Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi

UNIVERSITÀ DEGLI STUDI DI GENOVA

An Approach for Multi-Robot Opportunistic Coexistence in Shared Space

by

Muhammad Hassan Tanveer

Thesis Series
DIBRIS-TH-2018-XXXI

DIBRIS, Università di Genova
Via Opera Pia, 13, 16145 Genova, Italy http://www.dibris.unige.it/

Università degli Studi di Genova

Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi

Ph.D Thesis in Curriculum of Robotics and Autonomous Systems

An Approach for Multi-Robot Opportunistic Coexistence in Shared Space

by

Muhammad Hassan Tanveer

April 2019

Dottorato di Ricerca in Robotica ed Ingegneria dei Sistemi
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università degli Studi di Genova

DIBRIS, Univ. di Genova
Via Opera Pia, 13
16145 Genova, Italy
`http://www.dibris.unige.it/`

Ph.D. Thesis in Curriculum of
Robotics and Autonomous Systems
Submitted by Muhammad Hassan Tanveer
DIBRIS, Univ. di Genova
. . . .

Date of submission: April 2019

Title: An Approach for Multi-Robot Opportunistic Coexistence in Shared Space

Advisor: Dr. Antonio Sgorbissa
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università di Genova
. . .

Abstract

This thesis considers a situation in which multiple robots operate in the same environment towards the achievement of different tasks. In this situation, please consider that not only the tasks, but also the robots themselves are likely be heterogeneous, i.e., different from each other in their morphology, dynamics, sensors, capabilities, etc. As an example, think about a "smart hotel": small wheeled robots are likely to be devoted to cleaning floors, whereas a humanoid robot may be devoted to social interaction, e.g., welcoming guests and providing relevant information to them upon request.

Under these conditions, robots are required not only to co-exist, but also to coordinate their activity if we want them to exhibit a coherent and effective behavior: this may range from mutual avoidance to avoid collisions, to a more explicit coordinated behavior, e.g., task assignment or cooperative localization.

The issues above have been deeply investigated in the Literature. Among the topics that may play a crucial role to design a successful system, this thesis focuses on the following ones:

(i) An integrated approach for path following and obstacle avoidance is applied to unicycle type robots, by extending an existing algorithm [1] initially developed for the single robot case to the multi-robot domain. The approach is based on the definition of the path to be followed as a curve $f(x,y)$ in space, while obstacles are modeled as Gaussian functions that modify the original function, generating a resulting safe path. The attractiveness of this methodology which makes it look very simple, is that it neither requires the computation of a projection of the robot position on the path, nor does it need to consider a moving virtual target to be tracked. The performance of the proposed approach is analyzed by means of a series of experiments performed in dynamic environments with unicycle-type robots by integrating and determining the position of robot using odometry and in Motion capturing environment.

(ii) We investigate the problem of multi-robot cooperative localization in dynamic environments. Specifically, we propose an approach where wheeled robots are localized using the monocular camera embedded in the head of a Pepper humanoid robot, to the end of minimizing deviations from their paths and avoiding each other during navigation tasks. Indeed, position estimation requires obtaining a linear relationship between points in the image and points in the world frame: to this end, an Inverse Perspective mapping (IPM) approach has been adopted to transform the acquired image into a bird eye view of the environment. The scenario is made more complex by the fact that Pepper's head is moving dynamically while tracking the wheeled robots, which requires to consider a different IPM transformation matrix whenever the attitude (Pitch and Yaw) of the camera changes. Finally, the IPM position estimate returned by Pepper is merged with the estimate returned by the odometry of the wheeled robots through an Extened Kalman Filter. Experiments are shown with multiple robots moving along different paths in a shared space, by avoiding each other without onboard sensors, i.e., by relying only on mutual positioning information.

Software for implementing the theoretical models described above have been developed in ROS, and validated by performing real experiments with two types of robots, namely: (i) a unicycle wheeled Roomba robot (commercially available all over the world), (ii) Pepper Humanoid robot (commercially available in Japan and B2B model in Europe).

Acknowledgements

I am thankful to Almighty Allah who gave me this opportunity and strength to complete this research to the best of my efforts. There were stranded moments but there were divine help too. I would like to thank my research advisor Professor Antonio Sgorbissa for his continuous guidance and encouragement throughout my PhD tenure.

I am also very grateful to my whole family, specially my wife who believed in my efforts, motivating in all difficult times storing my hurts and worries with their inestimable prayers.

Last but not least, to Dr. Carmine T. Recchiuto for his precious time and in depth discussions that helped me to understand things and for his kind and generous support in all phases of the research.

Contents

Acknowledgements **iv**

Contents **v**

List of Figures **vii**

List of Tables **x**

1 Introduction **1**

1.1 Path following and obstacle avoidance 2

1.1.1 Objectives 2

1.1.2 Motivation 2

1.1.3 Thesis Contribution 3

1.2 Multi-robot cooperation for localization 4

1.2.1 Objectives 5

1.2.2 Motivation 5

1.2.3 Thesis contribution 7

1.3 Outline 9

1.4 List of Publications 10

2 Related Works **11**

2.1 Path following and obstacle avoidance 11

2.1.1 Model Predictive control (MPC) 12

2.1.2 Velocity Obstacle (VO) 12

2.1.3 Artificial Potential Field (APF) 13

2.1.4 The edge detection approach 13

2.1.5 The Vector Field Histogram (VFH) 14

2.1.6 Virtual Force Field algorithm 14

2.1.7 Wall following algorithms 15

2.1.8 Dynamic Window Approach (DWA) 15

2.2 Multi-robot cooperation 16

2.2.1 Multi-robot Systems 17

2.2.2 Challenges of multi-robot cooperation 18

2.2.2.1 Relative Poses of Robots 18

2.2.2.2 Uncertainty of the Relative Poses 18

2.2.2.3 Updating Maps and Poses 19
2.2.2.4 Line-of-sight Observations 19
2.2.2.5 Loop closure 19
2.2.2.6 Complexity 20
2.2.2.7 Communications 20
2.2.2.8 Heterogeneous Vehicles and Sensors 20
2.2.2.9 Synchronization 21
2.2.2.10 Performance Measure 21
2.2.3 Perception techniques for Multi-robot cooperation 21
2.2.3.1 Monocular vision 21
2.2.3.2 Stereo vision 22
2.2.3.3 Light detection and ranging (LIDAR) 23
2.2.3.4 Radar 23

3 Path following and obstacle avoidance 25
3.1 Materials and Methods 25
3.1.1 Path Following 25
3.1.2 Stability Analysis 28
3.1.3 Obstacle Avoidance 29
3.1.4 Shaping the obstacle function to avoid collisions 31
3.2 Experimental Setup 37
3.2.1 Results and Discussion 38
3.2.2 Scenario 1: Without Obstacles 38
3.2.3 Scenario 2: With Static Obstacle 40
3.2.4 Scenario 3: With moving Obstacle 44
3.2.4.1 Sine wave Path 49
3.2.4.2 Straight Line Path 52
3.2.5 Scenario 4: With Persons 56

4 Multi-robot cooperation 61
4.1 Materials and Methods 61
4.1.1 Problem Definition 61
4.1.2 Inverse Perspective Mapping (IPM) 63
4.1.3 Detection Phase 66
4.2 Results and Discussion 67
4.2.1 Static Pepper Head with moving robot in plane 68
4.2.2 Moving Pepper Head with moving robot(s) in plane 74
4.2.2.1 Tests with wheeled robots 78

5 Conclusion and Future work 84

Bibliography 89

List of Figures

1.1 Pepper tracking Roomba in plane 7

1.2 Illustration for Pepper robot tracking Roomba's in the plane. The green and yellow color blobs are used to pronounce the Roomba's unique. . . 8

3.1 Path through surface intersection 26

3.2 Deformation of the path due to the presence of an obstacle $\mathcal{O}_j$ with $A_j > 0$. 30

3.3 Deformation of the path due to the presence of multiple obstacles $\mathcal{O}_j$ with $A_j < 0$. 33

3.4 Structure of the system 35

3.5 Architecture of the system 38

3.6 Robot response without obstacle $u_1(t) = 0.3, K_1 = 15, K_2 = 2$ 39

3.7 Plot of e_1 versus time in absence of obstacles 39

3.8 Robot response with one static obstacle $u(t) = 0.3, K_1 = 15, K_2 = 2, A_j = 0.8, \sigma = 0.5$ 40

3.9 Plot of e_1 versus time with one static obstacle 41

3.10 Robot response with two static obstacle placed together $u(t) = 0.3, K_1 = 15, K_2 = 2, A_j = 0.8, \sigma = 0.5$ 42

3.11 Plot of e_1 versus time with two static obstacle placed together 42

3.12 Robot response with 2 obstacles placed at different position $u(t) = 0.3, K_1 = 15, K_2 = 2, A_j = 0.8, \sigma = 0.5$ 43

3.13 Plot of e_1 versus time with two static obstacle placed apart 43

3.14 Two Robots with same speed recognizing each other as an obstacle $u(t) = 0.3, K_1 = 15, K_2 = 2, A_j = 0.5, \sigma = 0.45$ 44

3.15 Plot of e_1 and e_2 versus time 44

3.16 Plot of $dist_{1,2}$ versus time 45

3.17 2 Robots moving with different speed recognizing each other as an obstacle Robot 1: $u_1(t) = 0.6, K_1 = 10, K_2 = 1, A_j = 0.4, \sigma = 0.55$ Robot 2: $u_2(t) = 0.5, K_1 = 10, K_2 = 1.2, A_j = 0.4, \sigma = 0.55$ 45

3.18 Plot of e_1 and e_2 versus time 46

3.19 Plot of $dist_{1,2}$ versus time 46

3.20 Three Robots with different speed and different σ recognizing each other as an obstacle Robot 1: $u_1(t) = 0.6, K_1 = 10, K_2 = 1, A_j = 0.45, \sigma = 0.6$ Robot 2: $u_2(t)$=0.5, K_1=10, K_2=1.2 , A_j=0.4, σ=0.55 Robot 3: $u_3(t) = 0.3, K_1 = 15, K_2 = 2, A_j = 0.6, \sigma = 0.65$ 47

3.21 Plot of e_1, e_2 and e_3 versus time 48

3.22 Plot of $dist_{1,2,3}$ versus time 48

3.23 Two robots moving along two intersecting sine wave paths. Robot 1: $u_1(t) = 0.3, K_1 = 35, K_2 = 5, A_j = 0.5, \sigma = 0.5$ Robot 2: $u_2(t) = 0.2, K_1 = 40, K_2 = 7, A_j = 0.5, \sigma = 0.5$ 50
3.24 Plot of $e_{1,2}$ versus time 50
3.25 Plot of $dist_{1,2}$ versus time 50
3.26 Two robots moving along the same sine wave path, but in opposite directions. Robot 1: $u_1(t) = 0.1, K_1 = 45, K_2 = 10, A_j = 0.5, \sigma = 0.5$ Robot 2: $u_2(t) = 0.1, K_1 = 45, K_2 = 10, A_j = 0.5, \sigma = 0.5$ 51
3.27 Plot of $e_{1,2}$ versus time 51
3.28 Plot of $dist_{1,2}$ versus time 52
3.29 Two robots moving along two intersecting straight paths. Robot 1: $u_1(t) = 0.1, K_1 = 25, K_2 = 6, A_j = 0.5, \sigma = 0.5$ Robot 2: $u_2(t) = 0.1, K_1 = 25, K_2 = 6, A_j = 0.5, \sigma = 0.5$ 53
3.30 Plot of $e_{1,2}$ versus time 53
3.31 Plot of $dist_{1,2}$ versus time 53
3.32 Two robots moving along the same straight path, but in opposite directions. Robot 1: $u_1(t) = 0.1, K_1 = 25, K_2 = 6, A_j = 0.5, \sigma = 0.5$ Robot 2: $u_2(t) = 0.1, K_1 = 25, K_2 = 6, A_j = 0.5, \sigma = 0.5$ 54
3.33 Plot of $e_{1,2}$ versus time 54
3.34 Plot of $dist_{1,2}$ versus time 55
3.35 Two robots moving along the same straight path, but in opposite directions. Robot 1: $u_1(t) = 0.3, K_1 = 35, K_2 = 5, A_j = 0.5, \sigma = 0.5$ Robot 2: $u_2(t) = 0.2, K_1 = 40, K_2 = 7, A_j = 0.5, \sigma = 0.5$ 55
3.36 Plot of $e_{1,2}$ versus time 55
3.37 Plot of $dist_{1,2}$ versus time 55
3.38 Robot with Persons 56
3.39 1 Robot with 1 Person 57
3.40 Plot of $dist_{1,4}$ versus time 57
3.41 Plot of e_1 versus time 57
3.42 1 Robot with 2 Persons 58
3.43 2 Robots with 1 Person 59
3.44 3 Robots with 1 Person 59

4.1 Image coordinate system in relation to world coordinate system. (X_c, Y_c, Z_c) and (X_w, Y_w, Z_w) denotes the camera and world frame respectively, (u, v) is the image plane, h is the height of the Pepper's camera from the ground and θ is the mapping angle of a point on the world frame to its projection on the image plane. 63
4.2 An overview of object position detection during Pepper's head movement. When an object is not detected in the IPM frame, the NO branch is followed, requiring to track the object; when the object is detected, the YES branck is followed, finally leading to world frame coordinates computation. 64
4.3 Pepper RedBall API response at static points 69
4.4 Pepper Redball API response while tracking a circular response 70

4.5 Pepper camera view and transformed IPM view 70
4.6 Distant object as shown by Pepper Top-Down view, Top Down view, distance in this image may be measureable 71
4.7 IPM view pixel values in different distances 72
4.8 Comparison of IPM view and Redball API at static point 72
4.9 Distant and near moving robot in Pepper camera view, IPM view and Bianary view 73
4.10 Fixed object IPM view with different Pepper Head Orientation (i.e: (yaw,pitch) = (36,0),(36,-5),(36,-15),(36,20)), left and right column shows the Pepper camera view the IPM views respectively. 75
4.11 Estimated positions in the world frame for 36 different locations. 76
4.12 Position estimation of an object by varying Pitch and Yaw of the Pepper's head. 77
4.13 One frame showing Pepper's camera view and robot detection in IPM view. 78
4.14 Multiple frames showing a robot moving along a circular path in Pepper's camera view and IPM view. 78
4.15 Plot of the robot's trajectory and error with odometry only. 79
4.16 Plot of the robot's trajectory and error by merging odometry and IPM. . 80
4.17 One frame showing Pepper's camera view and multi-robot detection in IPM view. 81
4.19 Plot of the trajectory of two Robots moving along a circular path with different speed, plot of their distance, and the corresponding positioning errors. Robot 1: $u_1(t) = 0.1m/s, K_1 = 20, K_2 = 4, A_j = 0.5, \sigma = 0.5$. Robot 2: $u_2(t) = 0.2m/s, K_1 = 18, K_2 = 3.5, A_j = 0.5, \sigma = 0.5$. 82
4.18 Multiple frames showing two robots moving along circular paths in Pepper's camera view and IPM view. 82
4.20 Plot of the trajectory of two Robots moving along a circular path with the same speed, plot of their mutual distance, and the corresponding positioning errors. Robots: $u(t) = 0.1m/s, K_1 = 20, K_2 = 4, A_j = 0.5, \sigma = 0.5$. 83

List of Tables

3.1 Response of Robot with different velocities and without obstacle 40
3.2 Response of Robot with one fixed obstacle 41
3.3 Response of Robot with two fixed obstacle placed together 42
3.4 Response of Robot with two fixed obstacle placed apart 43
3.5 Response of Two Robot with same speed and varying σ 45
3.6 Response of Two Robot with different speed 46
3.7 Response of Three Robot with speed = 0.3 and varying σ, $K_1 = 15, K_2 = 2$ 47
3.8 Response of Three Robot with speed $Robot_{1,2,3} = 0.3, 0.4, 0.5$, $A_j = 0.4$ and $\sigma = 0.55$ 47
3.9 Response of Two Robot back and forth with same/different speed following sine wave path intersecting each other 51
3.10 Response of Two Robot back and forth with same/different speed following sine wave path parallel to each other in opposite direction 52
3.11 Response of Two Robot back and forth with same/different speed following straight line path intersecting each other 54
3.12 Response of Two Robot back and forth with same/different speed following straight line path parallel to each other 56
3.13 One Robot with speed = $0.3m/s$, $K_1 = 15$, $K_2 = 2$, $A_j = 0.4$, $\sigma = 0.45$. 58
3.14 Two Robots with speed = $0.3m/s$, $K_1 = 15$, $K_2 = 2$, $A_j = 0.4$, $\sigma = 0.45$ 59
3.15 Three Robots with speed = $0.3m/s$, $K_1 = 15$, $K_2 = 2$, $A_j = 0.4$, $\sigma = 0.45$ 60

4.1 Summary: $av|e|andstd|e|$, circular path. 80
4.2 Summary: $av|e|andstd|e|$, to and fro path. 81
4.3 Summary: Response of Two Robots by varying speed and $A_j = 0.4$, $\sigma = 0.55$. 82

Chapter 1

Introduction

Twenty years ago the idea of having robots moving around like humans or performing tasks in human environments was only a futuristic thought. Perhaps the time when fully autonomous cooperative robots coexist with humans is still a distant reality; yet what is becoming true is the human acceptance of such a coexistence. The cooperation between heterogeneous team of robots is one of the principal reasons for the great attention that these autonomous agents are receiving from both the academic and the industrial community. Indeed robots are endowed with a number of peculiarities that makes them extremely interesting in many scenarios: ease of construction, low cost, flexibility, possibility to carry a payload (usually cameras or other sensors). The contexts of their applications include agriculture and vegetation mapping, monitoring of wetland ecosystems and coastal management, traffic monitoring, recording of sport events and damage assessment after a disaster (earthquakes, floods, avalanches, fires).

Cooperative approaches pose several challenges. Among the most important ones are those directly related to the safety issues: path following, obstacle avoidance and knowing the position of each others for localization. The control of mobile robots have been the subject of much research in recent years, due to robots being frequently used in dangerous or inaccessible environments where human beings can hardly intervene.

This chapter elucidates the motivation for our research followed by the contribution. The research is divided into 2 phases: i) Path following and obstacle avoidance (Section 1.1). ii) Multi-robot cooperation for localization (Section 1.2).

1.1 Path following and obstacle avoidance

In this section we briefly discuss the objectives, followed by the motivation and the contributions of our path following and obstacle avoidance approach.

1.1.1 Objectives

- To identify relevant errors in odometry and overcome it while the robot operates for longer periods of time.
- To investigate and analyze path following and dynamic obstacle avoidance with Motion Capture systems.
- To compare and evaluate the proposed techniques with conventional techniques.

1.1.2 Motivation

Path following is the process of generating a sequence of trajectories that enables the robot to perform the assigned tasks. State-of-the-art methods successfully perform path following in real world scenarios. For example, deployment of wheeled robots (WRs) in disaster missions have already been proven successful. Few of these methods include, the utilization of WRs for the surveying of the damaged area [2], efficient deployment in the context of planetary exploration [3], and more recently in vineyard navigation [4] and climbing steel structures [5]. In all these situations, WRs assessment was an effective method for rescue teams, and disaster regions were efficiently and accurately identified, helping human operators to find survivors and target locations. In the context of disaster scenarios, other robots like multirotors are required to fly in complex environments, in which buildings are likely to be completely or partially collapsed [6, 7]. Therefore, pre-existing maps are not very useful: debris, damaged furniture and various objects are likely to occupy pre-existing free space, while new free spaces can be produced as well [8].

Most often, obstacle avoidance strategy is required to circumvent this issue. So, an appropriate strategy for obstacle avoidance is a key factor for achieving safe navigation [9–11]. Many research works have focused on developing safe navigation algorithms in static environments [12–16]. However, in real world scenarios, assuming that the environment is static can be catastrophic, and the environment should more realistically be modelled as dynamic: therefore, in the field of mobile robotics, the problem of safe navigation in dynamic environments is one of the most important challenges to be addressed [17, 18]. It becomes more complex and tough when the information about dynamic obstacles and the environment is not available, even tough such information is often assumed to be available in many of the developed approaches [19–23]. This information may include the complete map of the environment, the position and the orientation of the obstacles in the map, the nature of the obstacles (whether the shape of the obstacles is constant or varies over time) and the motion of the obstacles (whether the obstacle is moving with a constant or time-varying velocity) etc.

1.1.3 Thesis Contribution

The approach described in this thesis is based on a previously proposed framework for path-following [1], which introduces the idea of representing a path in 2D through the implicit equation of a curve in the plane $f(x,y) = 0$, and includes the definition of a feedback controller that takes the equation of the curve and the robot's pose to compute the path-following error. In [1] it has been formally demonstrated that the approach guarantees asymptotic convergence to the path (the approach has been extended, among the others, to N-trailers [24] and UAVs [25]).

The main contribution of this thesis is to consider – for the first time in this framework – multiple dynamic obstacles during path following (e.g., other robots or persons). The thesis describes a formal procedure to model obstacles in such a way that collisions are guaranteed to be avoided, by producing a deformed path $f'(x,y) = 0$ that roughly follows the original one while avoiding all obstacles (the convergence to the deformed path being guaranteed, once again, by the feedback control law adopted [1]). The performance of the proposed approach in terms of closeness to the original path and farness from obstacles has been validated by quantitatively measuring its performance in a $3m \times 3m$ meters arena crowded with robots and persons. Even if the approach may partially resemble to Artificial Potential Fields or similar force field-based methods for navigation [26, 27], the presented approach has many peculiarities: among the others, it allows for setting the control variables (linear and angular speed) as a unique continuous function of the deformed path represented as a curve $f'(x,y) = 0$, the robot's pose x, y, θ, and the relative position x_j, y_j of all the locally-sensed obstacles with respect to the robot. The differences between the present approach and force-field methods are better discussed in Chapter 2.

The approach is applicable to path following and obstacle avoidance of unicycle–like mobile systems with bounded speed and angular velocity. It is well-known that the motion of many wheeled robots and unmanned aerial vehicles can be described by this model; see [28] and the references therein. A similar approach for modelling a path in presence of obstacles has been proposed in [29] for controlling a multicopter: however, the latter approach is different from the present article since it is not based on the feedback controller proposed in [1] and it has been only tested with a small number of static obstacles.

The merits and demerits of previous work in the literature have been considered and a method allowing mobile robots to follow a predetermined path while avoiding fixed and moving obstacles has been used [1] and expanded by quantitatively measuring its performance in $3m \times 3m$ arena crowded with robots and humans. The method allows for representing a path in 2D through the implicit equation of a curve in the plane $f(x,y) = 0$, and includes the definition of a feedback controller that takes the equation of the curve and the robot's pose to compute the path-following error.

In addition to this algorithm, obstacle avoidance can then be achieved by adding a positive (or negative) contribution to the curve equation for each detected obstacle, thus producing a deformed path that is guaranteed to not collide with any of the obstacles. The algorithm is applicable for path following and obstacle avoidance of nonholonomic unicycle–like mobile systems with bounded speed and angular velocity. It is well-known that the motion of many wheeled robots and unmanned aerial vehicles can be described by this model; see [28] and the references therein.

1.2 Multi-robot cooperation for localization

The problem of cooperating multi-robots has received considerable attention in the robotics literature. Whenever several robots are deployed in the same environment there is the need for coordinating their movements and knowing the position of each robot with respect to the other. Trajectories for the individual robots have to be computed such that collisions between the robots and static obstacles as well as those between the robots are avoided. In this section we briefly discuss the objectives, followed by the motivation and the contributions of our multi-robot cooperative localization approach.

1.2.1 Objectives

- To propose a cooperative approach for localization between heterogeneous team of robots.
- To analyze the robust IPM method in order to estimate the position of other robots in plane.
- To validate the performance and effectiveness of the proposed IPM method with and without EKF.
- To assess the computational complexity of the proposed algorithm.
- To propose a novel descriptor that accurately measure the position and reduce positioning errors for robot cooperation.

1.2.2 Motivation

One of the core elements in the field of robotics is to implement the cooperation between robots and thereby facilitating efficient coordination in shared space. Such an efficient coordination requires each robot to know the pose of other robots in the environment. In this regard, the cooperation of robots have been explored extensively, resulting in a number of approaches that includes localization [30].

Localization is a fundamental problem in robotics, whose aim is to determine the pose of the robot with respect to an environment map. Achieving high-accuracy localization is an indispensable requirement for autonomy in a variety of scenarios including mapping, target tracking, autonomous driving, surveillance and smart-homes [31, 32]. Yet, these real world scenarios presents different challenges. For example, localization approaches vary according to the initial knowledge of the robot pose (known or unknown), environment (static or dynamic) and control (passive or active). Most localization approaches have been developed in the context of single robot scenarios. However, many real world applications require different robots working in tandem and the past two decades has witnessed an increasing number of research in the area of multi-robot cooperative localization [33].

The problem of pose estimation is related to obstacle avoidance, since accurate localization can help predicting and avoiding collisions when mobile robots are not equipped with onboard sensors, or have limited sensing range [34]. Collision detection systems provide robots with an alert prior to a collision that allows robot to take preventive actions [35]. In [36], the robot are included in the security system, which immediately stops the robot processes if an imminent object is detected in the surroundings nearby. In [37] it is shown that, determining the mutual position of robots could also allow the robots to provide alerts and to make cautious comments to the user. [38] introduces instructions that may be used to prevent or create a security setting for sudden collision. In [39], pose correction activities have been carried out using optical cameras that contributes to intervention in comparison with sensors.

Hence, localization and pose estimation are challenging problems that become even more challenging in a multi-robot setting that has gained a broad attention over the recent years among the robotics community [40]. In comparison to a single robot system, multiple robots working in a team can be more robust to individual failures and more efficient in time to accomplish a job [41, 42]. Multi-robot systems have wide potential applications including large area surveillance, search and rescue operation, perimeter protection, and intruder detection [43–45]. For a large number of robots working together to achieve a common goal, an efficient controller is required to coordinate the motion of robots. Furthermore, for such controllers to efficiently exercise control in real world, the knowledge of the state of the robot itself and its nearby robots is usually desirable. Typically, the state includes the pose of the robot (the pose consists of the position and bearing). This thesis will address multi-robot cooperation in shared space.

