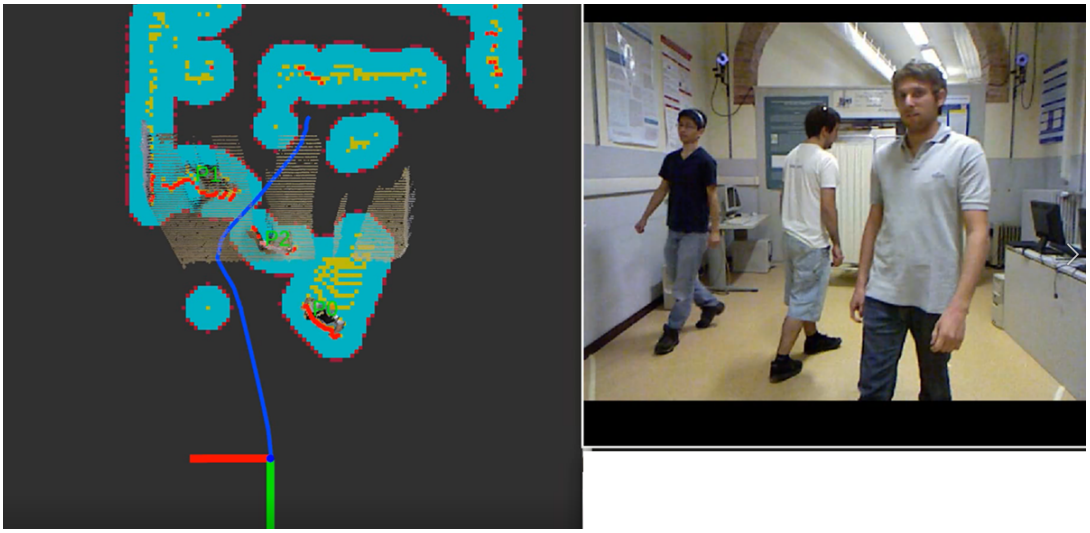


Figure 4.2: When people walk shoulder to shoulder, independent planners navigates around crowd intead of allowing them to make room in cooperation [51].

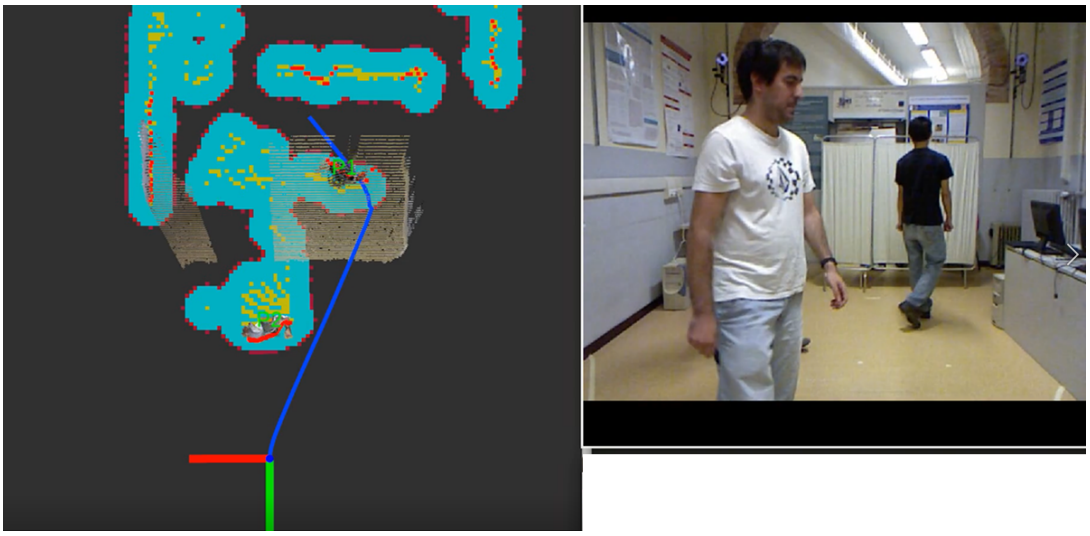
For the test of the planner, we implemented a 2D costmap which consists of grids with 0.05 m side length. This allows us to discretize the area to create plans on. Each grid has a cost varying between 0 and 255. A cost of 0 means free space and 255 means a lethal obstacle. To assign costs to grids, we generated 2D laser scan from 3D point cloud data of KTP dataset. Laser scan is simulated as if it is at the same height with RGBD camera. Laser readings of obstacles (and humans in this case) correspond to cost of 255. Costs of the surrounding area of that grid decay exponentially.

We defined a goal behind the three people moving around and A* path planner tried to plan a path to this goal starting from sensor location. As people move around randomly, costs in costmap are updated. Some of the grids which were free before became occupied. Therefore, the planner updated the plan at every instance. Even in some cases it could not find any possible path because there was no empty space left to move through. Few screenshots of movement of people and changing plans are shown in Figure 4.3.

As our test results show, classical path planning methods, if they can find any, always change the planned path due to dynamic movement of people. If all free space is occupied by people, classical methods fail to find a path to the goal. As a result,



(a)



(b)

Figure 4.3: Classical A* planning to a predefined target. Abrupt plan changes (blue lines) are observed between two different frames which are 3 seconds apart.

the robot becomes halted and as mentioned early, this is called as “freezing robot problem” in the literature [51]. We aim to solve this problem with developed methods throughout this study.

4.4 Related Work

After incorporating humans into navigation algorithms there has been a considerable amount of study that develop human motion models ([59], [60]). The underlying assumption was that if the human motion prediction is accurate enough, it will lead to a good navigation performance. Methods are developed to model goal-directed trajectories of pedestrians with collecting data from pedestrian movements [61]. In all of these studies, moving agents in the environment are modeled independently. Therefore, in dense crowds, robots fail to move which is called “the freezing robot problem” [51]. In this situation, once the environment becomes complex up to a certain level, the planner of the robot cannot find out a safe path, and the robot performs unnecessary movements or just freezes in place to avoid collisions.

According to the “social forces model” [53], people typically engage in joint collision avoidance i.e., they adapt their trajectories according to each other to make necessary space for navigation. This model suggests that human effort to avoid obstacles and to reach a goal (i.e., internal objectives) can model human motion. Even though the social forces model simulates crowds well, it is found that the model is insufficient for individual movement prediction. Especially, sudden evasive maneuvers are not captured satisfactorily with the social forces model. Therefore, Trautman developed Interacting Gaussian Processes (IGP), a non-parametric statistical model based on dependent output Gaussian processes that can estimate crowd interaction from data [51]. His model captures the non-Markov fashion of human trajectories while they are navigation to their goals.

Pradhan et al. developed a path planner to navigate in environments crowded with people [62]. Their method relies on potential fields with probabilistic data and motion information. Since their method is strongly sensitive on parameter changes, it is not applicable to general cases of the problem. In [63], Morioka used monocular

omnidirectional sequential images to gather features from the environment. These features are selected as stable features and a learning phase is implemented. In the learning phase, a human teleoperator controls a mobile robot and the robot learns the route. In the navigation phase, the mobile robot autonomously plans the path using the learned route and navigates to a position using that path. Kuderer et al. presented a model for predicting pedestrian movements using important features of the trajectories to determine the probability distribution representing human navigation behavior [64]. Learning methods rely on principle of maximum entropy in human navigation behavior. But, it is a fact that humans may behave differently to robots than to people. Yet, similarity (human likeness) in motion is accepted to be a factor that improves anthropomorphism [65].

In [66], it is argued that the navigation with obstacle avoidance in dynamic environments is a learning from demonstrations (LfD) without the use of a path planner but with proper feature sets. The main assumption of their approach is that the demonstrator (i.e., human) would always pick the same policy during the demonstration process. Kruse et al. stated that selecting the shortest path to the target destination for crossing scenarios seems to be the preferred human behavior in experiments [67]. Therefore, this behavior should be taken into account while developing algorithms for mobile robots. Kidokoro studied on a different aspect of the problem [68]. The author stated that even after operating long time, i.e. 3 years, people (especially children) still gathered around the robot. So congestion and crowd are caused by the robot itself.

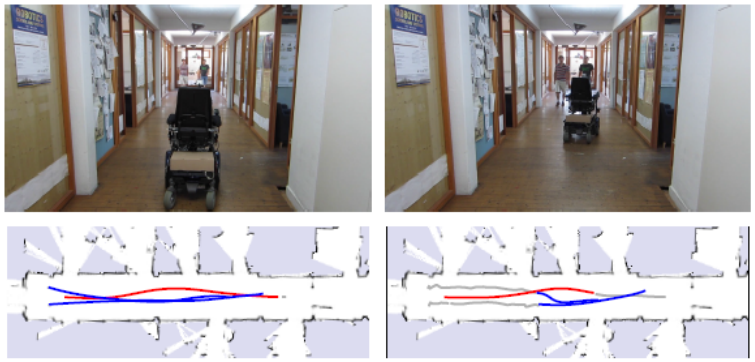
In [69], authors state that from an engineering point of view, robot navigation needs to be scalable to different crowd densities, effective in following trajectories, robust to sensor noises and errors, and not relying on any external infrastructure to navigate effectively. Also from a societal point of view, robot navigation needs to be human-friendly, predictable and safe. With these conclusions, they have developed a bio-inspired, local navigation algorithm which can generate human-aware and human-like trajectories. Trautman et al. used the concept of cooperation for navigation [52]. They argue that their experimental results, evaluated with previous results in [51], showed that a human-robot cooperation model is important for safe and efficient navigation in dense crowds. They tested their Interacting Gaussian Process (IGP) method in a

Figure 4.4: Overhead still of the crowded university cafeteria test bed in [52].

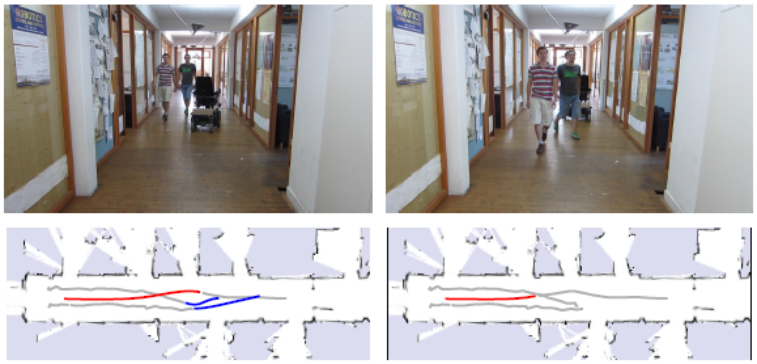
university cafeteria (Figure 4.4). IGP method assumes known final destinations (i.e., goals) for all agents. This is a strong assumption which is generally not the case except some specially designed scenes. Kuderer et al. developed another cooperative approach which uses maximum entropy inverse reinforcement learning method to compute a policy of the desired interaction behavior [70]. In the experiment, first, the pedestrians walk side by side, blocking the corridor (Figure 4.5). It is obvious that a conventional path planner would not be able to find a clear path to the goal in such a situation. In contrast, with their method, the robot expects humans to cooperatively navigate with joint collision avoidance behavior. During the encounter, the robot constantly computes the most possible cooperative scenario with the humans, which makes the wheelchair able to navigate in natural joint collision avoidance. But this method is not tested with overcrowded environments.

In the literature there is also an outdoor study on navigating in crowded environments. Kümmerle et al. developed a navigation system for urban environments [71]. Their system navigates as pedestrian-like autonomously. They also incorporated a SLAM method for dealing with large city center maps. Terrain and traversable regions are recognized for safe navigation. Tests are conducted on a 3.3 km route successfully. However their fixed 2D laser configuration requires a movement to detect objects in 3D. Therefore, it may not be applicable to different robotic configurations. For crossing road the robot cannot act autonomously since traffic lights are not enough for a safe crossing. A police car or an emergence situation may be encountered,

therefore, the robot cannot rely on traffic lights. This is another drawback of this application.



(a)



(b)

Figure 4.5: Steps of the cooperative navigation experiment in [70].

Considering dynamic obstacles as individual rational agents which maintain social relations and follow rules of social behavior is the approach that is necessary to make mobile robots share environments with humans. In this aspect, there are different approaches in the literature. One group of researchers apply models from social psychology and cognitive sciences, while the second group rely on machine learning techniques. In [72], [73], [74] and [75] human sciences models are incorporated. Mostly used models are Proxemics Theory [76], social forces [53] and back space model [77]. This approach is promising but these methods may not be applicable to human-robot interaction cases in uncontrolled conditions. On the other hand, learning methods tend to incorporate human behavior by statistical learning from observa-

tional data from humans. Ziebart et al. studied the human movement in static environments [61]. They used maximum entropy inverse reinforcement learning (IRL) to predict movements. Henry et al. addressed dynamic environments in [78]. They use IRL to make a simulated agent learn to join the flow of pedestrians using features like average flow direction and density. Another study with the maximum-entropy algorithm was introduced by Kudererer et al. [64]. To predict socially compliant trajectories, their algorithm learns a behavior model with maximum-entropy method. Kim et al. in [79] applied a Bayesian method with IRL for social navigation. They incorporated depth-augmented optical flow features as input. Vasquez et al. published an experimental comparison of IRL algorithms and features for the robot navigation in crowds [80]. They argue that IRL allows modeling the factors that motivate people's actions instead of the actions themselves. They developed evaluation metrics and methodologies as a first effort to fully benchmark this kind of techniques. In [81], authors introduced non-cooperative game theoretic approach. They solved the decision process behind human interaction behavior using the Nash equilibrium. They argue that Nash's theory is suitable for the problem because humans behave rationally in a sense that while navigating they aim to minimize their own cost. They validated their assumption for five different cost functions. This approach is applied to decision making of multiple agents passing each other. They concluded that the solution with Nash equilibria in non-cooperative games selects trajectories similar to humans. Authors also extended their work and used this method to develop a path planner using rapidly-exploring random trees (RRTs) [82]. Vemula et al. [83] model cooperative human navigation by learning interaction from real data using Gaussian processes. They use an occupancy grid in close vicinity of each agent and a goal to train and predict future velocities, thus, trajectories. This method requires known final destination information of each agent during training, which is not the case in realistic applications. Also, this assumption limits the generalizability of the approach, since the model needs to be trained for each different environment.

Deep learning methods are also used in crowd prediction problem. Alahi et al. [84] proposed a method incorporating Long-Short Term Memory (LSTM) for human trajectory prediction. They used one LSTM for each trajectory also exchanging information among LSTMs through a pooling layer. For this purpose, they collect millions of

training samples [85]. Since the method does not consider uncertainty, it is possible to have assertive predictions resulting in unsafe robot navigation among humans. Also, in [86], Trautman asks the question of *"can social navigation be learned?"* and concludes that due to the combinatorics of social navigation explained in [86] and [87], it becomes infeasible to have tractable inference about trajectories especially without exploiting the structure of the space.

4.5 Cooperative Trajectory Prediction

Our main idea in this thesis is to develop a human-aware navigation method for mobile robots. This human awareness includes joint collision avoidance. We base our study on the idea of joint collision avoidance of humans which is thoroughly showed to be existent in human navigation. Without regarding this cooperation among humans, human trajectory prediction and mobile robot path planning studies fail to represent human-likeness.

Postulate 1: *There is a joint collision avoidance and cooperation between humans while navigating [51, 88].*

Since we aim to solve human-aware robot navigation problem as our ultimate goal, modeling the mobile robot as one of other agents is required. Questions may arise on human reaction and cooperation with a mobile robot instead of another human. However, it is clear in the literature that if a mobile robot acts similar to a human while navigating, it is accepted as a human-like rational agent by other humans. This can be stated as,

Postulate 2: *Similarity (human-likeness) in motion increases the scale of anthropomorphism [65].*

We conclude that, if we can generate human-like trajectories, we can increase the perception of anthropomorphism so that, a mobile robot can be treated as just another human.

Previously described people detection and tracking algorithms provide a trajectory history for each agent on the scene. We use these past trajectories to estimate ref-

erence future trajectories. Using a cost based method on occupancy grid maps, we update these trajectories at each future step. At the end, we generate a collision-free, cooperative trajectories for each agent (including the robot). Details of each step are given on upcoming sections. We have tested our method on a simulation environment before experiments on real robot platform. Results also show that our model-based cooperative path planner performs superior to comparable methods in the literature.

4.5.1 Human-like Trajectories

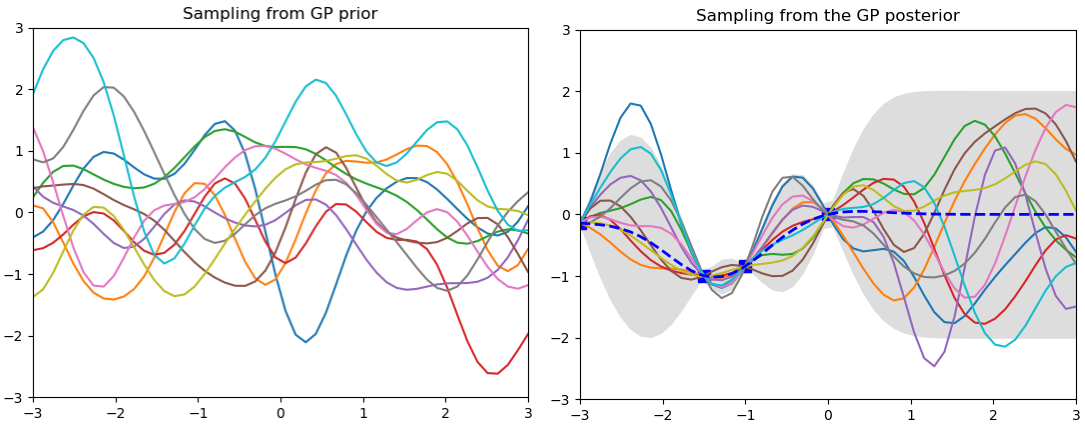
We start our pipeline with generating human-like trajectories for each agent based on their trajectory before the processed time step. Humans generally move on smooth paths minimizing their path length but avoiding collision while maintaining their speed [81]. We state this behavior as:

Postulate 3: *People move on smooth paths minimizing their path length to a goal [81, 89, 90].*

This composite nature of inherent human path planning brings non-parametric methods forward for trajectory prediction. At this point, Gaussian Process (GP) can represent or help to generate human-like trajectories if kernel functions are selected properly and hyper-parameters are tuned accordingly, as in [91], [92] and [88].

4.5.1.1 Gaussian Processes

A Gaussian Process (GP) defines a prior distribution over functions. It is a stochastic process relating each point in input space with a target variable that is normally distributed. Every finite subset of random variables is in the form of multivariate normal distribution [93–97]. Prior distribution can be converted into a posterior over functions once we gather data and condition GP accordingly. GP is analogous to a Gaussian distribution over random variables. Scalar random variables are defined by a mean and a variance, and vectorial random variables are defined by a mean vector and a covariance matrix. A Gaussian Process can be perceived as a distribution over functions, defined by a mean function $\mu(\mathbf{x})$ and a covariance $\mathbf{\Sigma}(\mathbf{x})$ with $\Sigma_{ij} = \kappa(\mathbf{x_i}, \mathbf{x_j})$,



(a) (b)

Figure 4.6: Ten samples from GP prior (a) and posterior (b). Note decrease in variances (shaded area) around observed points at $-3, -1.5, -1$ and 0.

where κ is a positive definite kernel function for two inputs of x_i and x_j. We can denote prior distribution over function $f(\mathbf{x})$ by

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')) \tag{4.1}$$

where,

$$m(\mathbf{x}) = \mathbb{E}\left[f(\mathbf{x})\right] \tag{4.2}$$

$$\kappa(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))\right] \tag{4.3}$$

It is a common practice to set $m(\mathbf{x}) = 0$, and still, model the mean arbitrarily without loss of generality. If we have noiseless observations, we will have the joint distribution of

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(\mathbf{x}) \\ m(\mathbf{x}_*) \end{pmatrix}, \begin{pmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^{\mathbf{T}} & \mathbf{K}_{**} \end{pmatrix}\right) \tag{4.4}$$

If we have n sized observations for training the GP and n_* sized test set, $K = \kappa(\mathbf{x}, \mathbf{x})$ will have size of $n \times n$ while $\mathbf{K}_* = \kappa(\mathbf{x}, \mathbf{x}_*)$ is $n \times n_*$ and $\mathbf{K}_{**} = \kappa(\mathbf{x}_*, \mathbf{x}_*)$ is $n_* \times n_*$. We can write the posterior as,

$$p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{x}, \mathbf{f}) = \mathcal{N}(\mathbf{f}_*|\mu_*, \mathbf{\Sigma}_*) \tag{4.5}$$

$$\mu_* = m(\mathbf{x}_*) + \mathbf{K}_*^{\mathbf{T}}\mathbf{K}^{-\mathbf{1}}(\mathbf{f} - m(\mathbf{x})) \tag{4.6}$$

$$\mathbf{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^{\mathbf{T}}\mathbf{K}^{-\mathbf{1}}\mathbf{K}_* \tag{4.7}$$

If we incorporate noise for observations, we can define

$$y = f(\mathbf{x}) + \epsilon \tag{4.8}$$

and the new joint distribution as,

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K_y} & \mathbf{K}_* \\ \mathbf{K}_*^{\mathbf{T}} & \mathbf{K}_{**} \end{pmatrix}\right) \tag{4.9}$$

where $\mathbf{K_y} = \mathbf{K} + \sigma_y^2\mathbf{I}$. Note the zero mean and the noise term, $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$. Now, we can update the posterior density,

$$p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{x}, \mathbf{y}) = \mathcal{N}(\mathbf{f}_*|\mu_*, \mathbf{\Sigma}_*) \tag{4.10}$$

$$\mu_* = \mathbf{K}_*^{\mathbf{T}}\mathbf{K}_{\mathbf{y}}^{-\mathbf{1}}\mathbf{y} \tag{4.11}$$

$$\mathbf{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^{\mathbf{T}}\mathbf{K}_{\mathbf{y}}^{-\mathbf{1}}\mathbf{K}_* \tag{4.12}$$

If we reduce the case to single input, posterior mean becomes $\bar{f}_* = \mathbf{k}_*^{\mathbf{T}}\mathbf{K}_{\mathbf{y}}^{-\mathbf{1}}\mathbf{y}$ and covariance becomes $\mathbf{k}_{**} - {\mathbf{k}_*}^T\mathbf{K}_{\mathbf{y}}^{-\mathbf{1}}\mathbf{k}_*$. Instead of inverting $\mathbf{K}_{\mathbf{y}}^{-\mathbf{1}}$ directly, we use Cholesky decomposition as $\mathbf{K_y} = \mathbf{L}\mathbf{L}^{\mathbf{T}}$ due to numerical and computational concerns. Then, the posterior function becomes $f_* \sim \mu + \mathbf{L}N(0, \mathbf{I})$ and calculating mean and variance reduces to Algorithm 1 [93].

Algorithm 1 GP regression

1: $\mathbf{L} = cholesky(\mathbf{K} + \sigma_y^2\mathbf{I})$
2: $\alpha = \mathbf{L}^{\mathbf{T}} \setminus (\mathbf{L} \setminus \mathbf{y})$
3: $\mathbb{E}[f_*] = \mathbf{k}_*^{\mathbf{T}}\alpha$
4: $\mathbf{v} = \mathbf{L} \setminus \mathbf{k}_*$
5: $var[f_*] = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^{\mathbf{T}}\mathbf{v}$

An example of prior and posterior sampling can be seen in Figure 4.6. Before gathering any observations and training the GP, ten samples are drawn. However, after

training (i.e. conditioning) the GP with four points ($-3, -1.5, -1.0$ and 0), another ten samples are drawn and it is seen that variance is bounded around these points. Since we plan paths for a mobile robot together with humans, we use two dimensional world with x and y coordinates. Therefore, we model two Gaussian processes for x and y separately for simplicity.

In contrast to [91] and [98], we use time steps as our finite points set similar to [88], i.e., $\mathbf{f}(t) : t \in [1, T]$. Trajectory up to time T for agent i is expressed as

$$\mathbf{f}_{1:T}^{(i)} = \left(\mathbf{f}^{(i)}(1), \mathbf{f}^{(i)}(2), \ldots, \mathbf{f}^{(i)}(T)\right) \tag{4.13}$$

where

$$\mathbf{f}^{(i)}(t) = (x(t), y(t)) \in \mathbb{R}^2$$

$$i \in \{1, 2, \ldots, N\}$$

Up to a time step t, we have observations for each agent as

$$\mathbf{z}_{1:t} = \left(\mathbf{z}_{1:t}^{(1)}, \mathbf{z}_{1:t}^{(2)}, \ldots, \mathbf{z}_{1:t}^{(N)}\right) \tag{4.14}$$

Then, the probability density is

$$p\left(\mathbf{f}_{1:T} \mid \mathbf{z}_{1:t}\right) \tag{4.15}$$

which represents distribution over each agent's trajectory. Random function (or trajectory), $\mathbf{f}^{(i)}$, can be represented via Gaussian process as

$$\mathbf{f}^{(i)} \sim GP\left(m^{(i)}, k^{(i)}\right) \tag{4.16}$$

where locations in different time steps, t and t', are related with the kernel function $k(t, t')$. Structure of this kernel function defines the behavior of resultant trajectories. To be consistent with the literature, we use similar kernel function with [88]. However, we do not consider joint distribution like $p(\mathbf{f}_{1:T}^{(R)}, \mathbf{f}_{1:T} \mid \mathbf{z}_{1:t})$ as in the literature, since we model the robot just as another agent (Postulate 2). This density function does not incorporate interaction and cooperation. It just suggests that if we know the measurements up to time t, we can predict each agent trajectory up to a final time T. This is not the case in human navigation due to necessity of interaction and cooperation modeling. Therefore, we need to model an interaction between each agent (including the robot). Trautman et al., with IGP [88], incorporated this interaction

by using an interaction potential. Use of this potential term results in non-Gaussian models which can not be sampled directly. Also, IGP strongly depends on known-goal assumption to be able to infer IGPs up to time T. Another drawback of IGP is that, large number of samples are drawn and weighted with an importance sampling approach which results in computationally ineffective prediction cycles. Since candidate trajectories are sampled already without any interaction process, resultant paths cannot satisfactorily cover abrupt avoidance maneuvers of pedestrians.