The motivation of this work can be elaborated by considering a scenario in which Roomba robots coexist with a humanoid Pepper robot, e.g., a hotel or a commercial centre. Roomba robots do not posses complex sensing capabilities, but they have the unique capability of cleaning the floor. On the other hand, Pepper is currently employed for verbal interaction with people, e.g., welcoming them and giving information. By taking into account their benefits, we exploit the different capabilities of Pepper and Roomba. This thesis will elucidate how Pepper can provide useful information for localization and navigation, that can be used by Roomba robots to improve their behaviour.

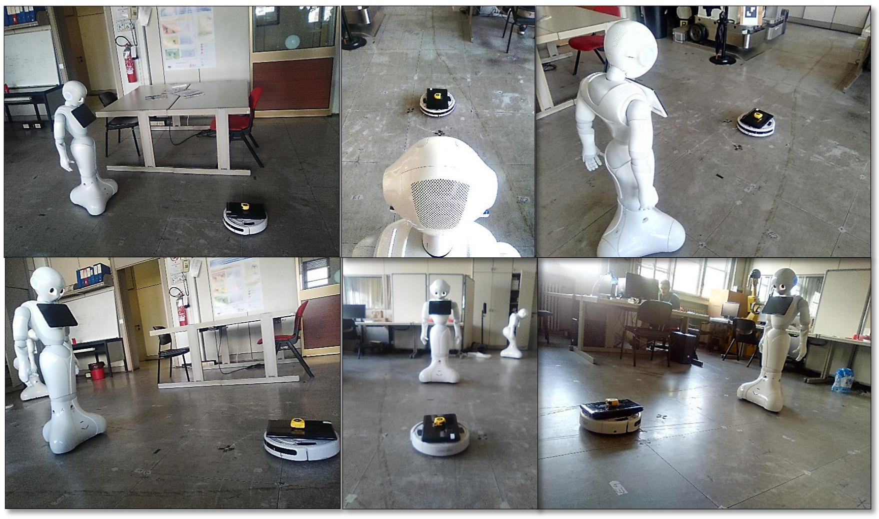

FIGURE 1.1: Pepper tracking Roomba in plane

1.2.3 Thesis contribution

In this phase of the thesis, two different robots have been considered, the Pepper Humanoid robot and iRobot Roomba. The Pepper robot has capabilities for vision which include facial and shape recognition (using inbuilt cameras). Secondly, the iRobot Roomba is a unicycle mobile robot that performs a wide range of task from autonomous cleaning, surveillance, to transportation and assistance to the elderly. By combining both of these robots, a solution that consists in the composition of IPM view (bird's eye view) of Pepper head camera has been proposed. This view will serve as a virtual map for the iRobot Roomba in the Pepper Frame of view (FOV) to help the user to adequately specify the commands to be sent to the other Roomba under its control, as in Figure 1.2 that is referred to Pepper Frame of view.

In particular, we are insterested in detecting the relative pose of a ground robot (i.e., Roomba) with respect to Pepper humanoid robot. The camera is mounted on Pepper head resulting in increased vertical distance between the camera plane and the object plane. The Pepper robot comes with few built-in APIs [46] that enables the detection of Roomba and thereby its pose. Yet, in our considered scenario, these APIs produce erroneous results. The Nao Mark API can only detect objects that are within 100 cm from Pepper's head and the markers further away become blurred. The Red Ball API suffers due to the height difference in the planes (as discussed above) resulting in a transient response, making it hard to determine the pose with any degree of certainity [47]. Furthermore, existing pose calculation approaches use forward facing cameras, sonar or laser scanners. However, these approaches fail in scenarios with perspective difference between the camera and the object in plane. Ma et al. [48] proposed a multi-camera setup, providing depth information by establishing feature correspondence and triangulation. However, it requires severe processing and configuration overheads, which are generally expensive.

To circumvent the issues of the perspective difference in the camera and objects in plane, we propose an Inverse Perspective Mapping (IPM) view of the Pepper head camera frame. This serves as a virtual map for the Roomba in the Pepper camera view, enabling the Pepper to localize all robots in the same frame and to adequately specify the commands to be sent to the other Roombas under its control. This scenario is illustarted in Figure1.2 that is referred to Pepper Frame of view. The proposed system consists of a single forward facing Pepper camera, capturing video images at 30 frames per second (fps).

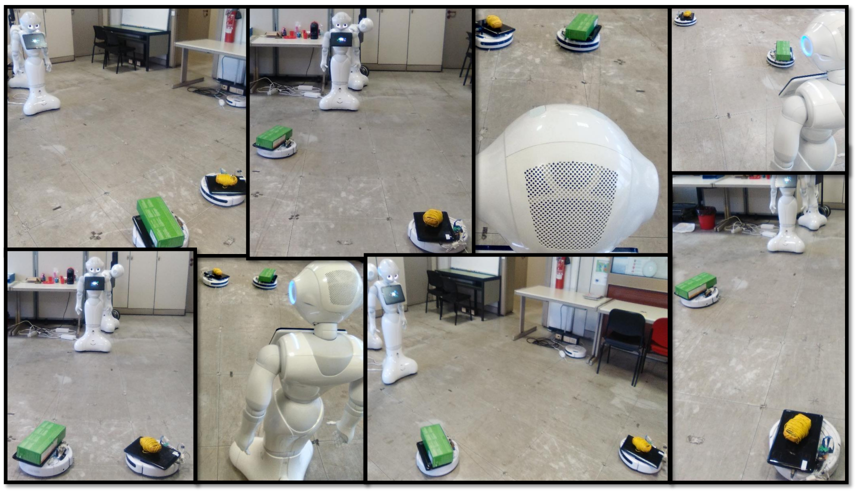

FIGURE 1.2: Illustration for Pepper robot tracking Roomba's in the plane. The green and yellow color blobs are used to pronounce the Roomba's unique.

1.3 Outline

This thesis describes in two phases a detailed implementation of generalized approach for position estimation of different robots in the plane. It is intended to serve as a base for the development of tools for multi-robot cooperation in field of robotics by using IPM information. To this end, this thesis is organized in logical and sequential chapters as follows:

- Chapter 2 introduces the state of the art. Phase 1: by summarizing different conventional technologies from which few are commercially available for the purpose of Path following, obstacle avoidance. Phase 2: we have discussed the different approaches that exist in literature for localization problem in multi-robot cooperation.
- Chapter 3 starts by defining the general flow of the kinematics of WRs. Then it presents the Path following and obstacle avoidance funtion with one or multiple moving obstacles, including experimetal results.
- Chapter 4 explains the cooperation between multi-robots using IPM technique. A detailed description of the software architecture and the hardware frameworks are also provided. Finally, it elucidates the performance of the algorithm as well as the results achieved after implementing our novel cooperation architecture for localization. Both static and dynamic obstacles are considered for empirical evaluation.
- Chapter 5 concludes the thesis, providing suggestions for further improvements.

1.4 List of Publications

This section list the publications based on this PhD research:

- Tanveer, M., Recchiuto, C., and Sgorbissa, A. (n.d.). Analysis of path following and obstacle avoidance for multiple wheeled robots in a shared workspace. Robotica, 1-29. doi:10.1017/S0263574718000875.
- Tanveer M.H, and Sgorbissa A., Generalized approach for multi-robot cooperation in shared space. Publication in a book of the " Lectures Notes in Electrical Engineering " (LNEE) series (https://www.springer.com/series/7818) (status: accepted).
- Tanveer M. and Sgorbissa A. (2018). An Inverse Perspective Mapping Approach using Monocular Camera of Pepper Humanoid Robot to Determine the Position of Other Moving Robot in Plane.In Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO, ISBN 978-989-758-321-6, pages 219-225. DOI: 10.5220/0006930002190225.
- M. H. Tanveer, A. Sgorbissa and C. T. Recchiuto, "Collision-free navigation of multiple unicycle mobile robots," 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN) Lisbon, 2017, pp.1365-1372. doi: 10.1109/ROMAN.2017.8172482.
- Tanveer, M. H., Sgorbissa, A., Recchiuto, C., Coordinated behaviour with a Pepper Humanoid robot to estimate the distance of other robot using Inverse Perspective Mapping, ICAROB 2018- IEEE International Conference on Automation and Robotics.

Chapter 2

Related Works

In this chapter we discuss the recent works carried out in the field of: i) path following and obstacle avoidance for autonomous control of mobile robots, ii) multi-robot cooperation for localization.

2.1 Path following and obstacle avoidance

The problem of collision avoidance integrated with path following, in presence of both static and moving obstacles, has been widely studied and solutions have been put forward [49–54]. While all the proposed solutions are able to generate a safe path (at least in presence of static obstacles), in order to evaluate the efficiency of a obstacle avoidance algorithm some additional aspects should be considered, e.g. computational cost, presence of limited or noisy information, necessity to estimate the position or velocity of moving obstacles and other robots, capability to find a path to the goal under any condition. In the following, some techniques in the literature for path following and obstacle avoidance are presented and compared with the approach proposed in this research, taking into account their strengthness and weaknesses:

2.1.1 Model Predictive control (MPC)

MPC is one of the most common solutions for obstacle avoidance [55]. Indeed, it has been applied to control various systems, including industrial systems [56–63]. Moreover, it has also been used to generate safe trajectories for robots by using simplified dynamics in an unknown environment. An example of its application is the work of [64], in which MPC is applied for online dynamic obstacle avoidance along with streamlined navigation towards the destination. In this framework, the controller predicts a future path and perform optimization to ensure collision-free trajectories. Variants of MPC have been proposed to allow mobile robots equipped with onboard sensors to avoid moving obstacles [65, 66]. However, MPC is applicable for vehicles with simple linear models, while most vehicles exhibit more complicated non-holonomic characteristics with constraints on the linear and angular velocities. For this kind of vehicles, non-linear MPC is a more suitable control approach [67]. In spite of its popularity, MPC requires prior knowledge of the robot model which increases the mathematical complexity: thus, the main drawback of this family of approaches is a significant computational burden associated with solving a set of nonlinear differential equations and a nonlinear dynamic optimization problem. On the contrary, the techniques proposed in this thesis requires very few computational resources, as a consequence of its simplicity.

2.1.2 Velocity Obstacle (VO)

This technique was first proposed in [68]. With some modifications, it is still extensively used in research related to different domains [18, 69, 70]. For motion planning, VO requires the set of all velocities of the robots and obstacles, assuming that the velocities to be constant. If a moving obstacle changes its velocity, then it could result in a collision, unless the path is not re-computed in real-time. The main disadvantage of this class of methods is that they take into account the obstacle velocities and that the robot behaviour does not change if the velocities of moving obstacles or other robots change (unless the path is periodically recomputed, which may be computationally expensive). Therefore, it is not well suited for highly dynamic scenarios. Also, estimating the velocities of moving objects and other robots using onboard sensors may be technically challenging. Our approach does not consider the velocity but only the position of other obstacles and robots (which is much easier to be estimated using onboard sensors), periodically recomputing the positions at high frequency. For this reason our approach may require obstacles and robots to move at a lower average velocity and is much less sensitive to sudden change in their velocity.

2.1.3 Artificial Potential Field (APF)

APF and its variants is still one of the most widespread techniques for obstacle avoidance [50, 71–74]. In APF, the robot is considered as a moving point in a potential field, where the goal generates an attractive force and the obstacles produce repulsive forces: the method is easy to implement, and it can be applied straightforwardly to avoid moving obstacles, by knowing only their position relatively to the robot. However, it is a well-know drawback of APF that the robot may be trapped in a local minimum, thus preventing it to find a path to the goal. With the approach proposed in this thesis, it is guaranteed that by appropriately tuning the distance of influence of obstacles, the robot will be never trapped in local minima and a path toward the goal will be always found. This property of the algorithm is formally proven in [75].

2.1.4 The edge detection approach

The edge detection approach, with its more recent variants [76, 77], is also worthy of mention. In this approach the robot takes into consideration the vertical edges of the obstacle; then, it looks for lines connecting edges, and considers them as the boundaries of the obstacles. As a result, it tries to move along the boundary. One of the main drawbacks of the method concerns its practical implementation: indeed, it is usually necessary for the robot to stop in front of the robot or obstacle to acquire accurate measurements of edges, since sensor data must be very accurate in order for the algorithm to work efficiently. Errors in sensor readings can result in the distortion of the original shape of the obstacle and hence a misreading may lead to a collision. On the contrary, the approach proposed in this thesis considers all sensor readings as if they belonged to different obstacles: the approach guarantees collision avoidance, thus being more robust to sensor noise [75] and allowing for real-time path-generation and updation while the robot is moving.

2.1.5 The Vector Field Histogram (VFH)

The VFH approach has been also extendedly applied to ground robots [52, 78]. In the VFH approach, the space around the robot is divided into sections of the same size, and every section has a value that represent the likelihood of the obstacle. The map is then translated into a polar histogram, that represents the space around the robot and the nearness of the obstacles. Finally, the robot direction is selected through heuristics, and can be straightforwardly applied to avoid both static and moving obstacles. The methods is suitable to work with sensors returning noisy information, however it has drawbacks similar to APF, in that - in complex environments - the method cannot guarantee that a path to the goal can be found even if it exists. With respect to VFH, the method proposed in this thesis has the same advantages that has been already mentioned when comparing it with APF [75, 79].

2.1.6 Virtual Force Field algorithm

It is an approach to find a target and to avoid objects in an environment [80]. It uses a positive or attractive force to guide the robot towards the target, while objects exert negative or repulsive forces to guide the robot away from obstacles. The algorithm also uses some kind of occupancy grid, called a certainty grid, in which each cell contains a certainty value. That value indicates how likely an object occupies that cell. To determine in which direction to move, a small window moves along with the robot. Each occupied cell exerts a repulsive force to the robot, and these forces are added to create a single repulsive force that tries to push the robot away from obstacles. At the same time, the target exerts an attractive force that tries to pull the robot closer to the target. From those two forces, a resultant force is calculated, which determines the next direction of the robot. Though this algorithm is quite smooth and accurate, it requires many sensors to scan the area around the robot for obstacles. To actually implement this as an embedded controller on a robot would be much more complicated than the earlier discussed methods. Other downsides are that the robot could get stuck if the resultant force becomes zero (this can occur, e.g., when there is an object directly between the robot and the target), and it can cause unstable behaviour in narrow pathways.

2.1.7 Wall following algorithms

They are well-known and simple algorithms for robot navigation [81]. In such an algorithm, the robot uses some sensor such as an infrared (IR) sensor or an ultrasonic sensor to follow walls at a certain distance. When the robot finds an obstacle, it follows the edges of the object like it would with a wall, until it can follow its original course again. When it drives into a wall, it turns and starts following the wall. This kind of algorithms are useful for navigation through environments such as mazes, but is not suitable for the kind of environments and mission we consider. In our case, the robot would only keep following the edges of the environment trying to find a path, while it should traverse the environment searching for the hill.

2.1.8 Dynamic Window Approach (DWA)

DWA relies on the idea of performing a local search for admissible velocities that allow the robot to avoid obstacles while meeting kinematics constraints [82]. In order to reduce computational complexity, the search is performed within a dynamic window which is centred around the current velocities of the robot in the velocity space, and only circular curvatures are considered. A solution to avoid local minima is proposed in [83, 84] by introducing a planning stage in DW which produces collision-free local paths with a given velocity profile. Recently, [85] has proposed the Forbidden Velocity Map, a generalization of the Dynamic Window concept that considers the obstacle's and robot's shape, velocity and dynamics, to deal with navigation in unpredictable and cluttered scenarios. To take into account kinematics constraints, obstacle avoidance has been fully integrated with path following in [86] in which path following is achieved by controlling explicitly the rate of progression of a "virtual target" to be tracked along the path [87, 88], and obstacle avoidance relies on the deformable virtual zone principle, that defines a safety zone around the vehicle, in which the presence of an obstacle drives the vehicle reaction. However, as stated by the authors, the combination of path following with a reactive obstacle avoidance strategy has a natural limitation coming from the situation where both controllers yield antagonist system reactions. This situation leads to a local minimum problem similar to APF, where a heuristic switch between controllers is necessary. The method proposed in this thesis includes an algorithm for path deformation in presence of obstacles and for path following, which at the same time guarantees goal-reachability [75] as well as Lyapunov convergence to the deformed path [1].

Summary

Other techniques for obstacle avoidances in which moving obstacles are considered, such as Neural Control System [89, 90] that requires the exact distance to the obstacle is still difficult to obtain as it may require more ultrasonic sensor, gyroscope, wheel encoder to extract further position information of robots. Some graph based search algorithms include: Simulation annealing approach that ignores the robot dimensions [91], A* search that disregards dynamic obstacle constraints [92]. Differential Game approach that only considers the static obstacles [93], Switching Formation strategy that works with very slow linear speed, considering static obstacle avoidance [94]. Sampling-based algorithms (like RRT method works in a manner that after the robot passed through two obstacles, route planning disregarded the safe border, and the recommended method ignored any difference between physical and presumptive obstacle location. [95, 96], sensor based online technique that consider the obstacle velocity equal to or less than the robot velocity [97], Electrostatic approach that totally relies on the avaiability of visual information [98] have not been taken into account into our analysis, since they are directly related to path planning and obstacle avoidance, while the proposed approach, that does not need to know a global map of the environment and overcomes the drawbacks of the discussed algorithms (but still guarantee that a path to the goal may be found [75]).

In this section, we have discussed different approaches that exist in the literature for path following and obstacle avoidance. The limitations of these approaches have been elucidated.

2.2 Multi-robot cooperation

Multi-robot cooperation is motivated by the fact that localization tasks can be done faster and more accurately by multiple robots than by a single robot. In addition, a distributed system is more robust since the failure of one of the robots does not halt the entire mission [99]. Furthermore, most cooperative approaches need to meet real-time constraints, autonomously making decisions for localization.

Modern robots have numerous applications in industrial, military, and domestic settings. Often we hear news about workers trapped in mines or workers losing their lives in petroleum refineries or power plants. Autonomous swarm robotics is an inexpensive alternative to having humans perform risky and hazardous tasks in such environments. Other examples include space exploration over long periods of time. By deploying robots to conduct dangerous tasks, risks can be minimized. In all above discussed scenarios, reliable perception is a key aspect towards achieving the desired tasks.

2.2.1 Multi-robot Systems

In addition to multi-robot localization, the additional aspects of multi-robot systems considered attention in this domain of research are task allocation, formation, and coordination. To read in detail a comprehensive review of different multi-robot systems, we refer the readers to [100, 101]. In this context, the first initiatives on multi-robot systems was the Autonomous Vehicle Aerial Tracking and Reconnaissance (AVATAR) approach, developed by Defence Advanced Research Project Agency (DARPA). The job was targeted military surveillance and demonstrated the effectiveness of a multiple-robot system in cooperative localization [102]. Perception of Offroad Robotics (PerceptOR), DAPRPA focused on cooperation between an autonomous helicopter and an autonomous ground vehicle. The helicopter helped the ground robot to navigate autonomously in an unknown environment [103]. The Mobile Autonomous Robot Software (MARS) program, is another multiple-robot framework by DARPA, which addresses issues such as cooperative target localization, search and rescue, and maintaining connectivity during cooperative mapping [104, 105]. The Multi-Autonomous Ground-robotic International Challenge (MAGIC) is a multi-robot competition, where the robotics teams were requested to explore and map large indoor and outdoor environments while identifying and neutralizing threats. MAGIC was funded by the Australian Department of Defence and the U.S. Army, with more than USD 1 million in prize money [106].

Some other contributions in cooperation of multi-robot such as, a thorough review on swarm robotics done by Barca et al. [107]. Ren et al. ([108, 109]) and Olfati-Saber et al. ([110]) reviews the consensus problems in the cooperation of multi-agent systems. A comprehensive survey on cooperative control is provided by Murray [111]. Bullo has mathematically presented the motion and localizion cooperation problem in a book

[112]. Olfati-Saber has presented and reviewed three flocking algorithms [113]. Multi-robot exploration ([114–116]) is another intriguing topic which is tightly coupled with multi-robot localization. A review of the exploration methods can be found in [117].

2.2.2 Challenges of multi-robot cooperation

Multi-robot cooperative approaches presents itself with a lot of challenges. Here we provide a brief overview of the same.

2.2.2.1 Relative Poses of Robots

In multi-robot localization, the map obtained by each robot in its own reference frame is called the local map. Each local map is generated from coordinated measurements such as laser scans. Each robot tries to integrate all of the local maps provided by other robots to generate a global map of the environment. However, this is a difficult task because the required alignments or transformation matrices which relate these maps to each other are in general unknown. The problem of the relative pose of the robot is coupled with the multi-robot data association problem. Knowledge of one, renders the other one simple.

2.2.2.2 Uncertainty of the Relative Poses

Uncertainty is pervasive in real world problems and arises if the robot lacks critical information for carrying out its task and is identified as having five main roots [118]. Uncertainty mainly arise due to unknown or partial knowledge of the environment, sensor noise, process noise, inexact models and imperfect computations. Planning under uncertainty is an instantiation of Partially Observable Markov Decision Process (POMDP) [119], whose exact solution is computationally intractable [120].

The uncertainties render the transformation matrices (discussed in Section 2.2.2.1) uncertain, further complicating the problem at hand. The multi-robot planning problem, in most cases then transforms to an uncertainty reduction problem. These unceratinties are often quantified in terms of the state covariance matrix and most approaches find robot configurations that minimizes the matix via an objective function.

2.2.2.3 Updating Maps and Poses

Once the relative transformation (Section 2.2.2.1) is found, a procedure is required to fuse local maps. The resulting map should integrate all information from the local maps. As a result of updating the maps, poses of the robots should also be updated. This requires considering the trajectory of the robots and new information received from other maps. Due to the nature of multi-robot localization, updating poses and maps is a coupled problem. In feature-based localization, its not only about finding the duplicate landmarks across the robot but the important thing is to identify that they are duplicate and hence should be taken as unique and not 2 different.

2.2.2.4 Line-of-sight Observations

When robots can see each other through direct observations (i.e., line-of-sight), the estimates can be improved. This fact can help robots to reduce localization error. In most applications and especially in close range localization, line-of-sight observations are much more reliable than other indirect estimation techniques. Richardson et al. shows an application where the line-of-sight observation is its key component. In this application, a rotary wing UAV lands on a moving platform autonomously. The UAV is equipped with a camera which identifies a known target pattern on the moving platform. Then the distance and orientation of the UAV with respect to the moving platform is calculated. The results are used to control and land the UAV [121].

2.2.2.5 Loop closure

Loop closure, also called cycle detection, is defined as identifying a place observed previously but not very recently (recently is defined in relation to the mission duration). Loop closure for a single robot is challenging enough. Extending this problem for a team of multiple robots requires solving it using all resources of information from individual robots. In multi-robot localization, various events can trigger loop closure, such as direct encounter of the robots or rendezvous and indirect encounter, when the robots see the same area or features in the world.

2.2.2.6 Complexity

Robotics applications are usually real-time. Thus, it is very important to design an algorithm capable of solving the above mentioned problems with minimum time and memory requirements. In multi-robot localization, space complexity and time complexity are two important issues. The complexity of a multi-robot algorithm directly affects its scalability.

2.2.2.7 Communications

Availability of a medium for data sharing among robots is an important requirement in multi-robot localization. Information between robots can be exchanged via communication channels. The quality of the communication channels is dependent on the environment. For instance, communication issues are a challenging problem for a team of robots in underwater environments, where the environment imposes limitations on the bandwidth and data rate.

2.2.2.8 Heterogeneous Vehicles and Sensors

An important advantage of team based mapping is that different types of robots, equipped with different sensors, can provide a better model of the environment. For instance, a ground robot may see features that a quadrotor cannot, and at the same time, a quadrotor may have access to different areas that a ground robot does not. However, this advantage requires processing and integrating different types of information. For example, if a ground robot provides an occupancy grid map and a quadrotor generates a feature map, then these maps must be combined to generate a global and consistent map. This issue has been studied in [122]. Due to the variety of sensors and maps, this problem can be presented in many different forms, such as integrating topological maps with grid maps, integrating topological maps with feature maps, integrating scanning laser rangefinder measurements with camera measurements, integrating satellite and aerial views with ground views [123], and many more. Michael et al. present a very good example of a team of heterogenous robots, including a quadrotor and two ground robots, which map a multi-floor earthquake-damaged building collaboratively [124].

2.2.2.9 Synchronization

As a general rule, each acquired sensor reading should have a time stamp field, which shows the time of the acquisition of the data. An important issue in a system of multiple agents and multiple sensors is the synchronization of the clocks. Synchronization can be considered at two levels: first, local synchronization, which means the sensors of each robot should be synchronized and second, global synchronization, which means all robots on the team must have synchronized clocks.

To synchronize time on different robots, Chrony[1] is a suitable choice, which is also used in robot operating system (ROS) middleware. Chrony supports online and offline time adjustment by two different applications. In the online case, a network time protocol (NTP) daemon runs in the background and synchronizes the time with time servers. For an isolated machine, one might enter the time periodically. The synchronized time appears as a label in the header of each acquired data. A similar approach is used by Leung et al. in [125].

2.2.2.10 Performance Measure

In multi-robot localization, evaluating the accuracy of results is a challenging problem due to the lack of the model of the environment and the actual trajectory of the robots. Additionally, evaluating the accuracy of localization becomes more critical when the robots rely on localization to perform autonomous behaviors. Therefore, performance measure is always required to determine the reliability of multi-robot localization.

2.2.3 Perception techniques for Multi-robot cooperation

Different senors are used for multi-robot localization and mapping. Below, we briefly overview the most common senors that are currently being used.

2.2.3.1 Monocular vision

Monocular vision techniques are mainly used for edge detection of roads for autonomous driving. By obtaining the same image of the road as we humans see while driving, it is

[1]Curnow, R. (2014). Chrony 1.29.1, from http://chrony.tuxfamily.org/.

possible to detect the road via various methods using a camera. Edge detection [126] works basically by identifying where the object color changes (its HSV or RGB code) significantly. Via this method lane-markers, and edged of the road are easily identified. Yoo et all [127] and Assidiq et all [128] use this method in their algorithm. Both of them include the use of Hough Transform [129], which is used to identify participle shapes, such as circles or lines. Another method is to use difference in intensity. Well prepared roads have lane markers with a higher intensity than the road itself. Edge detection works well in this scenario, as well as representing the lane-markers using local maxima points. However, the smaller the resolution of the image, the less edge detection performs. High resolution images however increase the computation time, which is a disadvantage for real-time performance.

Edge detection also needs more pre-processing to obtain the lane-markers, as more edges than the lane-markers are often found. The algorithm has to distinguish the difference between lane-markers and objects. Finding lane-markers based on local maxima is therefore found a more suitable method for real-time operation.

Once the previous and current lane are detected and the vehicle position within it is known, this information can be used to predict the state of the next frame. The next time-frame is then a control factor to determine if the calculations are correct or not.

An advantage of this method is that it is easy to implement and allows for good cost-effective solutions on a consumer-level. However, the approach has few disadvantages. Due to the high-reliability demands, building a vision-based system, even for the simplest applications, is a big development effort. Many functional blocks are required for a stable system, and many different conditions and assumptions have to be identified and handled. In addition a large validation effort is required, as many of the failure cases are rare and hard to predict.

2.2.3.2 Stereo vision

Stereo imaging is done by using two cameras in order to obtain 3D information, which can be compared to the human vision. The range accuracy is a function of the stereo baseline (the distance between the two cameras). A larger baseline stereo system will provide better range accuracy, but often at the cost of lower reliability and higher computation cost (a tougher correspondence problem to solve). Generally speaking, stereo imaging poses a greater processing challenge compared with LIDAR system, with an increased probability of errors. Still, stereo imaging can be used for the same basic tasks as LIDAR, including obstacle identification, host to road pitch angle estimation, curb detection and 3D road geometry and slope estimation. Bertozzi and Broggi developed an algorithm (GOLD) that make use of stereo vision [130].

This method is easy to implement and allows for good cost-effective solutions at the consumer-level. It is also able to compute distances of objects relative to the car. However, the approach has few disadvantages, namely, range accuracy and lower reliability than LiDAR as depth measurements are texture dependent. Compared to monocular vision techniques, a large development effort is involved for creating a highly-reliable system.

2.2.3.3 Light detection and ranging (LIDAR)

LIDAR is a surveying technology that measures distance by illuminating a target with a laser light. Soren and Benjamin [131] created a lane marker detection and mapping algorithm purely based on LIDAR alone. Albert et al. [132] combine LIDAR and monocular vision in order to find lane markers. Basically, LIDAR is used to:

- Identify objects and boundaries of the road by using its 3D detection.
- Identify the type of road by determining the road roughness. This can also be used to identify the edge of the road.
- Identify potholes and bumps in the road surface.

LIDARs can report reflected intensity as well, providing a substitute to a visual camera with an advantage of being an active light source and thus independent of natural light sources. This specifically helps in coping with shadows and darkness. Since lane marks only have intensity information and no 3D structure, intensity measurement is required if the LIDAR is to be used as the only modality. However, the LIDAR equipment is still relatively very expensive and require large development effort.

2.2.3.4 Radar

Radar is an object-detection system that uses radio waves to determine the range, angle, or velocity of objects. A radar transmits radio waves or microwaves that reflect from any object in their path. A receive radar, which is typically the same system as the transmit radar, receives and processes these reflected waves to determine properties of the object(s). It is able to detect obstacles like other vehicles or buildings, and able to discriminate between road an off-road regions based on their large reflective difference. The advantage of a radar is that it is relatively cheap, and can be used well in combination with more than one modality. The disadvantage is that the properties are only a limited subset as compared to the capabilities of a LIDAR.

Summary

In this section, we have discussed different approaches that exist in the literature for multi-robot localization. In the Chapter 4 we provide the mathematical framework and describe in detail our methodology with results and discussion.

Chapter 3

Path following and obstacle avoidance

In this chapter the extension of an existing algorithm for path-following in presence of obstacles [1] has been modified to the multi-robot case. The approach is validated in a controlled environment by ignoring the localization problem.

3.1 Materials and Methods

In this work, a unicycle–like mobile robot has been considered as a case study. The kinematics of a unicycle–like robot comprises linear and angular motion which can be represented as:

$$\begin{aligned}\dot{x} &= u\cos\theta \\ \dot{y} &= u\sin\theta \\ \dot{\theta} &= r\end{aligned} \tag{3.1}$$

where x, y and θ correspond to the position and orientation of the robot with respect to a fixed frame, u is the linear velocity and r is the angular velocity (i.e., the control inputs).

3.1.1 Path Following

The control structure for the system described here is an extension of [1], which shows the implementation of the method for path following of ground robots. The aim of the current work is to prove the performance and capability of the control algorithm also in presence of fixed and moving obstacles. In the proposed method, the path to be followed is defined as a curve on the plane expressed in its implicit form $f(x,y) = 0$: Figure 3.1 shows this concept[1].

It may be noticed that the value of the function $f(x,y)$ while the robot is in position x,y represents the error from the path. Indeed, when the robot is following the curve (i.e., it is on the desired path) it holds $f(x,y) = 0$, whereas the value of $f(x,y)$ locally increases or decreases when the robot abandons the path.

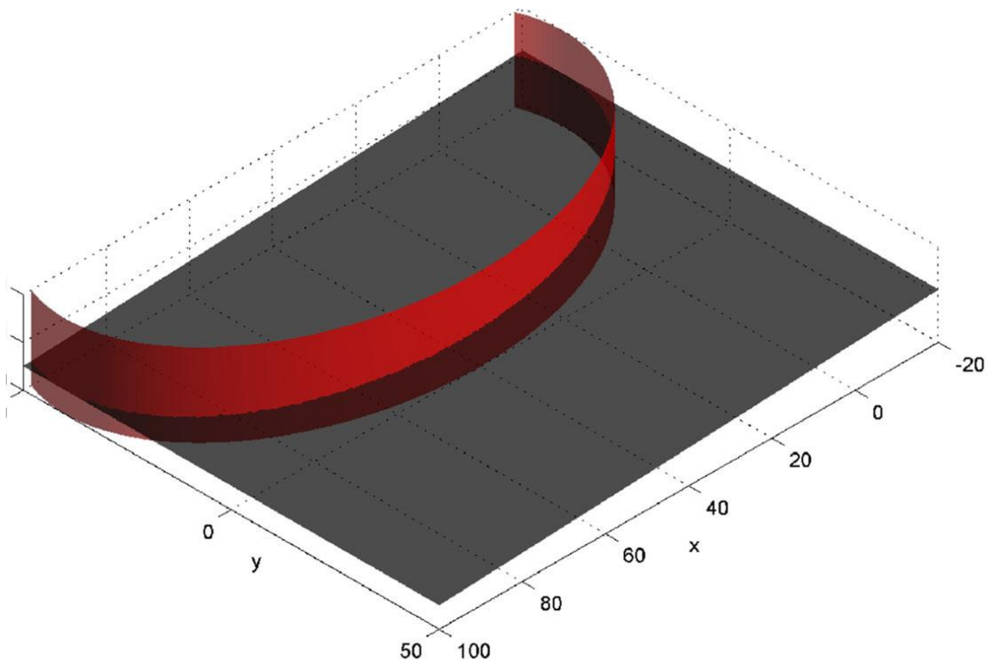

FIGURE 3.1: Path through surface intersection

The function $f(x,y)$ must meet the following constraints:

i. f has to be twice differentiable, with derivative f_x and f_y.

ii. The norm of the gradient $||\nabla f||^2 = f_x^2 + f_y^2 > 0$.

[1]To be more precise, the path in Figure 3.1 is given by the intersection of a cylindrical surface in the 3D space $f(x,y,z) = 0$ with a plane: however, in the rest of this thesis, we assume that such intersection is produced with the XY plane (i.e., the plane described by the implicit equation $z = 0$). In this case, it is possible to make the z variable disappear, and represent the path through a single implicit equation $f(x,y) = 0$ describing a planar 2D curve.

Under these assumptions, it has been shown that the robot can converge to the path by setting the control inputs as follows:

$$\begin{aligned} u &= u(t) \\ r &= K_1(-||\nabla f|| u S(f) - f_x |u| \cos\theta - f_y |u| \sin\theta) + \dot{\theta}_c \end{aligned} \tag{3.2}$$

$$\begin{aligned} S(f) &= \frac{K_2 f}{\sqrt{1+f^2}} \\ 0 < K_2 &\leq 10 \\ K_1 &> 0 \end{aligned} \tag{3.3}$$

where,

- $u(t)$ is a positive velocity profile;
- K_1 and K_2 are gains;
- $\dot{f} = \dot{f}(x, y, \theta) = f_x u \cos\theta - f_y u \sin\theta$ describes how f varies with time, i.e., it is a measure of how fast the vehicle is getting closer to / farther from the path[2];
- $S(f)$ is the C^n sigmoid function, where K_2 determines the shape of the sigmoid;
- $\theta_c = \arg(f_y - i f_x)$ is the orientation of the vector $(f_y, -f_x)$ normal to ∇f in (x, y), i.e., tangent to the level curve, and $\dot{\theta}_c$ is its derivative with respect to time, which takes into account the curvature of the path.

The control law in (3.2) can be intuitively interpreted as follows. If the vehicle is in (x, y, θ) and it is moving along a level curve $w = f(x, y)$ with $w > 0$, it holds $\dot{f} = 0$ and $\dot{\theta} = \dot{\theta}_c$: in this case, the controller sets $\dot{\theta} = \dot{\theta}_c - K_1 \|\nabla f\| uS(w)$, and the vehicle approaches the path by leaving the level curve with $w > 0$ on its left side. This follows the fact that $\dot{\theta} < \dot{\theta}_c$ since the second term is negative, i.e., $\dot{\theta}$ is set to a lower value than required to move on the level curve. Symmetrically, when the vehicle is moving along a level curve with $w < 0$, the controller sets $\dot{\theta} = \dot{\theta}_c + K_1 \|\nabla f\| uS(w)$ since $S(-w) = S(w)$, and the vehicle tends to the path by leaving the level curve on its right side as $\dot{\theta} > \dot{\theta}_c$. For a more detailed analysis of the control law in (3.2) and (3.3) see [1], which contains

[2] In [1], the absolute value of the velocity $|u|$ is used instead of u, which guarantees convergence to the path even when the vehicle is moving backward. Here we limit our analysis to positive values for sake of simplicity.

also an experimental evaluation of the impact of control gains K_1 and K_2 on the robot's trajectory.

3.1.2 Stability Analysis

The equation 3.2 can be rewritten as:

$$\begin{aligned} \dot{f} &= -\|\nabla f\|\, u\, sin\,(q_\pi) \\ \dot{q}_\pi &= K_1(-||\nabla f||uS(f) + (||\nabla f||\,|u|\sin(q_\pi) \end{aligned} \tag{3.4}$$

where,

$$\begin{aligned} q &= (2k+1)\,\pi\,|\,k\,\varepsilon\,Z \\ q_\pi &= q - \pi \end{aligned} \tag{3.5}$$

The solution to system 3.2 has unstable equlibrium points, where $||\nabla f|| > 0$, $lim_{t\to\inf} u \neq 0$, $and\, K_1 > 0$

To verify that the equilibrium point, i.e., ($f = 0$, q_π = 0) that corresponds to ($f = 0$, $q = \pi$) is unstable, a C^1 Lyapunov function, i.e., $V = V(f, q_\pi)$, is defined as

$$V = K_1K_2(\sqrt{1+f^2}+1) + 1 - cos(q_\pi) \tag{3.6}$$

arbitrarily close to the origin, can take straigtly positive values e.g., for ($f = 0$, $q_\pi \neq 0$), and $V(0,0) = 0$. Hence

$$\begin{aligned} \dot{V} = K_1 \frac{K_2 f}{\sqrt{1+f^2}} u\,||\nabla f||\,sin\,(q_\pi) \\ + sin\,(q_\pi)\,K_1(-||\nabla f||\,u\frac{K_2 f}{\sqrt{1+f^2}} + ||\nabla f||\,|u|\,sin\,(q_\pi) \end{aligned} \tag{3.7}$$

$$= K_1\,||\nabla f||\,|u|\,sin^2\,(q_\pi) \tag{3.8}$$

is positive definite. This implies that $(f = 0, q_\pi = 0)$ is unstable. The result can be extended to all points in the set $(f = 0, q_\pi = 2k\pi)\,|\,k\,\varepsilon\,Z$ as a consequence of the periodicity of q_π and consequntly to all points in the set $(f = 0, q = (2k+1)\pi)\,|\,k\,\varepsilon\,Z$.

3.1.3 Obstacle Avoidance

In order to avoid obstacles while following (as closer as possible) the desired path, the path itself may be deformed when the robot perceives the presence of any obstacle. This is done by introducing a Gaussian function with radial simmetry in correspondence of each obstacle $\mathcal{O}_j$ as in (3.9)

$$O_j(x,y) = A_j e^{\frac{-(x-x_j)^2+(y-y_j)^2}{\sigma^2}} \tag{3.9}$$

where,

- x_j, y_j represent the position of the obstacle;
- A_j is the amplitude of the Gaussian curve;
- σ is the standard deviation.

The idea is to add the obstacle function O_j to the left term of the equation $f(x,y) = 0$ defining the path to be followed. Please notice that the Gaussian curve in (3.9) is one of the possible candidates as an obstacle function (a different bell–shaped function may be adopted as well), and that the behavior of the robot in proximity of the obstacle can be modified by tuning the parameters σ and A_j. Figure 3.2 illustrates these concepts: the path $f(x,y) = 0$ obtained as the intersection of a cylinder with a plane is deformed by the presence of an obstacle. The result is a path $f'(x,y) = 0$ that avoids the obstacle, while staying as closer as possible to the original path[3].

By tuning the value of σ and A_j, it is possible to generate a collision free trajectory even in the presence of multiple moving obstacles. In presence of more obstacles, it is necessary to sum up all the individual obstacle contributions:

$$f'(x,y) = f(x,y) + \sum_{j=1}^{N} O_j(x,y) = 0 \tag{3.10}$$

where each obstacle j is modeled by its position and dimensions that influence the parameters x_j, y_j σ and A_j of the obstacle function O_j.

[3]Once again, the Figure shows the intersection of a cylindrical surface $f(x,y,z) = 0$ with a generic plane in 3D, and obstacles j are consequently modelled as 3D Gaussians $O_j(x,y,z) = 0$. However, in the thesis we consider only the plane $z = 0$ to make the z variable disappear in all equations, thus finally yielding obstacle functions expressed as $O_j(x,y) = 0$ and a deformed path $f'(x,y) = 0$.

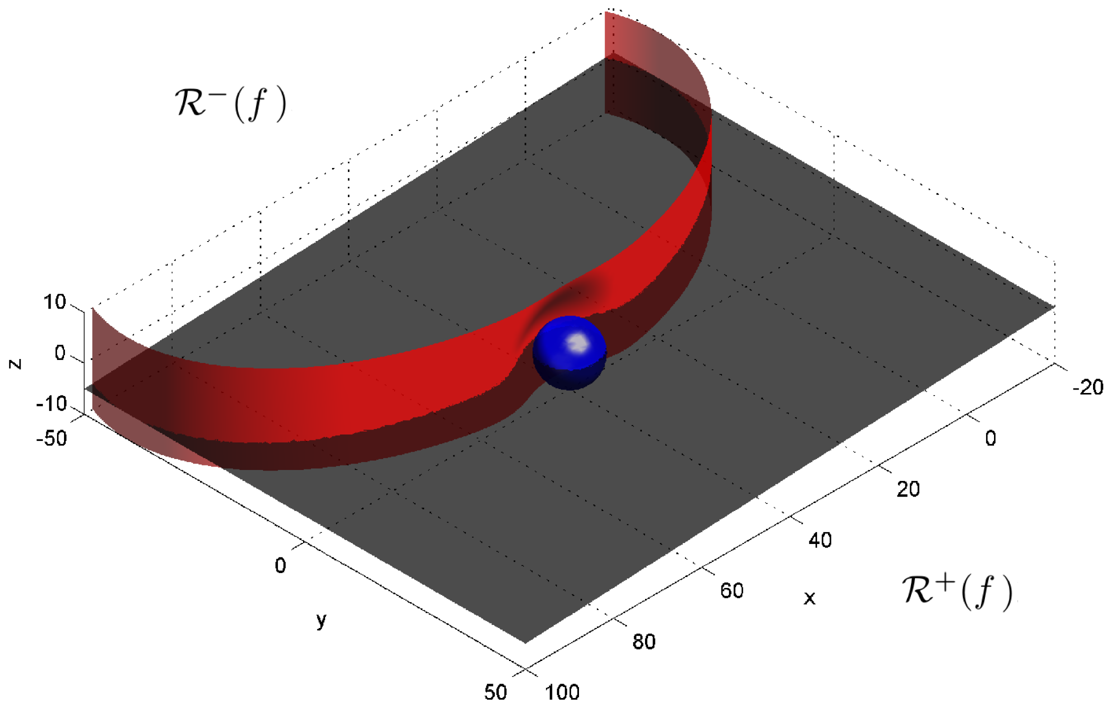

FIGURE 3.2: Deformation of the path due to the presence of an obstacle $\mathcal{O}_j$ with $A_j > 0$.

The obstacle detection and consequent path modification can be performed in real-time since it is sufficient to use the deformed path function $f'(x,y)$ instead of $f(x,y)$ in (3.2): as usual, to compute the path following error, $f'(x,y)$ needs to be evaluated only in the current robot's pose and its expression can be updated as soon as an obstacle has been detected. For instance, in presence of N obstacle and a nominal path corresponding – respectively – to a straight line $y = 0$, a circumference $x^2 + y^2 - R^2 = 0$, and a sine wave $y - sin(x) = 0$, the deformed path $f'(x,y) = 0$ would be computed as follow:

$$f'(x,y) = y + \sum_{j=1}^{N} O_j(x,y) = 0 \tag{3.11}$$

$$f'(x,y) = x^2 + y^2 - R^2 + \sum_{j=1}^{N} O_j(x,y) = 0 \tag{3.12}$$

$$f'(x,y) = y - sin(x) + \sum_{j=1}^{N} O_j(x,y) = 0. \tag{3.13}$$

3.1.4 Shaping the obstacle function to avoid collisions

The procedure to choose the values of σ and A_j deserves a deeper discussion.

First of all, it shall be noticed that the sign of A_j shall be chosen a priori. The undeformed curve $f(x,y) = 0$ divides the plane in two half-planes R^+ and R^-, and the deformed curve $f'(x,y) = 0$ lies on either R^+ or R^- depending on the sign of A_j: this determines whether obstacles are avoided on the right or on the left (Figure 3.2).

Second, to compute the absolute value of A_j, consider an obstacle $\mathcal{O}_j$ centered in a position x_j, y_j close to the path. A safety margin r_j shall be introduced to take into account both the dimensions of the obstacle and the vehicle: a collision may happen if the distance $d_j(x,y)$ between the position x,y of the vehicle and x_j, y_j is less or equal than the safety margin r_j. That is, $\mathcal{O}_j$ is defined as a circle

$$\mathcal{O}_j = \{(x,y) \ s.t. \ |(x,y) - (x_j,y_j)| \leq r_j\}. \tag{3.14}$$

In order to choose the actual value of A_j to avoid collisions, the path should not intersect any obstacle region.

Then, in presence of N obstacles, the following must hold:

$$f'(x,y) \neq 0, \ \forall (x,y) \in \bigcup_{j=1}^{N} \mathcal{O}_j \tag{3.15}$$

From (3.15) and (3.10) it follows that

$$f(x,y) + \sum_{j=1}^{N} O_j(x,y) \neq 0, \ \forall (x,y) \in \bigcup_{j=1}^{N} \mathcal{O}_j \tag{3.16}$$

which can be be solved for A_j by requiring either of the two situations below to hold:

$$f(x,y) + \sum_{j=1}^{N} O_j(x,y) > 0, \ \forall (x,y) \in \bigcup_{j=1}^{N} \mathcal{O}_j \tag{3.17}$$

or

$$f(x,y) + \sum_{j=1}^{N} O_j(x,y) < 0, \ \forall (x,y) \in \bigcup_{j=1}^{N} \mathcal{O}_j. \tag{3.18}$$

Suppose now that the arbitrary choice $A_j > 0$ has been made. In this case, it is convenient to satisfy (3.17), which allows us to simplify computations for the following reasons:

(i) the condition (3.17) is always satisfied for those obstacles $\mathscr{O}_j$ that lie completely in the semispace with $f(x_j, y_j) > 0$;

(ii) if the condition (3.17) is satisfied for each individual obstacle taken separately, it is also verified when considering all the obstacles.

Both properties above are due to the fact that, when $A_j > 0$, each individual obstacle adds a positive contribution in (3.17): this allows for computing A_j and σ to satisfy (3.17) for each obstacle taken separately.

Let us consider a generic obstacle $\mathscr{O}_j$: it can be observed that, since the Gaussian contribution in (3.9) has a radial simmetry, the minima of $O_j(x,y)$ in $\mathscr{O}_j$ necessarily lie on the boundary $\partial\mathscr{O}_j$, i.e., the circumference with radius r_j centered in x_j, y_j:

$$\min_{(x,y)\in\mathscr{O}_j} O_j(x,y) = A_j e^{-r_j^2/\sigma^2} \tag{3.19}$$

Moreover, in Section 3.1.1 we set the constraint $\nabla f(x,y) \neq 0$, with the effect that also the minima of $f(x,y)$ in $\mathscr{O}_j$ lie on the boundary $\partial\mathscr{O}_j$.

Then, from (3.17) it must hold

$$\min_{(x,y)\in\partial\mathscr{O}_j} f_1(x,y) + A_j e^{-r_j^2/\sigma^2} > 0, \tag{3.20}$$

hence

$$A_j > -\min_{(x,y)\in\partial\mathscr{O}_j} f(x,y) e^{r_j^2/\sigma^2}, \tag{3.21}$$

that yields, for each given σ, a lower bound on A_j.

Notice that, if we make the a priori choice $A_j < 0$, it is convenient to focus on (3.18), which guarantees the properties (i) and (ii) whereas (3.17) does not. After some computations, this finally requires to satisfy a condition similar to (3.21), but with the opposite inequality. Whichever choice is made for the sign of A_j, properties (i) and (ii) hold if and only if A_j has the same sign for all obstacles.

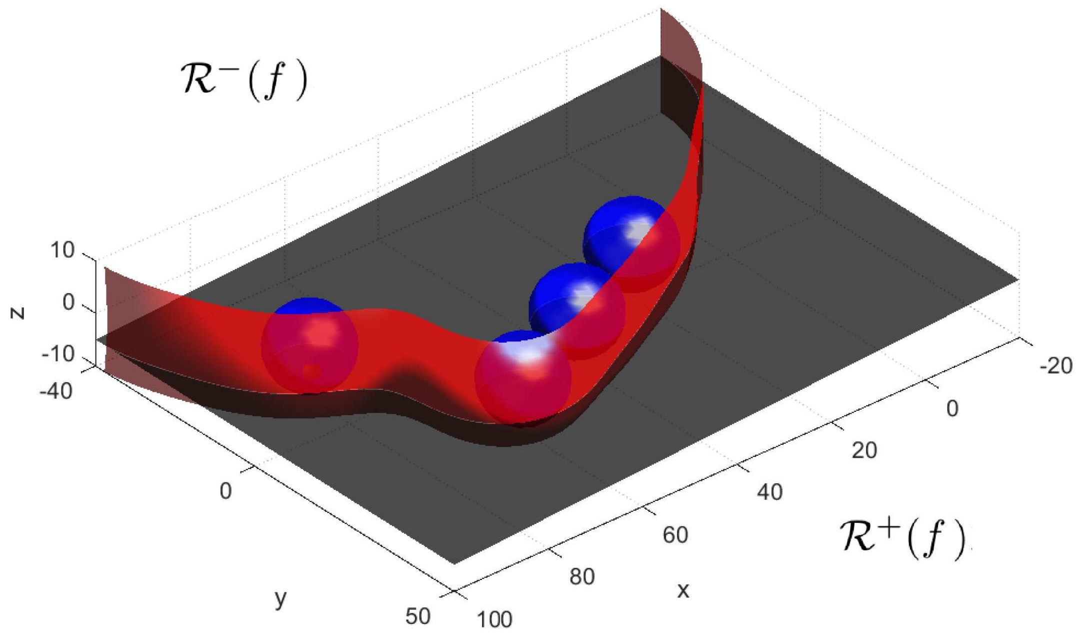

FIGURE 3.3: Deformation of the path due to the presence of multiple obstacles $\mathcal{O}_j$ with $A_j < 0$.

Lagrange multipliers can be used to find the minimum of $f(x,y)$ in $\partial\mathcal{O}_j$: this basically corresponds to finding the two level curves $f(x,y) = w_\alpha$ and $f(x,y) = w_\beta$ which are tangent to $\mathcal{O}_j$, respectively, in (x_α, y_α) and (x_β, y_β), and then taking the minimum between w_α and w_β.

In the case that the initial path is a straight line $f(x,y) = ax + by + c$, after some computations it holds:

$$\begin{aligned} f(x_\alpha, y_\alpha) &= -\|\nabla f\| r_j + a x_j + b y_j + c \\ f(x_\beta, y_\beta) &= \|\nabla f\| r_j + a x_j + b y_j + c \end{aligned} \tag{3.22}$$

with the minimum corresponding to (x_α, y_α). Then, in order for the path not to intersect $\mathcal{O}_j$, the following relation must hold between A_j and σ:

$$A_j > -f(x_\alpha, y_\alpha) e^{r_j^2/\sigma^2}. \tag{3.23}$$

The procedure above must be reiterated for all obstacles $\mathcal{O}_j$ by substituting in (3.22), (3.23) the corresponding value of x_j, y_j. This allows for computing an admissible value for A_j depending on σ, thus guaranteeing that the deformed surface $f'(x,y) = 0$ does not collide with any obstacle, see Figure 3.3. In the case that the path $f(x,y) = 0$ is not a straight line, using the Lagrange multipliers to find $f(x_\alpha, y_\alpha)$ and $f(x_\beta, y_\beta)$ is not as computationally efficient as in the linear case. Therefore, a slightly different procedure is adopted which is based on the same rationale, but requires to approximate $f(x,y) = 0$ with a straight line (the whole procedure is not shown here for sake of brevity).

Finally notice that – depending on the value of σ and A_j – different paths are obtained: all of them guarantee that the constraint in (3.15) is met, but have different shapes. When σ is higher, the vehicle is influenced by obstacles at a greater distance, thus avoiding the obstacle along a lower curvature path.