Instead of sampling trajectories from Gaussian process posteriors as predictions, we implicitly model interactions (and cooperation) as future measurements again for conditioning a temporary duplicate of GP of each agent.

$$p\left(\mathbf{f}_{1:T} \mid \mathbf{z}_{1:t}, \tilde{\mathbf{z}}_{t+1:T}\right) = GP\left(m_T^{(i)}, k_T^{(i)}\right) \tag{4.17}$$

Future measurements, $\tilde{\mathbf{z}}_{t+1:T}$, are generated using a cost-based interaction between all agents. Note that, $\tilde{\mathbf{z}}_T$ designates final destinations of agents. If known, they can be used to condition GPs for better prediction, but our model does not rely on known goal assumption. This density allows us to reduce planning to inference as,

$$\underset{\mathbf{f}_{1:T}}{\operatorname{argmax}}\, p\left(\mathbf{f}_{1:T} \mid \mathbf{z}_{1:t}, \tilde{\mathbf{z}}_{t+1:T}\right) \tag{4.18}$$

4.5.1.2 Kernel Functions

Gaussian process kernel is a real-valued function that relates two inputs, say $\mathbf{t}$ and $\mathbf{t}'$, i.e., $\kappa(\mathbf{t}, \mathbf{t}') \in \mathbb{R}$. It may be stated as measure of similarity between two objects. In other words, if the kernel relates $\mathbf{t}$ and $\mathbf{t}'$ as similar, we expect the output of the function at these points to be similar too. In Figure 4.6b, one can see that as we move towards training points we get values similar to them also. This is the visualization of the similarity measure property of kernel functions. Kernels must be positive-definite by definiton (i.e., $\kappa(\mathbf{t}, \mathbf{t}') = \kappa(\mathbf{t}', \mathbf{t})$ and $\kappa(\mathbf{t}, \mathbf{t}') \geqslant 0$).

There are some commonly used kernels in the literature. Squared-exponential (SE) kernel, Matern kernel and linear kernel are three of these most widely used kernels.

SE kernel is defined as,

$$\kappa(\mathbf{t}, \mathbf{t}') = exp\left(-\frac{1}{2}(\mathbf{t} - \mathbf{t}')^T \mathbf{\Sigma^{-1}} (\mathbf{t} - \mathbf{t}')\right) \tag{4.19}$$

or for single dimensional input,

$$\kappa(t, t') = exp\left(-\frac{(t - t')^2}{2l^2}\right) \tag{4.20}$$

Here, l is the characteristic length scale (or hyper-parameter). Lengthscale parameter specifies the width of the kernel and the smoothness of the samples.

Matern kernel can be stated as generalized form of SE kernel. Its general form is,

$$\kappa(\mathbf{t}, \mathbf{t}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}||\mathbf{t} - \mathbf{t}'||}{l}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}||\mathbf{t} - \mathbf{t}'||}{l}\right) \tag{4.21}$$

and $\nu > 0, l > 0$ where K_ν is a modified Bessel function. ν controls the smoothness of the samples, i.e., as ν gets smaller, paths get rougher. If $\nu \to \infty$, Matern kernel approaches to SE kernel. We will use $\nu = 5/2$ and for single input, Matern kernel will be,

$$\kappa(t, t') = s\left(1 + \frac{\sqrt{5}(t - t')}{l} + \frac{5(t - t')^2}{3l^2}\right) exp\left(-\frac{\sqrt{5}(t - t')}{l}\right) \tag{4.22}$$

with s and l being hyper-parameters of the kernel.

Linear kernel is defined as,

$$\kappa(\mathbf{t}, \mathbf{t}') = \mathbf{t}^T \mathbf{t}' \tag{4.23}$$

or in single input form,

$$\kappa(t, t') = t \cdot t' \tag{4.24}$$

To account for noises of measurements, generally a noise kernel is incorporated,

$$\kappa(\mathbf{t}, \mathbf{t}') = \sigma^2 \delta(\mathbf{t}, \mathbf{t}') \tag{4.25}$$

We aim to generate human-like trajectories using GPs. The shape of trajectories is defined primarily by the kernels used and their parameters. Among commonly used kernels; we use the linear kernel since people tend to move on a straight line to a goal if there are no obstacles, the Matern kernel to account for abrupt directional changes

due to obstacles and other elements, and the noise kernel to incorporate noises from measurements, etc. Summing these three kernels gives us the final composite kernel as,

$$\kappa(t,t') = t \cdot t' + s\left(1 + \frac{\sqrt{5}(t-t')}{l} + \frac{5(t-t')^2}{3l^2}\right) exp\left(-\frac{\sqrt{5}(t-t')}{l}\right) + \sigma^2\delta(t,t') \tag{4.26}$$

Squared-exponential kernel is a stationary kernel [93]. Therefore, it tends to move towards the mean wherever there is no data. This behavior does not resemble a human trajectory yet it may be used for velocities as in [91] and [98].

4.5.1.3 Hyper-parameters

Hyper-parameters are other factors defining the shape of trajectories. Modifying these parameters will vary the output significantly. Therefore, we need to tune them for human-like trajectories. One way of determining hyper-parameters is learning from a training data. If we have a set of input and outputs as t and z pairs, we can optimize hyper-parameters and also validate the kernel functions we use. To allow faster optimization, we consider maximizing the *marginal likelihood.*

If θ is a vector of hyper-parameters, we can write log marginal likelihood as,

$$\log p(\mathbf{z}|\mathbf{t},\theta) = -\frac{1}{2}\log|\mathbf{K}_z| - \frac{1}{2}(\mathbf{z}-\mu)^T\mathbf{K}_z^{-1}(\mathbf{z}-\mu) - \frac{t}{2}\log(2\pi) \tag{4.27}$$

Note that, $\mathbf{K_z} = \mathbf{K} + \sigma^2_{noise}\mathbf{I}$. The first term of log marginal likelihood, $-\frac{1}{2}\log|\mathbf{K}_z|$ is called a complexity penalty term. As the model gets complex, this term applies penatly, i.e., $\mathbf{K}_z$ becomes almost diagonal and $\log|\mathbf{K}_z|$ becomes large. Second term, $-\frac{1}{2}(\mathbf{z}-\mu)^T\mathbf{K}_z^{-1}(\mathbf{z}-\mu)$ is data fit measure while the last term, $-\frac{t}{2}\log(2\pi)$ is log normalization term.

For maximizing the log likelihood, we use partial derivatives with respect to hyper-parameters,

$$\frac{\partial}{\partial\theta_j}\log p(\mathbf{z}|\mathbf{t},\theta) = \frac{1}{2}\mathbf{z}^T\mathbf{K}_z^{-1}\frac{\partial\mathbf{K}_z}{\partial\theta_j}\mathbf{K}_z^{-1}\mathbf{z} - \frac{1}{2}\operatorname{tr}\left(\mathbf{K}_z^{-1}\frac{\partial\mathbf{K}_z}{\partial\theta_j}\right) \tag{4.28}$$

After having derivative of log marginal likelihood, hyper-parameters can be optimized using a gradient based optimizer.

4.5.1.4 Goal Estimation

Generating realistic, human-like trajectories depends on goal selection also. If the goal position is known for an agent, Gaussian process can be trained with goal position associated with last time step as input. This significantly increases the accuracy of generated trajectories. Trautman et al. assumes that a goal is known for each agent [88]. For example, if a scene includes a narrow corridor, the inlet and the outlet to the corridor can be accepted (with relatively high uncertainty) as goal positions depending on the moving direction of agents. However, this is generally not the case for a generalized navigation scene. At this point, goal estimation techniques are present in the literature. However, accurate goal estimation is not in the scope of this study. Therefore, we use a very simple goal estimation using past velocities of agents for a short prediction horizon.

$$\tilde{\mathbf{v}}^{(i)} = \frac{1}{n} \sum_{j=0}^{n-1} \mathbf{v}_{t-j}^{(i)} \tag{4.29}$$

$$\mathbf{z}_T^{(i)} = \tilde{\mathbf{v}}^{(i)} h \tag{4.30}$$

$\tilde{\mathbf{v}}$ being the estimated velocity for agent i using last n measurements, h being the prediction horizon (seconds) and $\mathbf{z}_T^{(i)}$ being goal position (i.e., measurement at last time step T).

As agents move, the planner updates the plan at high frequency, therefore, goal position constantly changes (i.e., brought towards the real goal). Accordingly, GP is trained again at each step using the new goal position estimation (without the previous one).

4.5.1.5 Gaussian Process Prior Testing

In this test, our purpose is to generate reference trajectories in the absence of any other agents or obstacles. Assumption is that; we can generate trajectories for people

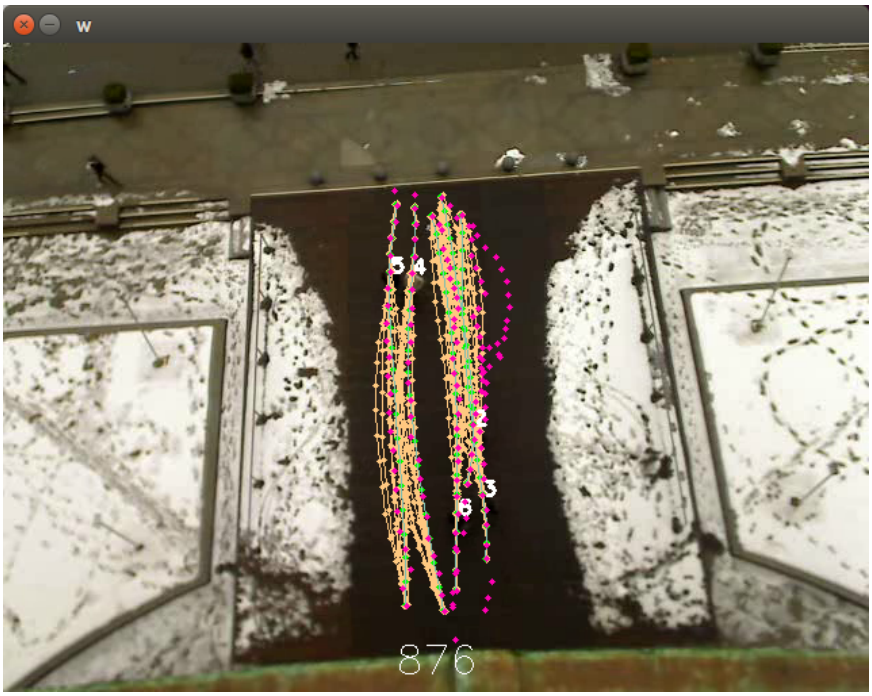


Figure 4.7: Ten samples generated for each pedestrian.

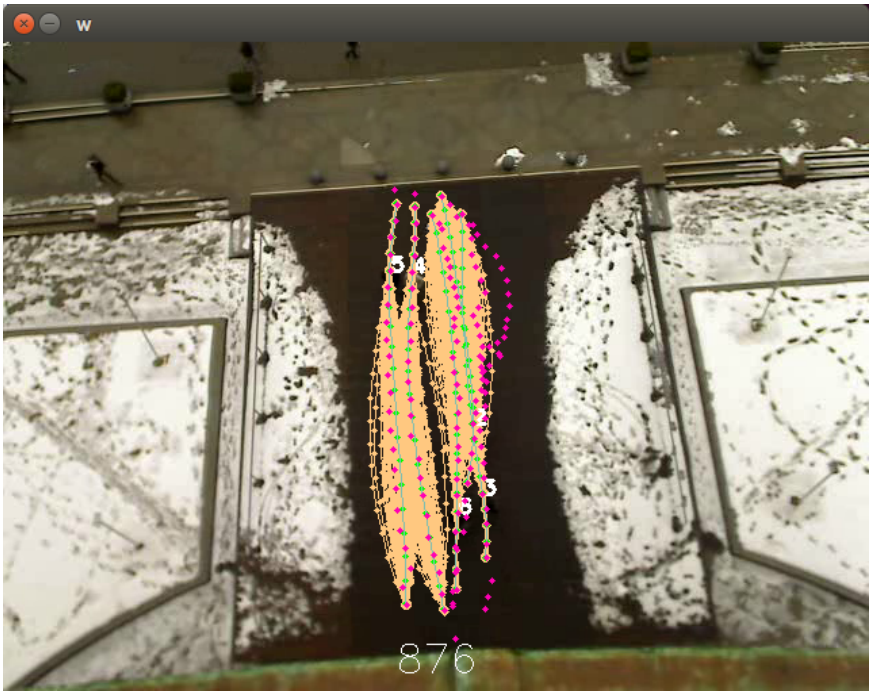


Figure 4.8: Hundred samples generated for each pedestrian.

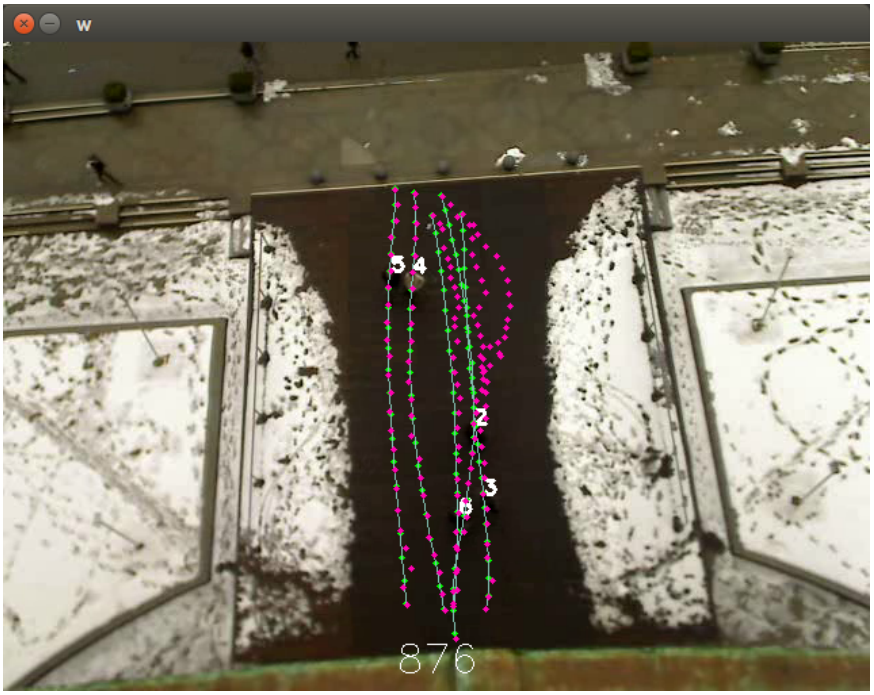


Figure 4.9: The means of samples (green) and actual paths (pink) are shown.

without cooperation if there is no obstacle (or another human) on the scene between the agent and its goal. Then, we use these reference trajectories for assigning costs during cooperation.

We test selected kernels and optimized hyper-parameters on a dataset [57]. ETH BIWI Walking Pedestrians dataset consists of hand-annotated frames with people moving in and out from a library. As goal positions are known already, we used these information also while training GPs. Figure 4.7 shows ten samples from posteriors of each agent. When this sample size is increased (for example to hundred as in Figure 4.8), it is nearly guaranteed that one of the samples coincides with the real trajectory in expense of serious computational resource. Even the mean of GPs may represent the true trajectories (Figure 4.9, agents 4 and 5). However, this is of course not the case when there exists an obstacle or moving agent crossing the trajectory. In [88], authors use an interaction potential function to weight samples with a measure of inter-agent distance and path length.

4.5.2 Individual Cost Hypotheses

Each agent on the scene has its own belief (hypothesis) about upcoming poses of other agents and adjusts its own trajectory accordingly. This is somehow an implicit handshake between people. Other people's past trajectories and especially last velocity vectors give insight about their possible future trajectories. If they tend to intersect our planned trajectory at a future time step, we change our course accordingly. Reciprocal Velocity Obstacles [99] which is extended from the Velocity Obstacles [100] emerges from a similar idea however, amount of noise in human navigation causes erratic behaviors. A simple but similar approach to ours is taken in [101] with use of potential fields. They generate potential fields depending on a few factors like proxemic distances, distance to goal, etc. Then, they use RRT ([102, 103]) for finding a path to the goal. However, this planner is not cooperative, i.e., does not account for other people to cooperate with the robot for better navigation for all. Our method defines a costmap for each agent and each agent considers predictive cooperation while predicting others and its own trajectory.

4.5.2.1 Costmaps

Occupancy grid represents a grid structure with value of each grid being the probability that there is an obstacle there [104, 105]. It is used extensively for mobile robot localization research [106,107]. In costmaps, instead of probability of being occupied, a cost value generated from various inputs is used for each grid. These costmaps can be binary (occupied or not) or can represent more complex cost assignments. Using different cost assignment methods, one can put constraints on traversable area [108,109].

We use byte-type for our grid structure as in ROS navigation stack [6] (Figure 4.10). Each grid can have a cost ranging from 0 (free) to 254 (occupied), where 255 is regarded as unknown. Using values between 0-253, we can put soft constraints on traversable area. In our implementation, we have two sources of costs: mean trajectory valleys and Gaussian agent costs. Combining these two for each agent generates their individual costmap for each time step. Therefore, any cost at specific time, t, in

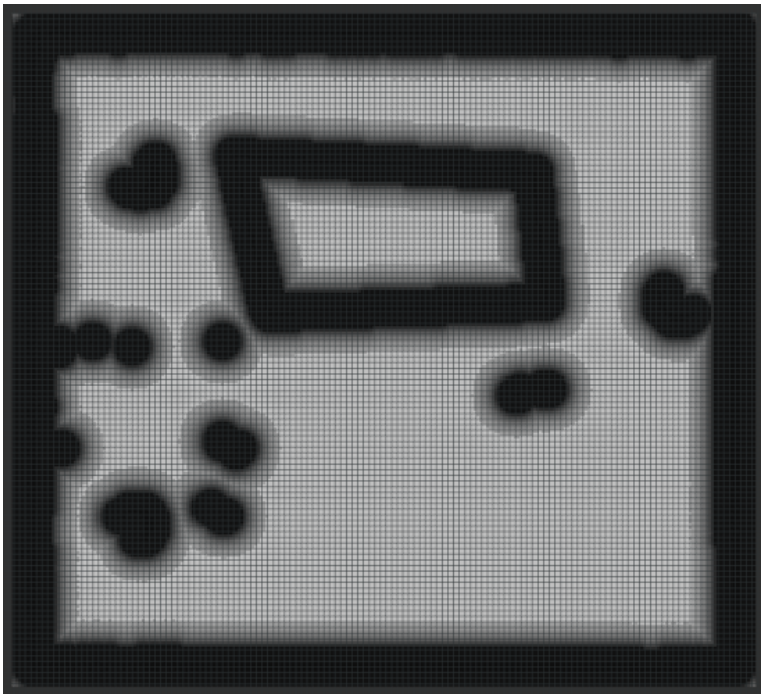


Figure 4.10: An example costmap where darker grids mean higher costs.

costmap is modeled as,

$$C(\mathbf{z}_t) = \max\left(c_m(\mathbf{z}_t), c_a(\mathbf{z}_t)\right) \tag{4.31}$$

Here, $c_m(\mathbf{z}_t)$ is the cost component generated using mean trajectory valleys (MTV) while $c_a(\mathbf{z}_t)$ is generated using Gaussian costs of other agents. Details of these cost components are given in next two sections.

4.5.2.2 Mean Trajectory Valleys

People tracking stage explained in Chapter 3, provides tracks for n people in the scene up to time step t (i.e., recent time),

$$\mathbf{T} = \left[\mathbf{z}_{1:t}^{(1)} \;\; \mathbf{z}_{1:t}^{(2)} \;\; \dots \;\; \mathbf{z}_{1:t}^{(n)}\right]^T \tag{4.32}$$

where measurements for each human are provided as tuples of x and y coordinates with respect to a fixed reference frame as,

$$\mathbf{z}_{1:t}^{(i)} = \left\{(x,y)_1^{(i)}, (x,y)_2^{(i)}, \dots, (x,y)_t^{(i)}\right\} \tag{4.33}$$

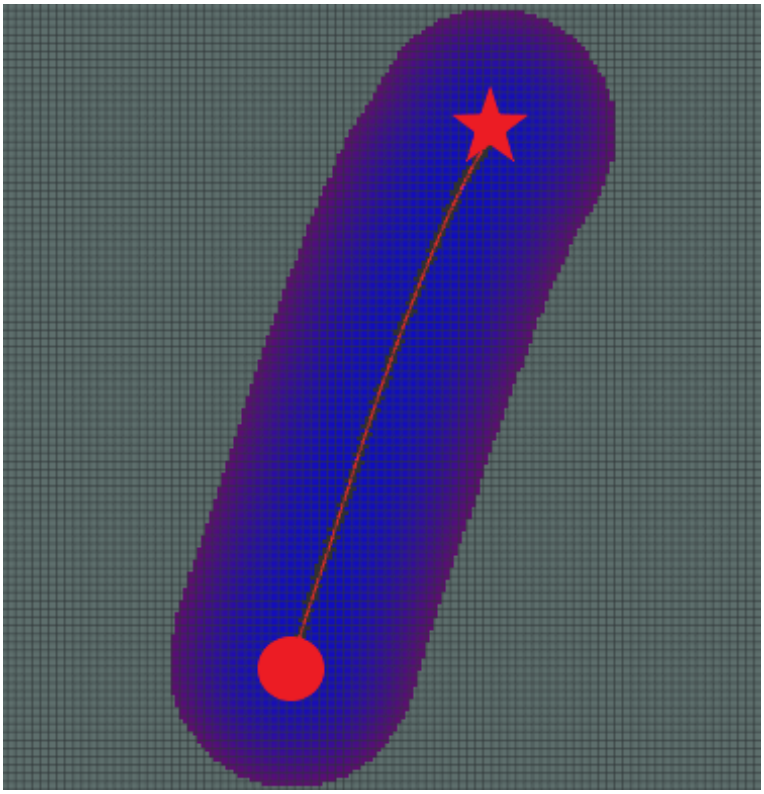


Figure 4.11: Mean trajectory (red curve) is starting from red dot (current pose) and ending at red star (goal). Assigning gradual costs around this curve forms a valley shape.

Goal position, explained in Section 4.5.1.4, is added to these measurements as the measurement taken at time step T,

$$\mathbf{z}_{1:t,T}^{(i)} = \left[\mathbf{z}_{1:t}^{(i)} \;\; \mathbf{z}_{T}^{(i)}\right] \tag{4.34}$$

In guidance of Postulate 1, we generate smooth and human-like trajectories from start to goal. Each individual (including the robot) has its own Gaussian process, conditioned with previous tracking information, $p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t})$. Trajectories are generated by sampling the most probable trajectory (i.e., mean, $m_t^{(i)}$) for each agent from their corresponding GPs. Since humans tend to move on smooth paths while minimizing the path length, we regard this mean sample as reference trajectory that an agent prefers to follow if no interaction is required. When we predict future trajectories using step by step cost-based interaction and cooperation processes, we condition GPs as $p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}, \tilde{\mathbf{z}}_{t'})$ and re-sample their means for further steps. In final prediction step at time step T, these mean functions become final predicted trajectories as

$p(\mathbf{f}_{1:T}^{(i)} \mid \mathbf{z}_{1:t}, \tilde{\mathbf{z}}_{t+1:T})$.

We implement a valley shaped cost distribution around these mean trajectories which we call *mean trajectory valleys*. Costs are assigned in an occupancy grid structure (i.e., costmap) around each agent. Cost is zero for grids intersecting $m_t^{(i)}$ and increases gradually as we move further away. For any agent, i, and any point $\mathbf{x} = (x, y)$ on the costmap, cost of the corresponding grid $c_m^{(i)}(\mathbf{x})$ is,

$$c_m^{(i)}(\mathbf{x}) = \frac{d_{\min}(\mathbf{x}, m_t^{(i)}) \left(c_m^{\max} - c_m^{\min}\right)}{d_m^{\max}} + c_m^{\min} \tag{4.35}$$

Here, $d_{\min}$ function provides minimum Euclidean distance between a grid point, $\mathbf{x}$, and the mean trajectory $m_t^{(i)}$. $c_m^{\max}$ and $c_m^{\min}$ are maximum and minimum cost values to be assigned for mean trajectory valleys while $d_m^{\max}$ is the maximum distance that the valley will expand to (i.e., the distance from the man trajectory which will have maximum cost). By assigning different values to $d_m^{\max}$, $c_m^{\max}$ and $c_m^{\min}$, i.e., changing the slope, we can alter how aggressive the valley tends to pull an agent towards the valley floor, i.e., the mean trajectory. m_t for one agent can be seen as red curve in Figure 4.11. This curve is the initial (or ideal) reference for an agent. We assume that this trajectory represents the possible trajectory for a human if no obstacles or other agents are present in the scene.

4.5.2.3 Gaussian Agent Costs

In the literature, there has been an extensive research on human personal space, i.e., proxemics. Space around a human is categorized in several studies as [76, 110],

- Intimate space (up to 0.45m)
- Personal space (0.45m to 1.2m)
- Social space (1.2m to 3.6m)
- Public space (3.6m to 7.6m)

Of course, these distances can vary depending on the geography and culture but the main idea is that; people attempt to keep a personal space around themselves and others [111, 112]. These space is assumed to be asymmetric, changing with the moving

Figure 4.12: Two people moving one after another with defined cost functions. Note the lethal cost circles at human positions.

direction, speed, etc. [109, 113]. Personal space is generally modeled as a Gaussian distribution or mixture of multiple Gaussian distributions. Also in [75, 114], authors concluded that people tend to assume similar personal space for robots in the same environment. We summarize this assumption as

Postulate 4: *Each individual (including robots) has a personal space which can be modeled as an asymmetric Gaussian function [109, 110, 113].*

Kirby et al. divides personal space into two, symmetrical function at the back and an elliptical function at the front which their sizes are altered according to the relative velocity between the person and the robot [109]. We use this model with an extension for marking other agents in individual costmaps. Instead of defining only a Gaussian cost distribution, we add a circular lethal cost (i.e., 254) centered at agent positions (Figure 4.12) because path planning through a circle centered at human position which is about 0.30m in radius means colliding the people. Therefore, a planned path should never go through this lethal cost circle. Gaussian cost distribution with this lethal circle is together called *Gaussian agent cost* (GAC). Two dimensional Gaussian function can be defined as,

$$f(x, y) = \exp\left(-\left(\frac{(x - x_0)^2}{2\sigma_x^2} + \frac{(y - y_0)^2}{2\sigma_y^2}\right)\right) \tag{4.36}$$

Two 2D Gaussian functions are used one for the front one for the back while sharing the same σ_x but having two different σ_y. Of course, this is generalized to any orientation with x and y axes. Variance along heading direction is σ_h, along sides is σ_s and

towards back side is σ_r with a relation as,

$$\sigma_h = \max(2v, 1/2) \tag{4.37}$$

$$\sigma_s = \frac{2}{3}\sigma_h \tag{4.38}$$

$$\sigma_r = \frac{1}{2}\sigma_h \tag{4.39}$$

where v is the velocity of the agent. The combined Gaussian cost of other agents for i^{th} agent is

$$c_g^{(i)}(x,y) = \bigcup_{i \neq j}^{N} \exp\left(-\left(a\left(x - x_a^{(j)}\right)^2 + b\left(x - x_a^{(j)}\right)\left(y - y_a^{(j)}\right)\right) + c\left(y - y_a^{(j)}\right)^2\right) \tag{4.40}$$

with $i \neq j$. Terms a, b and c are defined as,

$$a = \left(\frac{(\cos\theta)^2}{2\sigma^2} + \frac{(\sin\theta)^2}{2\sigma_s^2}\right) \tag{4.41}$$

$$b = \left(\frac{2\sin 2\theta}{4\sigma^2} - \frac{2\sin 2\theta}{4\sigma_s^2}\right) \tag{4.42}$$

$$c = \left(\frac{(\sin\theta)^2}{2\sigma^2} + \frac{(\cos\theta)^2}{2\sigma_s^2}\right) \tag{4.43}$$

$$\sigma = \begin{cases} \sigma_h & if \;\; \alpha > 0 \\ \sigma_r & otherwise \end{cases} \tag{4.44}$$

where, $\alpha = \text{atan2}(y - y_a^{(j)}, x - x_a^{(j)}) - \theta + \pi/2$, θ is the heading angle and $(x_a^{(j)}, y_a^{(j)})$ is the center point coordinates for agents other than the agent who owns the costmap. α being larger than zero means that we are talking about the Gaussian distribution for the front side and otherwise for the back side.

Total Gaussian agent cost becomes,

$$c_a^{(i)}(x,y) = \begin{cases} c_g^{(i)}(x,y) & if \;\; \sqrt{(y - y_a^{(j)})^2, (x - x_a^{(j)})^2} > 0.3 \\ 254 & otherwise \end{cases} \tag{4.45}$$

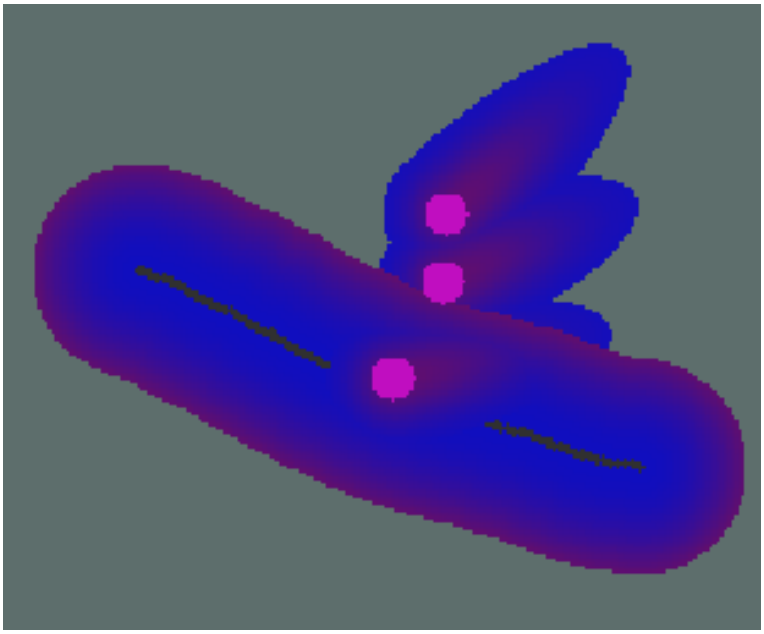


Figure 4.13: Three people with their cost functions where one of them is combined with the mean trajectory valley.

We combine two costs, c_m and c_a, as c_f according to,

$$c_f^{(i)} = \max\left(c_m^{(i)}, c_a^{(i)}\right) \tag{4.46}$$

We should note that, if an agent is behind another agent, i.e. $\text{atan2}(y^{(i)} - y^{(j)}, x^{(i)} - x^{(j)}) - \theta^{(j)} + \pi/2 < 0$, we disregard agent i in agent j's costmap. This is due to fact that, people generally do not adjust their plan w.r.t. people behind them. This also simplifies computational load.

A sample cost combination can be seen in Fig. 4.13. Three people are present in the scene other than the robot. Their Gaussian costs are defined according to their velocity and these costs are merged with the mean trajectory valley of the robot.

4.5.3 Preferred Speed and Cost Gain

Humans tend to walk or run at some particular speeds which are related mainly to metabolic energy consumption of body. This can be stated as:

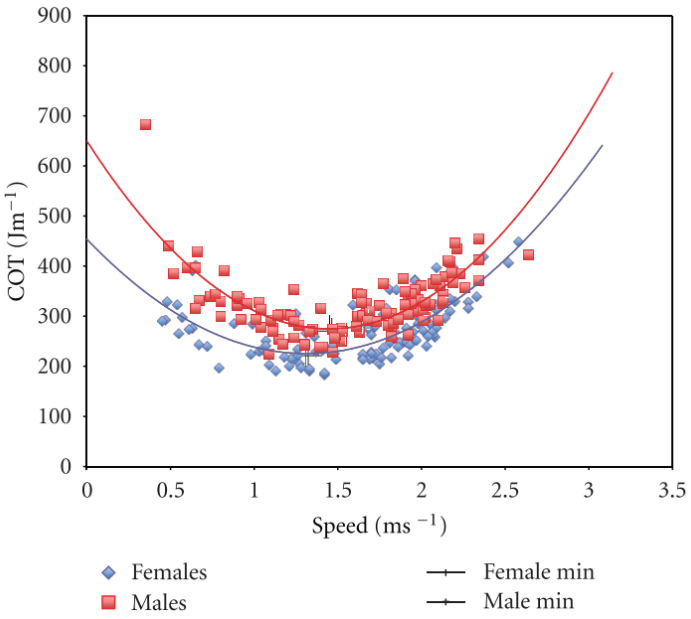


Figure 4.14: CoT values for male and female individuals at their self-selected walking speeds [115].

Postulate 5: *People tend to walk and run at certain preferred speed and they are eager to maintain their navigation speed to minimize their energy consumption [115–117].*

The metabolic energy cost to travel a certain distance is called *cost of transport* (CoT). There are numerous research in the literature showing that CoT is related with preferred running and walking speeds in a curvilinear (i.e., U-shaped) fashion (Figure 4.14) [118, 119]. In different studies ([115, 119, 120]) the relation between walking speed (V in m/s) and metabolic power ($\mathcal{P}$ in W/kg) and as a result, the cost (C in $J/kg \cdot m$) are shown to be best described by a second-order polynomial function:

$$C = aV^2 + bV + c \tag{4.47}$$

In the light of these studies, it is apparent that, humans tend to minimize their CoT while walking or running. They do this by preferring optimal speeds, thus, reducing their energy expenditure to maintain fertility [116, 117, 121, 122]. Deviation from these optimal speeds results in higher metabolic energy expenditure meaning higher CoT. Body anthropometrics like mass, lower limb length, etc. are shown to be the main factors determining the optimal speed of an individual [115]. This results in

different optimal speeds for different people, yet, what they have in common is the trend of CoT versus speed. Although different individuals with different physical properties, including their sex, have different preferred speeds, this relation is shown to be consistent among all humans [115]. Coefficients from the same study result in average optimal speeds of $1.45\ m/s$ and $1.31\ m/s$ for male and female, respectively and an average of $1.38\ m/s$. Since the optimal speed changes among individuals, we assume that each person in a scene moves with their own preferred speed. Due to increasing costs when moving slower or faster than preferred speed, we penalize speed changes via a cost gain in prediction steps. We choose a quadratic model for this cost gain to be consistent with human metabolic dynamics and behavior as $\alpha = ad^2 + c$ where d is the difference between last track velocity and candidate prediction velocity divided by step time (time between frames). We limit the cost gain, α, to maximum value of 2.0 due to costmap structure. Therefore, we set coefficients as $a = 1.6$ and $c = 1$.

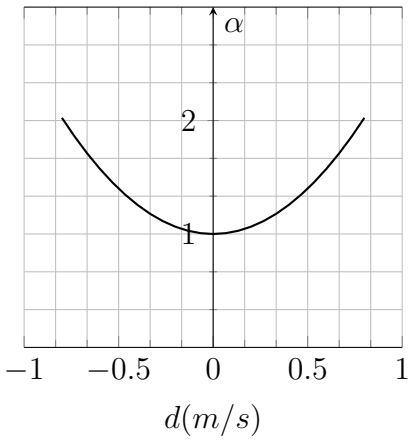


Figure 4.15: Cost gain function with respect to speed difference.

4.5.4 Velocity Samples and Best Velocity

After marking costs on costmap for each individual agent at a time step t, we need to predict a velocity and corresponding position for each future time step t' up to final time T. If we predict the full trajectory at once for each agent, we would disregard the interaction and cooperation which we believe to be the key point in human-aware navigation. Instead, at each future time step, we update each agent's Gaussian process

with the prediction of the previous time step. All other agents use this updated version of Gaussian process for estimating the trajectories of other agents. We encode the cooperation between agents by using this step-wise technique.

Generated samples should cover possible step-wise movements as much as possible. We use an arc in front of each agent at a radius computed using the difference of last predicted position and the current mean position (Figure 4.16) while keeping the magnitude same (as described in Section 4.5.3). We derive this magnitude (i.e., walking speed) from Gaussian process of agents. Base sample vector and corresponding velocity are defined as,

$$\mathbf{r}_{t'}^{(i)} = \frac{\mathbf{m}_{t'}^{(i)} - \tilde{\mathbf{z}}_{t'-1}^{(i)}}{||\mathbf{m}_{t'}^{(i)} - \tilde{\mathbf{z}}_{t'-1}^{(i)}||} ||\mathbf{m}_{t'}^{(i)} - \mathbf{m}_{t'-1}^{(i)}|| \tag{4.48}$$

$$\mathbf{v}_{t'}^{(i)} = \mathbf{r}_{t'}^{(i)} / \Delta t \tag{4.49}$$

$$\mathbf{v}_{t'}^{(i)} \in \mathbf{V}^{(i)} \tag{4.50}$$

where Δt is the time step and $\mathbf{V}^{(i)}$ is the set of admissible velocities for the i^{th} agent. Samples are generated by changing the direction of this vector uniformly, while keeping its magnitude same. Note that we use difference of mean trajectory, $\mathbf{m}_{t'}^{(i)}$, and last predicted position, $\tilde{\mathbf{z}}_{t'-1}^{(i)}$, divided by time step instead of directly using the last velocity from the previous time step. The rationale behind this is that, people tend to return back to the shortest path to their goal after maneuvering to avoid obstacles due to energy minimization principle [117]. Using the last predicted velocity would make samples move away from the mean trajectory. Therefore, we generate velocity samples towards to the next step of mean trajectory.

Besides $\tilde{\mathbf{r}}_{t'}^{(i)}$, we also account for acceleration, deceleration and stopping, which add additional costs with calculated cost gains. For acceleration, we generate another set of samples at radius r_a, for deceleration at r_d. Empirically, we use a ratio of $\%20$ w.r.t $\tilde{\mathbf{r}}_{t'}^{(i)}$ for acceleration and deceleration though it can be altered for other cultural variations, etc. Note that, any sampling ratio can be used since they will be weighted with cost gain. Finally, we add previous predicted position, $\tilde{\mathbf{z}}_{t'-1}^{(i)}$, as another sample in case an agent may want to stop to avoid collision. To prevent an

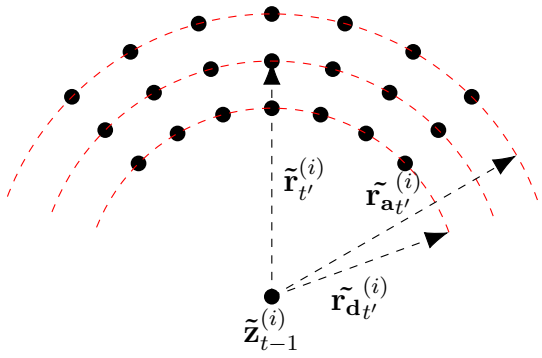


Figure 4.16: Velocity samples generated for prediction of next position.

agent to stop always due to a low cost, we assign a cost of 253 to this stopping sample. Assigning 253 means that, this is only selectable if a collision will be present with all other samples. A sample view for generated samples can be seen in Fig. 4.17. Number of samples and angles between them can be changed according to available computational resources.

Selection of best velocity among $\mathbf{V}$ is carried out by multiplying cost of each sample with its cost gain, and selecting the one with lowest cost. This is an expected behavior and derived from,

Postulate 6: *Individuals behave rationally if they maximize their expected utility or minimize expected cost [123].*

Since the best velocity, $\tilde{\mathbf{v}}_{t'}^{(i)}$, is the one with lowest cost, we express this minimization problem as,

$$\tilde{\mathbf{v}}_{t'}^{(i)} = \underset{\mathbf{v}_{t'}^{(i)} \in \mathbf{V}_{t'}^{(i)}}{\operatorname{argmin}} \left(\left(a(||\tilde{\mathbf{v}}_{t'-1}^{(i)}|| - ||\mathbf{v}_{t'}^{(i)}||)^2 + 1 \right) c_f^{(i)}(\mathbf{v}_{t'}^{(i)} \Delta t) \right) \tag{4.51}$$

We multiply cost of each velocity sample with corresponding cost gain, α, described in Section 4.5.3. Among generated velocity samples, we select the one with the lowest cost as our next prediction. The combined individual costmap of each agent multiplied with cost gain determines the sample with the lowest cost. Therefore, this selected sample becomes the next predicted position, $\tilde{\mathbf{z}}_t^{(i)}$, of the agent. Cost gain also functions as a clamp and prevents continuous acceleration or deceleration, similar to explained human behavior.

Algorithm 2 Proposed human-aware navigation method

function PREDICTTRAJECTORIES($\mathbf{T}$)
  $\mathbf{T} \leftarrow Tracks,\ \mathbf{P} \leftarrow Predictions$
  $t \leftarrow Prediction\ horizon$
  $\mathbf{GP} \leftarrow TrainGP(\mathbf{T})$ ▷ Train predictions with tracks
  $\mathbf{m} \leftarrow SampleGP(\mathbf{GP}, t)$ ▷ Generate initial mean trajectories
  $steps \leftarrow t/step\_time$
  $\mathbf{GP'} \leftarrow \mathbf{GP}$ ▷ Copy GP for temporary prediction steps
  for $i \leftarrow 1, steps$ **do**
    for $j \leftarrow 1, size(\mathbf{P})$ **do**
      $\mathbf{v_s}(j) \leftarrow velocity\ samples$
      $costmap(j) \leftarrow MTV(m(j))$ ▷ Own mean trajectory valley
      for $k \leftarrow 1, size(\mathbf{P})$ **do**
        if $j \neq k$ **then**
          $costmap(j) \leftarrow GAC(P(k))$ ▷ Other Gaussian agent costs
        end if
      end for
      $\mathbf{p_s}(j) = \mathbf{v_s}(j) \times step\_time$
      $costmap(j) \leftarrow \alpha(\mathbf{v_s}(j)) \times costmap(\mathbf{p_s}(j))$ ▷ Apply cost gains
      $p(i) = \mathrm{argmin}_{costmap(j)}(\mathbf{p_s}(j))$ ▷ Predicted position for step i
      $P(j) \leftarrow p(i)$ ▷ Add new predicted position to Prediction
      $GP'(j) \leftarrow TrainGP(p(i))$ ▷ Condition GP with new prediction
      $m(j) \leftarrow SampleGP(GP'(i), t)$ ▷ Update mean with new GP
    end for
  end for
  for $i \leftarrow 1, size(\mathbf{P})$ **do** ▷ Generate final trajectories
    $f(i) \leftarrow SampleGP(GP'(i), t)$
  end for
end function

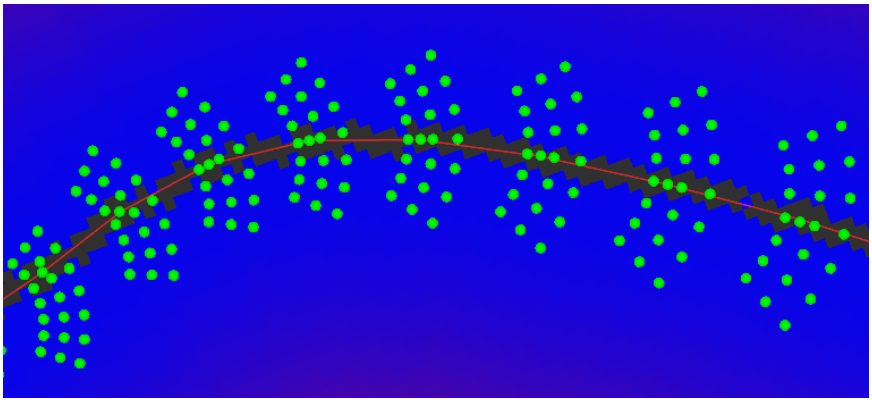


Figure 4.17: Generated velocity samples throughout a full prediction cycle. Red curve is the mean trajectory for the agent.

4.5.5 Path Planning from Inference

When a predicted position, $\tilde{\mathbf{z}}_{t'}^{(i)}$, is computed for a time step for agent i, we condition the temporary Gaussian process belonging to that agent with this new prediction. This results in better estimated mean trajectories for generating valleys on own costmap and marking own Gaussian agent cost in other agents' costmaps for upcoming prediction steps. In other words, we do not use the initial mean trajectories throughout the prediction horizon, instead, at each prediction step we update mean trajectories with latest predictions from previous steps. After each step, we condition temporary GPs of each individual. This process is carried out until we reach time step T. When all steps are predicted up to the time step, T, belonging to goal, we get the mean from each Gaussian process for each individual as our final predicted trajectory, $p\left(\mathbf{f}_{1:T} \mid \mathbf{z}_{1:t}, \tilde{\mathbf{z}}_{t+1:T}\right) = GP\left(m_T^{(i)}, k_T^{(i)}\right)$. This reduces planning to inference so that we can use predicted trajectory for the mobile robot directly as planned path. The algorithm for the whole method is presented in Algorithm 2. Reasons we use Gaussian process of individuals are explained in Section 4.5.1. We can generate human-like trajectories with these Gaussian processes. Additionally, since we train GPs with predicted positions, generated trajectories will go through these positions while keeping human-likeness between these steps.

4.6 Dataset Experiments

The developed trajectory prediction method should be validated with real data to be able to deploy it on a physical platform. People detection and tracking methods are not tested in this stage since they are well-tested in previous chapters, and implemented on a physical platform as a whole in the next section.

As in Section 4.5.1.5, we used ETH BIWI Walking Pedestrians dataset [57], since it is used commonly in the literature to validate pedestrian trajectory prediction methods. Dataset consists of hand annotated frames with people moving in and out from a library building. About 12000 frames are present in the dataset with two dimensional position and velocity annotation in a world coordinate frame. However, annotation is carried out in 2.5 Hz (i.e., 0.4 seconds between). In total, there are 365 different persons identified.

Since we have full ground truth trajectories for each person, we make use of the last position in trajectory as goal position and we train Gaussian processes with these goals. Known goal for each person is a strong assumption and helps for generating better estimates. In the literature there are different methods for human aware trajectory prediction with both known ([57, 88]) and unknown goals ([83, 84]). To be able to compare our method with different methods with different assumptions, we prepared two versions of experiments: with known goal, and with unknown goal. For the case with known goal, goal information is taken from dataset observation matrix. For the case with unknown goal, we simply average last three velocity vectors and use this velocity for a certain period of time (or number of time steps) to estimate a goal position. Final goal estimation research is not in the scope of this study, therefore, we did not put a strong effort to estimate final goal positions using different inputs like trajectory history, environmental conditions, crowd density, etc.

Challenging and descriptive frames are selected within this dataset and these frames are used to validate our method. We tried to select frames with high number of people moving towards each other, expressing cooperation to avoid and make room to each other.

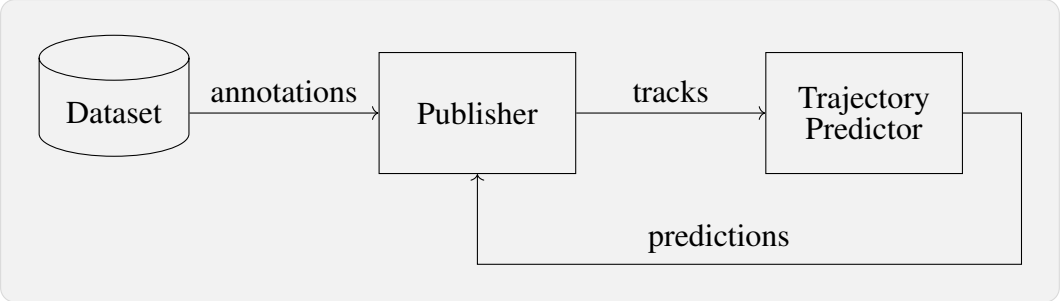


Figure 4.18: Test setup for validation of the trajectory prediction method.

4.6.1 Setup

Tracks are the inputs for Trajectory Predictor (TP). In the complete system, People Detector (PD) provides positions of people with respect to a world-fixed coordinate frame. Then, People Tracker (PT) keeps track of each person with a specific ID and publishes tracks. Finally, TP gathers these tracks and predicts trajectories of each person. In the test setup, since we only test TP, tracks should be generated from the dataset. As seen in Figure 4.18, a publisher node is developed for this purpose. This node reads annotation data from the dataset, generates tracks for each frame, and publishes this track information as if they are provided by PT.

Trajectory Predictor is not aware of the dataset or real data. It just processes the track information supplied. After processing, TP generates a predictions list for further usage. In the test setup we fed these predictions back to Publisher, so that it can compare predictions with ground truth to generate results.

For validating our method, we select different frames regarding their crowd configuration and their presence in the literature. Selected frames generally consist of configurations with multiple persons encountering in different directions which require interaction and cooperation for navigation. Simple scenes which do not require cooperation are omitted.

4.6.2 Results

A sample scene from ETH BIWI Walking Pedestrians dataset can be seen in Figure 4.19. There are four people moving towards the library (down) and three people in the opposite direction (up). Small squares designate ground truth annotations from dataset which are connected with straight lines. On the other hand, circles designate predictions generated by our trajectory predictor. Person ID numbers are positioned with recent position of the corresponding person. Thus, predicted trajectory and ground truth start from the recent time step position.

We evaluate test results in terms of three different metrics:

1. *Trajectory length error,* $\mathbf{\Delta_L}$: The absolute difference between the length of true trajectory and predicted trajectory. We use this metric to measure how well the predicted trajectories resemble human trajectories because humans move in a way to minimize the global length of their trajectories [124]. Even observed for infants, it is accepted that a reasonable person would take the shortest trajectory to reach a goal [125]. Therefore, difference in length of a trajectory is taken as a metric to measure human-likeness.