In order to evaluate the performance of the described approach, experiments have been performed with the robots Create, manufactured by iRobot. In particular, a variable number (from 1 to 3) of Create Robots move along predetermined and intersecting paths, so that the robots need to modify their paths in order to avoid colliding with each other. Additionally, a variable number of persons (from 1 to 3) are instructed to walk randomly in the same area, acting as mobile obstacles for the robots. All experiments have been performed within a $3m \times 3m$ area, inside a motion capture (MoCAP) environment (i.e: Motive Cap) that provides positioning feedback of any rigid body inside its perceptual field calculated by using 8 cameras located at the ceiling of the area. The system is based on the usage of reflective markers that are placed on the robots and the obstacles. The required 4 markers are placed on top of each robot and person as shown in Figure 3.4 and each set of markers is initialized as a rigid body with respect to the frame of reference. By measuring the size and the shape of the rigid body by using the markers, the system is able to precisely estimate the position and the orientation of the wheeled robots.

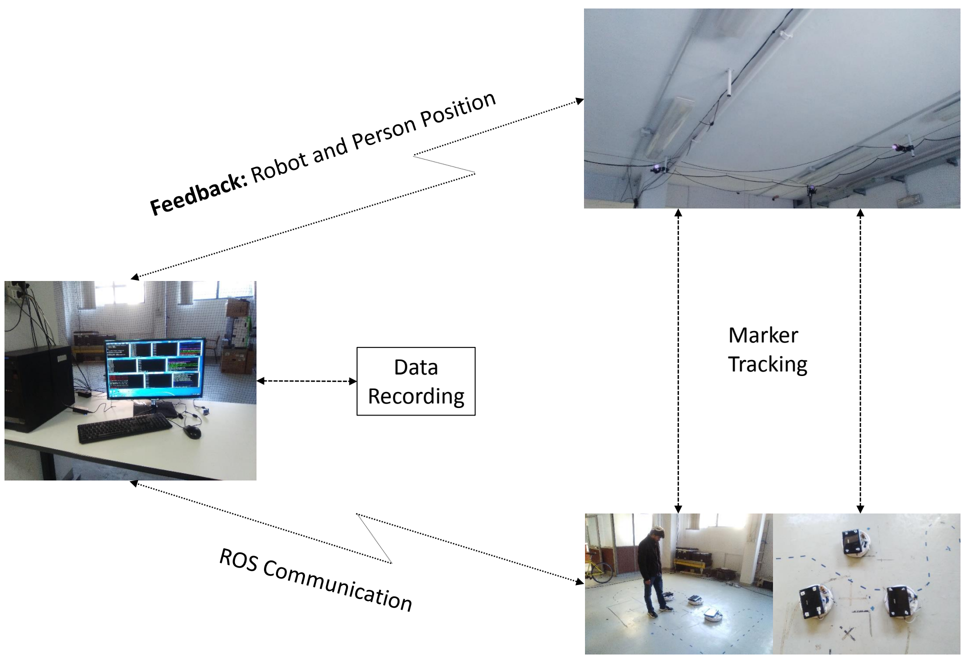

FIGURE 3.4: Structure of the system

The whole architecture of the Robot includes Create and a processing board with Ubuntu Linux (version 12.04-2). Robots are wirelessly connected with a ground station, that receives feedback by the MoCap and works as a Master for all rigid bodies. The ROS (Robot Operating System) environment has been used in order to allow the robots and the ground station to communicate with each other. The positions of robots and obstacles in space are used as inputs for the proposed method, allowing the calculation of safe linear and angular velocities that can be generated through (3.2), (3.3).

Of course, more accurate systems or other additional velocity estimation algorithms based on pose information can be used [133]. The choice of a motion capture system instead of using on-board sensors for localization and obstacle detection was mainly due to the fact that sensing and localization was not among the objectives of this work. Obviously, the method can be applied also by estimating the robot and obstacles position by using on-board sensors: in both cases, only relative pose information is required concerning the surrounding obstacles (including other robots), which makes the proposed method very effective when limited information is available.

In order to perform experiments, paths corresponding to a straight line, a sine wave and a circumference have been considered. Experiments have been grouped in 4 scenarios:

1. with no obstacles, to define a baseline to setup gains (A and σ) and compare the results with different speed.

2. with static obstacles, by changing the navigation speed from 0.1 to $0.6 m/s$ and with three different configurations:
   - 1 static obstacle.
   - 2 obstacle placed together.
   - 2 obstacle placed apart from each other.

3. with moving obstacles:
   - circular path with more robots moving in the same area with different speed and gains.
   - sine wave, crossing, back and forth: with two robots moving along intersecting sine wave paths, by crossing each other path.
   - sine wave, back and forth: with two robots moving along the same sine wave path, but starting in opposite direction.
   - straight line, crossing, back and forth: with two robots moving along intersecting straight line paths, by crossing each other path.
   - straight line, back and forth: with two robots moving along the same straight line path, but starting in opposite direction.

4. with robots and persons, varing the number of persons from 1 to 3.

The requirement for all cases is that the robots and persons should remain inside the predefined rectangular MoCap arena during the entire process. The results are depicted in following figures and Tables. Notably, all the collision-avoidance cases designed are based on a fundamental assumption that the position of the static and moving obstacles in the environment can be carefully detected, thus temporarily ignoring the problem of sensor noise. Before proceeding to the results section it would be worthwhile to consider the following:

- A varying number of robot $k = 1....K$ has been considered, each moving along an actual path described as x_k, y_k.

- The equation used to calculate the error at time t of each robot k between its actual path and the desired (undeformed) path is:

$$e_k = f(x_k, y_k) \tag{3.24}$$

where x_k, y_k are the coordinate of the robot at time t. The error e_k should be interpreted as a measure of how far the robot is forced to deviate from the desired (undeformed) path due to the presence of surrounding obstacles.

- The average over time and standard deviation of $|e_k|$ are calculated and denoted by $av(|e_k|)$ and $std(|e_k|)$

- The Euclidean distance between two agents k and i at time t (k and i can either be persons or robot) is computed and denoted by $dist_{i,k}$, with k,i being integer numbers. When $dist_{i,k}$ is greater than a safety distance d_{safe}, this means that there have been no collisions between agents (the Create robots used in experiments can be modelled as circles with radius $0.17m$, and we model persons in the same way for the purpose of the present analysis: then we assume $d_{safe} = 0.34m$).

3.2 Experimental Setup

To evaluate the performance of the proposed approach, various experiments have been performed with the unicycle-type robots Create, manufactured by IRobot. The following experimental setup has been used (Figure 3.5):

- All experiments have been performed within a $3m \times 3m$ area, inside a motion capture (MoCAP) environment (i.e: Motive Cap) that provides positioning feedback of any rigid body inside its range calculated by using 8 cameras in the MoCAP arena.

- A static ground station receives feedback by the MoCAP and works as a Master for all robots.

- A variable number (from 1 to 3) of Create Robots has been adopted. The robots are connected with a portable PC running Ubuntu Linux (version 12.04-2) and wireless connected with the Master ground station via ssh protocol.

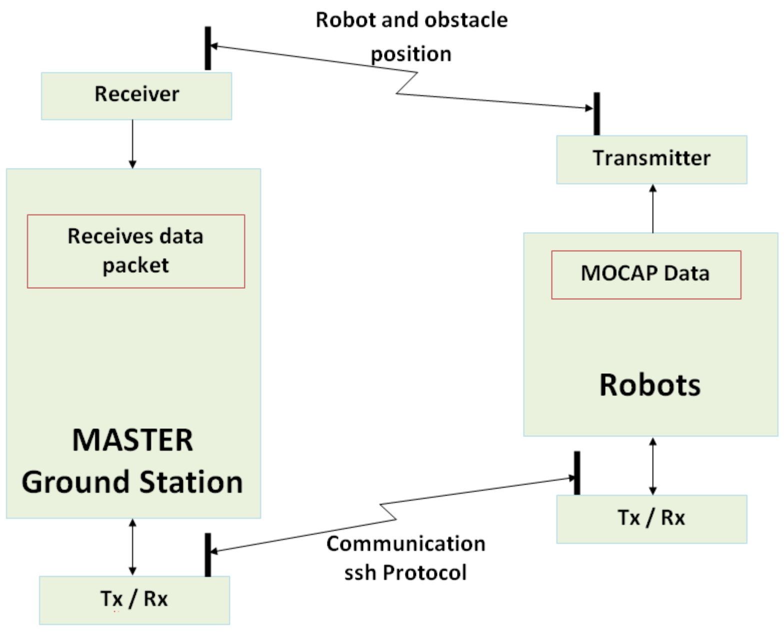

FIGURE 3.5: Architecture of the system

- The ROS (Robot Operating System) environment has been used in order to allow robots and ground station to communicate with each other.

3.2.1 Results and Discussion

In this section, three different scenarios and seven cases have been designed to test the efficiency of algorithm. The scenario details, together with related Figures and Tables are listed.

Notably, all the collision-avoidance cases designed are based on a fundamental assumption that the position of the static and moving obstacles in the environment can be carefully detected, thus temporarily ignoring the problem of sensor noise.

3.2.2 Scenario 1: Without Obstacles

The first experiment is aimed at validating the approach without any obstacles. The robot follows a circular reference trajectory of radius $R = 0.7m$. The response of the robot while following the reference path is shown in Figure 3.6. The distance of robot from its circular path is shown in Figure 3.7.
Table 3.1 corresponds to the experiments performed in absence of obstacles and by variating the linear velocity from $0.1m/s$ to $0.6m/s$. The first column reports the velocity

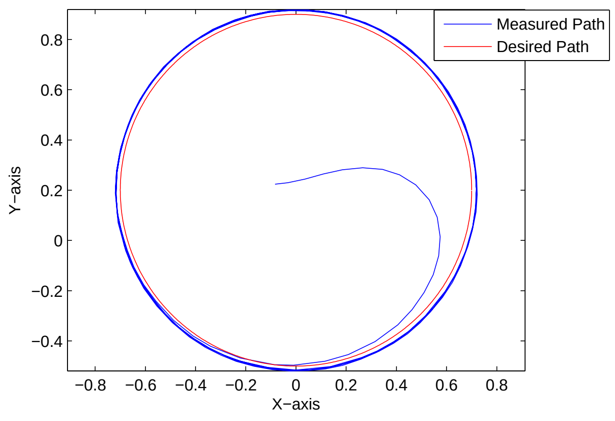

FIGURE 3.6: Robot response without obstacle
$u_1(t) = 0.3, K_1 = 15, K_2 = 2$

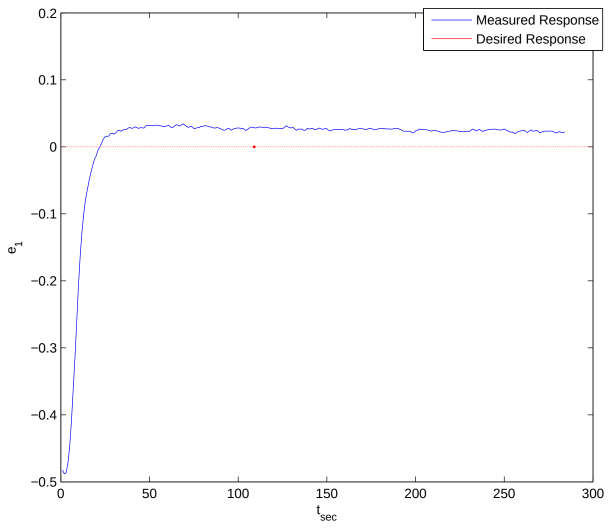

FIGURE 3.7: Plot of e_1 versus time in absence of obstacles

$u_1(t)$ of the robot (that is constant within each experimental run), the second column reports the average $av(|e_k|)$ and the standard deviation $std(|e_k|)$ of the error between the actual robot path and the path to be followed. The third column reports the control gains that shall be properly tuned depending on the velocity. It can be noticed that, when moving in absence of obstacles along a circular path, the value of the linear velocity has no significant impact on of e_k when in the range $0.1m/s - 0.6m/s$.

TABLE 3.1: Response of Robot with different velocities and without obstacle

$u_1(t)$ m/s	$av\ (\lvert e_1\rvert)$, $std\ (\lvert e_1\rvert)$	**Gains** K_1 , K_2
0.1	0.0380 , 0.0871	20 , 4
0.2	0.0542 , 0.0855	18 , 3.5
0.3	0.0343 , 0.0850	15 , 2
0.4	0.0600 , 0.0848	12 , 1.6
0.5	0.0485 , 0.0862	11 , 1.2
0.6	0.0659 , 0.0841	10 , 1

3.2.3 Scenario 2: With Static Obstacle

The second experiment is performed adding a static obstacle in a selected point along the reference path, while the robot moves along a circle with radius $R = 0.9m$.

The response of robot is shown in Figure3.8, while the plot of e_1 versus time is plotted in Figure3.9. Table 3.2 corresponds to the same experiment repeated with different linear velocities in the range between $0.1m/s$ to $0.6m/s$. It also shows the corresponding values of σ and A_j, that have been properly tuned to avoid the obstacles as required. The analysis of the values of $av(|e_1|)$ and $std(|e_1|)$ shows how the robot diverges from the path in presence of the single static obstacle.

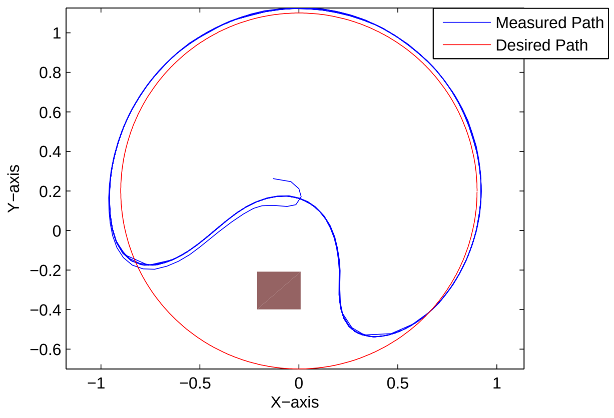

FIGURE 3.8: Robot response with one static obstacle
$u(t) = 0.3, K_1 = 15, K_2 = 2, A_j = 0.8, \sigma = 0.5$

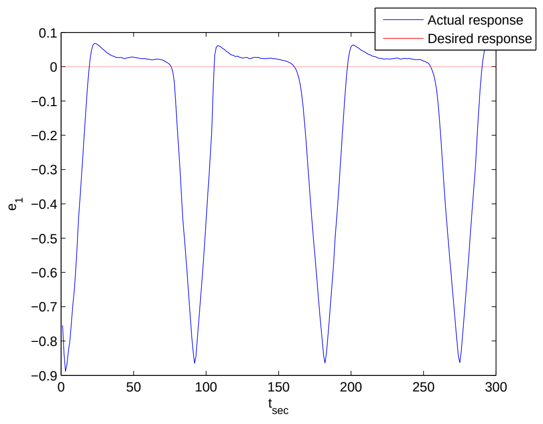

FIGURE 3.9: Plot of e_1 versus time with one static obstacle

TABLE 3.2: Response of Robot with one fixed obstacle

$u_1(t)$ m/s	$av\ (\lvert e_1 \rvert)$, $std\ (\lvert e_1 \rvert)$	**Gains** K_1, K_2	A_j , σ
0.1	0.1781 , 0.2625	20 , 4	0.8 , 0.5
0.2	0.2011 , 0.2646	18 , 3.5	0.8 , 0.5
0.3	0.1826 , 0.2634	15 , 2	0.8 , 0.5
0.4	0.2102 , 0.2630	12 , 1.6	0.8 , 0.5
0.5	0.1963 , 0.2647	11 , 1.2	0.8 , 0.5
0.6	0.1871 , 0.2641	10 , 1	0.8 , 0.5

In the third case, two static obstacles are placed together. The robot response is shown in Figure3.10 and the distance of the robot from its defined path is shown in Figure3.11. Table 3.3 shows at a glance the behaviour of the robot with different speed from $0.1m/s$ to $0.6m/s$. It can be seen how the addition of an obstacle, keeping the same value of σ and A_j of the previous case, reduce the deviation of the robot from the predetermined trajectory, given by the standard deviation $std(\lvert e_1 \rvert)$. Indeed, the robot starts deviating earlier, but it remains closer to the predefined circular path.

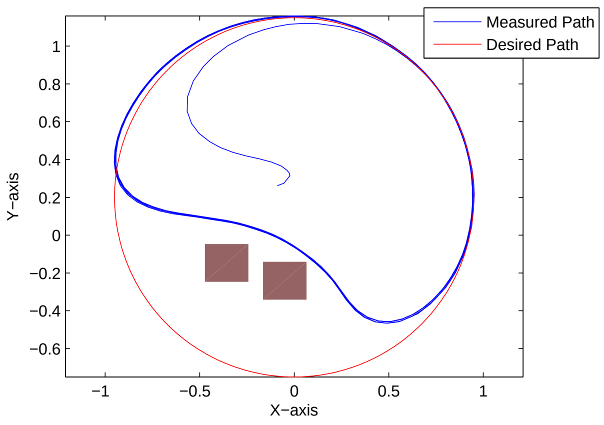

FIGURE 3.10: Robot response with two static obstacle placed together
$u(t) = 0.3, K_1 = 15, K_2 = 2, A_j = 0.8, \sigma = 0.5$

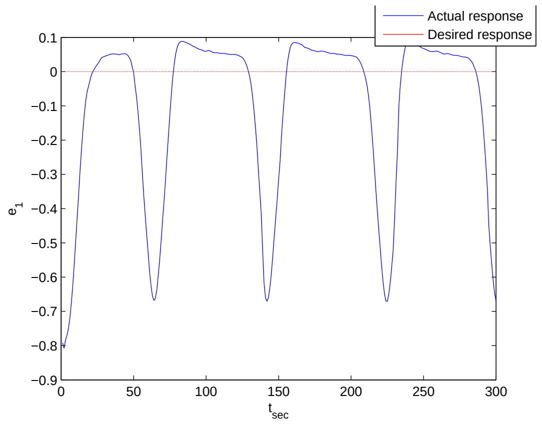

FIGURE 3.11: Plot of e_1 versus time with two static obstacle placed together

TABLE 3.3: Response of Robot with two fixed obstacle placed together

$u_1(t)$ m/s	$av\ (\lvert e_1\rvert)$, $std\ (\lvert e_1\rvert)$	**Gains** K_1, K_2	A_j , σ
0.1	0.1866 , 0.2053	20 , 4	0.8 , 0.5
0.2	0.2386 , 0.2029	18 , 3.5	0.8 , 0.5
0.3	0.1651 , 0.2061	15 , 2	0.8 , 0.5
0.4	0.2235 , 0.2037	12 , 1.6	0.8 , 0.5
0.5	0.2134 , 0.2042	11 , 1.2	0.8 , 0.5
0.6	0.1984 , 0.2049	10 , 1	0.8 , 0.5

In the last experiment of this scenario, static obstacles are placed along the path but apart from each other. The response in Figure3.12 confirms the ability of the robot of following the circular path by avoiding both obstacles and Figure3.13 shows the distance of the robot from the given circular path. Table 3.4 summarizes the results related to this case study. It can be seen how in this situation the standard deviation of the error between the reference and the actual path is further reduced.

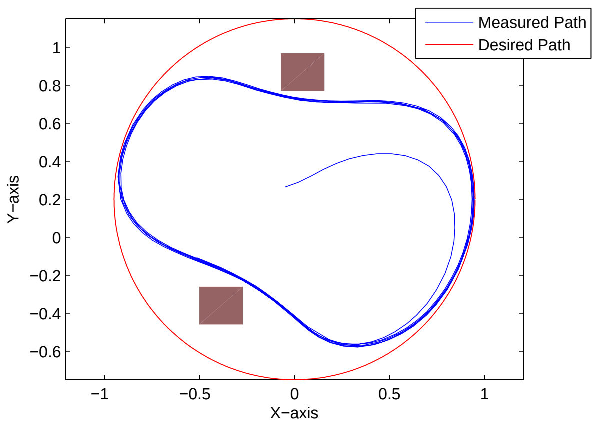

FIGURE 3.12: Robot response with 2 obstacles placed at different position $u(t) = 0.3, K_1 = 15, K_2 = 2, A_j = 0.8, \sigma = 0.5$

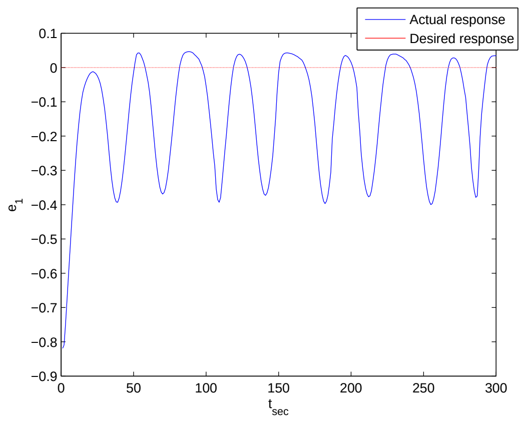

FIGURE 3.13: Plot of e_1 versus time with two static obstacle placed apart

TABLE 3.4: Response of Robot with two fixed obstacle placed apart

$u_1(t)$ m/s	$av\ (\lvert e_1\rvert)$, $std\ (\lvert e_1\rvert)$	**Gains** K_1, K_2	A_j , σ
0.1	0.1592 , 0.1594	20 , 4	0.8 , 0.5
0.2	0.1560 , 0.1553	18 , 3.5	0.8 , 0.5
0.3	0.1552 , 0.1561	15 , 2	0.8 , 0.5
0.4	0.1685 , 0.1627	12 , 1.6	0.8 , 0.5
0.5	0.1784 , 0.1677	11 , 1.2	0.8 , 0.5
0.6	0.1620, 0.1604	10 , 1	0.8 , 0.5

3.2.4 Scenario 3: With moving Obstacle

The last scenario involves the presence of moving obstacles, i.e. more WRs are placed in the arena, moving along their respective path and avoiding each other. Indeed, paths are intersecting each other and each robot recognizes the other robots as obstacles.

In the first test of this scenario, two robots are used. The robots start moving with a distance of $0.8m$, and follow two circular paths with radius $R = 0.6m$ (Figure 3.14). Figure3.15 shows the plot of e_1 and e_2 versus time while Figure3.16 shows $dist_{1,2}$, i.e., the mutual distance between the two robots, versus time. Finally, Table 3.5 summarizes results, including the average and standard deviation of $|e_1|$ and $|e_2|$, as well as the minimum distance $\min(dist_{1,2})$. This value is particularly significant since it is a measure of the safety of the approach, and shows that no collisions have been detected during the experiments.

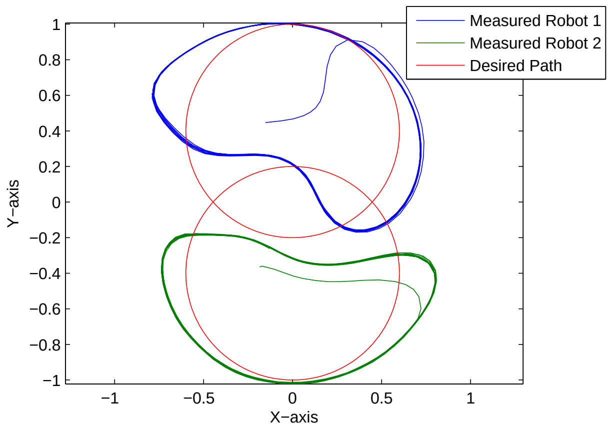

FIGURE 3.14: Two Robots with same speed recognizing each other as an obstacle $u(t) = 0.3, K_1 = 15, K_2 = 2, A_j = 0.5, \sigma = 0.45$

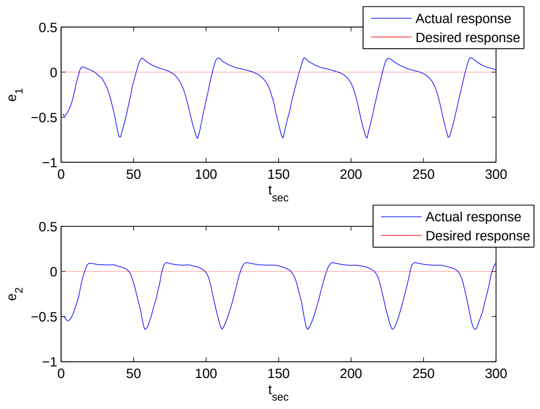

FIGURE 3.15: Plot of e_1 and e_2 versus time

The same experiment has been repeated by letting the robots move with different linear speeds $u_1(t)$ (for robot 1) and $u_2(t)$ (for robot 2), keeping A_j and σ constant. The plots

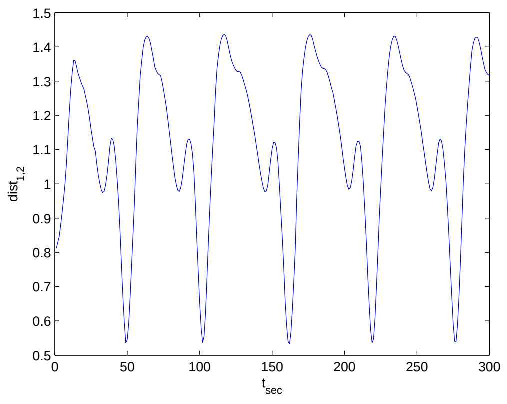

FIGURE 3.16: Plot of $dist_{1,2}$ versus time

TABLE 3.5: Response of Two Robot with same speed and varying σ

$u(t)$ m/s	$av\ (\lvert e_1\rvert)$, $std\ (\lvert e_1\rvert)$	$av\ (\lvert e_2\rvert)$, $std\ (\lvert e_2\rvert)$	**min** $dist_{1,2}$	**Gains** K_1, K_2	A_j , σ
0.3	0.1380 , 0.0976	0.1344 , 0.1167	0.5358	15 , 2	0.4 , 0.4
0.3	0.1439 , 0.0896	0.1363 , 0.0996	0.5369	15 , 2	0.4 , 0.45
0.3	0.1586 , 0.1218	0.1726 , 0.1358	0.5542	15 , 2	0.4 , 0.5
0.3	0.1780 , 0.1286	0.1916 , 0.1358	0.5797	15 , 2	0.4 , 0.55
0.3	0.1799 , 0.1112	0.1562 , 0.0839	0.5915	15 , 2	0.4 , 0.6

of the robot paths are shown in Figure3.17, while the error e_1 and e_2 and the distance $dist_{14}$ are plotted in Figure3.18 and in Figure3.19. Finally, Table VI summarizes the results. Notice that, when changing the linear speed, it is also necessary to properly tune the control gains K_1 and K_2, that now turn out to be different for the two robots.