2. *Average displacement error,* $\mathbf{\Delta_{ave}}$: The mean squared error of all time-indexed points in the predicted and corresponding true trajectory. This metric is introduced in [57]. This error is calculated for each person in each frame and used to measure how much a predicted trajectory is deviated from the corresponding true trajectory.

3. *Final displacement error,* $\mathbf{\Delta_f}$: The Euclidean distance between the predicted last position and the corresponding true last position for a predefined prediction horizon (time steps). This metric is used in [84] and [83]. If remaining time step size is smaller than the prediction horizon for an agent, final displacement error is calculated as the distance between the last positions of prediction and true trajectories regardless of prediction horizon.

First, we present results for sample challenging scenes with different configurations: a) *1 person vs. 8 people*, b) *1 person vs. 7 people*, c) *7 people vs. 11 people*, d)

overtaking. Then, we provide overall results for the dataset. These tests are carried out for both known goal and unknown goal cases. Finally, we compare our results with two state-of-art methods for the more challenging unknown goal case.

(a) (b) (c)

Figure 4.19: A sample scene from the dataset. On the left (a), original scene is shown while on the right, ground truth positions (squares connected with lines) and predicted trajectories (circles connected with lines) are shown for the original scene (b), and for 12 frames later (c).

4.6.2.1 Known Goal

In the first case, we assume that we know final goal positions and corresponding time steps for each person in the scene. This may not be the case for open areas but for structured environments there may be scenarios that make the algorithm know (or at least, estimate precisely) the goal positions. For example, a scenario in a hall with two entry points suggests assuming these two entry points as goal positions for persons moving through that hall. Another example can be a scenario with certain food buffets [88] or payment points which also can be regarded as goal points of

persons moving around.

In ETH library walking pedestrians dataset, we use last position of each true trajectory as final goals. Since we train the GPs with this information, in results we notice that even a prediction deviates from the true trajectory, as time step approaches to end, prediction converges to true trajectory. This is a natural result of Gaussian processes and selected kernel functions. In all tests carried out under known goal assumption, we did not limit the prediction horizon, i.e., predictions are calculated throughout the full trajectory (up to final goal) for each pedestrian.

1 Person vs. 8 People Cooperation: We provide one sample scene to fully demonstrate our method. Figure 4.20 shows a scene with interactions and corresponding predictions in five different frames. One person (#176) is moving against other eight people. In the beginning there is no room for #176 to pass. A non-cooperative planner would plan a trajectory on left or right side of this people cluster. However, our cooperative planner predicts trajectories for each person and truly predicts that people #183 and #184 will cooperate and make room for #176 to pass between them (Figure 4.20b to 4.20f). Besides this interaction and cooperation, note that, we predict future trajectory of each person with a high accuracy. In this scene, we ignore person #171 since he just wanders around with an unrealistic navigation pattern. In Table 4.1, individual results for Figure 4.20b are given. In this frame, 10 people are annotated as moving downwards while person #176 is moving upwards. It is shown that, trajectory lengths are very close to true trajectories (i.e., the difference in trajectory length, $\mathbf{\Delta_L}$, is relatively low for each person, and averaging at about 0.04 meters). Average displacement errors are also below 1 meter, while for interacting agents #176, #183 and #184, this value becomes 0.329, 0.236 and 0.273 respectively. These results, with final displacement errors, show that our method behaves similar to humans in terms of trajectory selection and joint collision avoidance. Further results with total averages are also given in upcoming sections.

For the scene presented in Figure 4.20b, individual displacement errors are plotted in Figure 4.21. Errors are relatively low (less than 1 meter) throughout agents' trajectories. Only persons #182 and #178 deviate from true trajectory about 1.5 meters and then converge again. In early prediction steps, errors for these people are still

low, however, after prediction step 8, errors become larger than 1 meter. Then again, errors decrease and converge to final goal. This behavior is a natural result of Gaussian processes. Since we train the GP with known goal information, as time steps approach to final step, trajectory approaches to final goal. Yet, note that displacement errors shown in Figure 4.21 do not represent path distance error, instead they show spatio-temporal (trajectory) errors. This is because, we compare predicted and true trajectories as time indexed series. Even if prediction and true paths match spatially, accelerations and decelerations yield displacement errors.

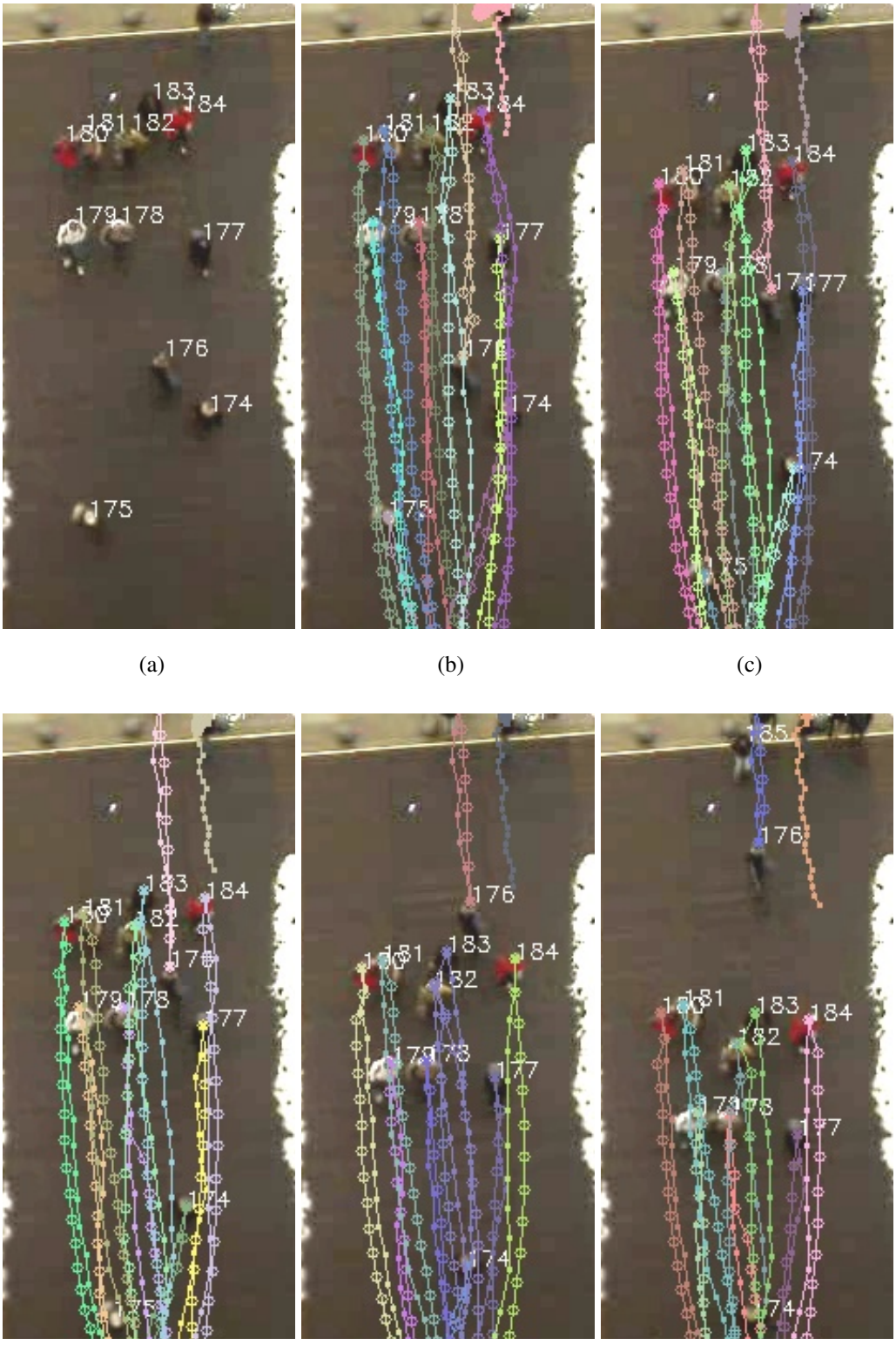


Figure 4.20: A sample scene shows cooperative navigation. Note the predicted trajectory of person #176 which shows the cooperative human navigation prediction ability of our method in consecutive frames.

Table 4.1: Results for scenes shown in Figure 4.20.

(a) Scene in Figure 4.20b.

Person	$\Delta_{\mathbf{L}}$	$\Delta_{\mathbf{ave}}$	$\Delta_{\mathbf{f}}$
176	0.028	0.329	0.354
177	0.004	0.408	0.638
178	0.040	0.768	1.372
179	0.054	0.165	0.323
182	0.103	0.802	1.544
184	0.087	0.273	0.408
183	0.004	0.236	0.341
181	0.098	0.419	0.168
180	0.006	0.282	0.293
Ave.	0.039	0.369	0.522

(b) Scene in Figure 4.20c.

Person	$\Delta_{\mathbf{L}}$	$\Delta_{\mathbf{ave}}$	$\Delta_{\mathbf{f}}$
176	0.081	0.143	0.015
177	0.090	0.307	0.313
178	0.095	0.652	1.066
179	0.009	0.267	0.269
182	0.178	0.638	1.021
184	0.083	0.129	0.077
183	0.036	0.284	0.496
181	0.083	0.296	0.006
180	0.026	0.231	0.273
Ave.	0.070	0.297	0.364

(c) Scene in Figure 4.20d.

Person	$\Delta_{\mathbf{L}}$	$\Delta_{\mathbf{ave}}$	$\Delta_{\mathbf{f}}$
176	0.038	0.192	0.262
177	0.145	0.289	0.225
178	0.037	0.486	0.591
179	0.054	0.272	0.157
182	0.189	0.646	0.971
184	0.054	0.147	0.170
183	0.100	0.332	0.494
181	0.069	0.401	0.520
180	0.018	0.333	0.013
Ave.	0.072	0.323	0.354

(d) Scene in Figure 4.20e.

Person	$\Delta_{\mathbf{L}}$	$\Delta_{\mathbf{ave}}$	$\Delta_{\mathbf{f}}$
176	0.030	0.224	0.313
177	0.037	0.301	0.371
178	0.029	0.480	0.497
179	0.002	0.259	0.204
182	0.109	0.382	0.304
184	0.006	0.296	0.267
183	0.015	0.422	0.404
181	0.098	0.282	0.375
180	0.002	0.274	0.211
Ave.	0.043	0.306	0.312

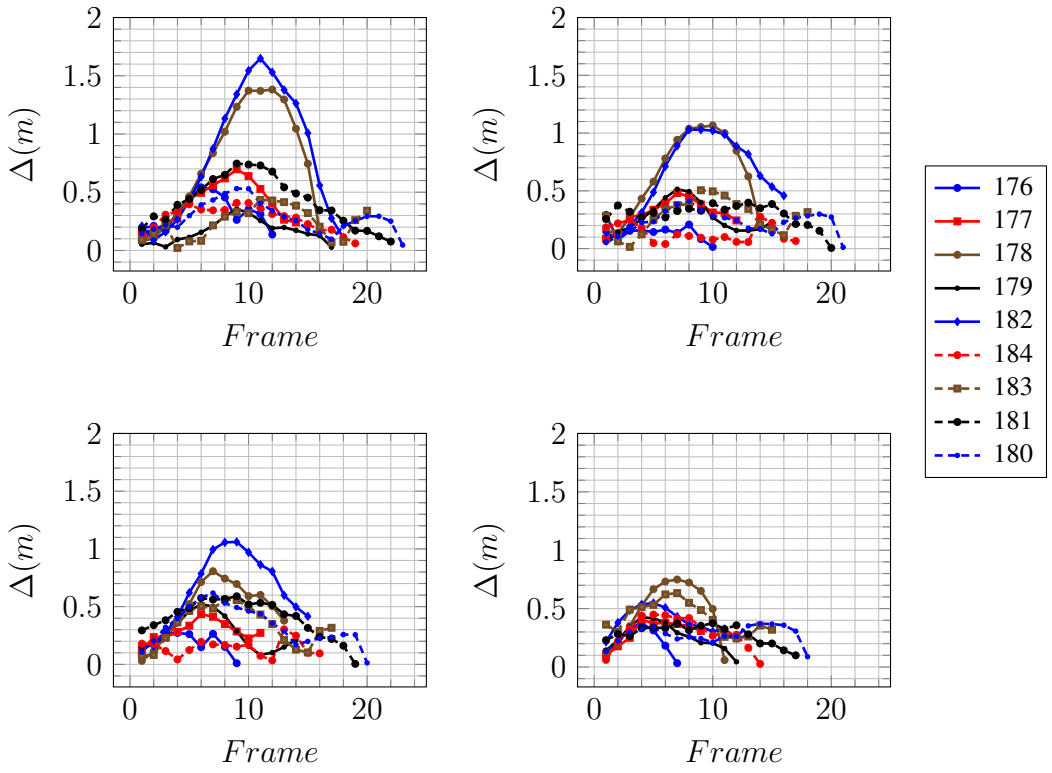


Figure 4.21: Individual displacement errors for scene shown at Figure 4.20b (upper left), Figure 4.20c (upper right), Figure 4.20d (lower left) and Figure 4.20e (lower right).

1 Person vs. 7 People Cooperation: Another scenario for demonstrating the performance of our method is shown in Figure 4.22. Seven people are moving downwards (towards the library entrance), while one person (#236) is moving against them. A non-cooperative planner would pass to the left of this people cluster, however as humans, we do not avoid such people clusters. Instead, we prefer to cooperate and gain room to pass. Also, in the figure, even there is no room to pass, eventually, we see that person #236 passes between persons #238 and #239. This is a good example of interaction and cooperation between people. Predicted trajectories foresee this behavior steps ago (about 5 seconds). In later steps, non-interacting agents deviate from the prediction a bit (but still lower than 2 meters). Especially, short-term (up to 10 steps) prediction of trajectories matches with true trajectories. Changes in displacement error of each individual are plotted in Figure 4.23. Also average values of our metrics (trajectory length error, average displacement error and final displacement error) are listed for each person in Table 4.2. We show that we can capture this natural human behavior with our proposed method.

Table 4.2: Results for scenes shown in Figure 4.22.

(a) Scene in Figure 4.22b.

Person	$\mathbf{\Delta_L}$	$\mathbf{\Delta_{ave}}$	$\mathbf{\Delta_f}$
236	0.174	0.201	0.148
241	0.004	0.424	0.747
242	0.002	0.347	0.622
243	0.033	0.975	0.093
240	0.141	0.874	0.778
237	0.131	0.833	0.897
239	0.045	0.495	0.945
238	0.261	0.688	1.076
Ave.	0.129	0.536	0.606

(b) Scene in Figure 4.22c.

Person	$\mathbf{\Delta_L}$	$\mathbf{\Delta_{ave}}$	$\mathbf{\Delta_f}$
236	0.093	0.146	0.121
241	0.035	0.270	0.064
242	0.025	0.418	0.053
243	0.102	0.376	0.595
240	0.060	0.552	0.996
237	0.037	0.680	0.080
239	0.109	0.527	0.589
238	1.034	0.458	0.729
Ave.	0.187	0.429	0.403

Figure 4.22: The predicted and true trajectories for 1 person vs. 7 people scene.

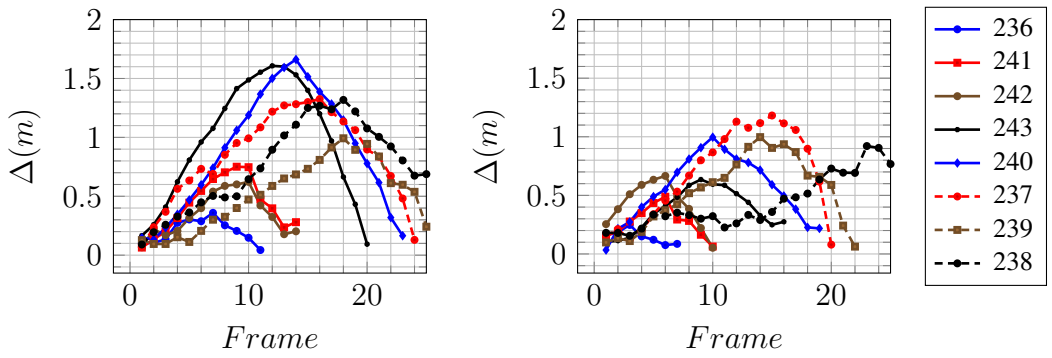
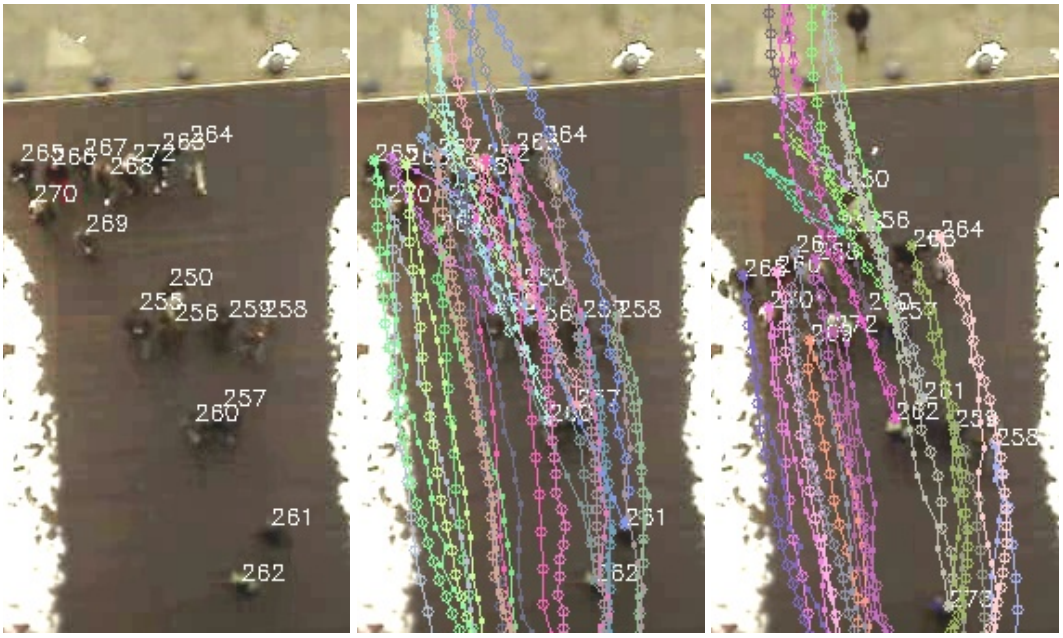


Figure 4.23: Individual displacement errors for scene shown at Figure 4.22b (left) and Figure 4.22c (right).

7 People vs. 11 People Cooperation: The most crowded and challenging scene in the dataset is shown in Figure 4.24. In this scene, there are seven people moving upwards (i.e., #250, #255, #256, #257, #260, #261 and #262) while eleven people are moving in opposite direction. Interactions of these 18 people are predicted by our algorithm, and their predicted trajectories are drawn. However, due to the excessive crowd, it is very difficult to interpret the predicted trajectories visually. Also, providing errors for each pedestrian is not efficient due to high number of people, therefore, for this scene we provide averages for each frame. In Table 4.3, frame number 1 is corresponding to Figure 4.24a and consecutive figures are 6 frames apart from each other (due to annotation frequency of the dataset). The scene lasts for seven annotated frames, and their total averages are also provided in the last row. Average path length error is 0.226 meters while average displacement error and final displacement error are 0.667 and 0.793, respectively. Results show that even with 18 people interacting with each other, our method is able to predict each individual trajectory with high accuracy averaging with much less than 1 meter. Also, note that these errors are calculated with full prediction horizon (i.e., from starting point to goal point of each individual).



(a) (b) (c)

Figure 4.24: The predicted and true trajectories for 7 people vs. 11 people scene.

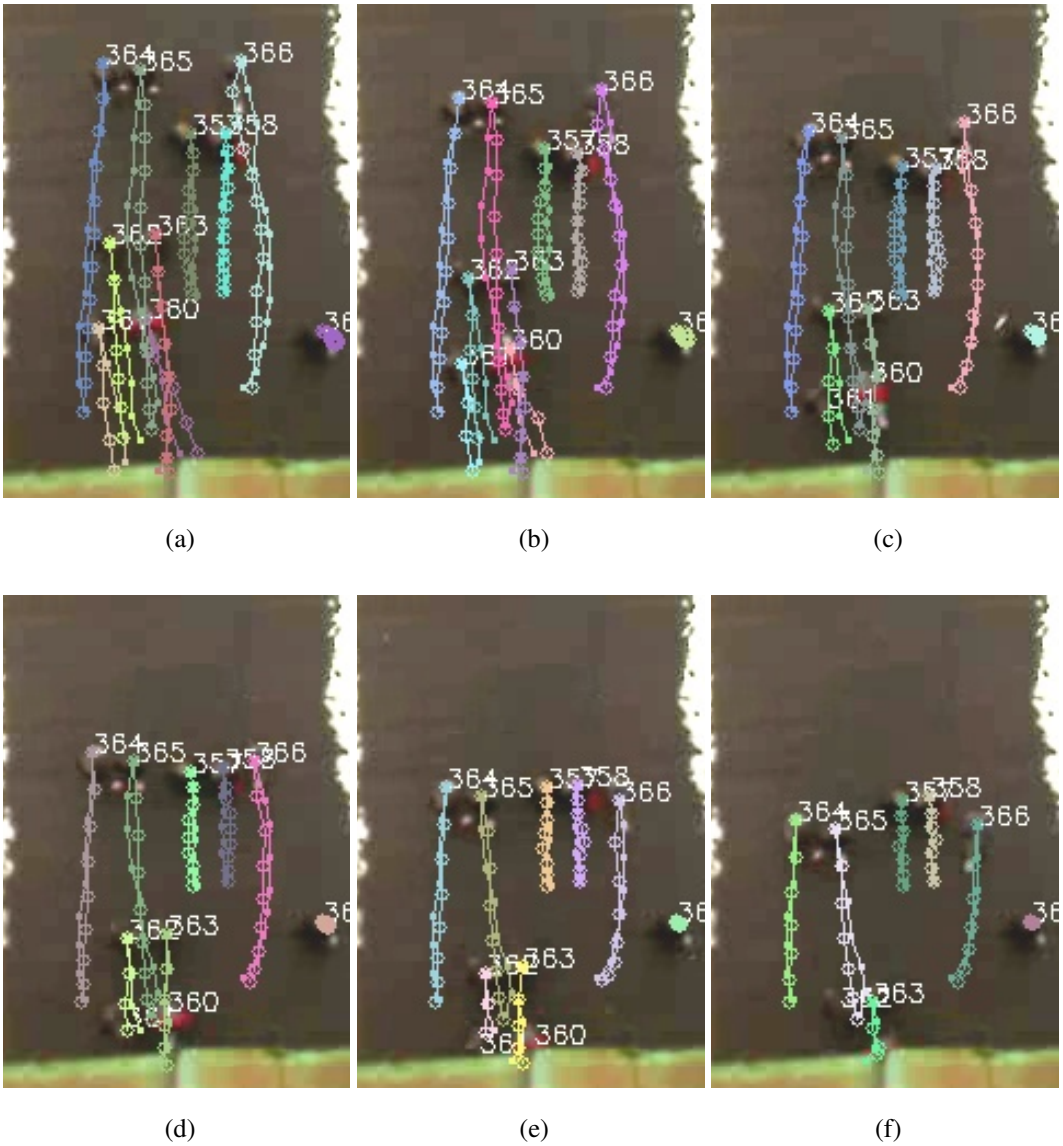
Table 4.3: Results for consecutive 7 annotated frames starting with the scene shown in Figure 4.24b.

Frame	$\mathbf{\Delta_L}$	$\mathbf{\Delta_{ave}}$	$\mathbf{\Delta_f}$
1	0.303	0.915	1.017
2	0.262	0.799	0.989
3	0.249	0.768	0.998
4	0.174	0.577	0.710
5	0.200	0.532	0.614
6	0.221	0.537	0.584
7	0.171	0.543	0.642
Ave.	0.226	0.667	0.793

Overtaking: One interesting example scene is shown in Figure 4.25. Here, all people are moving downwards, but with considerably different speeds. Especially, person #357 and #358 are walking rather slowly (they may be elders). Meanwhile, person #366 is walking faster and approaching from behind of these two people. Obviously, to be able to reach his target in time, person #366 should overtake slow-walking #357 and #358. Figure 4.25a to 4.25f show that our method successfully predicts that #366 will overtake other two people. Figures are from consecutive scenes which are 0.4 seconds apart from each other. We can see that predictions are very consistent over consecutive frames and overtaking is well presented even before there is an intention to overtake. This is an interesting example because here we do not have a cooperation. The slow-moving people, possibly, even do not know the presence of the person #366 before the overtaking. This proves that, our method also captures avoiding and overtaking behaviors of humans.

Individual errors for the six scenes present in Figure 4.25 are listed in Table 4.4. In this table, we regard only five people present throughout the scene (i.e., #357, #358, #364, #365 and #366). Especially, note the average displacement errors of slow-moving pedestrians (#357 and #358) and the person #366 overtaking them. These

low errors indicate that a reasonable human-aware trajectory predictor is developed. Average displacement errors of all pedestrians are also around 0.18 meters which prove very good trajectory prediction even in this overtaking scenario.



(a) (b) (c)

(d) (e) (f)

Figure 4.25: The predicted and true trajectories for the overtaking scene.

Table 4.4: Results for scenes shown in Figure 4.25.

(a) Scene in Figure 4.25a.

Person	$\Delta_{\mathbf{L}}$	$\Delta_{\mathbf{ave}}$	$\Delta_{\mathbf{f}}$
358	0.015	0.200	0.151
357	0.029	0.215	0.132
366	0.094	0.484	0.275
364	0.009	0.098	0.087
365	0.002	0.183	0.161
Ave.	0.030	0.236	0.161

(b) Scene in Figure 4.25b.

Person	$\Delta_{\mathbf{L}}$	$\Delta_{\mathbf{ave}}$	$\Delta_{\mathbf{f}}$
358	0.023	0.190	0.019
357	0.031	0.194	0.071
366	0.076	0.414	0.144
364	0.018	0.141	0.057
365	0.011	0.260	0.061
Ave.	0.032	0.240	0.070

(c) Scene in Figure 4.25c.

Person	$\Delta_{\mathbf{L}}$	$\Delta_{\mathbf{ave}}$	$\Delta_{\mathbf{f}}$
358	0.029	0.208	0.296
357	0.031	0.152	0.240
366	0.091	0.201	0.241
364	0.022	0.129	0.148
365	0.001	0.161	0.136
Ave.	0.035	0.170	0.212

(d) Scene in Figure 4.25d.

Person	$\Delta_{\mathbf{L}}$	$\Delta_{\mathbf{ave}}$	$\Delta_{\mathbf{f}}$
358	0.043	0.106	0.162
357	0.038	0.137	0.230
366	0.031	0.197	0.332
364	0.037	0.182	0.306
365	0.011	0.162	0.150
Ave.	0.032	0.157	0.236

(e) Scene in Figure 4.25e.

Person	$\Delta_{\mathbf{L}}$	$\Delta_{\mathbf{ave}}$	$\Delta_{\mathbf{f}}$
358	0.044	0.074	0.073
357	0.041	0.105	0.125
366	0.023	0.369	0.452
364	0.038	0.166	0.250
365	0.028	0.132	0.194
Ave.	0.035	0.169	0.219

(f) Scene in Figure 4.25f.

Person	$\Delta_{\mathbf{L}}$	$\Delta_{\mathbf{ave}}$	$\Delta_{\mathbf{f}}$
358	0.029	0.034	0.004
357	0.042	0.065	0.051
366	0.004	0.186	0.120
364	0.048	0.137	0.151
365	0.036	0.134	0.235
Ave.	0.032	0.111	0.112

Overall Results for Known Goal: After presenting different cases with their results, we test our algorithm for the whole dataset. For each annotated frame, full trajectory predictions are generated. Errors are averaged for each prediction step. Then, these results are averaged for a frame (for all pedestrians). Finally, a grand average is calculated for all annotated frames in the dataset. This result generation method is also used in [83]. Final results are listed in Table 4.5. We calculate averages with respect to the prediction horizon to see results with changing prediction steps. 20+ prediction horizon means that there is no limit for the number of steps for prediction. Each pedestrian trajectory is predicted up to their known goal step.

It is important to note that, while changing the prediction horizon, we did not changed the final goal information for training. In other words, for example, for 5-steps-horizon we still used the true final goal information. Of course, if we use 5th true position as our goal for 5-step-horizon, we would have smaller errors. However, we thought that always using the final goal for changing horizons is more applicable in real world scenarios because in some cases, true final goals can be extracted from environmental conditions.

Table 4.5: Overall average results for known goal case.

Steps	$\mathbf{\Delta_L}$	$\mathbf{\Delta_{ave}}$	$\mathbf{\Delta_f}$
1	0.093	0.147	0.147
2	0.145	0.180	0.213
5	0.269	0.260	0.365
10	0.370	0.355	0.493
20	0.227	0.411	0.374
20+	0.203	0.425	0.153

As prediction horizon increases, naturally, errors are expected to increase. It becomes harder to predict future positions similar to true trajectory as prediction steps increase. In known goal case, as we approach to a goal, errors should start decreasing since we train GP with the goal information. In Table 4.5, we interpret the results for 20+ prediction horizon case (since it is the most challenging one). The path length error

is about 0.2 meters, while our average displacement error is 0.425 meters which is a low for such a prediction. Final displacement error average is 0.153 which is quite low due to known goal assumption as explained before. Final displacement error is more meaningful for other defined prediction horizons (i.e., 1, 2, 5, 10 and 20) and for the unknown goal case.

4.6.2.2 Unknown Goal

After testing our method with known goal case, we move further to see generalization performance. Known goal assumption may not be the case in all scenarios. Open areas or environments with high number of possible movement directions require trajectory predictor to still function with unknown goal assumption. For these cases, we need to estimate a temporary goal position and time step. However, we do not go deep in goal estimation since it is not in the scope of this study. Instead, we employ a simple goal estimator using past n trajectory points. Details of our simple goal estimator is explained in Section 4.5.1.4.

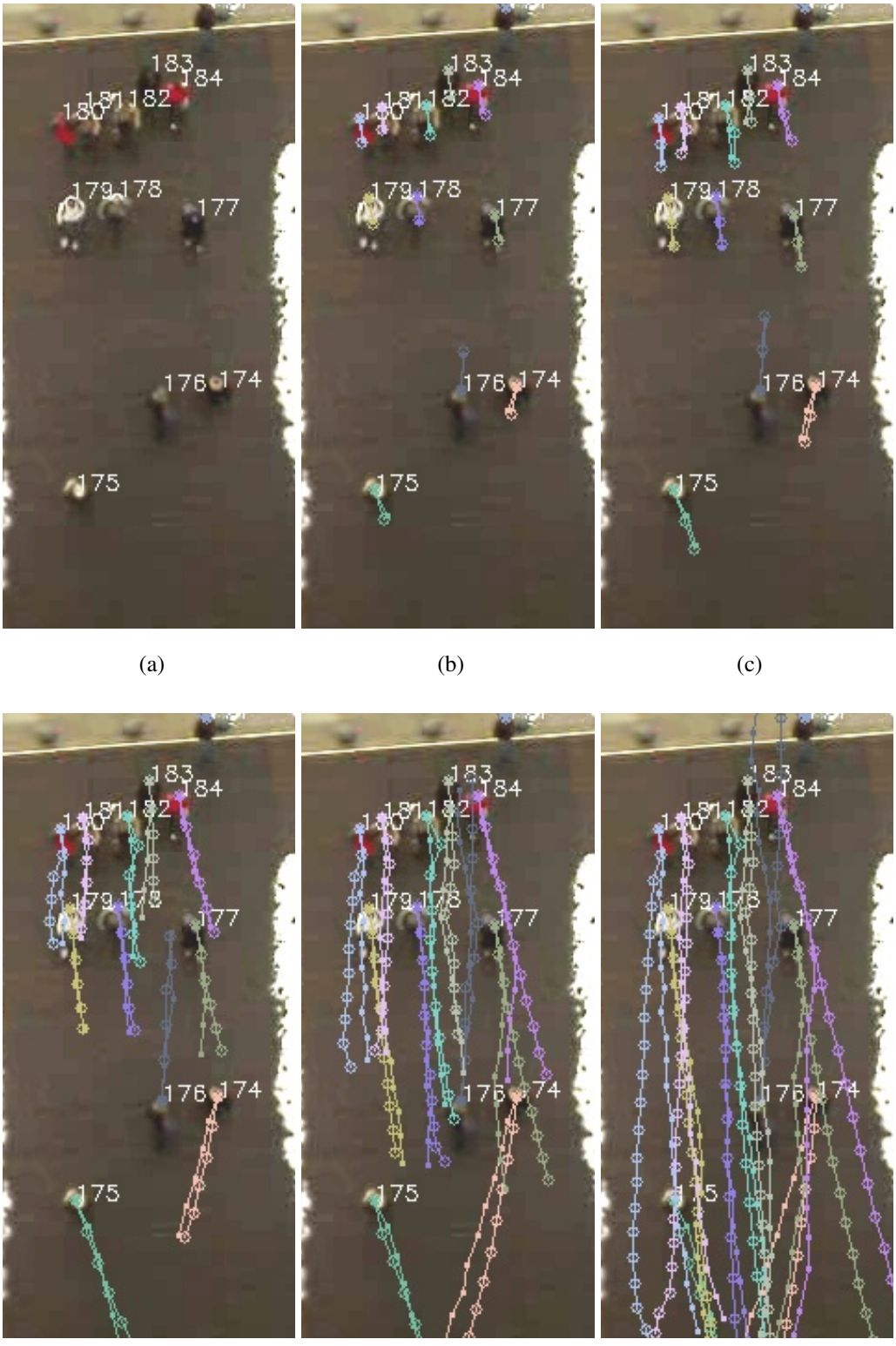
In this section, we present test results for the same scenarios presented in Section 4.6.2.1. For unknown goal case, since we do not train each Gaussian process with true final goal information, errors are expected to increase as prediction steps increase. This is the case because estimated temporary goal deviates more and more from the true goal as steps increase. We show the results for different prediction horizons, and we compare our results with other important algorithms in the literature.

We set five different prediction horizons for our unknown goal tests. First, we predict just one step further, which nearly always results in very close predictions to true trajectory. Then, we increase the prediction horizon to 2 steps, then 5, 10 and 20 steps. Since we have different prediction horizons besides different pedestrians, in contrast to known goal case, we provide only averages instead of pedestrian-based results. Also, for the figures, we provide the same scene for different horizons, where for the known goal case we provide consecutive frames.

1 Person vs. 8 People Cooperation: Same scenario tested in known goal case is also tested here with five different prediction horizons and unknown goal assumption. In Figure 4.26, 1 vs 8 people cooperative navigation scenario is shown with changing prediction steps. Predictions for 1 step and 2 steps are very close to true trajectories. Predictions for 5 steps (Figure 4.26d) and 10 steps (Figure 4.26e) are still representing the true trajectory reasonably. Also, in Figure 4.26d, we see that cooperative navigation takes place between persons #176, #183 and #184. Still, our algorithm predicts trajectories including this interaction. As prediction steps increase, errors start to increase as expected due to invalidation of estimated goal positions. However, if estimated goal is close to the true goal, then the behavior is similar to known goal case and errors are still low. If we have abrupt direction changes like we have with person #177, of course, the goal estimation becomes invalid and errors become large with 20 steps prediction horizon (Figure 4.26f). Average errors for the scene shown in Figure 4.26 are listed in Table 4.6a. Errors increase with increasing prediction horizon. However, even with 20 steps, average displacement error is below 1 meter. Final displacement error becomes larger than 1 meter when we have 20 steps due to unknown goal assumption.

1 Person vs. 7 People Cooperation: In this scenario, 1 vs 7 people are interacting with each other in opposite directions (Figure 4.27). Even with prediction horizon of 20 steps, interacting agents #236, #238 and #239 follow similar pattern to their true trajectories. Our prediction algorithm still predicts that #238 and #239 will make room for #236 to pass in between. Starting from Figure 4.27d, interaction and cooperative movements can be observed both from the true and the predicted trajectories. Average results are provided in Table 4.6b.

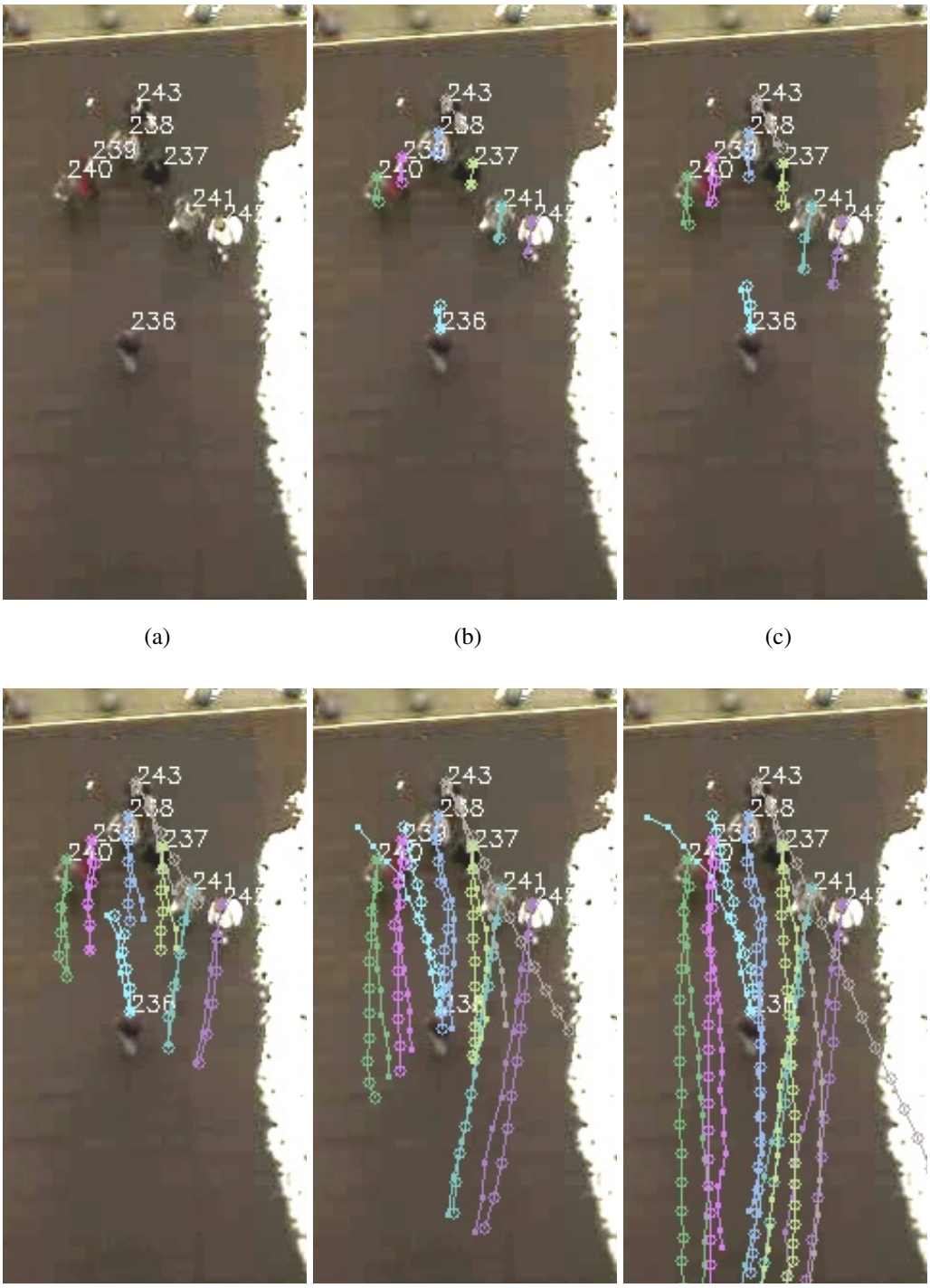
7 People vs. 11 People Cooperation: The most crowded and challenging scene in the dataset is again shown in Figure 4.28. In this scene, there are seven people moving upwards (i.e., #250, #255, #256, #257, #260, #261 and #262) while eleven people are moving in opposite direction. With prediction horizon of 5 (Figure 4.28d), interactions begin. However, due to people density of the scene, figures become overcrowded and hard to interpret. Average errors are listed in Table 4.6c.



(a) (b) (c)

(d) (e) (f)

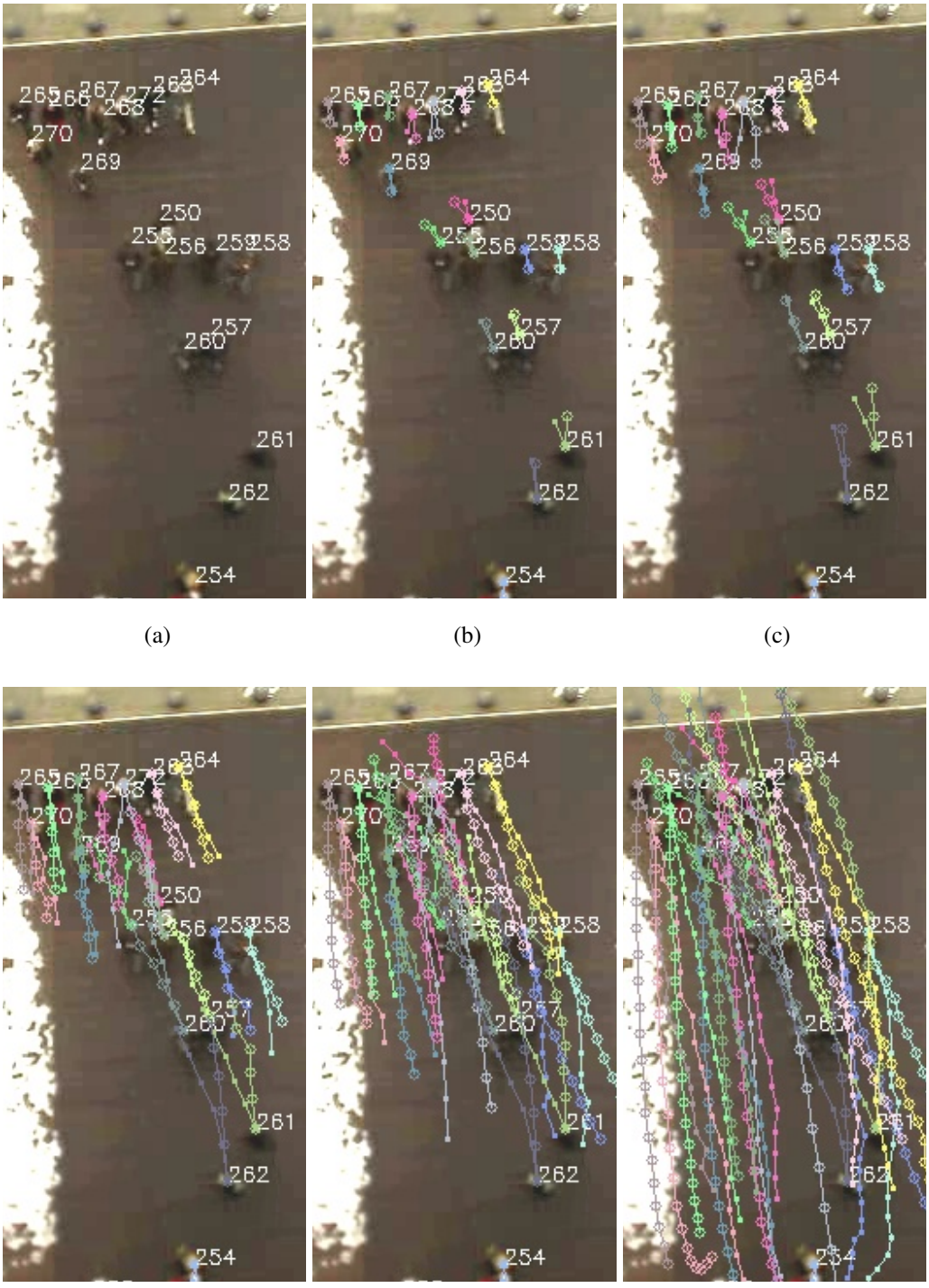
Figure 4.26: 1 person vs 8 people scene with unknown goal assumption for different prediction horizons.



(a) (b) (c)

(d) (e) (f)

Figure 4.27: 1 person vs 7 people scene with unknown goal assumption for different prediction horizons.



(a) (b) (c)

(d) (e) (f)

Figure 4.28: 7 vs 11 people scene with unknown goal assumption for different prediction horizons.

Table 4.6: Average results for scenes shown in Figures 4.26, 4.27, 4.28 and 4.29 respectively.

(a) Scene in Figure 4.26.

Steps	$\mathbf{\Delta_L}$	$\mathbf{\Delta_{ave}}$	$\mathbf{\Delta_f}$
1	0.066	0.087	0.087
2	0.077	0.103	0.108
5	0.174	0.200	0.270
10	0.320	0.355	0.569
20	0.768	0.814	1.263

(b) Scene in Figure 4.27.

Steps	$\mathbf{\Delta_L}$	$\mathbf{\Delta_{ave}}$	$\mathbf{\Delta_f}$
1	0.053	0.074	0.074
2	0.082	0.116	0.148
5	0.294	0.219	0.225
10	0.433	0.354	0.461
20	0.838	0.787	1.277

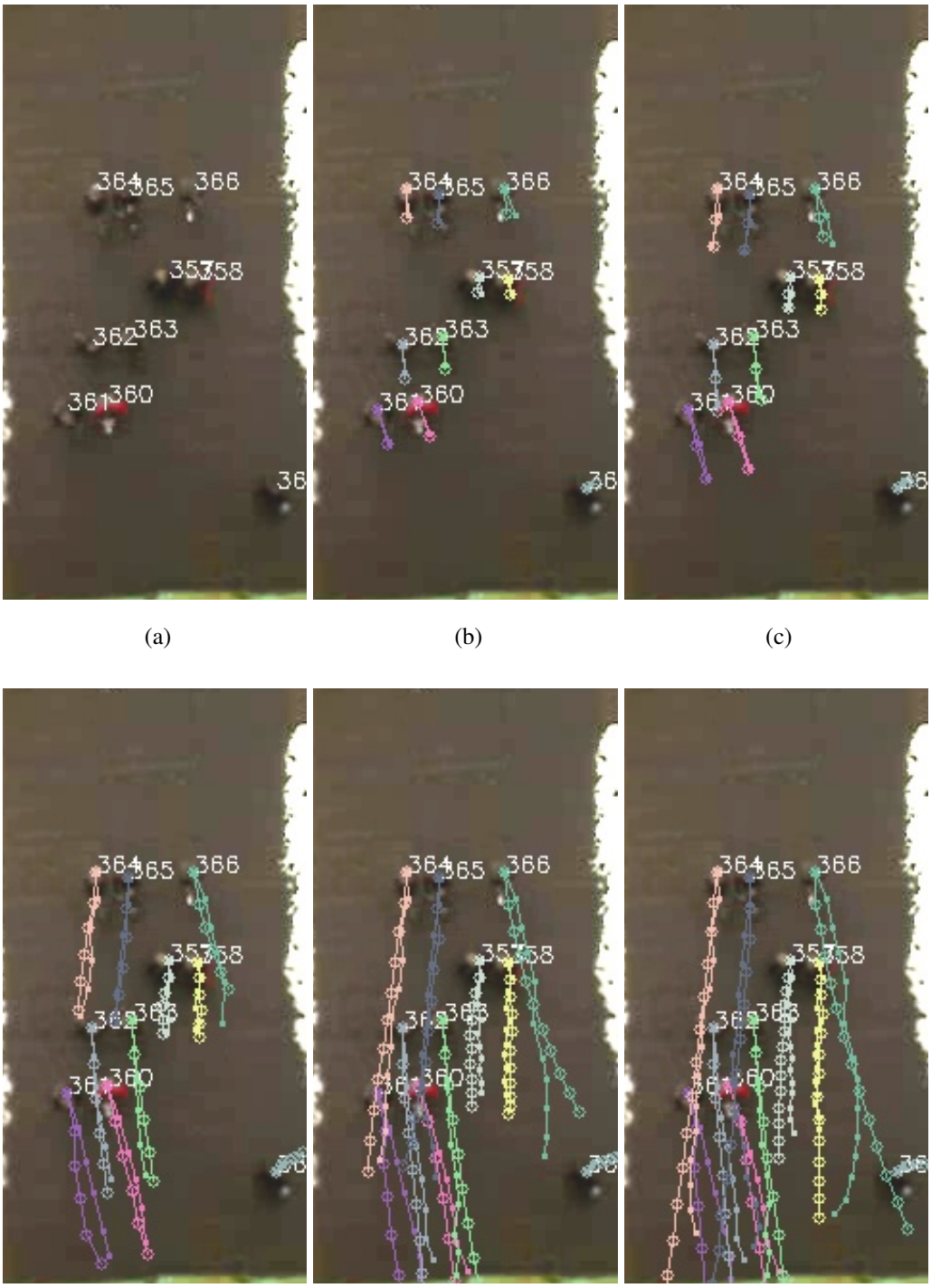
(c) Scene in Figure 4.28.

Steps	$\mathbf{\Delta_L}$	$\mathbf{\Delta_{ave}}$	$\mathbf{\Delta_f}$
1	0.074	0.145	0.145
2	0.112	0.186	0.233
5	0.273	0.322	0.505
10	0.549	0.565	0.840
20	0.873	1.051	1.454

(d) Scene in Figure 4.29.

Steps	$\mathbf{\Delta_L}$	$\mathbf{\Delta_{ave}}$	$\mathbf{\Delta_f}$
1	0.045	0.073	0.073
2	0.083	0.089	0.097
5	0.236	0.189	0.273
10	0.649	0.404	0.717
20	1.298	0.656	1.020

Overtaking: Overtaking scene provided in known goal case is also shown in Figure 4.29. Here, all people are moving downwards, but with considerably different speeds. Person #357 and #358 are walking slowly and person #366 is walking faster and approaching from behind of these two people. Also in unknown goal case, due to velocity difference, the need to overtake becomes obvious and our algorithm predicts this behavior. Only difference is that, after deviating for overtake, person #366 does not return back to his original path in predicted trajectory because we do not know the exact final goal in this case. Still, average errors are relatively low as provided in Table 4.6d.



(a) (b) (c)

(d) (e) (f)

Figure 4.29: The overtaking scene with unknown goal assumption for different prediction horizons.