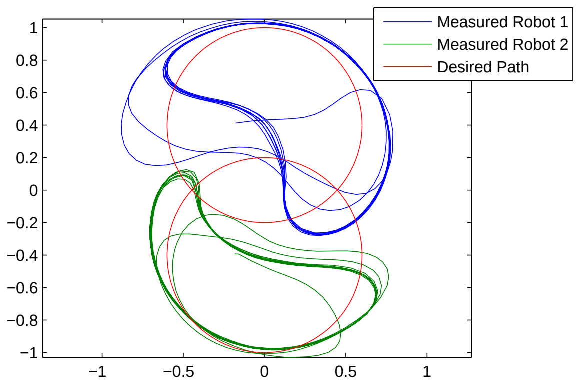

FIGURE 3.17: 2 Robots moving with different speed recognizing each other as an obstacle
Robot 1: $u_1(t) = 0.6, K_1 = 10, K_2 = 1, A_j = 0.4, \sigma = 0.55$
Robot 2: $u_2(t) = 0.5, K_1 = 10, K_2 = 1.2, A_j = 0.4, \sigma = 0.55$

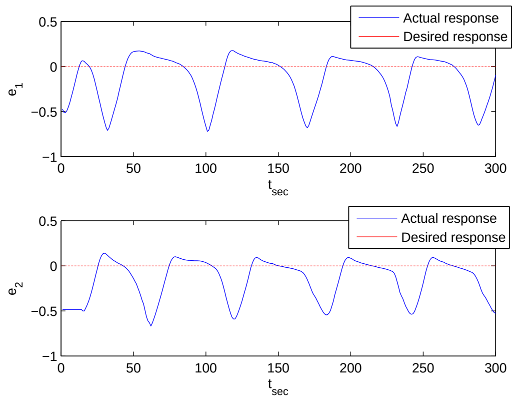

FIGURE 3.18: Plot of e_1 and e_2 versus time

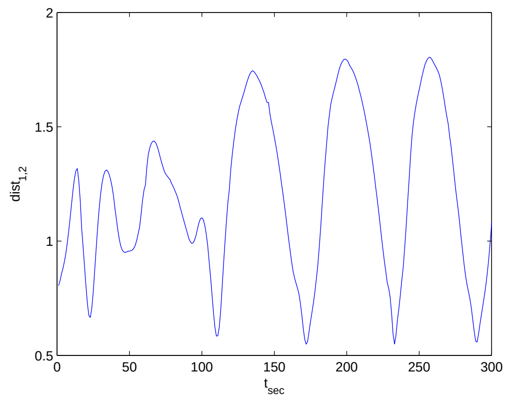

FIGURE 3.19: Plot of $dist_{1,2}$ versus time

TABLE 3.6: Response of Two Robot with different speed

$u_1(t), u_2(t)$ m/s	$av\ (\lvert e_1 \rvert)$, $std\ (\lvert e_1 \rvert)$	$av\ (\lvert e_2 \rvert)$, $std\ (\lvert e_2 \rvert)$	**min** $dist_{1,2}$	$Robot_1$ **Gains** K_1, K_2	$Robot_2$ **Gains** K_1, K_2	A_j , σ
0.3 , 0.1	0.1631 , 0.1342	0.1550 , 0.1363	0.5489	15 , 2	20 , 4	0.4 , 0.55
0.4 , 0.5	0.1823 , 0.1226	0.1676 , 0.1121	0.5323	12 , 1.6	11 , 1.2	0.4 , 0.55
0.6 , 0.5	0.2083 , 0.1298	0.1821 , 0.1073	0.5390	10 , 1	11 , 1.2	0.4 , 0.55

Finally, the last test has been performed with a more complex scenario, using three robots, moving along three intersecting circular paths, using different σ and having different speeds.

Figs. 3.20, 3.21 and 3.32 respectively show the plot of the robot paths, the error e_1, e_2 and e_3 versus time and the plot of distance distances between robots (i.e., $dist_{12}$, $dist_{13}$, $dist_{23}$). Table 3.7 and Table 3.8 summarize the results of the experiments with three robots, firstly executing by variating σ (Table 3.7) and then having three different linear velocities $u_1(t)$, $u_2(t)$, $u_3(t)$ for the three robots (Table 3.8).

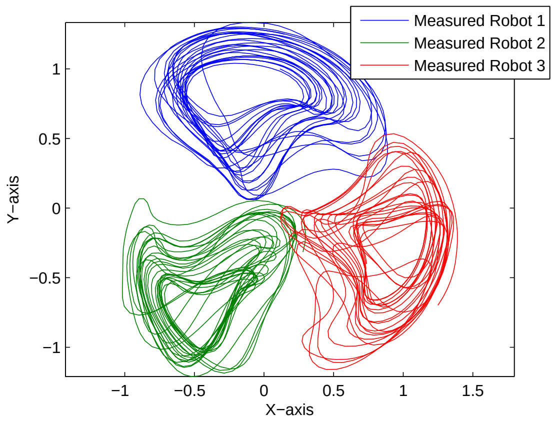

FIGURE 3.20: Three Robots with different speed and different σ recognizing each other as an obstacle
Robot 1: $u_1(t) = 0.6, K_1 = 10, K_2 = 1, A_j = 0.45, \sigma = 0.6$
Robot 2: $u_2(t)$=0.5, K_1=10, K_2=1.2 , A_j=0.4, σ=0.55
Robot 3: $u_3(t) = 0.3, K_1 = 15, K_2 = 2, A_j = 0.6, \sigma = 0.65$

TABLE 3.7: Response of Three Robot with speed = 0.3 and varying σ, $K_1 = 15, K_2 = 2$

$av(\|e_1\|)$, $std(\|e_1\|)$	$av(\|e_2\|)$, $std(\|e_2\|)$	$av(\|e_3\|)$, $std(\|e_3\|)$	**min** $dist_{1,2}$	**min** $dist_{1,3}$	**min** $dist_{2,3}$	A_j , σ
0.1790 , 0.1463	0.2083 , 0.1316	0.3645 , 0.2612	0.7286	0.6864	0.7092	0.4 , 0.4
0.1866 , 0.1374	0.2206 , 0.1335	0.3436 , 0.2514	0.7521	0.6986	0.7213	0.4 , 0.5
0.2021 , 0.1239	0.2285 , 0.1374	0.3344 , 0.2412	0.7552	0.6998	0.7338	0.4 , 0.6

TABLE 3.8: Response of Three Robot with speed $Robot_{1,2,3} = 0.3, 0.4, 0.5$, $A_j = 0.4$ and $\sigma = 0.55$

$av(\|e_1\|)$, $std(\|e_1\|)$	$av(\|e_2\|)$, $std(\|e_2\|)$	$av(\|e_3\|)$, $std(\|e_3\|)$	**min** $dist_{1,2}$	**min** $dist_{1,3}$	**min** $dist_{2,3}$	$Robot_{1,2,3}$ **Gains** K_1, K_2
0.2113 , 0.1207	0.2306 , 0.1453	0.3335 , 0.2387	0.7612	0.5642	0.7305	15 , 2 12 , 1.6 11 , 1.2

From the analysis of the results, it can be seen how the proposed approach results robust even in presence of moving osbtacles. Moreover, the following considerations can be done:

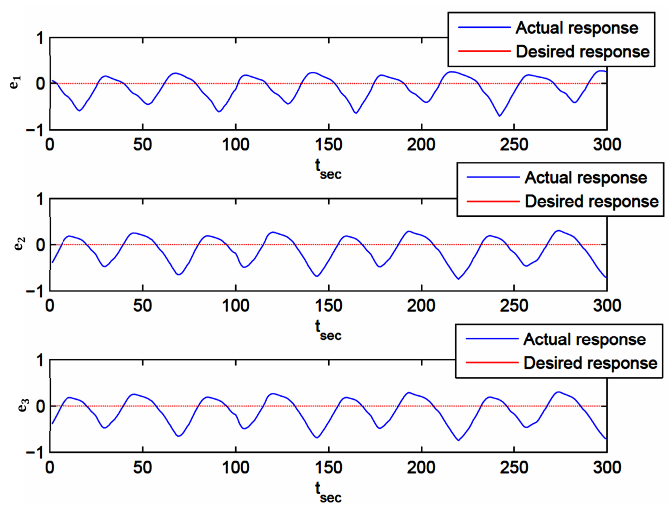

FIGURE 3.21: Plot of e_1, e_2 and e_3 versus time

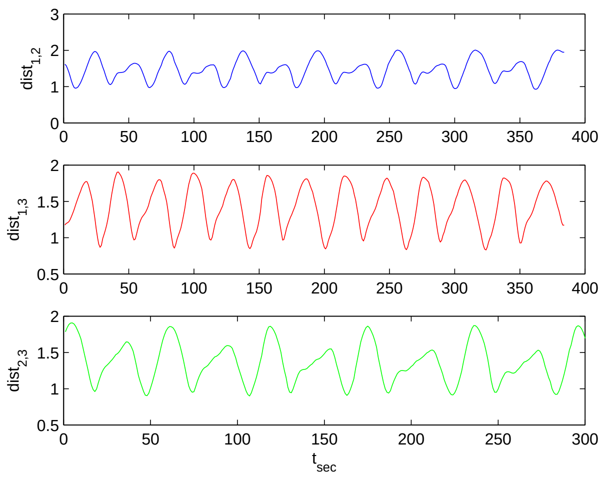

FIGURE 3.22: Plot of $dist_{1,2,3}$ versus time

- The parameters σ and A_j of the Gaussian (obstacle) function should be opportunely tuned in order to avoid the obstacles. If they are given a too small value, the robot could collide with obstacles.
- By increasing σ and A_j, the collision possibility between robot and obstacle decreases but the average error between the performed path and desired path increases. The appropriate value of σ and A_j has to be chosen in order to allow the robot to avoid the obstacle in an advantageous way.
- As robot speed increases, the appropriate gain tuning parameters must be selected in order to stabilize the robot response.

3.2.4.1 Sine wave Path

In this experiment, a sine wave path Eq. (3.13) has been considered as a reference. Figures 3.23 and 3.26 show the plot of two robots moving back and forth along a sinusoidal path. In the first experiment two intersecting paths are considered, which requires the robots (starting from $x_1 = 0, y_1 = 0$ and $x_2 = 0, y_2 = -1.2$) to avoid each other when they are in proximity of the intersection at the same time (similarly to what would happen to two cars approaching a crossroad). In the second experiment, the reference path is the same for the two robots, but they move in opposite directions (starting from $x_1 = 0, y_1 = 0$ and $x_2 = 1.5, y_2 = 0$), which requires them to avoid each other whenever they meet somewhere along the path (similarly to what would happen to two cars moving along the same road but in opposite directions). Figures 3.24, 3.25, 3.27, 3.28 show the errors e_1, e_2 and distance $dist_{1,2}$ between robots while they follow their path back and forth. The control parameters to properly follow the sinusoidal path, along with the results in terms of average error and minimum distance are shown in Table 3.9 and 3.10.

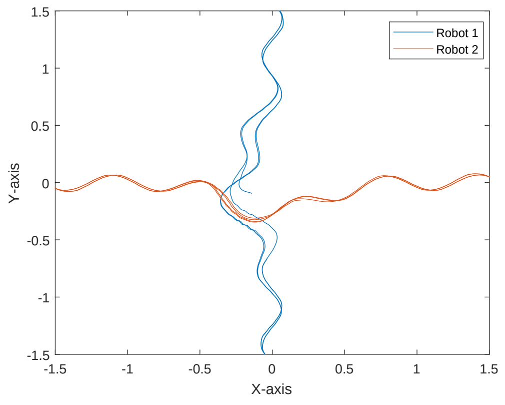

FIGURE 3.23: Two robots moving along two intersecting sine wave paths.
Robot 1: $u_1(t) = 0.3, K_1 = 35, K_2 = 5, A_j = 0.5, \sigma = 0.5$
Robot 2: $u_2(t) = 0.2, K_1 = 40, K_2 = 7, A_j = 0.5, \sigma = 0.5$

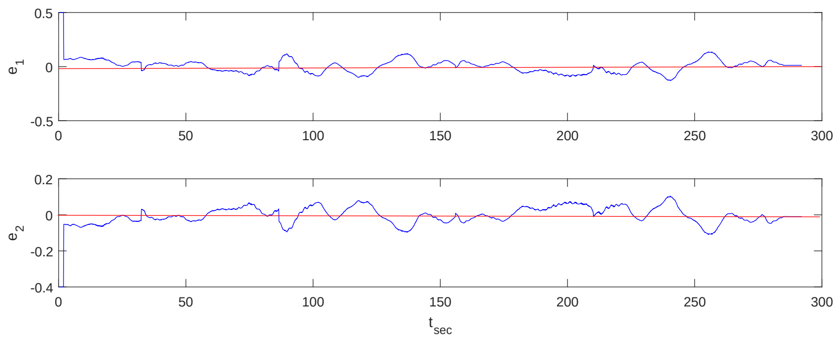

FIGURE 3.24: Plot of $e_{1,2}$ versus time

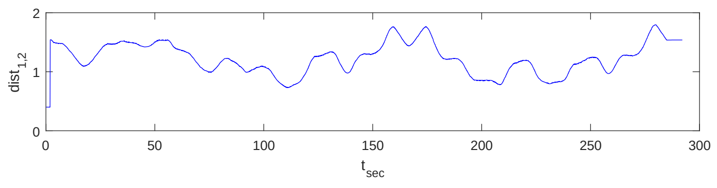

FIGURE 3.25: Plot of $dist_{1,2}$ versus time

TABLE 3.9: Response of Two Robot back and forth with same/different speed following sine wave path intersecting each other

$u_1(t), u_2(t)$ m/s	$av\ (\lvert e_1\rvert)$, $std\ (\lvert e_1\rvert)$	$av\ (\lvert e_2\rvert)$, $std\ (\lvert e_2\rvert)$	**min** $dist_{1,2}$	$Robot_1$ **Gains** K_1, K_2	$Robot_2$ **Gains** K_1, K_2	A_j , σ
0.1 , 0.1	0.076, 0.069	0.075, 0.071	0.707	45, 10	45, 10	0.5, 0.5
0.2 , 0.1	0.107, 0.099	0.081, 0.073	0.681	40, 7	45, 10	0.5, 0.5
0.3 , 0.2	0.121, 0.117	0.113, 0.109	0.669	35, 5	40, 7	0.5, 0.5

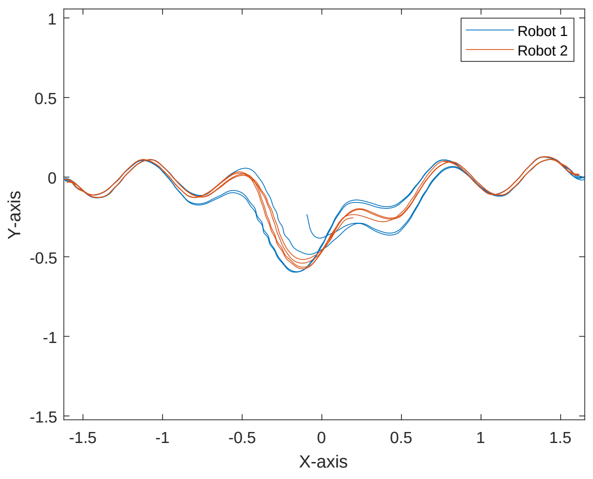

FIGURE 3.26: Two robots moving along the same sine wave path, but in opposite directions.
Robot 1: $u_1(t) = 0.1, K_1 = 45, K_2 = 10, A_j = 0.5, \sigma = 0.5$
Robot 2: $u_2(t) = 0.1, K_1 = 45, K_2 = 10, A_j = 0.5, \sigma = 0.5$

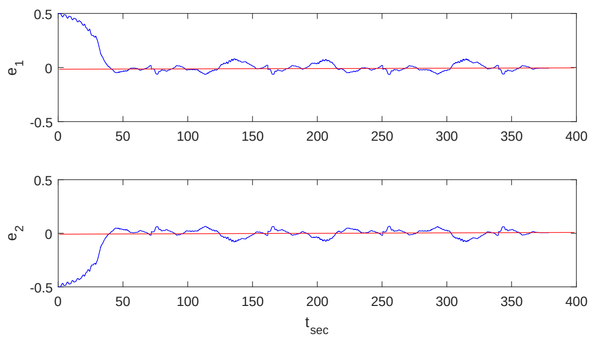

FIGURE 3.27: Plot of $e_{1,2}$ versus time

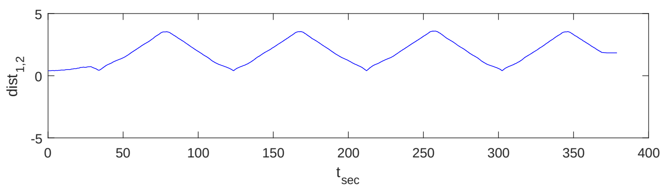

FIGURE 3.28: Plot of $dist_{1,2}$ versus time

TABLE 3.10: Response of Two Robot back and forth with same/different speed following sine wave path parallel to each other in opposite direction

$u_1(t), u_2(t)$ m/s	av $(\lvert e_1 \rvert)$, std $(\lvert e_1 \rvert)$	av $(\lvert e_2 \rvert)$, std $(\lvert e_2 \rvert)$	**min** $dist_{1,2}$	$Robot_1$ **Gains** K_1, K_2	$Robot_2$ **Gains** K_1, K_2	A_j , σ
0.1 , 0.1	0.102, 0.097	0.111, 0.102	0.648	45, 10	45, 10	0.5, 0.5
0.2 , 0.1	0.148, 0.133	0.107, 0.101	0.532	40, 7	45, 10	0.5, 0.5
0.3 , 0.2	0.173, 0.166	0.153, 0.132	0.439	35, 5	40, 7	0.5, 0.5

3.2.4.2 Straight Line Path

This experiment is almost identical to the previous one, with the only difference that a straight line Eq.(3.11) has been considered as a reference: in the first case (Figure 3.29), two intersecting lines are considered (the robots start from $x_1 = 0, y_1 = 0$ and $x_2 = 0, y_2 = -1.2$); in the second case (Figure 3.32), the two robots move along the same line but in opposite directions (starting from $x_1 = 0, y_1 = -1.2$ and $x_2 = 0, y_2 = 1.2$); in the third case (Figure 3.35), the two robots move along the same line but in opposite direction and with a different speed (starting from $x_1 = -1.2, y_1 = 0$ and $x_2 = 1.2, y_2 = 0$). Figures 3.30, 3.31, 3.32, 3.33, 3.36, 3.37 show the errors e_1, e_2 and distance $dist_{1,2}$ between robots while they follow their path back and forth. The control parameters to properly follow the sinusoidal path, along with the results in terms of average error and minimum distance are shown in Table 3.11 and 3.12.

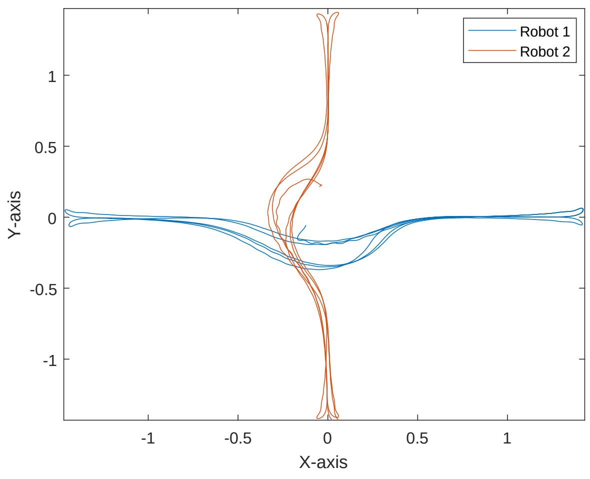

FIGURE 3.29: Two robots moving along two intersecting straight paths.
Robot 1: $u_1(t) = 0.1, K_1 = 25, K_2 = 6, A_j = 0.5, \sigma = 0.5$
Robot 2: $u_2(t) = 0.1, K_1 = 25, K_2 = 6, A_j = 0.5, \sigma = 0.5$

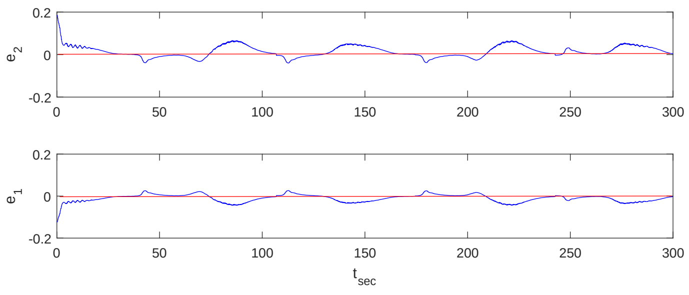

FIGURE 3.30: Plot of $e_{1,2}$ versus time

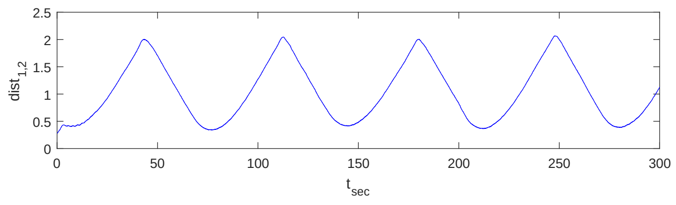

FIGURE 3.31: Plot of $dist_{1,2}$ versus time

TABLE 3.11: Response of Two Robot back and forth with same/different speed following straight line path intersecting each other

$u_1(t), u_2(t)$ m/s	av $(\|e_1\|)$, std $(\|e_1\|)$	av $(\|e_2\|)$, std $(\|e_2\|)$	**min** $dist_{1,2}$	$Robot_1$ **Gains** K_1, K_2	$Robot_2$ **Gains** K_1, K_2	A_j , σ
0.1 , 0.1	0.097, 0.089	0.089, 0.082	0.567	45, 10	45, 10	0.5, 0.5
0.2 , 0.1	0.125, 0.119	0.131, 0.122	0.452	40, 7	45, 10	0.5, 0.5
0.3 , 0.2	0.170, 0.153	0.159, 0.148	0.439	35, 5	40, 7	0.5, 0.5

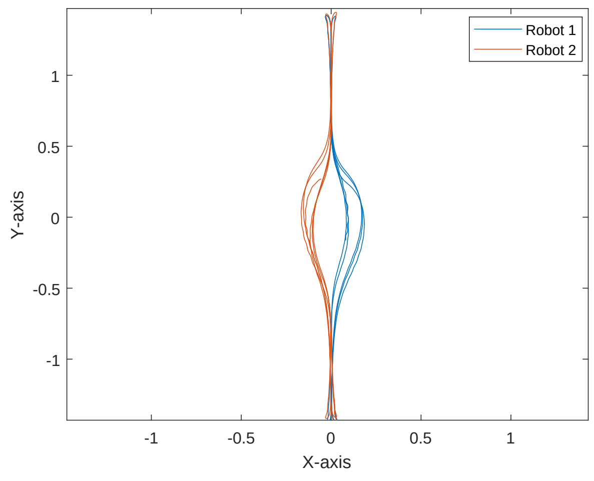

FIGURE 3.32: Two robots moving along the same straight path, but in opposite directions.
Robot 1: $u_1(t) = 0.1, K_1 = 25, K_2 = 6, A_j = 0.5, \sigma = 0.5$
Robot 2: $u_2(t) = 0.1, K_1 = 25, K_2 = 6, A_j = 0.5, \sigma = 0.5$

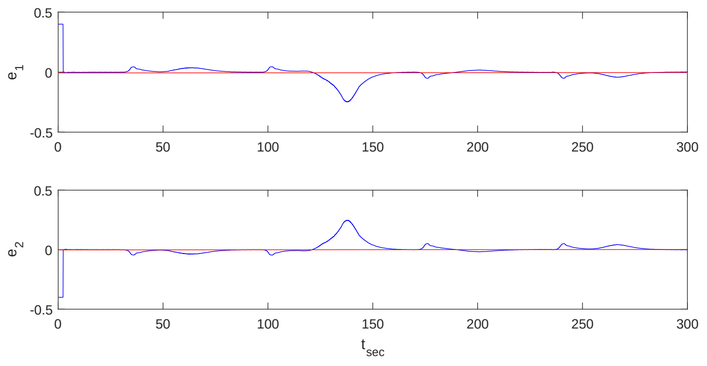

FIGURE 3.33: Plot of $e_{1,2}$ versus time

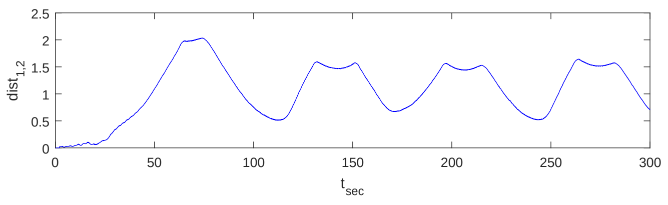

FIGURE 3.34: Plot of $dist_{1,2}$ versus time

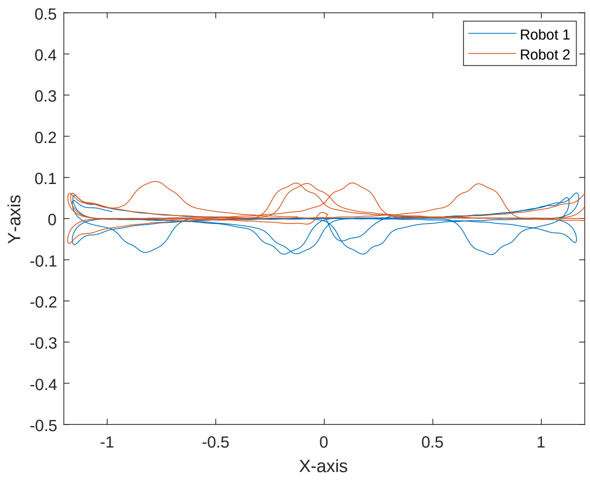

FIGURE 3.35: Two robots moving along the same straight path, but in opposite directions.
Robot 1: $u_1(t) = 0.3, K_1 = 35, K_2 = 5, A_j = 0.5, \sigma = 0.5$
Robot 2: $u_2(t) = 0.2, K_1 = 40, K_2 = 7, A_j = 0.5, \sigma = 0.5$

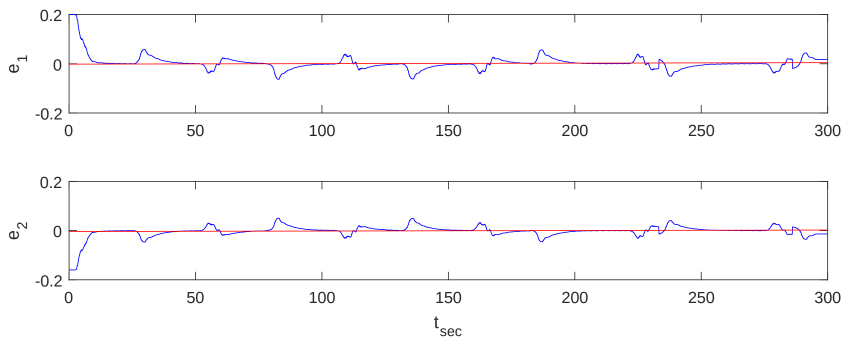

FIGURE 3.36: Plot of $e_{1,2}$ versus time

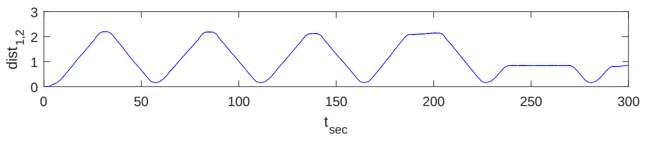

FIGURE 3.37: Plot of $dist_{1,2}$ versus time

TABLE 3.12: Response of Two Robot back and forth with same/different speed following straight line path parallel to each other

$u_1(t), u_2(t)$ m/s	av $(\lvert e_1\rvert)$, std $(\lvert e_1\rvert)$	av $(\lvert e_2\rvert)$, std $(\lvert e_2\rvert)$	**min** $dist_{1,2}$	$Robot_1$ **Gains** K_1, K_2	$Robot_2$ **Gains** K_1, K_2	A_j , σ
0.1 , 0.1	0.086, 0.077	0.079, 0.068	0.492	45, 10	45, 10	0.5, 0.5
0.2 , 0.1	0.122, 0.116	0.092, 0.081	0.337	40, 7	45, 10	0.5, 0.5
0.3 , 0.2	0.127, 0.123	0.119, 0.115	0.266	35, 5	40, 7	0.5, 0.5

3.2.5 Scenario 4: With Persons

In this scenario, one or more robots move along a circular path at constant speed in the motion capture area, but this time concurrently with operators (people/persons) walking in the same area: the robots are obstructed in following the path by the presence of walking persons, as shown in Figure 3.38. While the persons walk in a random way, the robots have been given a speed of 0.3m/s, and the gains are set according to the analysis performed in the previous experiments. Indeed, persons have been considered in the experiments because their motion is less predictable (differently from the moving robots case) and they give the possibility of increasing the number of moving obstacles without adding other autonomous robots.