Overall Results for Unknown Goal: After presenting different cases with their results, we test our algorithm for the whole dataset. For each annotated frame, full trajectory predictions are generated for different prediction horizons. Errors are averaged for each prediction step. Then, these results are averaged for a frame (for all pedestrians). Finally, a grand average is calculated for all annotated frames in the dataset. Final results are listed in Table 4.7.

Table 4.7: Overall average results for unknown goal case.

Steps	$\mathbf{\Delta_L}$	$\mathbf{\Delta_{ave}}$	$\mathbf{\Delta_f}$
1	0.074	0.111	0.111
2	0.120	0.143	0.175
5	0.239	0.251	0.359
10	0.453	0.430	0.645
20	0.867	0.746	1.104

4.6.2.3 Comparison of Known and Unknown Goal

First, we compare two cases of our trajectory prediction algorithm: known goal and unknown goal. For prediction horizons of 1, 2 and 5 steps, average results are similar for both cases. Even the unknown goal case results in slightly better results. This can be originated from the fact that, for small horizons even we estimate it, the goal is not far from the starting position. Training Gaussian processes with this low uncertainty goal information results in better predictions for short horizons. However, after prediction horizon of 5, the known goal case becomes superior due to trained exact goal information, as expected. Results are plotted for both cases in Figure 4.30. In both cases, average errors are quite low compared to other methods in the literature. Now we compare our method with other methods.

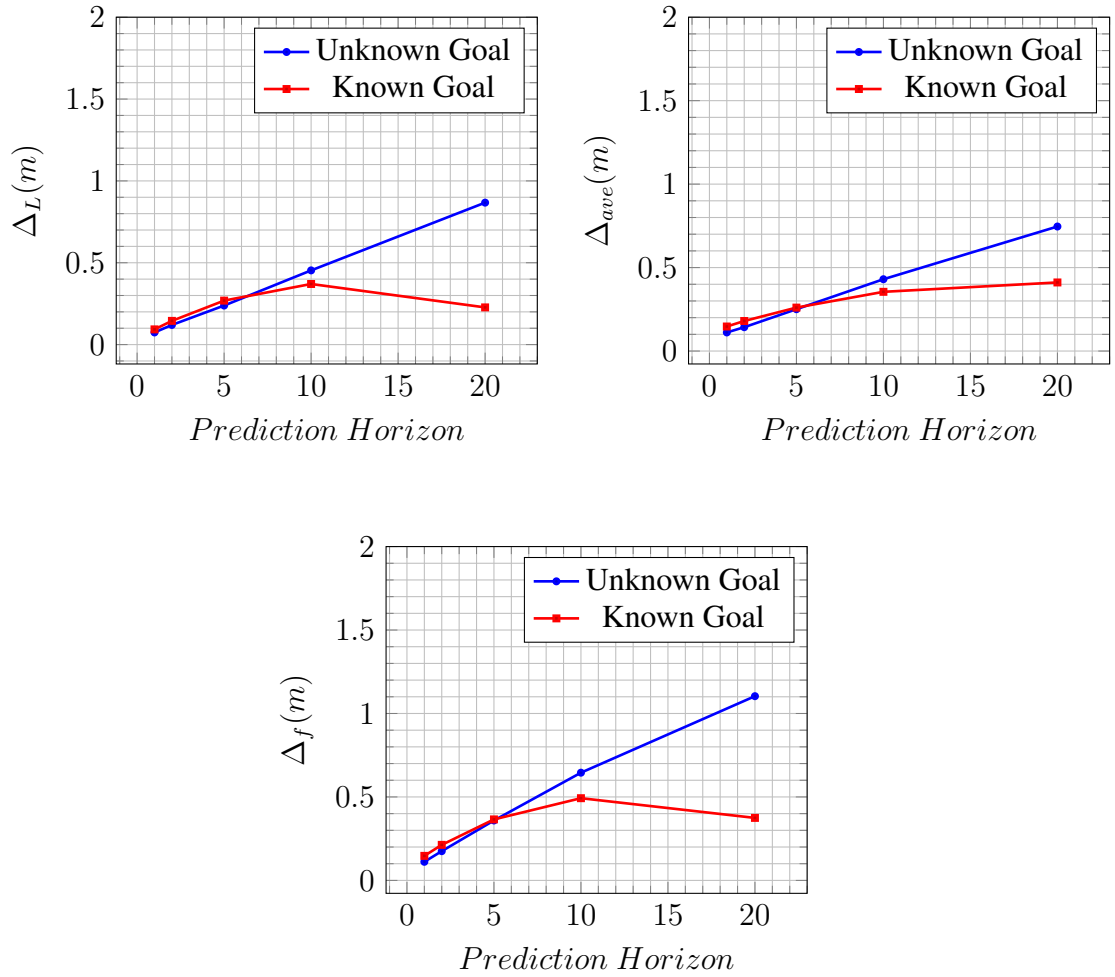


Figure 4.30: Error comparison of Known Goal and Unknown Goal cases.

4.6.2.4 Comparison with Other Methods

Among human-aware navigation methods mentioned in Section 4.4, we compare our method with two most cited and promising algorithms; Interacting Gaussian Processes (IGP) model by Trautman et al. [88] and local interactive model by Vemula et al. [83]. Original IGP relies on known goal assumption. However, to be consistent and not to create an unfair comparison, we used estimated goal for all three methods (unknown goal case for our method). Evaluation is carried out on the same dataset (i.e., ETH Walking pedestrians dataset) [57]. Method and metrics are the same as described in Section 4.6.2.

Errors for different metrics and prediction horizons are listed in Table 4.8 and for better visualization, plotted in Figure 4.31. All three methods perform similar for short horizons like 1 step and 2 steps. In short horizons, possibility to have interactions is low, and velocity change of each individual is small. Therefore, it is likely to predict 1 or 2 steps further close to the true trajectory. However, generally starting with a horizon of 5, interaction and cooperative navigation takes place and modeling of these behaviors becomes the key point. IGP method by Trautman et al. performs satisfactorily for short horizons but as the horizon increases it becomes worse because IGP generates smooth paths up to a predicted goal coupled with an interaction potential. It does not deal with local interactions. On the other hand, Vemula et al. train their model from real data and infer agent velocities depending on the surrounding people configuration. Therefore, their method depends on the quality and generalization performance of training data. Using only spatial orientations of other people may not be the only input for inferring agent velocities. Our interaction model is well-defined and does not rely on a training data. As a result, our method generates lower errors meaning that it predicts trajectories closer to true ones.

4.7 Conclusions

Incorporating navigation behaviors of humans is the key point in achieving human-aware and human-like navigation methods. Cooperative and interaction-based navigation of humans can be modeled and blended in robot navigation. In this chapter,

Table 4.8: Comparison of our method with Trautman et al. [88] and Vemula et al. [83].

Metric	Horizon	Trautman et al.	Vemula et al.	Our method
Traj. length error	1	0.08	0.10	0.07
	2	0.12	0.14	0.12
	5	0.43	0.40	0.24
	10	0.79	0.72	0.45
	20	1.30	1.14	0.87
Average disp. error	1	0.10	0.12	0.11
	2	0.16	0.17	0.14
	5	0.44	0.34	0.25
	10	0.58	0.60	0.43
	20	1.16	0.97	0.75
Final disp. error	1	0.10	0.12	0.11
	2	0.20	0.22	0.18
	5	0.65	0.55	0.36
	10	1.08	1.09	0.65
	20	1.89	1.52	1.10

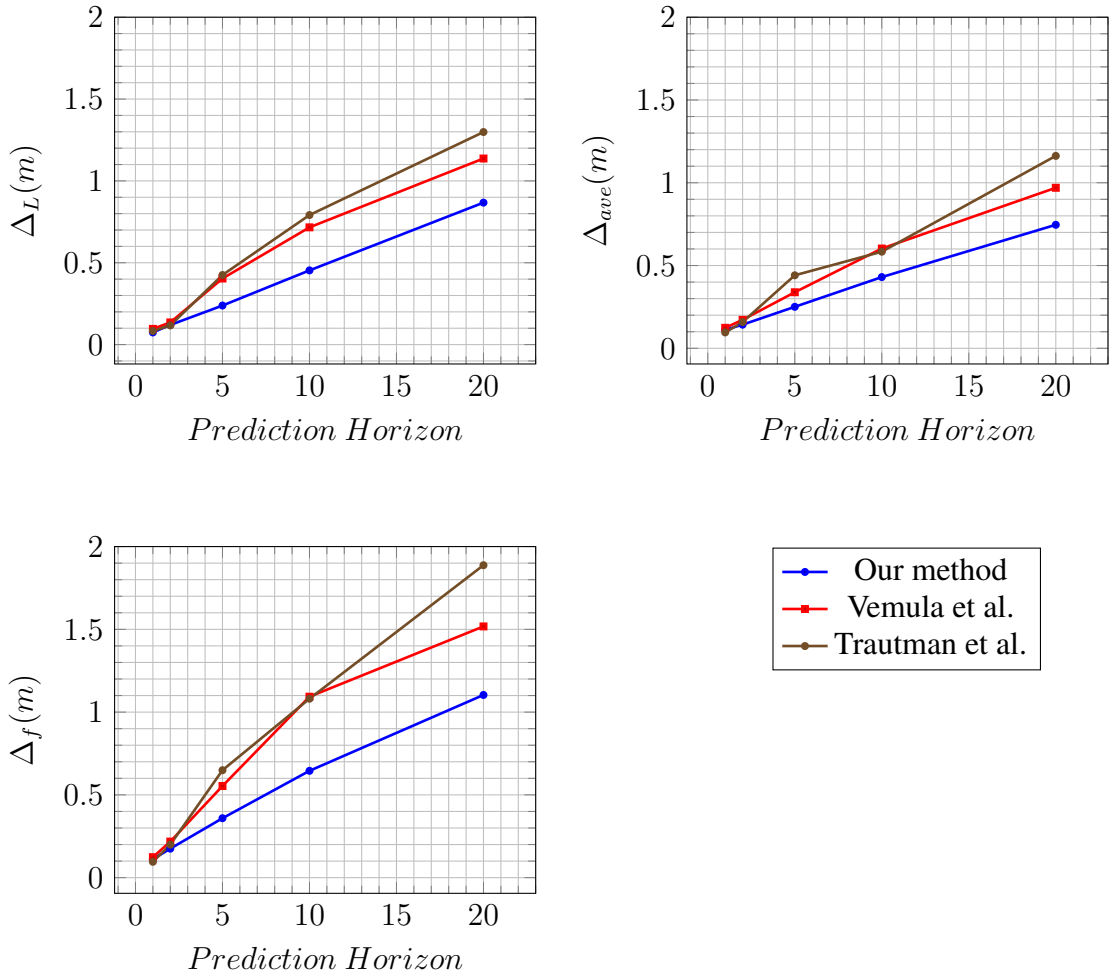


Figure 4.31: Error comparison of our method with two other state-of-art methods.

we developed a trajectory prediction and path planning method for mobile robots. The proposed method makes use of Gaussian processes for sampling trajectories in intermediate and final prediction steps. Cooperation is modeled as cost-based interaction on a costmap. Each individual (including the robot) has a costmap representing its belief about the other agents in the scene. Costs are derived from two different sources: mean trajectory valleys and Gaussian agent costs. Mean trajectory valleys account for the deviation from obstacle-free, ideal trajectories while Gaussian agent costs deal with personal space of other agents. Costs are also multiplied with a cost gain originated from human metabolic power consumption model. Deviations from optimum speeds for agents are penalized using this cost gain and the best velocity is selected with lowest cost for each agent. Gaussian processes of each agent is conditioned with these predicted measurements. After reaching to final step, means are sampled from conditioned Gaussian processes to obtain final predicted trajectories. The planned path of the robot becomes the one predicted for itself. Therefore, there is no need to have additional algorithms and methods to generate a plan.

The developed method is tested on a public dataset with two different assumptions: known goal and unknown goal. With both assumptions, results are quite similar the true trajectories. Even in long prediction horizons, we can achieve low errors in terms of defined evaluation metrics. As the final step in this chapter, the method is also compared with two state-of-art methods. Results show that the proposed algorithm performs superior especially for long prediction horizons.

CHAPTER 5

EXPERIMENTS AND RESULTS

5.1 Introduction

Human-aware navigation has different aspects as people detection, tracking, trajectory prediction and path planning. Throughout the study, we developed different novel methods for these components. We tested people detection and tracking on a dataset and obtained high performance even for non-standard poses. Then, we tested our trajectory prediction and path planning algorithm on a different dataset and compared our results with state-of-art techniques. We concluded that our method can predict trajectories similar to that of humans. This ability is the key for human-aware navigation. Now we combine all developed algorithms and implement them on a mobile robot base. This is the ultimate test to show that all developed components of this study work in harmony with high success rate.

5.2 Experimental Setup

To test the whole system, we needed to implement our algorithms on a mobile robot and then conduct experiments with people while navigating in a human-aware fashion. Three different scenarios are designed, and results are evaluated in terms of important metrics from literature.

Figure 5.1: SEIT 100 robot from Milvus Robotics [126].

5.2.1 The Robot

As the mobile robot base, we used SEIT 100 (Figure 5.1)) from Milvus Robotics [126]. SEIT 100 is developed primarily for logistics purposes. It can carry loads up to 100 kg. It has differential drive. Its size is suitable for indoor navigation in environments crowded with people.

Due to high internal computation needs, SEIT 100 is equipped with a powerful computer. It also provides extension ports for other attachments and applications.

Table 5.1: Specifications of SEIT 100.

Payload:	100 kg
Drive:	Differential
Dimensions:	890 mm x 650 mm x 297 mm
Maximum Speed:	1.5 m/s
Processor:	Intel Core i7 3.4 GHz
Communication:	Wi-Fi

We installed a Microsoft Kinect v2 camera on top of this robot (Figure 5.2). To

resemble a human in size, Kinect is placed at 1.60 m height from the ground plane. Higher positions for Kinect would help for better field of view but making the robot higher than an average human height may result in artificial reactions to robot by humans (like aggressively avoiding it). Kinect is tilted down about 10° from the horizontal plane for better scene coverage.

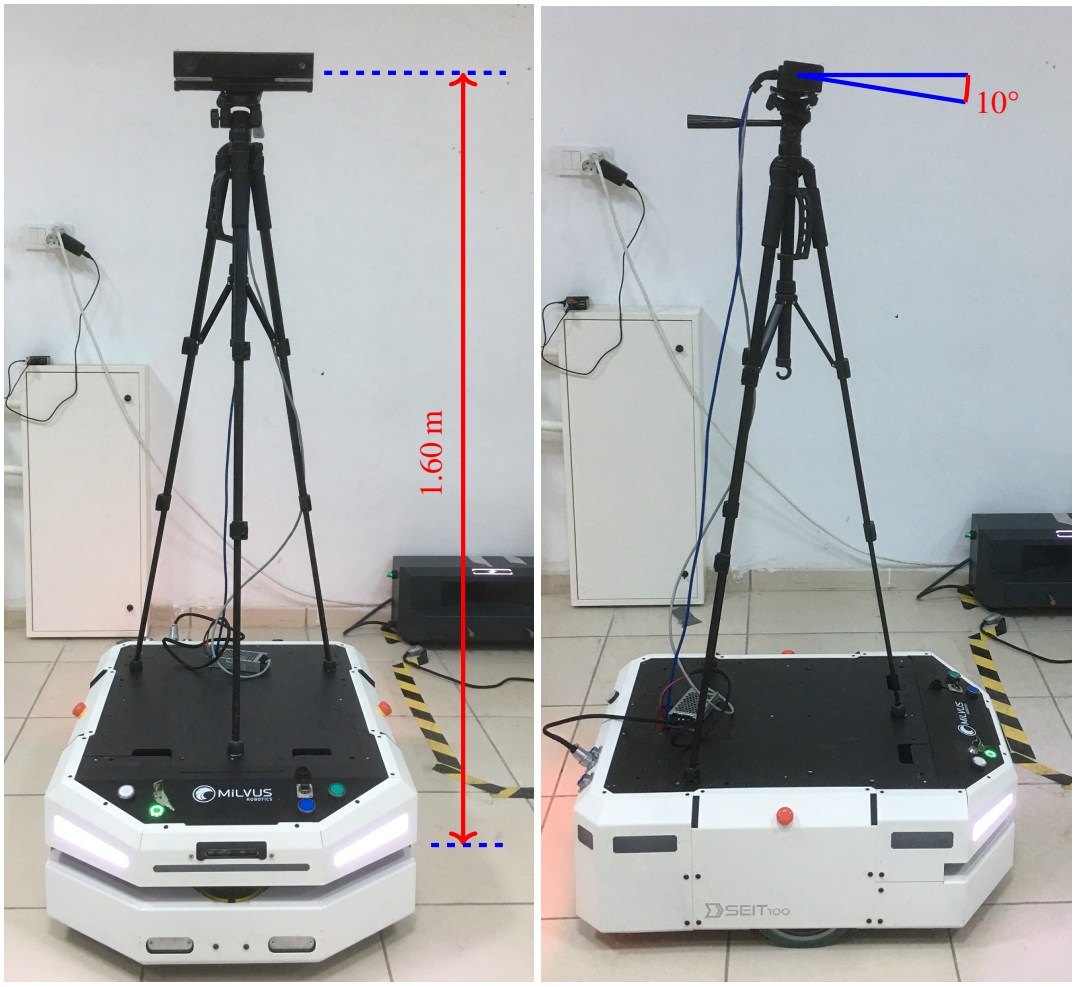


Figure 5.2: Kinect camera is mounted on SEIT 100 mobile robot.

SEIT 100 is equipped with a commercial navigation algorithm for safe and efficient navigation while carrying loads. For our experiments, we changed its path planner to our human aware path planner and used pure pursuit algorithm ([127–129]) for path tracking purposes.

5.2.2 Implementation

For experiments, the robot is equipped with our algorithms for people detection, tracking, trajectory prediction and path planning. We used Robot Operating System (ROS) as a base to implement our algorithms. Each component of this study is developed as a ROS package and at the end these packages are made to function together.

People detector runs continuously and processes each new frame gathered from Kinect which provides frames at 30 Hz. When people detector processes a frame and extracts people coordinates on the scene, it publishes this information over the network. People tracker, which also runs continuously, gets this message from people detector, and matches people identities with existing tracks or creates new tracks for new people in the scene. Generated tracks message is again published over the network. At this point, trajectory predictor and path planner gathers this tracks message. Additionally, besides people, trajectory planner adds the robot itself as another track to the tracks list. Trajectory predictor incorporates past trajectory information of each track to train predictions' Gaussian processes. With applying steps described in previous chapter, trajectory predictor estimates trajectories for each track. The prediction calculated for the robot itself is used as the planned path and sent to the mobile base controller to make it follow. The whole implementation is shown in Fig. 5.3.

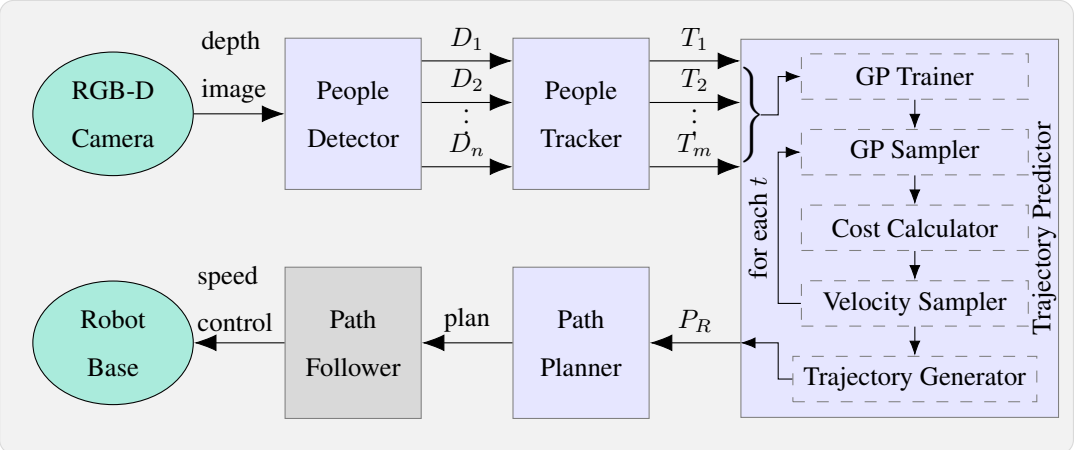
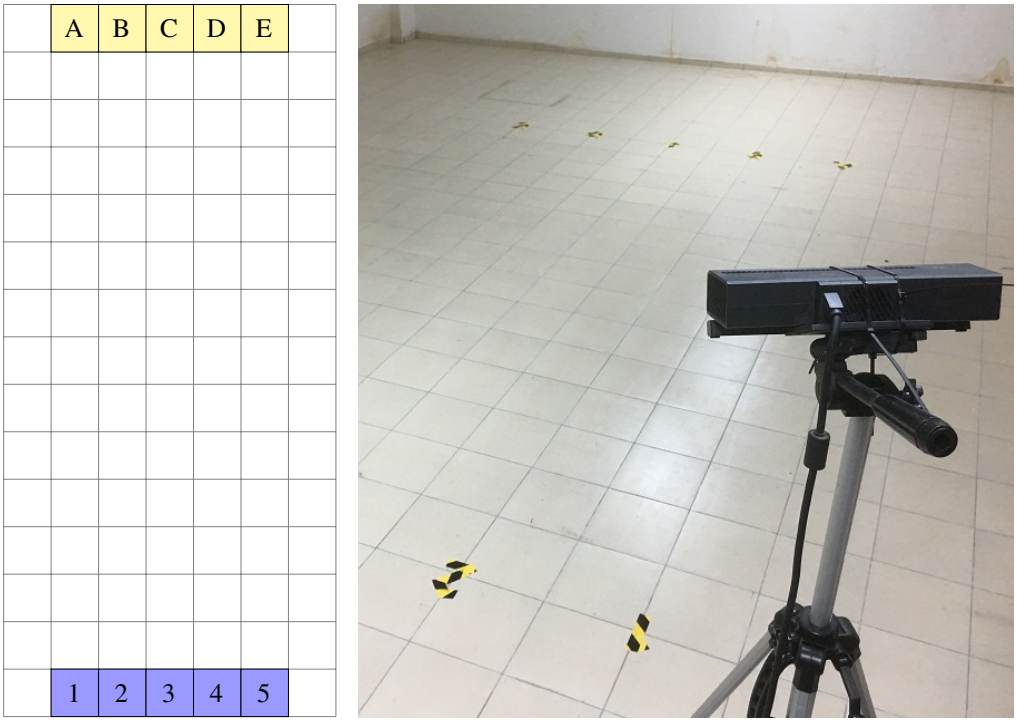


Figure 5.3: The implementation diagram on the mobile robot.



(a) (b)

Figure 5.4: Grid-based illustration of the scene (a) and developed tracking system for ground truth positions (b).

5.2.3 Scene

As the test scene, an empty room is used. Possible start and goal positions are marked on the ground as A, B, C, D, E on one side and as 1, 2, 3, 4, 5 on the other side. Position markers on one side are placed 0.4 meters apart from each other. Opposite sides are positioned 6 meters apart from each other. People (or robot) starting from positions marked with letters ended up at those marked with numbers and vice versa.

For gathering ground truth positions of people during experiments, another Kinect-based setup is implemented. Kinect sensor is positioned at a certain height and angle towards the scene and configured to detect and track people using its proprietary software. Coordinates of people taken from Kinect are transformed into world coordinates and then projected to the ground plane. Algorithm runs at 30 Hz which is high enough to extract ground truth positions.

Figure 5.5: Test scene and ground markings.

5.2.4 Scenarios

We design three different experiments to verify human-likeness and human-awareness of our predicted trajectories. For the first configuration, two people walk directly towards each other and expected to avoid collision and reach final positions (Fig. 5.7a). With this configuration, the aim is to show avoidance capabilities of the method in direct encounter in opposite directions. Second configuration again includes two people however, this time they move diagonally which again results in collision around the middle of their trajectories (Fig. 5.7b). In third setup, two people move one way while one person moves opposite to them (Fig. 5.7c). This is the most representative scene to demonstrate human-aware navigation ability of the proposed method.

After all-human experiments, one of the participants is replaced with the mobile robot. In the third configuration, the participant moving towards two people is replaced, not one of the two people moving the same direction. Same tests are repeated with the robot, and results are recorded.

Finally, navigation algorithm of the robot is switched to a reactive planner based on dynamic window approach by Fox et al. [130] and all three configurations are tested again. The purpose is to compare and evaluate the improvement of the navigation with predicted cooperation included in the proposed method.

In experiments, there were 12 participants with 10 males and 2 females. They are informed about their target destinations for each configuration. Also, they are told to

Figure 5.6: When the lights of the robot turn from red to green and a buzzer sound is heard, experiment starts.

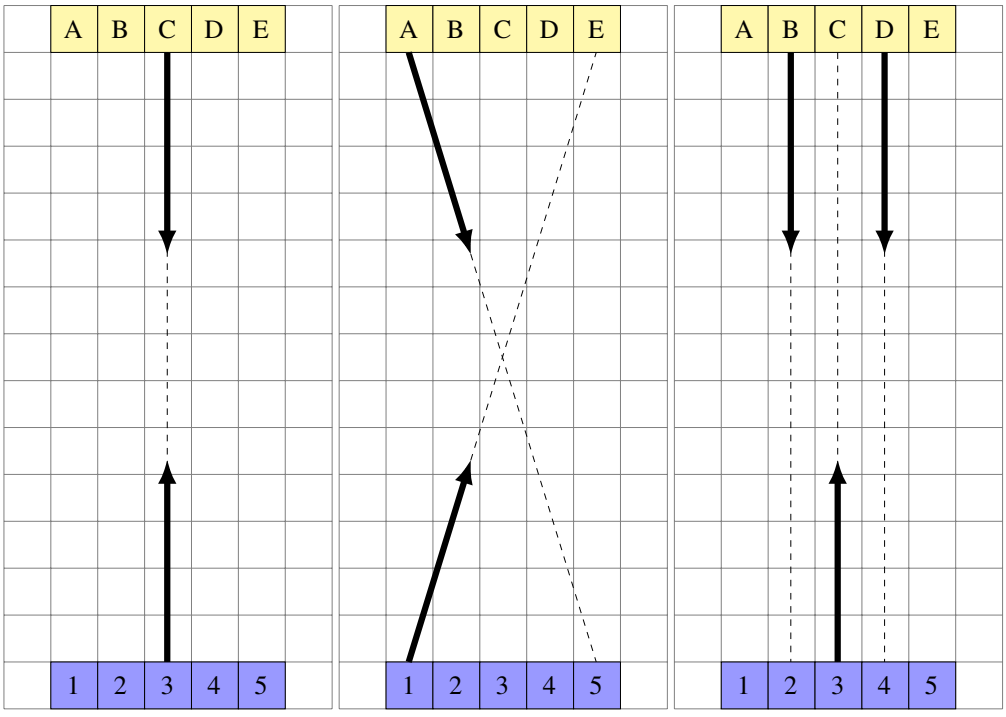


Figure 5.7: Three test configurations: one-to-one direct (a), one-to-one diagonal (b) and two-to-one direct (c).

behave naturally as if there is a human-being instead of the robot. Participants are positioned at starting points opposite to the robot. The robot is programmed to lit red indicator lights when idle (Fig. 5.6). The experiment start when red lights on the robot turn green (Fig. 5.6) and a buzzer sound is heard simultaneously. With both audio and visual indicators, participants start walking towards their goal while naturally avoiding other agents around. When both the robot and the participants reach their goal, the experiment is ended.

Every participant carries out each experiment two times. For the third scenario, participants changed starting position at each iteration. In total, each scenario is tested for three different methods 24 times which results in 216 runs in overall.

5.2.5 Evaluation

To evaluate the navigation performance of the algorithm on a mobile robot, two performance metrics from [52] are used: path length and safety margin. Path length is the Euclidean distance traveled by the robot in 2D plane, and safety margin is the closest Euclidean distance of the robot ever came to a pedestrian during a trial. Safety margin is also shown to be an important element in human-likeness perception in [82]. These two metrics are evaluated for experiments with and without the robot and results are compared with each other. We expect to have similar average results for the tests carried out with the robot equipped with our cooperative planner and for those with a human instead of the robot.

During each trial, the data from the robot and the camera (i.e., positions of the robot and participants) are recorded. In the post-processing step, positions of the robot and participants are synchronized using their time stamps. Then, the path lengths and safety margins are calculated for each run. Finally, they are averaged for each scenario with and without the robot, and results are compared.

5.3 Results

Among three test configurations, third one is the most demonstrative configuration for human-aware and human-like navigation. Second configuration shows the predictive capability of the algorithm. First configuration is a kind of obstacle avoidance scenario, yet, trajectory prediction on human, results in more user-friendly and human-like paths. Therefore, we will discuss results of all three configurations but we will mostly emphasize the results of the third configuration.

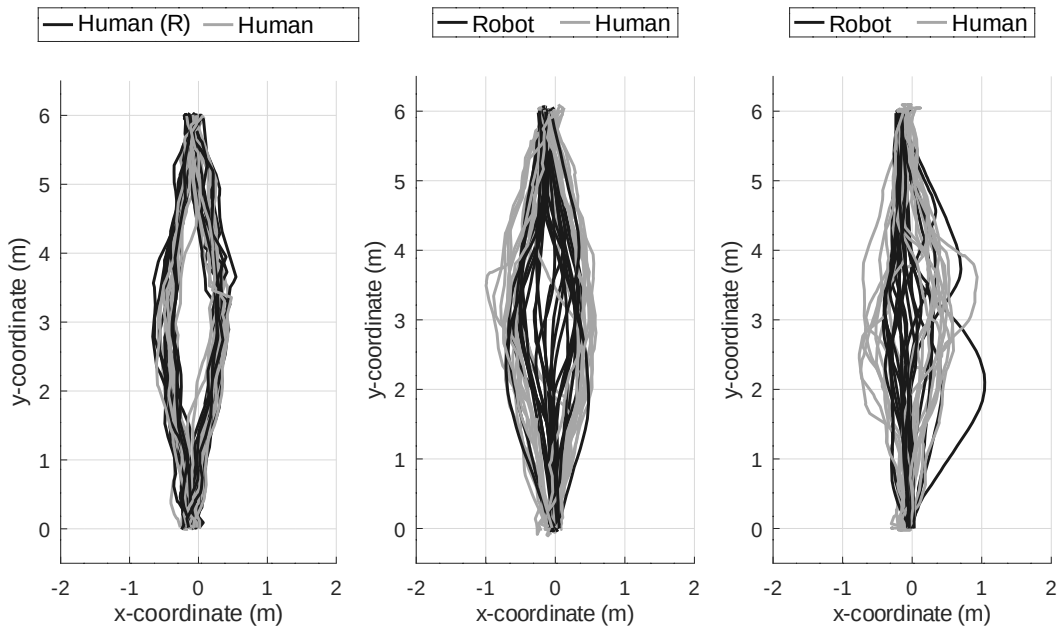


Figure 5.8: First test scenario with two persons (a), one person and the robot with proposed method (b), and one person and the robot with reactive planner (c).

First scenario is one-to-one direct encounter case. Two persons started walking towards starting positions of each other, avoiding collision while moving. Then, one of the people is replaced with the robot equipped with the proposed method and the test is repeated. Finally, same test is carried out again, this time, with the reactive planner. Note that, paths belong to people are a bit wavy in contrast to the robot's paths which are smoother. This is due to walking dynamics of people. Since we track heads for ground truth position, swinging motion of head in biped locomotion results in such

path trends.

Fig. 5.8 shows recorded paths of humans and the robot for all three methods: human-to-human, human-to-robot with the cooperative planner, and human-to-robot with the reactive planner. In Fig. 5.8a, paths traveled by humans are nearly symmetric and well-formed. Our method (Fig. 5.8b) records similar paths to that of human-to-human. We should note that, as also verified numerically in Table 5.2, average safety margin between the robot and the human is a bit larger in the cooperative robot test. This difference may be originated from the human behavior towards the robot especially in head-on encounter case. People tend to keep larger distance from the robot than from another human. The improvement of the navigation over a classic reactive planner is obvious when we compare Fig. 5.8b and Fig. 5.8c. The reactive planner, since it is not human aware, tries to navigate to its goal directly (see path lengths in Table 5.2). Paths of the robot are concentrated at the center of the image in Fig. 5.8c. This behavior obviously disrupts human, and causes them to take longer paths. Also since the robot is not predictive, it approaches human too much resulting in smaller safety margins (see safety margins in Table 5.2).

The second configuration again has two people, this time, navigating diagonally. The scenario is set up in a way that in normal navigation two people will have to avoid collision at around the middle of their trajectories. The expected human behavior is to

Table 5.2: Path lengths and minimum distances achieved between agents for the first test scenario with different methods: Human to Human (HH), Human to Robot - Cooperative (HR-C), and Human to Robot - Reactive (HR-R).

Method	Agent	Safety Margin	Path Length
HH	Human 1	0.774	6.197
	Human R	-	6.211
HR-C	Human	0.894	6.338
	Robot	-	6.107
HR-R	Human	0.714	6.494
	Robot	-	6.118

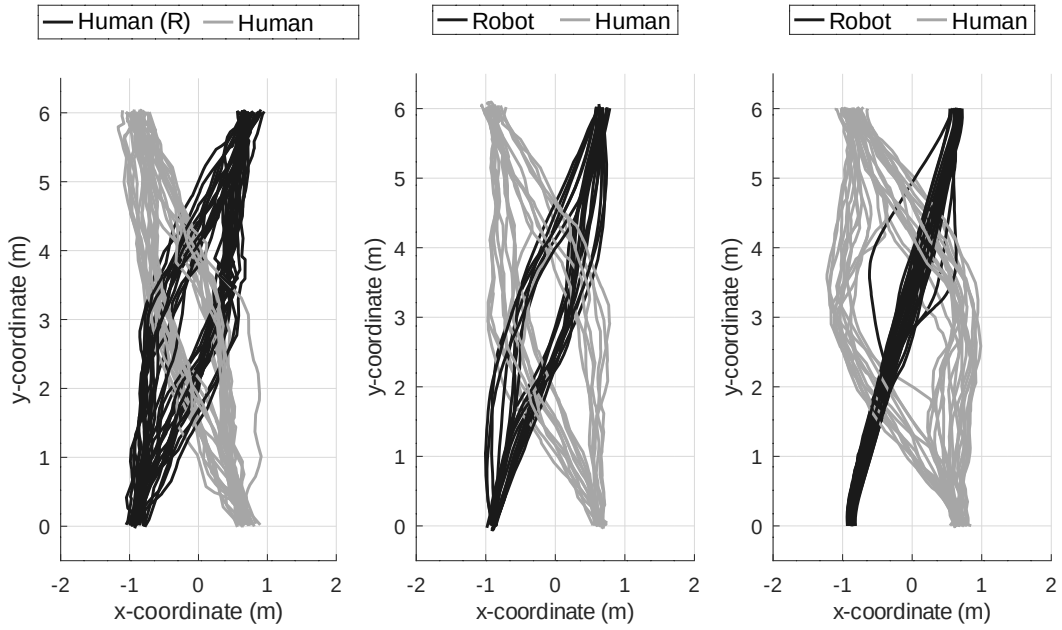


Figure 5.9: Second test scenario with two persons (a), one person and the robot with proposed method (b), and one person and the robot with reactive planner (c).

Table 5.3: Path lengths and minimum distances achieved between agents for the second test scenario with different methods: Human to Human (HH), Human to Robot - Cooperative (HR-C), and Human to Robot - Reactive (HR-R).

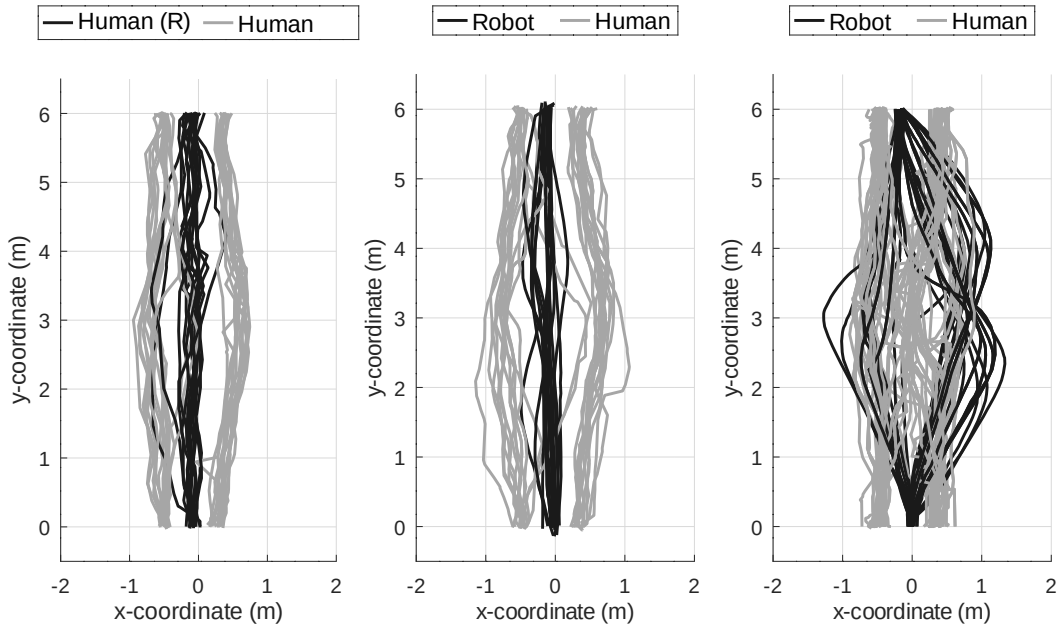
Method	Agent	Safety Margin	Path Length
HH	Human 1	0.829	6.418
	Human R	-	6.424
HR-C	Human	0.910	6.475
	Robot	-	6.297
HR-R	Human	0.929	6.643
	Robot	-	6.233

estimate the trajectory of the other human and adjust his/her path accordingly, keeping a safety margin in between. This expected behavior is verified with human-to-human tests as seen in Fig. 5.9a. Path lengths given in Table 5.3 for human-to-human case are also representative of this behavior as they are very close to each other. Safety margin is larger than the previous configuration because agents move diagonally, allowing short period of time to come close to each other. Here, the proposed method (i.e., human-to-robot with the cooperative planner, Fig. 5.9b) can be seen as very similar to that of human-to-human case. Our cooperative planner estimates the trajectory of the human and adjusts its path accordingly, and resembles the human-to-human case. Even though there is a small difference (about 0.15 meters), path lengths are still close to each other. Increased path length of human is due to the larger safety margin the human keeps to the robot than to a human. Yet, safety margins are similar to each other. The reactive planner fails to adjust any path, since it does not regard the human until it approaches to close vicinity. Robot paths with the reactive planner are mostly linear, disregarding the human, and the human takes evasive maneuvers keeping a larger safety margin due to uncooperative behavior of the robot.

Table 5.4: Path lengths and minimum distances achieved between agents for the third test scenario with different methods: Human to Human (HH), Human to Robot - Cooperative (HR-C), and Human to Robot - Reactive (HR-R).

Method	Agent	Safety Margin	Path Length
HH	Human 1	0.741	6.178
	Human 2	0.727	6.183
	Human R	-	6.180
HR-C	Human 1	0.721	6.274
	Human 2	0.795	6.228
	Robot	-	6.127
HR-R	Human 1	1.120	6.401
	Human 2	0.933	6.592
	Robot	-	6.428

The success of the proposed method is best represented in the third configuration



(a) (b) (c)

Figure 5.10: Third test scenario with three persons (a), two persons and the robot with proposed method (b), and two persons and the robot with reactive planner (c).

where two people move towards the robot without any room in between for the robot to pass. Here, the robot has to predict and mimic the cooperative behavior of people and navigate accordingly. As seen in Fig. 5.10a, even though there is no room initially, while approaching, two people generally cooperate and make room for the other human to pass in between. This is the expected and the verified behavior of human navigation. Only in few trials, the human, representing the robot, preferred to pass on the outer side. At Table 5.4, it is seen that path lengths of all three humans are similar. Also, safety margins kept to the human representing the robot ("Human R") are also nearly the same. This navigation behavior is replicated by our cooperative human-aware path planner. Paths traveled by the robot and humans are very similar to the human-to-human case as seen in Fig. 5.10a and 5.10b. Numerical results on path lengths and safety margins (Table 5.4) also prove the method to be human-like and human-aware. Safety margins in human-to-human and human-to-robot with the cooperative planner scenarios are nearly the same. This indicates that the robot is accepted as a rational agent by other people and they did not need to employ unnatural navigation behavior. Human-to-robot with the reactive planner scenario demonstrates

failure of classic non-cooperative planners. The reactive planner does not account any human intention or cooperation, therefore, it generally plans on the outer parts of the scene since the direct path to the goal is blocked by two people. If the number of people increases to an amount that will block the whole passage, the reactive planner obviously will halt and "the freezing robot problem" will occur. Also, with the reactive planner, people tend to keep larger distances to the robot, and the robot tends to move farther away than people which increases safety margins above 1 meter.

5.3.1 Statistical Analysis

The third configuration, two-to-one direct encounter, is the one which proves the cooperation and interaction prediction capability of the proposed method. Recorded trajectories are very similar with the all-human case. Even though the results in Table 5.4 show the human-like trajectory prediction and path planning performance of our method with mean values of safety margin and path length, we further carried out a statistical analysis for this configuration to validate the results in terms of statistical significance. We state two hypotheses as,

Hypothesis 1: *Path lengths, traveled by humans and the robot equipped with our human-aware navigation method, cannot be distinguished from each other. They are formed as equally human-like motions.*

Hypothesis 2: *Safety margins, occurred between humans and the robot equipped with our human-aware navigation method, cannot be distinguished from each other. They are formed as equally human-like motions.*

Samples of path length and safety margin are tested with a Shapiro-Wilk test. The p values of all tests are larger than 0.05; hence, all data are normally distributed. Then, independent t-test is applied to see if there are statistically significant differences between path lengths and safety margins in three methods.

Table 5.5: Enumeration of path lengths of humans (located on left, middle and right of the scene) and the robot for interpretation of statistical analysis.

HH Case	HR-C Case	HR-R Case
$L_1 \rightarrow$ human (left)	$L_4 \rightarrow$ human (left)	$L_7 \rightarrow$ human (left)
$L_2 \rightarrow$ human (middle)	$L_5 \rightarrow$ the robot	$L_8 \rightarrow$ the robot
$L_3 \rightarrow$ human (right)	$L_6 \rightarrow$ human (right)	$L_9 \rightarrow$ human (right)

Table 5.6: Pairwise comparisons of path lengths of the robots and the corresponding human, and resulting *p* values.

Variable	L_2	L_8
L_5	0.074	< 0.001
L_6	< 0.001	–

Table 5.7: Pairwise comparisons of path lengths of humans on the left and the right, and resulting *p* values.

Variable	L_1	L_4	L_7
L_3	0.903	–	–
L_6	–	0.307	–
L_9	–	–	0.012

To test the Hypothesis 1, path lengths of three agents in three methods are enumerated in Table 5.5. Calculated *p* values using independent t-test are tabulated in Table 5.6. There is no significant difference between path lengths of the human in the middle of the all-human experiment and the robot in the cooperative case. Also, it can be seen that the reactive planner produces significantly different paths as *p* values are even less than 0.001. These results confirm the Hypothesis 1, yet, we still examine the differences of people on the right and the left of scenes (i.e., symmetry) in experiments

Table 5.8: Enumeration of safety margins of humans (located on left, middle and right of the scene) and the robot for interpretation of statistical analysis.

HH Case	HR-C Case	HR-R Case
$S_1 \rightarrow$ left & middle	$S_3 \rightarrow$ left & robot	$S_5 \rightarrow$ left & robot
$S_2 \rightarrow$ right & middle	$S_4 \rightarrow$ right & robot	$S_6 \rightarrow$ right & robot

Table 5.9: Pairwise comparisons of safety margins between humans on the left and agents in the middle in all three methods, and resulting *p* values.

Variable	S_1	S_5
S_3	0.322	0.046
S_5	0.005	–

with three different methods. In Table 5.7, it is shown that there is no statistically significant difference on path lengths of people on each side of the scenes for the all-human and the cooperative robot cases. This also supports the Hypothesis 1. In the reactive robot method, there is a difference ($p < 0.05$) due to unpredictable behavior of the robot and necessary avoidance maneuvers of people.

To test the Hypothesis 2, similarly, safety margins between different agents in three methods are enumerated in Table 5.8. Calculated *p* values using independent t-test are tabulated in Table 5.9 and 5.10. Safety margin between the human on the left and at the middle in the all-human case is not different significantly ($p > 0.05$) from the margin between the human on the left and the robot in cooperative case. Similarly, the safety margin between the human on the left and at the middle in the all-human case is not different significantly ($p > 0.05$) from the margin between the human on the left and the robot in cooperative case. This approves the Hypothesis 2. For the two cases, however, differences are significant for the reactive robot ($p < 0.05$). Safety margins for two humans walking in the same direction in the all-human and cooperative robot cases do not differ significantly (Table 5.11). This also supports the Hypothesis2.

Table 5.10: Pairwise comparisons of safety margins between humans on the right and agents in the middle in all three methods, and resulting p values.

Variable	S_2	S_6
S_4	0.696	< 0.001
S_6	0.001	–

Table 5.11: Pairwise comparisons of safety margins for humans on the left and the right to agents in the middle, and resulting p values.

Variable	S_1	S_3	S_5
S_2	0.841	–	–
S_4	–	0.138	–
S_6	–	–	0.021

5.4 Conclusions

In previous chapters, different components of the proposed human-aware navigation method is presented. People detection and tracking methods are tested and verified. The developed trajectory prediction and path planning algorithm is described in the previous chapter and tested on a publicly available dataset. Results are also compared with state-of-art methods and shown that our proposed method performs superior.

In this chapter, the aim was to combine all components, implement on a mobile robot and prove its success for human-aware navigation. For this purpose, we developed a software package which includes all components proposed in this study and conducted experiments with real conditions and humans. Three different experiment configurations are tested with human-to-human, human-to-robot with our cooperative planner and human-to-robot with the reactive planner cases. During tests, people detection, tracking, trajectory prediction and path planning components were all

active and functioning. Obtained results are compared and it is shown that our cooperative planner behaves very similarly to real humans in terms of human-awareness and human-likeness during navigation. Also, by comparing with a reactive planner, improvement of navigation over classical reactive methods is shown. Overall results obtained in this chapter verify and conclude this study successfully.

CHAPTER 6

CONCLUSIONS

As mobile robots become more integrated into our daily life, classical navigation methods become insufficient. Environments crowded with people are dynamic and stochastic so that to be able to navigate effectively, mobile robots should inherit human navigational behaviors. In this study, our aim was to develop an efficient and robust navigation method which is human-like and human-aware. Mobile robots have to detect people around to be able to predict their movements and navigate accordingly. Therefore, in Chapter 2, we developed a lightweight people detection method which uses 3-D depth information gathered from an RGB-D camera. Algorithm runs in real-time, solely on a CPU without any GPU implementation. This allows implementation of the method on low-resource mobile robots. Its success is verified on a public dataset first, and then on a custom one. Besides being real-time and depending only on depth (not on RGB), contributions of the Chapter 2 include pose invariant people detection. Using eigenvectors and eigenvalues of point clouds, we processed clusters in a special pipeline and used support vector machines to classify them as human or not.

In Chapter 3, a people tracking method is implemented. Detected people in the previous chapter need to be tracked for trajectory estimation. Occlusions and direction changes may result in identity switching, however, implemented tracking method with Kalman filter and nearest neighbor algorithm running in real-time made it possible to track people around a mobile robot. This implementation is also tested and verified on a public dataset.

When we can detect and track people, we can predict their trajectories and navigate accordingly. Chapter 4 is devoted to this core component of the study. Research in

the literature showed that even with perfect detection and tracking, without human interaction and cooperation modeling, a navigation algorithm will fail with so-called "freezing robot problem". Therefore, the aim was to model this cooperation which, as humans, we have and use in our daily life. To generate human-like trajectories, we used Gaussian processes which is trained with real observed data (i.e., the past trajectory of people) taken from the people tracker. However, generating human-like trajectories is not sufficient without cooperation modeling. We developed a step-based interactive prediction stage. In addition to past trajectories of people, in each future time step, we trained Gaussian processes with estimated future positions of people extracted from a cost-based interaction method. Costs of an individual map of each agent are penalized with accelerations and deceleration, and final costs are used in next best velocity selection. We modeled the mobile robot as just another human in the scene. Therefore, when we predict cooperative trajectories of all agents in the scene, path planning of the mobile robot becomes just an inference from probability density function of the robot. The mean (the most likely sample) of the Gaussian process is used as the planned path of the robot.

Findings of Chapter 4 are tested on a public dataset which has pedestrians walking in front of a library, recorded from a top camera. Positions of people are labeled with their identities. The developed trajectory prediction method is tested on challenging scenes of this dataset. Especially, crowded scenes which require cooperation and interaction between people, are selected. Developed method does not need to incorporate true goal information prior to prediction. Still, tests are conducted on two different configurations: known goal and unknown goal cases. Results for the most challenging scenes are presented in detail. Then, overall results are presented for both known and unknown goal cases. Finally, our method is compared with two state-of-art methods in the literature and shown to perform superior than these methods even in long prediction horizon. Development of a scene and data independent interaction and cooperation model for humans using future conditioning of Gaussian processes with cost-based velocity selection is the main contribution of the Chapter 4.

Three different components, i.e., people detection, people tracking, and trajectory prediction and path planning, are developed and tested individually during the study. In Chapter 5, all components are brought together and implemented on a mobile

robot. An RGB-D camera is mounted on the robot and developed algorithms are implemented. To test all components together, experiments are designed. Three different scenarios are prepared: one-to-one direct movement, one-to-one diagonal movement, and two-to-one direct movement. These three configurations are tested for three different cases: human-to-human, human-to-robot with the cooperative planner, and human-to-robot with the reactive planner. Human-to-human cases are taken as benchmarks. Human participants instructed, experiments are carried out and data are recorded. In the Results section of the Chapter 5, it is shown that, paths traveled in human-to-human and human-to-robot with the cooperative planner cases are very similar. Findings are supported in terms of path lengths and safety distances. Also, results are compared with the reactive planner, and improvement in navigation is indicated. Results of Chapter 5 verify that output of this study, with all components, successfully makes a mobile robot be able to navigate in human-like and human-aware fashion. We believe that; this study made a considerable contribution to achieve more human-aware and human-like robots in near future.