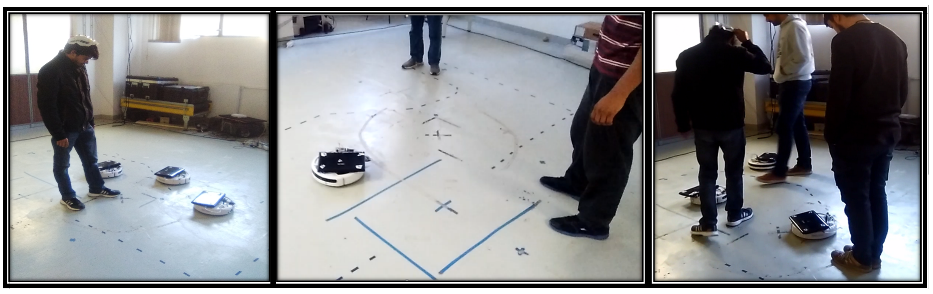

FIGURE 3.38: Robot with Persons

In the first case study of this scenario, the response of one robot with one walking person is taken into account. The actual pose of the robot in presence of one person is shown in Figure 3.39. The distance between robot and person is shown in Figure 3.40. The path followed by the robot is in this case much more subject to disturbances (with respect of scenario 3), because the random interference of the walking person tends to be more frequent with respect of the multi-robot case (Figure 3.41).

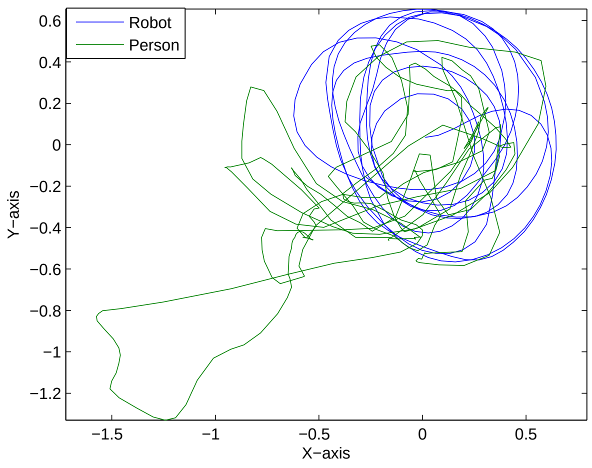

FIGURE 3.39: 1 Robot with 1 Person

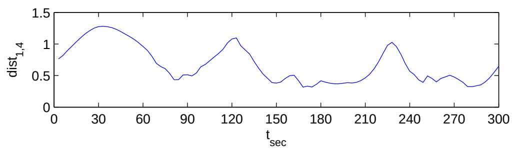

FIGURE 3.40: Plot of $dist_{1,4}$ versus time

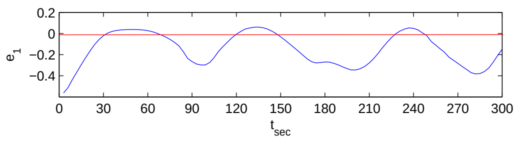

FIGURE 3.41: Plot of e_1 versus time

In the next case, a robot and two persons are taken into account. The path followed by the robot and the two persons is shown in Figure 3.42. As in the previous case, the distance between robot and the two persons and the error between the path followed by the robot and the desired path are reported in Table 3.13.

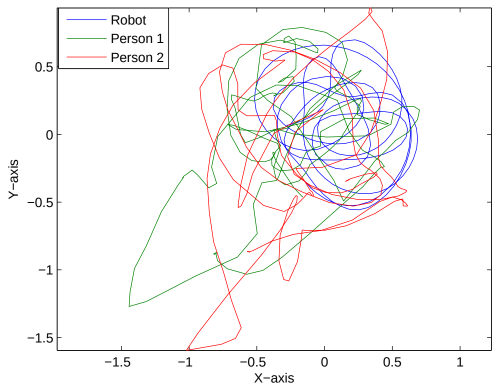

FIGURE 3.42: 1 Robot with 2 Persons

The same experiment has been performed 3 times for each case study (1, 2 and 3 persons). All results, averaged along the three runs, are presented in Table 3.13. The analysis of the values of $av(|e_1|)$ and $std(|e_1|)$ shows that the robots diverges from their path while persons are crossing it. The minimum distance of the robot from each person is reported in the table.

TABLE 3.13: One Robot with speed = $0.3m/s$, $K_1 = 15$, $K_2 = 2$, $A_j = 0.4$, $\sigma = 0.45$

No. of Persons	$av(\lvert e_1\rvert)$, $std(\lvert e_1\rvert)$	**min** $dist_{1,4}$	**min** $dist_{1,5}$	**min** $dist_{1,6}$	av (min_{dist})
1	0.092, 0.067	0.457	—	—	0.457
2	0.142, 0.101	0.437	0.418	—	0.427
3	0.266, 0.198	0.380	0.551	0.274	0.402

Similar tests have been performed by adding more robots (totally 2 and 3) moving along predetermined circular paths with 1, 2 and 3 persons were walking randomly in the same area. Figs. 3.43 - 3.44 and Tables 3.14 and 3.15 show the obtained results. As usual, $dist_{i,j}$ is the distance between robot (or person) i and robot (or persons j), while $av(|e_i|)$ and $std(|e_i|)$ are the average error and the standard deviation between the predefined path and the actual path of robot i. As before, $av(|e|)$ and $std(|e|)$ are averaged along the three runs.

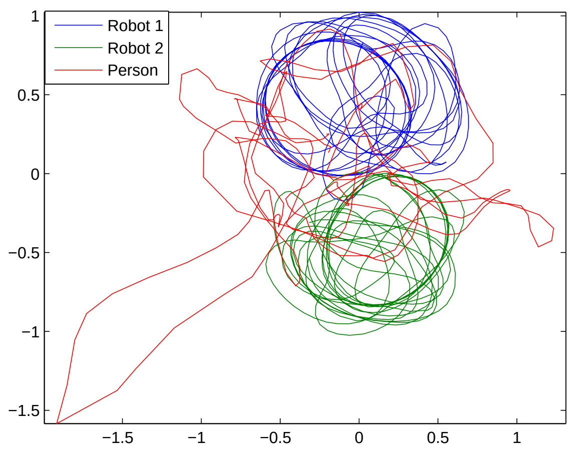

FIGURE 3.43: 2 Robots with 1 Person

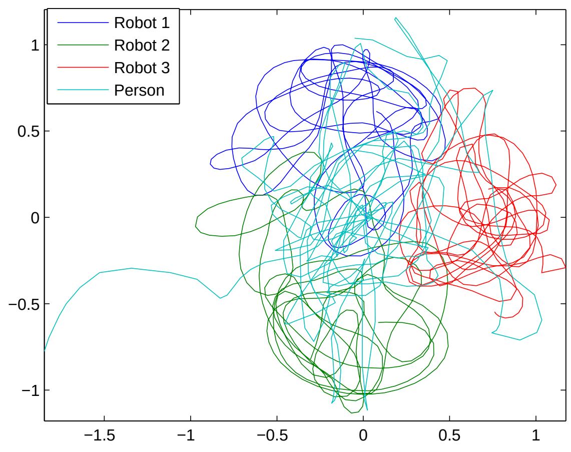

FIGURE 3.44: 3 Robots with 1 Person

TABLE 3.14: Two Robots with speed = $0.3m/s$, $K_1 = 15$, $K_2 = 2$, $A_j = 0.4$, $\sigma = 0.45$

No. of Persons	$av(\|e_1\|)$, $std(\|e_1\|)$	$av(\|e_2\|)$, $std(\|e_2\|)$	$av(\|e\|)$	**min** $dist_{1,2}$	**min** $dist_{1,4}$	**min** $dist_{2,4}$	**min** $dist_{1,5}$	**min** $dist_{2,5}$	**min** $dist_{1,6}$	**min** $dist_{2,6}$	av (min_{dist})
1	0.160, 0.119	0.219, 0.137	0.189	0.397	0.452	0.413	—	—	—	—	0.421
2	0.181, 0.124	0.242, 0.155	0.211	0.297	0.391	0.431	0.417	0.521	—	—	0.411
3	0.211, 0.156	0.294, 0.196	0.253	0.503	0.427	0.419	0.321	0.403	0.381	0.410	0.409

TABLE 3.15: Three Robots with speed = $0.3m/s$, $K_1 = 15$, $K_2 = 2$, $A_j = 0.4$, $\sigma = 0.45$

No. of Persons	$av(\lvert e_1\rvert)$, $std(\lvert e_1\rvert)$	$av(\lvert e_2\rvert)$, $std(\lvert e_2\rvert)$	$av(\lvert e_3\rvert)$, $std(\lvert e_3\rvert)$	$av(\lvert e\rvert)$	**min** $dist_{1,2}$, **min** $dist_{1,3}$, **min** $dist_{2,3}$	**min** $dist_{1,4}$, **min** $dist_{2,4}$, **min** $dist_{3,4}$	**min** $dist_{1,5}$, **min** $dist_{2,5}$, **min** $dist_{3,5}$	min $dist_{1,6}$, min $dist_{2,6}$, min $dist_{3,6}$	av (min_{dist})
1	0.166, 0.104	0.336, 0.215	0.121, 0.087	0.208	0.595, 0.521, 0.505	0.404, 0.391, 0.457	—	—	0.479
2	0.237, 0.153	0.214, 0.149	0.209, 0.177	0.220	0.472, 0.495, 0.517	0.532, 0.438, 0.427	0.507, 0.465, 0.323	—	0.464
3	0.118, 0.072	0.298, 0.116	0.411, 0.192	0.276	0.432, 0.497, 0.322	0.288, 0.216, 0.411	0.439, 0.337, 0.355	0.468, 0.506, 0.444	0.392

Chapter 4

Multi-robot cooperation

In this chapter, we implemented a system for multi-robot cooperation that incudes heterogeneous robots; a humanoid robot helps mobile robots to localize themselves.

4.1 Materials and Methods

4.1.1 Problem Definition

The problem of multi-robot collaborative localization to the end of mutual avoidance is presented using two robots, but the approach can be trivially be extended to multiple robots.

Consider two Roombas that follow a path using odometry with increasing errors, andassume that their position is also monitored using IPM images acquired through Pepper's head camera: to reduce positioning errors, the observations from Pepper IPM view can be merged with odometry to correct the Roombas estimated position using and Extended Kalman Filter (EKF).

We briefly recall the basic EKF principles. Let x_k denote the robot state at any time instant: for example, in mobile robot navigation, x_k denotes the robot pose. The observation obtained at time k will be denoted by z_k. The robot motion model is given by:

$$x_{k+1} = f(x_k, u_k) + w_k \ , \ w_k \sim \mathcal{N}(0, R_k) \tag{4.1}$$

where w_k and R_k are the zero mean Gaussian noise and process covariance respectively. We also assume a general observation model with zero mean Gaussian noise,

$$z_k = h(x_k) + v_k \ , \ v_k \sim \mathcal{N}(0, Q_k) \tag{4.2}$$

The function $h(x)$, in the case of a camera, may be defined as the operator that maps image plane coordinates onto world frame coordinates, discussed in Section 4.1.2.

The workflow of the process required to determine an object position using a moving camera (i.e., due to changes in Pepper's head orientation) is summarized in Figure 4.2 and includes two elements, an Acquisition and IPM Phase on the left (Tracking, Image Acquisition and IPM transform, BGR to HSV conversion) and a Detection Phase (ROI selection and Remapping). In the Figure, notice also that robot tracking may require to update the Yaw and Pitch of Pepper's head, which has an impact on the following process for world frame coordinate estimation. These aspects are detailed in the remainder of this section.

4.1.2 Inverse Perspective Mapping (IPM)

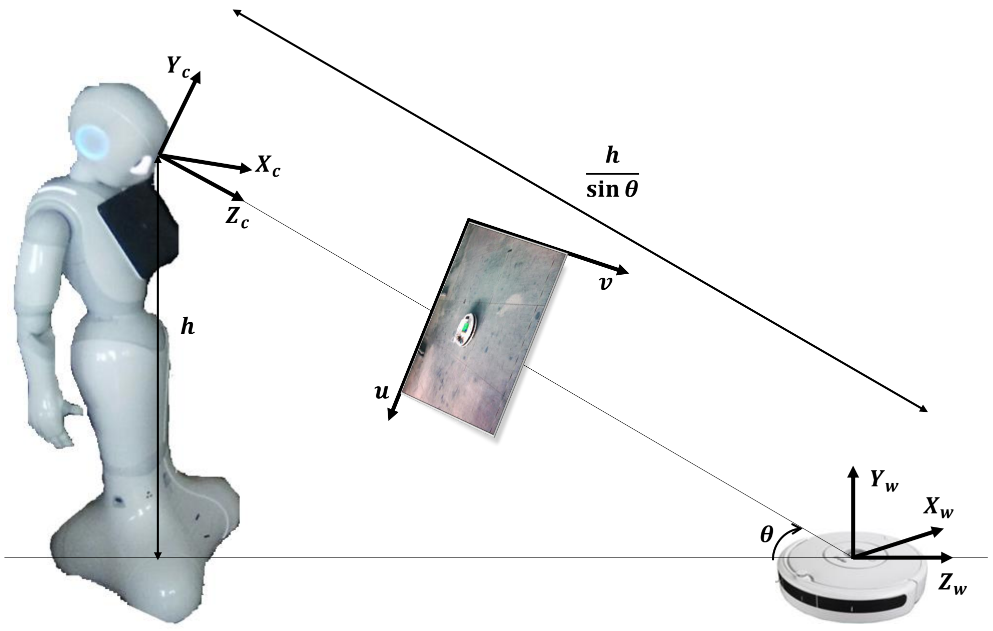

FIGURE 4.1: Image coordinate system in relation to world coordinate system. (X_c, Y_c, Z_c) and (X_w, Y_w, Z_w) denotes the camera and world frame respectively, (u, v) is the image plane, h is the height of the Pepper's camera from the ground and θ is the mapping angle of a point on the world frame to its projection on the image plane.

Mathematically, IPM [134] produce a mapping from object coordinates on the image plane (u, v) to world frame coordinates (X_w, Y_w, Z_w), as shown in Figure 4.1.

To this end, IPM tranformation requires the knowledge of intrinsic camera parameters as well as the position and attitude of the camera frame (X_c, Y_c, Z_c) with respect to world frame (X_w, Y_w, Z_w), complemented with additional assumptions about how the image is presented – also referred to as a priori knowledge [135] [136]. Therefore, the IPM transform is usually adopted in a structured environment in which, for example, the camera is placed in a static position or in situations where the calibration can be automatically obtained from another sensor [137] [138] [139]. This is different from our scenario, in which Pepper's head is moving dynamically in order to track robots moving in the plane.

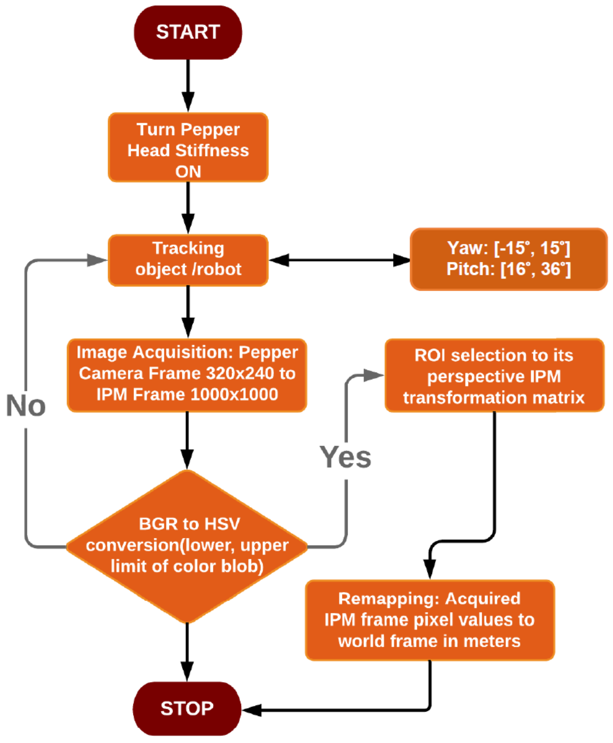
START
Turn Pepper Head Stiffness ON
Tracking object /robot
Yaw: [-15°, 15°] Pitch: [16°, 36°]
Image Acquisition: Pepper Camera Frame 320x240 to IPM Frame 1000x1000
ROI selection to its perspective IPM transformation matrix
No
Yes
BGR to HSV conversion(lower, upper limit of color blob)
Remapping: Acquired IPM frame pixel values to world frame in meters
STOP

FIGURE 4.2: An overview of object position detection during Pepper's head movement. When an object is not detected in the IPM frame, the NO branch is followed, requiring to track the object; when the object is detected, the YES branck is followed, finally leading to world frame coordinates computation.

Through IPM transform, the image acquired by the camera is projected onto a horizontal plane, with the final result that an image similar to a bird eye view of the environment is obtained. The method eliminates the non-linearity of relative distances between objects in the image plane, by mapping pixels in the original image onto world coordinate points. Below, we report the IPM equations, by reminding that the mapping depends on the rotation along camera optical axis (R), the translation along the camera optical axis (T), and a scaling by the camera parameter matrix (K) [140]. This can be expressed mathematically as follows:

$$[u, v, 1]^T = KTR[x, y, z, 1]^T \tag{4.3}$$

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\dfrac{h}{\sin\theta} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

$$K = \begin{bmatrix} f \times k_u & s & u_0 & 0 \\ 0 & f \times k_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.6}$$

where h is the height of camera on Pepper's head, f correspond to the focal length measured in horizontal and vertical pixel units k_u and k_v respectively. The position (u_0, v_0) is the camera frame point that fixes the image plane. We rewrite Eq. 4.3 as:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \end{bmatrix} \begin{bmatrix} X_L \\ Y_L \\ Z_L \\ 1 \end{bmatrix} \tag{4.7}$$

where the 3×4 matrix $M = KTR$ relates linear IPM coordinates (X_L, Y_L, Z_L) to the image plane (u_i, v_i). It is to be noted that a change in Yaw will not affect $M_{i,j}$ elements of the matrix, since the projection angle θ remains the same in (4.5)(4.6). On the opposite, a variation in Pitch implicitly influences $M_{i,j}$ due to change in h and therefore θ, Figure 4.1.

By ignoring the Z_L coordinate and inverting (4.7) we obtain:

$$\begin{bmatrix} X_L \\ Y_L \\ 1 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{14} \\ M_{21} & M_{22} & M_{24} \\ M_{31} & M_{32} & M_{34} \end{bmatrix}^{-1} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \tag{4.8}$$

which correspond to a bird eye view of the environment, see Figure 4.10. Finally, real world coordinates (X_w, Y_w, Z_w) are obtained from (X_L, Y_L) through a scaling factor as detailed in the next section.

4.1.3 Detection Phase

For object detection and segmentation in the IPM view (X_L, Y_L) obtained through (4.8), there are conventional techniques that need prior knowledge of the object [141]. However, we use a simpler and more effective approach based on Surface Subtraction (SS)[142]. The Roombas are uniquely distinguishable by putting different color markers on top of them (see Figure 1.2): the SS algorithm detects a Roomba by selecting the nearest part of the color blob in the IPM bird eye view (see Figure 4.10).

Algorithm 1 below summarizes the whole process. Once the color blob is detected, it is then converted to its corresponding RGB constituent (line 2). A pixel rectangle is then retrieved (line 3), thereby retrieving the pixels that match the required color and are closer to Pepper (line 4). The pixel coordinates are then mapped into the world frame and a low-pass filter is applied to reduce errors over subsequent acquisitions (line 5), thus estimating the Roomba's position in the real world.

Algorithm 1 Robot tracking using IPM

```
Require: IPM image, color blob, RGB values
1: Detect color blob
2: RGB constituent ← color blob
3: Pixel rectangle ← RGB constituent
4: Recover pixel coordinates of the edge center (rectangle in step 3) closest to Pepper

5: Scaled coordinates ← pixel coordinates
6: return Scaled coordinates
```

It is to be noted that the level of brightness and surface conditions effects the accuracy of camera based techniques, often rendering false negatives. To circumvent this issue, an OpenCV morphological tool is used to detect a particular color under different light conditions by manually selecting the RGB channel range for that color. Therefore, under different lighting conditions, the result will remain similar in the presence of a color due to the upper and lower limits of the RGB morphological tool, as discussed. This helps to identify potential objects by recognizing their colors in varying sizes under various lighting conditions.

It is to be noted that the level of brightness and surface conditions affects the accuracy of vision-based techniques, often returning false negatives. To circumvent this issue, an OpenCV morphological tool is used to detect a particular color under different light conditions by manually selecting the RGB channel ranges for that color. Therefore, under different lighting conditions, the result will remain similar during blob detection, given that RGB values are within the upper and lower limits of the RGB morphological tool. This helps to identify potential objects by recognizing their colors, even if their size varies and under different lighting conditions.

Notice also that, in order to compute the object's position in the world frame (X_W, Y_W) (line 5), it is necessary to compute a scaling factor from IPM coordinates (X_L, Y_L) to world frame coordinates (X_W, Y_W). To this end, there are different calibration methods, such as [143][144][145][146].

Once the object position has been estimated in the world frame, we use the EKF to merge such estimate with the estimate provided by the Roomba odometry to reduce the positioning errors. Using the standard equation of EKF, the predicted state $\hat{x}_{k+1}^{-}$ (i.e., the odometry) at interval $k+1$ can be predicted as:

$$\begin{aligned}\hat{x}_{k+1}^{-} &= f(\hat{x}_k, u_k)\\ \bar{\Sigma}_{k+1} &= F_k \Sigma_k F_k^T + R_k\end{aligned} \tag{4.9}$$

where $\hat{x}_k$ is the state estimate, Σ_k is the covariance and F_k is the Jacobian of $f(\cdot)$ with respect to x_k. Upon receiving a measurement from IPM view, i.e., z_k, the updated position of Roomba is given by:

$$\begin{aligned}K_k &= \bar{\Sigma}_{k+1} H_k^T (H_k \bar{\Sigma}_{k+1} H_k^T + Q_k)^{-1}\\ \hat{x}_{k+1} &= \hat{x}_{k+1}^{-} + K_k (z_{k+1} - h(\hat{x}_{k+1}^{-}))\\ \Sigma_{k+1} &= (I - K_k H_k) \bar{\Sigma}_{k+1}\end{aligned} \tag{4.10}$$

where H_k is the Jacobian of $h(\cdot)$ with respect to x, K_k is the Kalman gain and $I \in \mathbb{R}^{3\times 3}$.

4.2 Results and Discussion

The algorithm presented here was first investigated with a static camera and then implemented to Pepper in real time. For this purpose, the Python language, complemented by the OpenCV library of functions was chosen for implementation that allows the rapid development of image processing algorithms without the need to develop complex algorithms by using its open source library of functions [147].

4.2.1 Static Pepper Head with moving robot in plane

With Pepper robot, a database of video samples was captured at 30 frames per second (fps), at a resolution of 320×240 pixels. Considering that the change in distance of potential objects is relatively slow compared to the full frame rate of the system, a slower frame rate was considered for the purposes of calculating distance and computing position. Using a sampling rate of every 10 frames produced smooth and reliable results. A rate of 10 frames per second was chosen as it provides a good trade-off between computation time and number of calculations per second.

A comparative analysis has been shown in this section. The first experiment is being conducted with Pepper RedBall API. The Redball API works by considering the size in pixels of the red ball, and comparing it to the known size of the red ball to be detected. Obviously this works well only if the red ball is directly in front of the camera.

A Red ball is being placed at a static point in the FOV of Pepper Head camera. First it is placed at $0.7m$ on X_{axis} and $-0.22m$ on Y_{axis}, then the data have been recorded for 300 samples. The same acquisition has been performed in other three different points, as shown in Figure 4.3. It can be seen, how due to a continuous transient response, it is impossible to attain the position or determine distance of red ball.

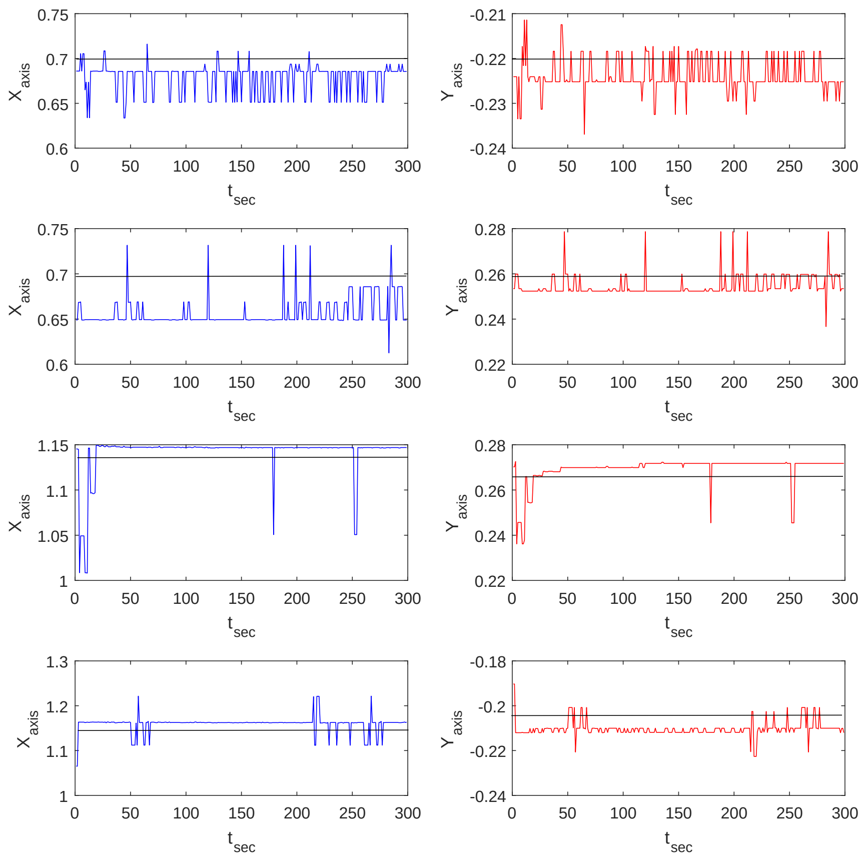

FIGURE 4.3: Pepper RedBall API response at static points

A second test has been conducted: in this case the red ball has been placed on board the wheeled robot, while it is performing a circular path as done in [1], shown in Figure 4.4. The average error in this case is $av|e| = 0.43m$.

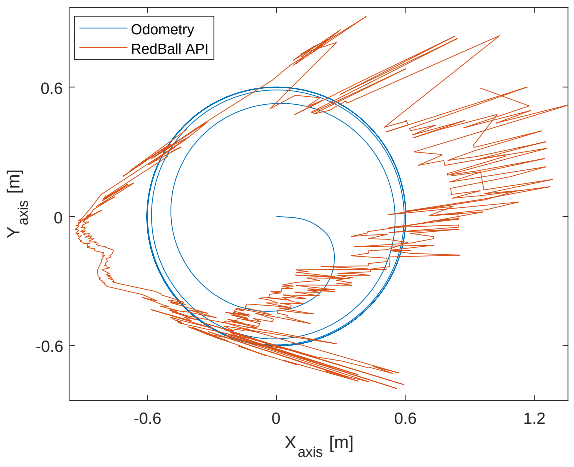

FIGURE 4.4: Pepper Redball API response while tracking a circular response

It is clear with the anaylsis that the API are not enough to determine the distance of objects in Pepper camera view. However, in order to determine the distance between the Pepper and Roomba in its FOV, the first step is to determine where the Roomba lies in the plane with respect to Pepper IPM view. So, Figure 4.5 shows the Pepper camera view and IPM view respectively.

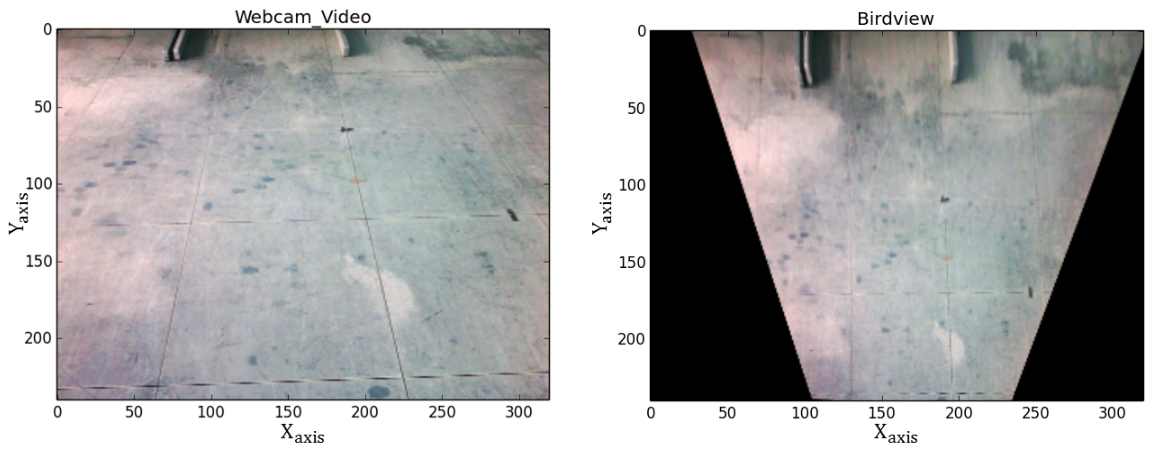

FIGURE 4.5: Pepper camera view and transformed IPM view

The same test have been performed, in this case a green box has been placed on top of the wheeled robot Roomba as a potential object. The green box has the same size as Red ball used in RedBall API experiment. As per discussed in methodology section, the use of a Pepper Head camera does not directly provide depth information in a scene so surface subtraction and IPM for getting a bird's-view image has been applied Figure4.6.

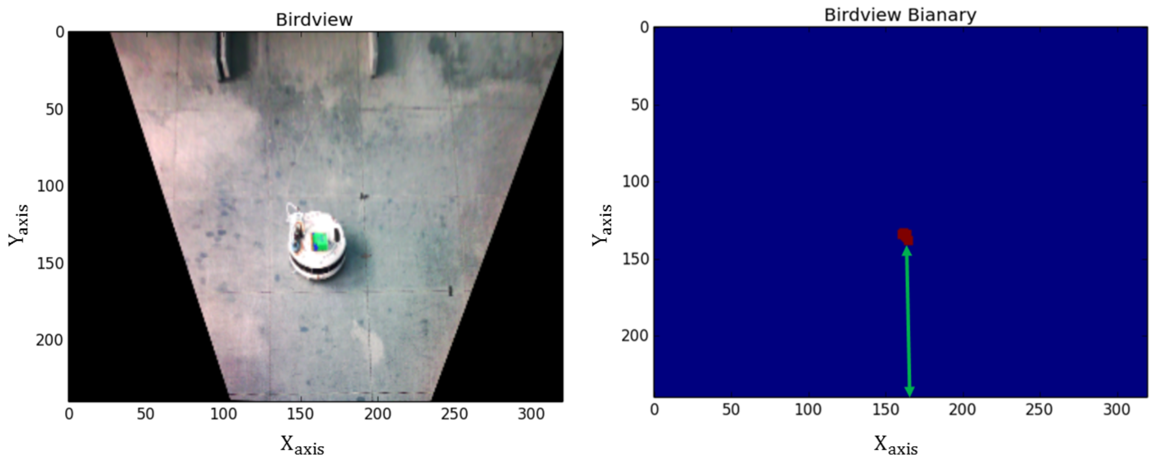

FIGURE 4.6: Distant object as shown by Pepper Top-Down view, Top Down view, distance in this image may be measureable

It can be seen from the green arrow in rectified bird's-view image, Figure 4.6, that the relationship between the potential object and its distance from the camera is linear in nature. However, the change in the length of the green arrow will proportionally reflect this difference in distance. In order to determine the position of potential object in bird eye view, the calibration with different distance on pixels are considered and sum up in a formula as in Eq. 4.11. We first calibrate the world points in Bird eye view frame with a known distance of object. Then the object is being placed at some distance from Pepper camera and by measuring at the same time the pixel (in bird's-view) corresponding to the object detection. The apparent pixels is measured corresponding to the detected object.

$$i_{n+1} = o + p_n + i_n + noise \tag{4.11}$$

where i is frame point, o is the offset, p is the pixel. The calibration result captured in Pepper FOV at different distances is shown in Figure 4.7.

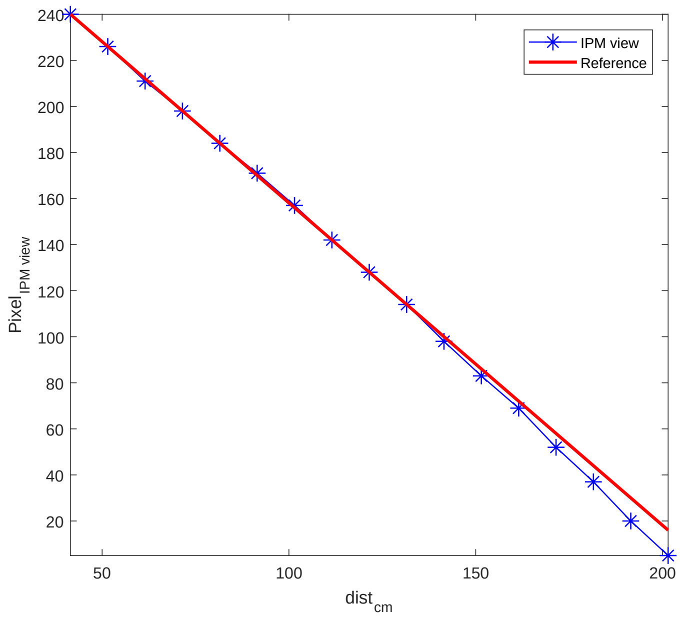

FIGURE 4.7: IPM view pixel values in different distances

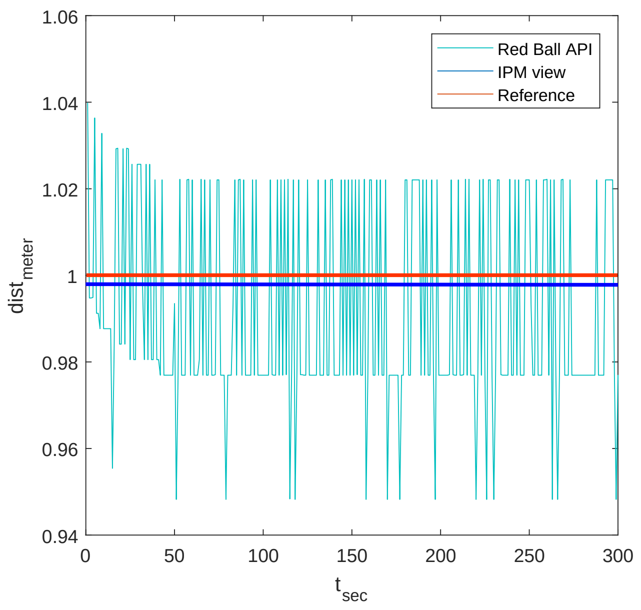

FIGURE 4.8: Comparison of IPM view and Redball API at static point

The static point comparison between Pepper Redball API and IPM view is shown in Figure 4.8.

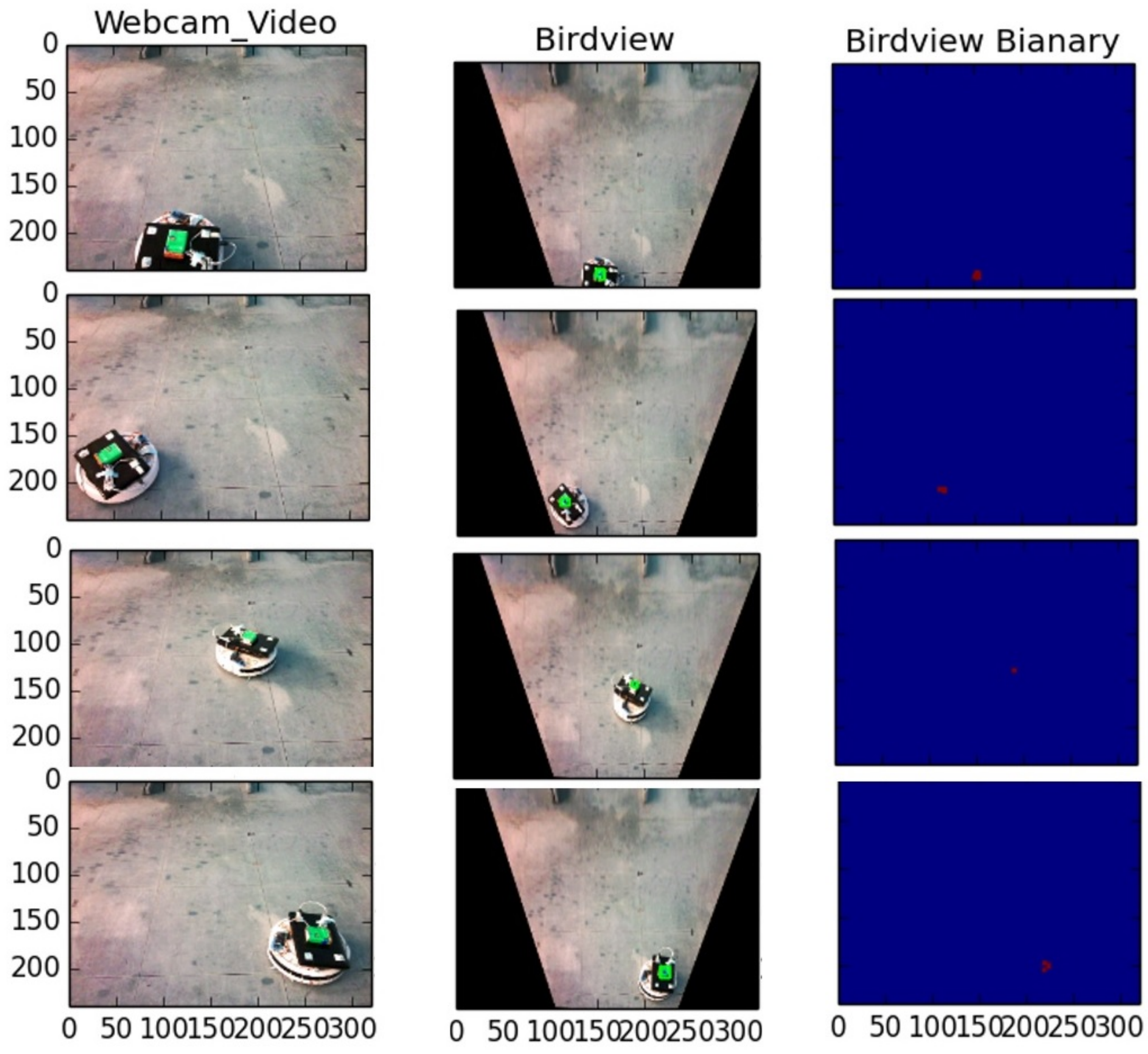

FIGURE 4.9: Distant and near moving robot in Pepper camera view, IPM view and Bianary view

Figure 4.9, shows the Pepper camera and IPM view to clarifies that while the robot is moving in a circular path, the roomba position is being calculated depending upon the potential objected tracked. So, it implies that the smaller the value of pixel the farther the distance.

The odometry of Roomba while following a circular path and Pepper IPM view response is presented to determine the distance and position, with an average error $av|e| = 0.086m$. It validates the IPM method (for static camera and moving objects on the floor). The successful comparison between RedBall and IPM view is shown in this section and obtained from a video samples of Pepper camera in real time operation. The experiments performed confirms that the implementation of an IPM approach with surface subtraction gives better results in comparison with the usage of standard RedBall API.

4.2.2 Moving Pepper Head with moving robot(s) in plane

To obtain the IPM bird eye view of the acquired image according to (4.8), we first utilize the *cv2.getPerspectiveTransform* function to compute the transformation Matrix M. This is done offline prior to operations, and requires two arguments, namely, the list of 4 ROI points in the original image and the corresponding points in the desired IPM frame. Notice that a different transformation matrix $M(\theta)$ needs to be computed for different θ values in (4.4)(4.5): this is done by quantisizing $\hat{\theta} \in \{36, 31, 26, 21, 16\}$, and executing *cv2.getPerspective-Transform* a number of times to produce a set of matrices $M(\hat{\theta})$. In run-time, this allows us to apply the right transformation matrix $M(\hat{\theta})$ depending on the current Pitch using the *cv2.warpPerspective* function to obtain the IPM bird eye view[1].

[1]https://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/

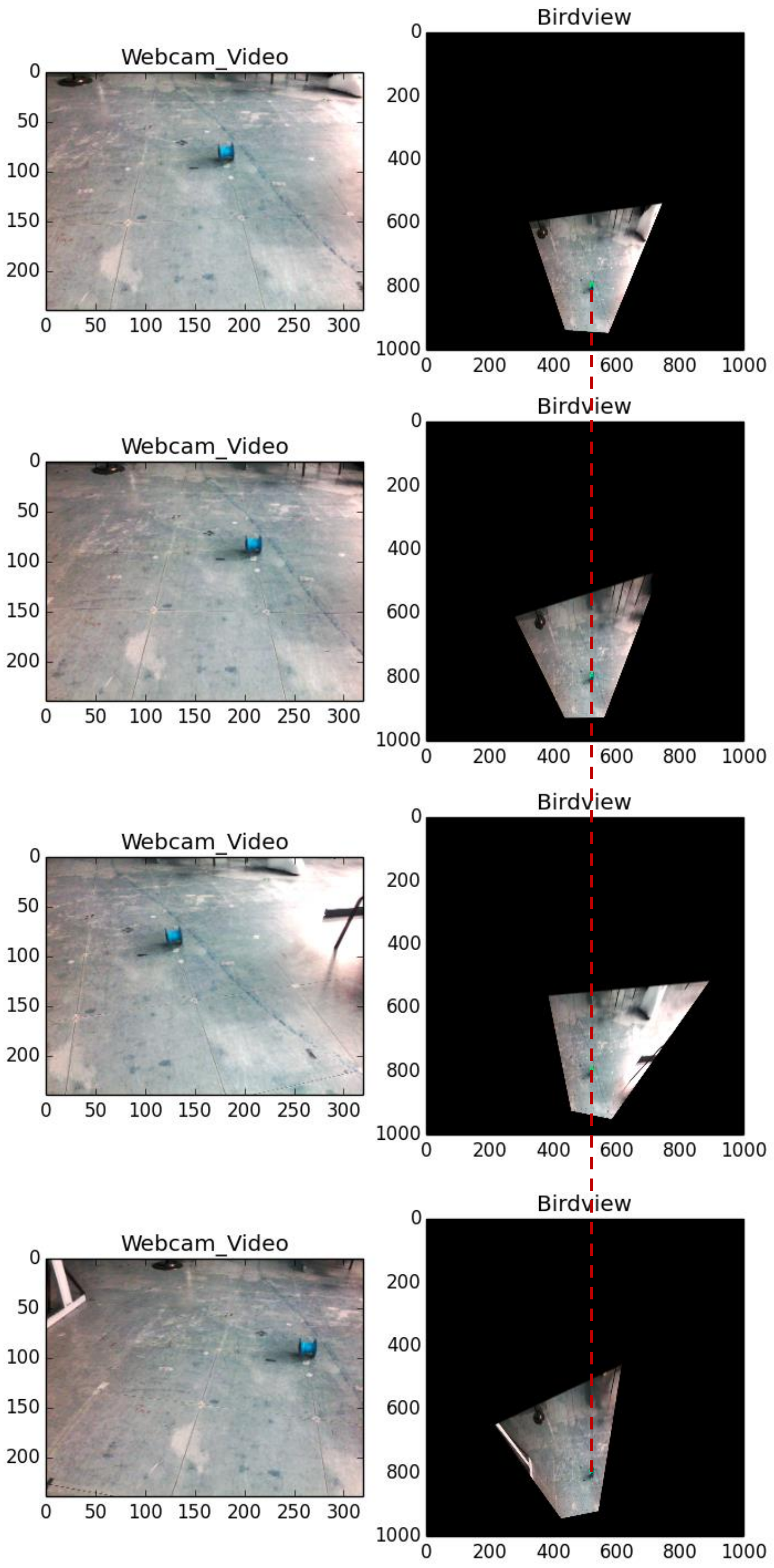

FIGURE 4.10: Fixed object IPM view with different Pepper Head Orientation (i.e: (yaw,pitch) = (36,0),(36,-5),(36,-15),(36,20)), left and right column shows the Pepper camera view the IPM views respectively.

The same acquisition has been performed with objects in different positions. See Figure

4.11: the average $av(|e|)$ and standard deviation $std(|e|)$ of the error e between the actual and the estimated position are calculated.

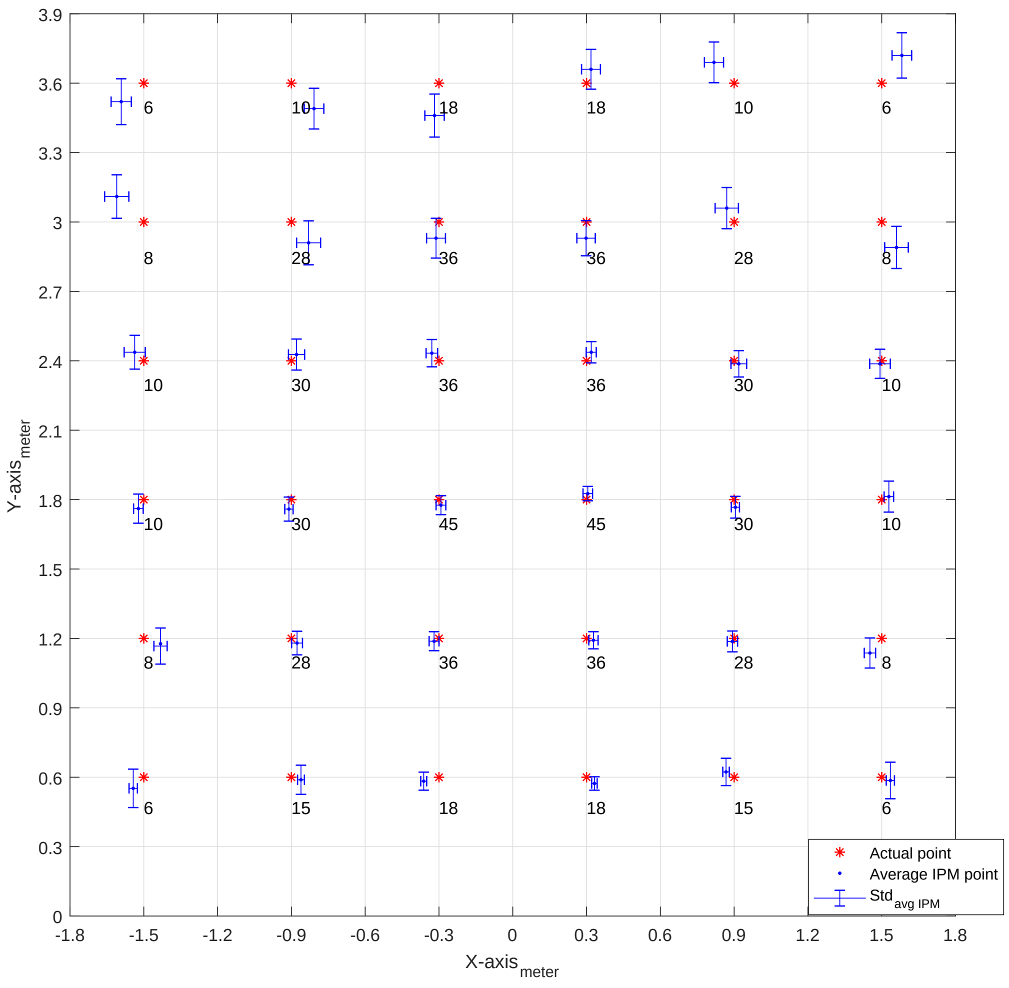

FIGURE 4.11: Estimated positions in the world frame for 36 different locations.

Figure 4.11 shows that, for covering a $3.6 \times 3.6m^2$ area, a total number of 36 different locations equally spaced at $0.6m$ have been considered. The red asterisk in Figure 4.11 is the actual location of the object on the floor and the blue dot is the estimated position averaged over all measurements. The standard deviation is expressed through error bars. The number below each red asterisk is the total number of acquisitions performed, corresponding to different values of Pitch and Yaw.

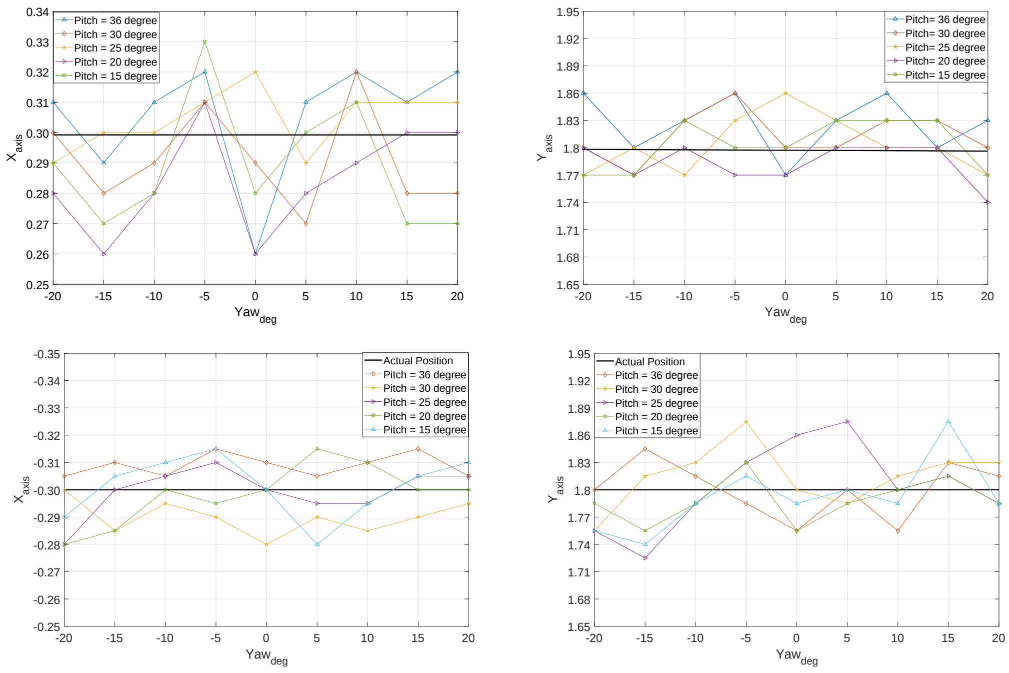

FIGURE 4.12: Position estimation of an object by varying Pitch and Yaw of the Pepper's head.