REFERENCES

[1] W. Burgard, A. B. Cremers, D. Fox, D. Haehnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Interactive museum tour-guide robot," *Proceedings of the National Conference on Artificial Intelligence*, pp. 11–18, 1998.

[2] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, H. Dirk, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "MINERVA: A Second-Generation Museum Tour-Guide Robot,"

[3] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, D. Schulz, and W. Steiner, "Experiences with two deployed interactive tour-guide robots," *Proceedings of the International Conference on Field and Service Robotics (FSR'99)*, 1999.

[4] I. R. Nourbakhsh, C. Kunz, and T. Willeke, "The mobot museum robot installations: A five year experiment," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 4, pp. 3636–3641, IEEE, 2003.

[5] M. Shiomi, T. Kanda, D. F. Glas, S. Satake, H. Ishiguro, and N. Hagita, "Field trial of networked social robots in a shopping mall," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2846–2853, 2009.

[6] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *International Conference on Robotics and Automation*, 2010.

[7] S. Srinivasa, D. Ferguson, M. Vande Weghe, R. Diankov, D. Berenson, C. Helfrich, and H. Strasdat, "The robotic busboy: Steps towards developing a mobile robotic home assistant," *Intelligent Autonomous Systems 10, IAS 2008*, pp. 129–136, 2008.

[8] M. Mori, K. F. MacDorman, and N. Kageki, "The uncanny valley [from the field]," *IEEE Robotics Automation Magazine*, vol. 19, pp. 98–100, June 2012.

[9] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pp. 886–893, 2005.

[10] C. Hua, Y. Makihara, Y. Yagi, S. Iwasaki, K. Miyagawa, and B. Li, "Onboard monocular pedestrian detection by combining spatio-temporal hog with structure from motion algorithm," *Machine Vision and Applications*, vol. 26, no. 2-3, pp. 161–183, 2015.

[11] a. Ess, B. Leibe, K. Schindler, and L. Van Gool, "A mobile vision system for robust multi-person tracking," *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008.*, pp. 1–8, 2008.

[12] L. Spinello and K. O. Arras, "People detection in RGB-D data," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 3838–3843, 2011.

[13] B. Choi, C. Mericli, J. Biswas, and M. Veloso, "Fast human detection for indoor mobile robots using depth images," *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1108–1113, 2013.

[14] F. Hegger, N. Hochgeschwender, G. K. Kraetzschmar, and P. G. Ploeger, "People Detection in 3D Point Clouds using Local Surface Normals," *Proc. of the Robocup Symposium*, 2012.

[15] A. P. Gritti, T. Oscar, J. Guzzi, G. A. Di Caro, C. Vincenzo, L. M. Gambardella, and A. Giusti, "Kinect-based People Detection and Tracking from Small-Footprint Ground Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4096–4103, 2014.

[16] J. Liu, Y. Liu, G. Zhang, P. Zhu, and Y. Q. Chen, "Detecting and tracking people in real time with RGB-D camera," *Pattern Recognition Letters*, vol. 53, pp. 16–23, 2014.

[17] G. Zhang, J. Liu, H. Li, Y. Q. Chen, and L. S. Davis, "Joint human detection and head pose estimation via multistream networks for rgb-d videos," *IEEE Signal Processing Letters*, vol. 24, pp. 1666–1670, 2017.

[18] T. E. Tseng, A. S. Liu, P. H. Hsiao, C. M. Huang, and L. C. Fu, "Real-time people detection and tracking for indoor surveillance using multiple top-view depth cameras," in *IEEE International Conference on Intelligent Robots and Systems*, no. Iros, pp. 4077–4082, 2014.

[19] B. K. Dan, Y. S. Kim, Suryanto, J. Y. Jung, and S. J. Ko, "Robust people counting system based on sensor fusion," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 3, pp. 1013–1021, 2012.

[20] C. Migniot and F. Ababsa, "Hybrid 3D–2D human tracking in a top view," *Journal of Real-Time Image Processing*, vol. 11, no. 4, pp. 769–784, 2016.

[21] M. Bajracharya, B. Moghaddam, A. Howard, S. Brennan, and L. H. Matthies, "A fast stereo-based system for detecting and tracking pedestrians from a moving vehicle," *International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1466–1485, 2009.

[22] M. Munaro and E. Menegatti, "Fast RGB-D people tracking for service robots," *Autonomous Robots*, vol. 37, no. 3, pp. 227–242, 2014.

[23] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System," in *IEEE International Conference on Robotics and Automation, Open-Source Software workshop*, 2009.

[24] R. B. Rusu and S. Cousins, "3D is here: point cloud library," in *IEEE International Conference on Robotics and Automation*, 2011.

[25] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[26] A. J. B. Trevor, S. Gedikli, R. B. Rusu, and H. I. Christensen, "Efficient organized point cloud segmentation with connected components," in *3rd Workshop*

on Semantic Perception Mapping and Exploration (SPME), Karlsruhe, Germany, 2013.

[27] R. B. Rusu, *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science Department, Technische Universitaet Muenchen, Germany, October 2009.

[28] M. H. Al-Haboubi, "Anthropometry for a mix of different populations," *Applied Ergonomics*, vol. 23, no. 3, pp. 191–196, 1992.

[29] R. M. H. W. Stoudt, A. Damon and J. Roberts, *National Health Survey 1962: Weight, Height and Selected Body Dimensions of Adults, United States 1960–1962*. Washington D.C.: U.S. Government Printing Office, 1965.

[30] J. Panero and M. Zelnik, *Human dimensions and interior space*. No. 39, Watson-Guptill, 1979.

[31] C.-c. Chang and C.-j. Lin, "LIBSVM : A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, pp. 1–39, 2013.

[32] J. C. Platt, "Probabilities for SV Machines," in *Advances in Large-Margin Classifiers* (A. J. Smola, P. J. Bartlett, B. Schölkopf, and D. Schuurmans, eds.), pp. 61–74, MIT Press, 2000.

[33] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the viewpoint feature histogram," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 2155–2162, 2010.

[34] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," in *IEEE International Conference on Robotics and Biomimetics*, pp. 2987–2992, 2011.

[35] S. Lazebnik, C. Schmid, and J. Ponce, "A sparse texture representation using local affine regions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1265–1278, 2005.

[36] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH)

for 3D registration," in *IEEE International Conference on Robotics and Automation*, pp. 3212–3217, 2009.

[37] Z. C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz, "General 3D modelling of novel objects from a single view," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 3700–3705, 2010.

[38] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[39] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 304–311, 2009.

[40] H. Zhang, C. Reardon, and L. E. Parker, "Real-time multiple human perception with color-depth cameras on a mobile robot," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1429–1441, 2013.

[41] J. Salas and C. Tomasi, "People detection using color and depth images," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6718 LNCS, no. 25288, pp. 127–135, 2011.

[42] J. Han, E. J. Pauwels, P. M. De Zeeuw, and P. H. De With, "Employing a RGB-D sensor for real-time tracking of humans across multiple re-entries in a smart environment," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, pp. 255–263, 2012.

[43] M. Bansal, S. H. Jung, B. Matei, J. Eledath, and H. Sawhney, "A real-time pedestrian detection system based on structure and appearance classification," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 903–909, 2010.

[44] D. M. Vo, L. Jiang, and A. Zell, "Real Time Person Detection and Tracking by Mobile Robots using RGB-D Images," *IEEE International Conference on Robotics and Biomometics*, pp. 689–694, 2014.

[45] M. Luber, L. Spinello, and K. O. Arras, "People tracking in RGB-D data with on-line boosted target models," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 3844–3849, 2011.

[46] R. Muñoz-Salinas, M. García-Silvente, and R. M. Carnicer, "Adaptive multimodal stereo people tracking without background modelling," *J. Visual Communication and Image Representation*, vol. 19, pp. 75–91, 2008.

[47] N. Bellotto and H. Hu, "Computationally efficient solutions for tracking people with a mobile robot: An experimental evaluation of Bayesian filters," *Autonomous Robots*, vol. 28, no. 4, pp. 425–438, 2010.

[48] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "People tracking with a mobile robot using sample-based joint probabilistic data association filters," *International Journal of Robotics Research*, vol. 22, no. 2, pp. 909–116, 2003.

[49] J. Correa, J. Liu, and G. Z. Yang, "Real time people tracking in crowded environments with range measurements," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8239 LNAI, pp. 471–480, 2013.

[50] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, pp. 83–97, 1955.

[51] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 797–803, oct 2010.

[52] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: the case for cooperation," *2013 IEEE International Conference on Robotics and Automation*, pp. 2153–2160, may 2013.

[53] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, no. 5, pp. 4282–4286, 1995.

[54] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, no. July, pp. 487–490, 2000.

[55] D. Helbing, P. Molnár, I. J. Farkas, and K. Bolay, "Self-organizing pedestrian movement," *Environment and Planning B: Planning and Design*, vol. 28, no. 3, pp. 361–383, 2001.

[56] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras, "People tracking with human motion predictions from social forces," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 464–469, 2010.

[57] S. Pellegrini, A. Ess, K. Schindler, and L. V. Gool, "You'll NeverWalk Alone: Modeling Social Behavior for Multi-target Tracking.pdf," no. Iccv, pp. 261–268, 2009.

[58] S. Pellegrini, A. Ess, M. Tanaskovic, and L. Van Gool, "Wrong turn - No dead end: A stochastic pedestrian motion model," *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010*, pp. 15–22, 2010.

[59] M. Bennewitz, "Learning Motion Patterns of People for Compliant Robot Motion," *The International Journal of Robotics Research*, vol. 24, pp. 31–48, jan 2005.

[60] S. Thompson, T. Horiuchi, and S. Kagami, "A probabilistic model of human motion and navigation intent for mobile robot path planning," *ICARA 2009 - Proceedings of the 4th International Conference on Autonomous Robots and Agents*, pp. 663–668, 2009.

[61] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, no. May 2014, pp. 3931–3936, 2009.

[62] N. Pradhan, T. Burg, and S. Birchfield, "Robot crowd navigation using predictive position fields in the potential function framework," *Proceedings of the 2011 American Control Conference*, pp. 4628–4633, jun 2011.

[63] H. Morioka, S. Yi, and O. Hasegawa, "Vision-based mobile robot's SLAM and navigation in crowded environments," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3998–4005, sep 2011.

[64] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," in *Robotics: Science and Systems*, 2012.

[65] N. Epley, A. Waytz, and J. T. Cacioppo, "On seeing human: a three-factor theory of anthropomorphism.," *Psychological review*, vol. 114, no. 4, pp. 864–86, 2007.

[66] C. C. Yu and C. C. Wang, "Collision- and freezing-free navigation in dynamic environments using Learning to Search," in *Proceedings - 2012 Conference on Technologies and Applications of Artificial Intelligence, TAAI 2012*, pp. 151–156, Ieee, nov 2012.

[67] T. Kruse, P. Basili, S. Glasauer, and a. Kirsch, "Legible robot navigation in the proximity of moving humans," *2012 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pp. 83–88, may 2012.

[68] H. Kidokoro and T. Kanda, "Will I bother here?-A robot anticipating its influence on pedestrian walking comfort," *2013 IEEE International Conference on Human-Robot Interaction (HRI)*, may 2013.

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

[70] M. Kuderer, H. Kretzschmar, and W. Burgard, "Teaching mobile robots to cooperatively navigate in populated environments," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3138–3143, nov 2013.

[71] R. Kummerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard, "A navigation system for robots operating in crowded urban environments," *2013 IEEE International Conference on Robotics and Automation*, pp. 3225–3232, may 2013.

[72] E. Pacchierotti, H. I. Christensen, and P. Jensfelt, "Embodied social interaction for service robots in hallway environments," in *FSR*, 2005.

[73] L. Scandolo and T. Fraichard, "An anthropomorphic navigation scheme for dynamic scenarios," *2011 IEEE International Conference on Robotics and Automation*, pp. 809–814, 2011.

[74] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "Understanding human interaction for probabilistic autonomous navigation using risk-rrt approach," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2014–2019, 2011.

[75] Y. Nakauchi and R. G. Simmons, "A social robot that stands in line," *Auton. Robots*, vol. 12, pp. 313–324, 2002.

[76] E. T. Hall, *Handbook For Proxemic Research*. Society for the Anthropology of Visual Communication, 1974.

[77] R. D. Middlemist, E. S. Knowles, and C. F. Matter, "Personal space invasions in the lavatory: suggestive evidence for arousal.," *Journal of personality and social psychology*, vol. 33 5, pp. 541–6, 1976.

[78] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," *2010 IEEE International Conference on Robotics and Automation*, pp. 981–986, may 2010.

[79] B. Kim and J. Pineau, "Human-like navigation: Socially adaptive path planning in dynamic environments," in *RSS 2013 Workshop on Inverse Optimal Control and Robotic Learning from Demonstration*, 2013.

[80] D. Vasquez, B. Okal, and K. O. Arras, "Inverse Reinforcement Learning algorithms and features for robot navigation in crowds: An experimental comparison," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1341–1346, sep 2014.

[81] A. Turnwald, D. Althoff, D. Wollherr, and M. Buss, "Understanding Human Avoidance Behavior: Interaction-Aware Decision Making Based on Game Theory," *International Journal of Social Robotics*, vol. 8, no. 2, pp. 331–351, 2016.

[82] A. Turnwald and D. Wollherr, "Human-Like Motion Planning Based on Game Theoretic Decision Making," *International Journal of Social Robotics*, 2018.

[83] A. Vemula, K. Muelling, and J. Oh, "Modeling cooperative navigation in dense human crowds," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1685–1692, 2017.

[84] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–971, 2016.

[85] A. Alahi, V. Ramanathan, and L. Fei-Fei, "Tracking Millions of Humans in Crowded Spaces," *Group and Crowd Behavior for Computer Vision*, no. i, pp. 115–135, 2017.

[86] P. Trautman, "Sparse interacting Gaussian processes: Efficiency and optimality theorems of autonomous crowd navigation," *2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017*, vol. 2018-January, no. May, pp. 327–334, 2018.

[87] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *International Journal of Robotics Research*, vol. 35, no. 11, pp. 1352–1370, 2016.

[88] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human-robot cooperation," *International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[89] A. V. Papadopoulos, L. Bascetta, and G. Ferretti, "Generation of human walking paths," *Autonomous Robots*, vol. 40, no. 1, pp. 59–75, 2016.

[90] S. M. LaValle, "Planning algorithms," *Planning Algorithms*, vol. 9780521862059, pp. 1–826, 2006.

[91] D. Ellis, E. Sommerlade, and I. Reid, "Modelling pedestrian trajectories with gaussian processes," *Ninth International Workshop on Visual Surveillance*, no. 1, pp. 27110–27110, 2009.

[92] K. Kim, D. Lee, and I. Essa, "Detecting regions of interest in dynamic scenes with camera motions," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1258–1265, 2012.

[93] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[94] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[95] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 3471–3476, 2008.

[96] H. Li, Y. Wu, and H. Lu, "Visual tracking using particle filters with gaussian process regression," in *PSIVT*, 2009.

[97] J. Ko and F. Dieter, "Learning GP-BayesFilters via Gaussian Process Latent Variable Models," *Autonomous Robots*, vol. 30, pp. 3–23, 2011.

[98] T. Klinger, F. Rottensteiner, and C. Heipke, "Probabilistic multi-person localisation and tracking in image sequences," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 127, pp. 73–88, 2017.

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

[100] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using the relative velocity paradigm," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[101] M. Svenstrup, T. Bak, and H. J. Andersen, "Trajectory planning for robots in dynamic human environments," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4293–4298, Oct 2010.

[102] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," tech. rep., 1998.

[103] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[104] L. Matthies and A. Elfes, "Integration of sonar and stereo range data using a grid-based representation," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pp. 727–733 vol.2, April 1988.

[105] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Magazine*, vol. 9, pp. 61–74, 1988.

[106] K. Konolige, "Improved occupancy grids for map building," *Auton. Robots*, vol. 4, pp. 351–367, 1997.

[107] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Auton. Robots*, vol. 15, pp. 111–127, 2003.

[108] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *IEEE Transactions on Robotics*, vol. 23, pp. 874–883, Oct 2007.

[109] R. Kirby, R. Simmons, and J. Forlizzi, "Companion: A constraint-optimizing method for person-acceptable navigation," in *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 607–612, Sep. 2009.

[110] E. T. Hall, *The Hidden Dimension*. New York: Doubleday, 1966.

[111] N. L. Ashton and M. E. Shaw, "Empirical investigations of a reconceptualized personal space," *Bulletin of the Psychonomic Society*, vol. 15, no. 5, pp. 309–312, 1980.

[112] J. R. Aiello, "Human spatial behavior," in *Handbook of Environmental Psychology*, pp. 359–504, New York: John Wiley & Sons, 1987.

[113] M. Gérin-Lajoie, C. L. Richards, and B. J. McFadyen, "The negotiation of stationary and moving obstructions during walking: anticipatory locomotor adaptations and preservation of personal space.," *Motor control*, vol. 9, no. 3, pp. 242–69, 2005.

[114] M. L. Walters, K. Dautenhahn, R. te Boekhorst, , C. Kaouri, S. Woods, C. Nehaniv, D. Lee, and I. Werry, "The influence of subjects' personality traits on personal spatial zones in a human-robot interaction experiment," in *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, pp. 347–352, Aug 2005.

[115] C. M. Wall-Scheffler, "Size and Shape: Morphology's Impact on Human Speed and Mobility," *Journal of Anthropology*, vol. 2012, pp. 1–9, 2012.

[116] J. Wagnild and C. M. Wall-Scheffler, "Energetic Consequences of Human Sociality: Walking Speed Choices among Friendly Dyads," *PLoS ONE*, vol. 8, no. 10, pp. 1–6, 2013.

[117] R. M. N. Alexander, "Energetics and optimization of human walking and running: The 2000 Raymond Pearl Memorial Lecture," *American Journal of Human Biology*, vol. 14, no. 5, pp. 641–648, 2002.

[118] S. M. O'Connor and J. M. Donelan, "Fast visual prediction and slow optimization of preferred walking speed," *Journal of Neurophysiology*, vol. 107, no. 9, pp. 2549–2559, 2012.

[119] G. J. Bastien, P. A. Willems, B. Schepens, and N. C. Heglund, "Effect of load and speed on the energetic cost of human walking," *European Journal of Applied Physiology*, vol. 94, no. 1-2, pp. 76–83, 2005.

[120] D. DeJaeger, P. A. Willems, and N. C. Heglund, "The energy cost of walking in children," *Pflugers Archiv European Journal of Physiology*, vol. 441, no. 4, pp. 538–543, 2001.

[121] R. C. Browning and R. Kram, "Energetic cost and preferred speed of walking in obese vs. normal weight women," *Obesity Research*, vol. 13, no. 5, pp. 891–899, 2005.

[122] R. C. Browning, E. A. Baker, J. A. Herron, and R. Kram, "Effects of obesity and sex on the energetic cost and preferred speed of walking," *Journal of Applied Physiology*, vol. 100, no. 2, pp. 390–398, 2006.

[123] A. M. Colman, "Cooperation, psychological game theory, and limitations of rationality in social interaction," *Behavioral and Brain Sciences*, vol. 26, no. 2, pp. 139–153, 2003.

[124] S. Bitgood and S. Dukes, "Not another step! Economy of movement and pedestrian choice point behavior in shopping malls," *Environment and Behavior*, vol. 38, no. 3, pp. 394–405, 2006.

[125] C. G, G. G, B. S, K. O, and B. M, "Goal attribution without agency cues: the perception of 'pure reason' in infancy," *Cognition*, vol. 72, no. 3, pp. 237–267, 1999.

[126] "Milvus robotics." `http://www.milvusrobotics.com`. Accessed: 2019-05-02.

[127] R. S. Wallace, A. Stentz, C. E. Thorpe, H. P. Moravec, W. Whittaker, and T. Kanade, "First results in robot road-following," in *IJCAI 1985*, 1985.

[128] O. Amidi, "Integrated mobile robot control," Tech. Rep. CMU-RI-TR-90-17, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pa, USA, 1990.

[129] R. C. Coulter, "Implementation of the pure-pursuit path tracking algorithm," Tech. Rep. CMU-RI-TR-92-01, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pa, USA, 1992.

[130] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, pp. 23–33, 1997.

CURRICULUM VITAE

Akif Hacinecipoglu
akifhno@gmail.com

EDUCATION

- **B.Sc.** Mechanical Engineering, Middle East Technical University, 2009
- **M.Sc.** Mechanical Engineering, Middle East Technical University, 2012
- **Ph.D.** Mechanical Engineering, Middle East Technical University, 2019

EXPERIENCE

- **Research and Teaching Assistant**, Mechanical Engineering Department, Middle East Technical University, 2010-2018
- **Co-Founder and Chief Technology Officer**, Milvus Robotics Ltd., 2011-Present

AREAS OF EXPERTISE

- Robotics
- System Dynamics and Control
- Autonomy
- Mechatronics
- Path Planning

- Robot Vision
- Human Computer Interaction

LANGUAGE SKILLS

- **Turkish** - Native
- **English** - Proficient
- **German** - Elementary

PROJECTS

- **Researcher**, Uzaktan Kullanıcı Etkileşimli Yarı-otonom Insansiz Kara Aracı Platformu, 2012-2014, *Funded by TUBITAK*.
- **Researcher**, Uzaktan Erişimli Yarı Otonom İnsansız Kara Aracı Platformunun Etkileşimli Kontrolü, 2010-2012, *Funded by BAP, METU*.
- **Project Coordinator**, Elektronik ve Yazılım Destekli Yarı-Otonom Hareketli Robot Platformu, 2011-2014, *Funded by KOSGEB*.
- **Project Coordinator**, Fabrika İçi İş Süreçleri İçin Otonom Hareket Destekli Lojistik Taşıma Aracı Robot Platformu, 2014-2017, *Funded by TUBITAK*.
- **Project Coordinator**, Tarım Alanlarına Yönelik Gözlem, Analiz, Çapalama Ve İlaçlama Gerçekleştirebilen Otonom Hareketli Robot Platformu, 2017-2019, *Funded by TUBITAK*.
- **Researcher**, Esnek Otomotiv Üretim Hatları İçin Değişken Üretime Uyumlu Akıllı Sanal Konveyör Sistemi, 2018-Present, *Funded by TUBITAK*.
- **Researcher**, Konum Bazlı Ve Seçilebilir Otonomi Seviyesine Sahip Tarım Aracı Yönlendirme Ve Kontrol Sistemi, 2019-Present, *Funded by TUBITAK*.

PUBLICATIONS

- Hacinecipoglu, A. 2012. "Development of Electrical and Control System of an Unmanned Ground Vehicle for Force Feedback Teleoperation", M.Sc. Thesis, METU

- Hacinecipoglu, A., Konukseven, E. I., Koku, A. B. 2013. "Evaluation of haptic feedback cues on vehicle teleoperation performance in an obstacle avoidance scenario", World Haptics Conference (WHC), 689-694.

- Hacinecipoglu, A., Koku B., Konukseven I. 2014. "Üç Boyutlu Haritalama ve Konumlama Uygulamaları için Donanım ve Yazılım Platformu Geliştirilmesi", Türk Otomatik Kontrol Konferansı, TOK14, 573-577.

- Ozutemiz, K. B., Hacinecipoglu, A., Koku, A. B., Konukseven, E. I. 2013-1. "Adaptive unstructured road detection using close range stereo vision", Asian Control Conference (ASCC), 1-6.

- Ozutemiz, K. B., Hacinecipoglu, A., Koku, A. B., Konukseven, E. I. 2013-2. “Self-learning road detection with stereo vision”, Signal Processing and Communications Applications Conference (SIU), 1-4).

- Ozutemiz K.B., Hacınecipoglu, A., Koku, A.B., Konukseven, E.İ. 2013-3. “A Comparative Study on Distance Metrics in Self-Supervised Unstructured Road Detection Domain”, Mechatronics and Machine Vision in Practice, M2VIP 2013.

- Hacinecipoglu, A., Demir, S. Ö., Ugurlu, M. Ç., Koku, A. B., Konukseven, E. İ. 2014. "Manuel Sürülen Platform Üzerinde SLAM Uygulaması", Türkiye Otonom Robotlar Konferansı, TORK 2014.

- Çakıcı Özkök, Ç., Lafcı, A., Şirin, H. O., Gürlek, T. T., Hacinecipoglu, A., Özden, B. Ş. 2018. Tarım Araçlarında Otomatik Dümenleme, Türkiye Robot Bilim Konferansı, TORK 2018.

- Hacinecipoglu, A., Konukseven, E. I., Koku, A. B. 2020. "Pose Invariant People Detection in Point Clouds for Mobile Robots", accepted to International Journal of Mechanical Engineering and Robotics Research, vol. 9, no. 4.

- Hacinecipoglu, A., Konukseven, E. I., Koku, A. B. 2019. "Fast People Detection in Arbitrary Poses using Depth Information", submitted to Sensor Review.
- Hacinecipoglu, A., Konukseven, E. I., Koku, A. B. 2019. "Multiple Human Trajectory Prediction and Cooperative Navigation Modeling in Crowded Scenes", submitted to Intelligent Service Robotics.