The data recording have been performed by rotating Pepper's head, i.e., by letting the Pitch assume values in the set (36, 31, 26, 21, 16) and the Yaw in the set (-20, -15, -10, -5, 0, 5, 10, 15, 20), which gives a total of 45 values for estimating the position of each point. Notice that an object is observed 45 times only in the best case: the more the object is closer to the center of the area, the more will be the number of recordings, since the object will be always in the field of view of the camera when Pitch and Yaw vary within the selected interval. On the other side, when the object is closer to the border of the region, it will be detected by the camera only with a limited set of (Pitch, Yaw) configurations.

Experiments show that the region in-between 1.2m to 3m along the Y (i.e, by changing Pitch) has the maximum number of camera views, and provides the lower error $av(|e|)$ and $std(|e|)$. The locations which are farther than 3m return a higher error deviation, because the object tends to become blurred in the image and then hard to detect; also, in this case, the object appears in the field of view of the camera only for a limited number of (Pitch, Yaw) configurations, for the same reason that has already been explained. Analogously, the error increases if Pepper's head rotates to the left/right more than ± 1.2m along the Yaw. Figure 4.11 shows also that the error deviation is, generally speaking, higher along the Y axis.

Figure 4.12 shows how the estimated position of a specific object varies during subsequent iteration of the algorithm with different Pitch and Yaw angles. The object is placed in position 0.3m along the X axis and 1.8m along the Y axis. The Figure clearly show that, in this location, the average error and the standard deviation are almost negligible (and they are due, among the other factors, to the distortion introduced by the camera lens).

4.2.2.1 Tests with wheeled robots

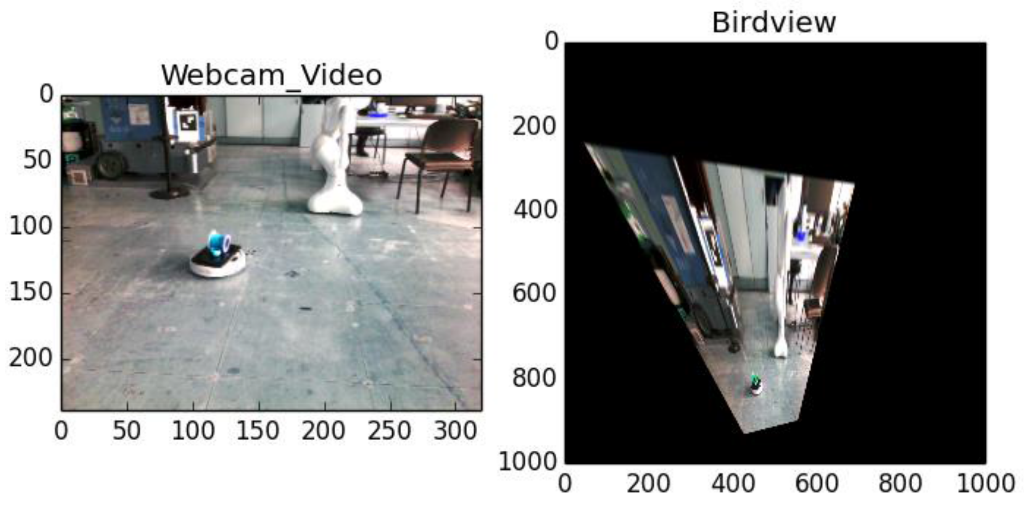

FIGURE 4.13: One frame showing Pepper's camera view and robot detection in IPM view.

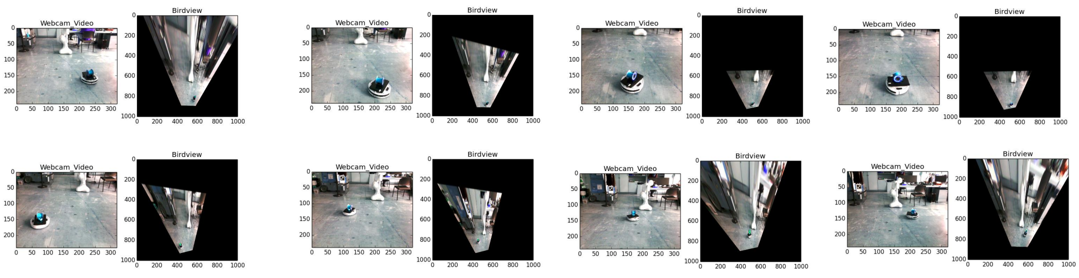

FIGURE 4.14: Multiple frames showing a robot moving along a circular path in Pepper's camera view and IPM view.

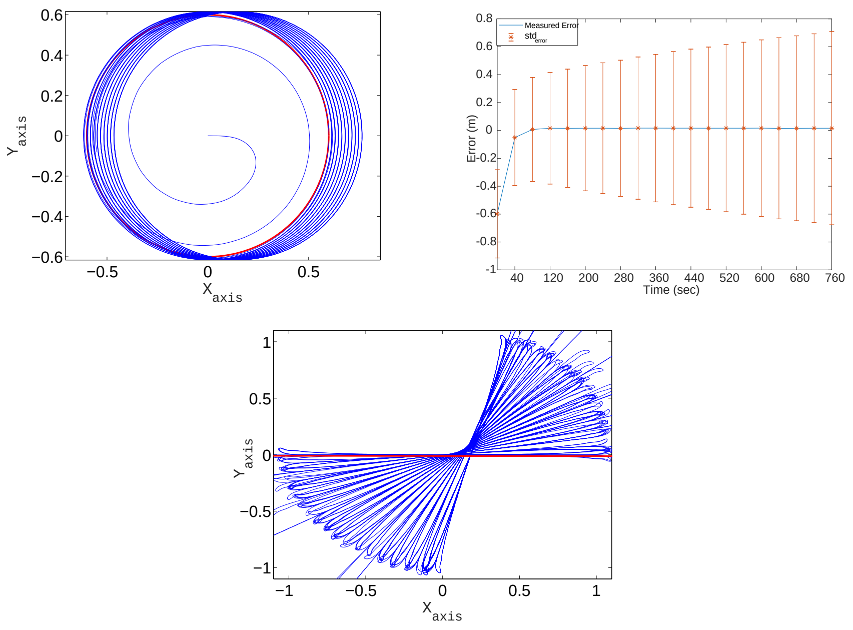

FIGURE 4.15: Plot of the robot's trajectory and error with odometry only.

In order to test the approach for run-time correction of the position of Roomba robots, the path following algorithm described in [1] has been adopted.

Figures 4.13 and 4.14 show the Pepper camera view and the IPM view while tracking a robot in the plane moving along a circular path. The Roomba position is being calculated by Pepper using a color marker mounted on the top of the robot, and communicated to the Roomba itself using the ROS framework and related communication tools[2] to facilitate multi-robot collaboration.

Figure 4.15 on the top shows the plot of the robot's trajectory as it is moving along a circular path, together with the measured error. Figure 4.15 on the bottom shows the plot of the robot while moving to and fro along a straight line. In both cases, the robot is relying on odometry only: it can be observed that the error monotonically increases as the robot moves along the path.

Figure 4.16 shows the same experiment by merging, through an EKF, the estimate provided by odometry with the estimate provided by Pepper through IPM. It can be observed that the error is significantly lower than the previous case and – more important – it does not increase as the robot moves along the path, which the wheeled robot is now able to accurately follow.

[2]http://www.ros.org/

Tables 4.1 and 4.2 report the average and stardard deviation of the error as the robot is moving, at different velocities $u(t) \in \{0.1msec, 0.2msec, 0.3msec\}$, without and with EKF correction. Table 4.1 refers to the circular path, whereas Table 4.2 refers to the straight line path. In all cases, the error is significantly higher in case that only odometry is considered for position estimation, whereas it dramatically decreases when considering the additional information provided by the IPM view.

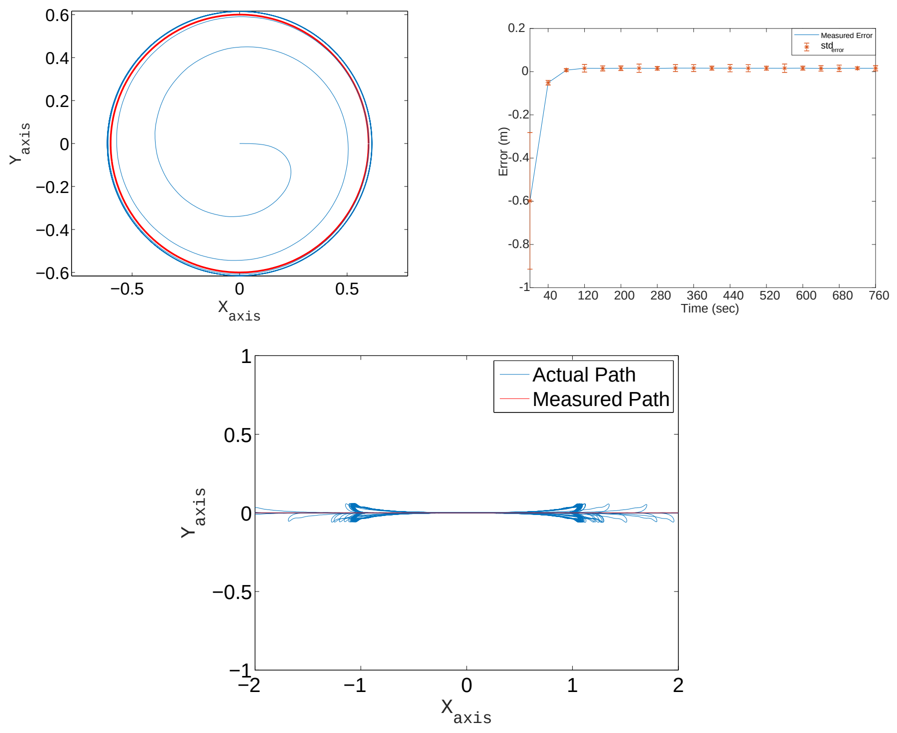

FIGURE 4.16: Plot of the robot's trajectory and error by merging odometry and IPM.

Figure 4.16 shows that with inclusion of IPM response and EKF, the is decreasing and the robot is following a circular and to-fro straight line path with minimizing the errors.

TABLE 4.1: Summary: $av|e|andstd|e|$, circular path.

Speed	Odometry Response	IPM and Odom without EKF	IPM and Odom with EKF
u(t)	$av(\|e\|)$, $std(\|e\|)$	$av(\|e\|)$, $std(\|e\|)$	$av(\|e\|)$, $std(\|e\|)$
0.1	0.267, 0.157	0.113, 0.108	0.021, 0.063
0.2	0.431, 0.193	0.146, 0.127	0.044, 0.081
0.3	0.512, 0.231	0.198, 0.144	0.057, 0.087

TABLE 4.2: Summary: $av|e|andstd|e|$, to and fro path.

Speed	Odometry Response	IPM and Odom without EKF	IPM and Odom with EKF
u(t)	$av(\|e\|)$, $std(\|e\|)$	$av(\|e\|)$, $std(\|e\|)$	$av(\|e\|)$, $std(\|e\|)$
0.1	0.307, 0.137	0.144, 0.116	0.033, 0.072
0.2	0.492, 0.188	0.167, 0.149	0.046, 0.091
0.3	0.637, 0.211	0.203, 0.175	0.064, 0.095

The next test has been performed by considering two robots. Figures 4.17 and 4.18 show snapshots where Pepper's camera is detecting multiple robots, by computing the corresponding IPM views in different time intervals that corresponds to different (Pitch, Yaw) configurations. Please notice that, in the following tests, robots avoid each other without any onboard sensors, i.e., by relying only on mutual position information received by Pepper. As in the previous case, position information is shared among all robots through ROS communication tools.

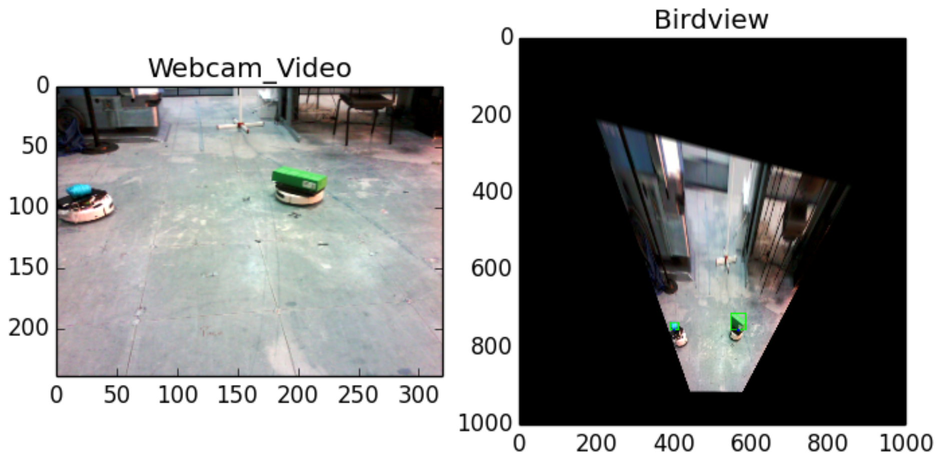

FIGURE 4.17: One frame showing Pepper's camera view and multi-robot detection in IPM view.

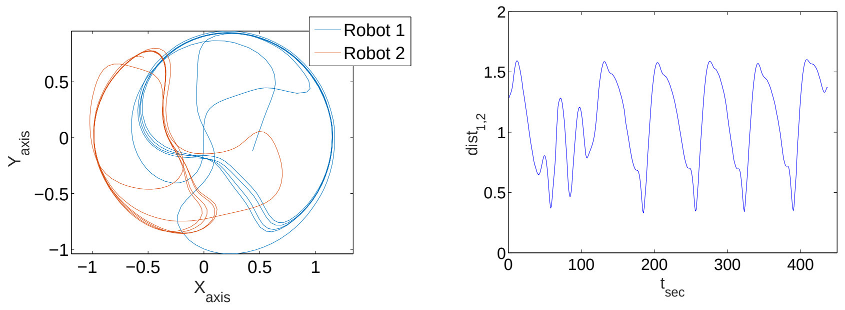

FIGURE 4.19: Plot of the trajectory of two Robots moving along a circular path with different speed, plot of their distance, and the corresponding positioning errors. Robot 1: $u_1(t) = 0.1m/s, K_1 = 20, K_2 = 4, A_j = 0.5, \sigma = 0.5$. Robot 2: $u_2(t) = 0.2m/s, K_1 = 18, K_2 = 3.5, A_j = 0.5, \sigma = 0.5$.

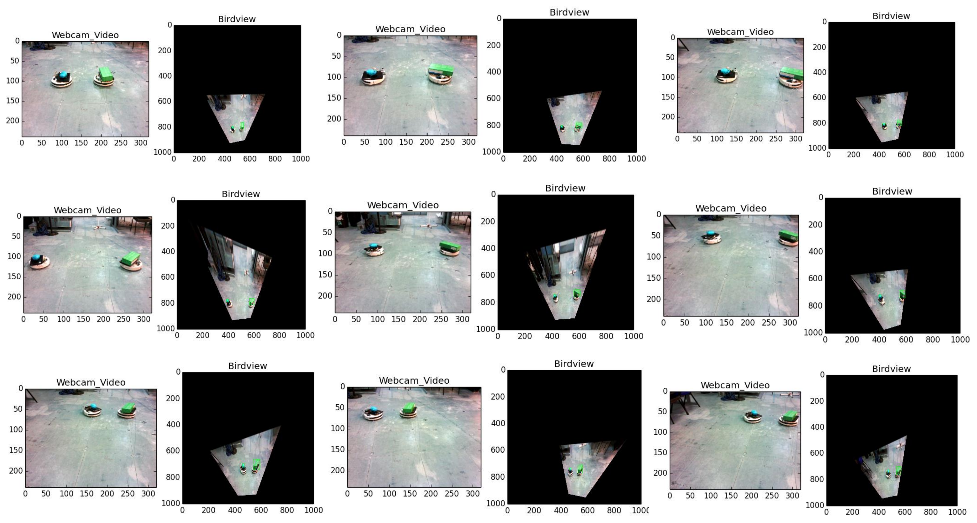

FIGURE 4.18: Multiple frames showing two robots moving along circular paths in Pepper's camera view and IPM view.

TABLE 4.3: Summary: Response of Two Robots by varying speed and $A_j = 0.4$, $\sigma = 0.55$.

$u_1(t)$, $u_2(t)$	$av(\|e_1\|)$, $std(\|e_1\|)$	$av(\|e_2\|)$, $std(\|e_2\|)$	$md_{1,2}$	$Robot1$ K_1, K_2	$Robot2$ K_1, K_2
0.1 , 0.1	0.133, 0.124	0.115, 0.132	0.503	20, 4	20, 4
0.1 , 0.2	0.141, 0.117	0.163, 0.143	0.487	20, 4	18, 3.5
0.1 , 0.3	0.178, 0.129	0.212, 0.133	0.329	20, 4	15, 2

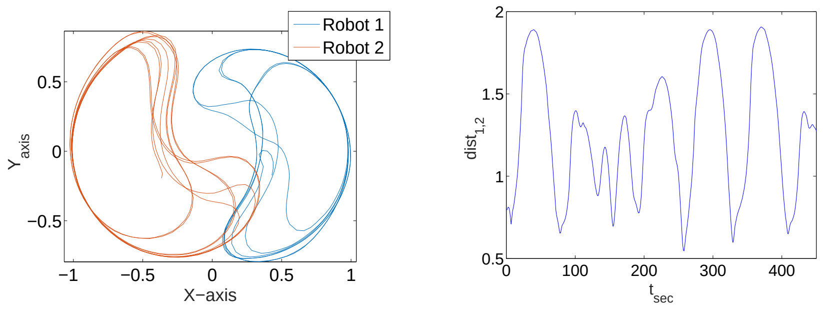

FIGURE 4.20: Plot of the trajectory of two Robots moving along a circular path with the same speed, plot of their mutual distance, and the corresponding positioning errors. Robots: $u(t) = 0.1m/s, K_1 = 20, K_2 = 4, A_j = 0.5, \sigma = 0.5$.

Figures 4.19 and 4.20 evaluate the performace of the two robots while they are correcting their own position and avoiding each other: please notice that, according to the algorithm proposed for obstacle avoidance, each robot is required to compute two gains K_1, K_2 as well as the influence region of obstacles A_j, σ.

Tests have been performed using different speeds, by measuring the errors e_1 and e_2 (corresponding to the two robots) from the circular path (which now depends also on the fact that robots diverge from the path to avoid each other), as well as the distance $md_{1,2}$ between robots. Results are summarized in Table 4.3.

Chapter 5

Conclusion and Future work

Overview

In this section the summary of the thesis is discussed. We have considered different scenarios for evaluating our algorithm, each of which are discussed therein. Initial experiments were performed using odometry feedback alone. However, in the long-run, due to the accrued errors, erratic deviations from the desired trajectory are observed, due to system and sensor noises.

To reduce the accumulated error and facilitate better localization, the same set of experiments were performed in a Motion Capture (MoCAP) environment. Since the MoCAP system localizes the robot, the desired trajectory is convincingly followed. Yet, using the MoCAP system limits the application to the MoCAP localizable area and prevents any outdoor applications.

We overcome this limitation by using the camera of the Pepper humanoid robot as an alternate to the MoCAP system. One can think of this as a "mobile" MoCAP environemnt, as opposes to the conventional "static" MoCAP environments. By using Pepper's single monocular camera, we first tried to detect object by taking into account Redball, NaoMark, Arucos markers and other built-in APIs. These tests failed to get accurate position due to camera distortion. As a solution to this problem, we propose using a bird-eye view also known as IPM view. To achieve this view the pepper camera view is converted to a bird-eye view, giving a linearized pixel point frame of the pepper image.

First, we perform tests using static pepper head, whose IPM transformation matrix always remains same, hence the FOV remains constant and we get the pixel values which can be converted to appropriate world frame. After failing with detection API (Naomark, Redcolor), we introduced openCV color detection tool, which detects any color blob in camera frame (i.e: red, green, yellow color) in different light condition. Once the color has been detected (as potential object), the pepper is able to know the position of colored object in both the frames (Pepper and IPM frame). So, we took into account the IPM frame that is linearized and calculated its respective position by doing manual calculation in real world (meters). This solves the object detetction problem when the camera and the objects are static. Now, we describe the procedure for object detection in case of camera and objects movements.

We have implemented an approach, an alternative of MoCAP system of 3m $\times$ 3m area. In our case, we change the orientation of Pepper head from maximum to minimum, using different combinations of Pitch and Yaw to see the plane of 3m $\times$ 3m area infront of Pepper. Pitch and Yaw points/pairs are obtained by discretizing the head orientations into different intervals with different transformation matrix every sample of time and calculated its world point with distance of 30 cm apart in each point, and detects the placed markers on the floor.

The Pepper sends position to Roomba and it updates itself with the IPM point instead of odometry so the rotation is similar as in MoCAP. Hence, a new way of robot cooperation has been developed, tested, calibrated and verified.

After this step, a calibration of world point in 3m $\times$ 3m area and is performed in order to carry out the experiment previously done in the MoCAP environment. A colored object is placed on a moving Roomba, in front of the Pepper, performing the path following and obstacle avoidance algorithm. The same is repeated with a couple of Roombas. In both the cases we obtain accurate results as obtained when performing the same in MoCAP.

Conclusion

In the first phase of the thesis the path following and obstacle avoidance has been achieved by extending the existing approach and the underlying methodology is discussed in detail. The environment we consider is dynamic in the sense that approach accommodates random movements of people and robots. Furthermore, the approach is extended to a heterogeneous multi-robot scenario, providing promising results.

As anticipated in Introduction, the reader may find some similarities with Artificial Potential Fields and other force field-based methods. In particular, this resemblance is a consequence of the fact that, similar to force field-based methods, the proposed approach reactively adds a contribution for each locally-sensed obstacle. However, the proposed approach is different for two main reasons, which follows from the fact that both the initial path as well as the deformed path are described as curves expressed through their implicit equations, respectively $f(x,y) = 0$ and $f'(x,y) = 0$:

- As opposed to force field based methods, the proposed approach guarantees that the robot can never be stuck in a position (x,y) without a preferred direction to move (in force field-based methods, this happens in correspondence of local minima, which are very frequent in presence of multiple obstacles). This property of the approach is due to the fact that the function $f(x,y)$ as well as the obstacle functions $O_j(x,y)$ are twice differentiable functions in $\mathscr{R}^2$, and therefore the deformed path $f'(x,y) = 0$ is necessarily continuous in $\mathscr{R}^2$, i.e., a direction to proceed along the path is always uniquely defined (see Figure 3.3).
- The error from the deformed path can be computed by simply evaluating $f'(x,y)$ in the robot's position (x,y), returning the result to a feedback controller [1] which guarantees asymptotic convergence to the path. This ultimately allows for setting the control variables (linear and angular speed) as a unique continuous function of the deformed path $f'(x,y) = 0$, the robot's pose (x,y,θ), and the relative position (x_j, y_j) of all the locally-sensed obstacles with respect to the robot.

At first, the proposed approach has been validated by quantitatively measuring its performance in a $3m \times 3m$ meters arena crowded with robots and persons. The experimental results, obtained with up to three mobile wheeled robots, confirm the robustness and the safety of the approach even in complex scenarios with moving obstacles and persons, in very narrow areas and at significant velocities.

Several conclusions can be drawn from the experimental analysis:

- The parameter K_1 and K_2 gains should be selected properly depending on the desired path and speed.
- The parameters σ and A_j of the Gaussian (obstacle) function should be opportunely tuned in order to avoid the obstacles. A procedure has been introduced

that makes it possible to set the two values in such a way as to guarantee that no collision can be produced. By arbitrarily increasing σ and A_j, the distance between the robot and the obstacles increases, but the average error between the actual path and the desired path increases as well.

- The approach proves to work correctly in static environments ($Scenarios\ 1,2$), since the error between the robots and its path is always less than 0.07m with no obstacle and less than 0.24m with static obstacle, as shown in Tables 3.1, 3.2, 3.3 and 3.4.

- While multiple robots are moving together ($Scenario\ 3$), varying A and σ has the effect of avoiding obstacle. As robot speed increases, the appropriate gain tuning parameters must be selected in order to stabilize the robot response. Results are shown in Tables 3.5, 3.6, 3.7, 3.8, 3.10, 3.9, 3.11 and 3.12.

- In the case of multiple robots and multiple walking persons ($Scenario\ 4$) increasing the number of robots and persons will increase the path following error ($av|e|$), while decreasing the average distance between persons and robots $av(|dist_{min}|)$. This is shown in Tables 3.13, 3.14 and 3.15.

- In scenario 4, the tracking error increases, but the performance can be still considered satisfactory. On the other hand, the proposed method shows significantly performance with static obstacles.

In the second phase of the thesis, localization is achieved for heterogeneous team of robots. The approach (in first phase) is further modified to a heterogeneous multi-robot cooperative approach using IPM. Cooperative localization is performed using a dynamically moving Pepper head. The key elements of our approach has been demonstrated using experiments on real robots. The odometry feedback, in addition with IPM scenario and EKF, helps to correct the position of robot(s) in the plane. Further, dynamic obstacle avoidance is a also achieved as shown in the results. It can be concluded that (i) the more the number of recording pairs (i.e: Yaw, Pitch) for each point, gives more accurate calculations. (ii) This method can trivially be extended to more than 2 robots. (iii) The experiment shows that this method can be an alternative to MoCAP environment.

Future work

The presented multi-robot cooperation algorithm can be used futher in scenarios with more complex paths and a higher number of moving obstacles. Currently, the Pepper is static which limits its perceptual field. This is easily overcome by incorporating Pepper translation to extend its perceptual field. Another solution to multi-robot cooperation is implemented where Pepper can monitor the position of multiple robots in same plane. If any wheeled robot exceeds its covariance while following its desired path, then Pepper starts updating its position by giving IPM observation values and vice versa to other robots.

In future work we intend to examine our methodology's benefit in supporting larger teams of robots, The algorithm can be used in many application like: autonomous driving, autonomous car parking, pick and place applications, drones having cameras to assist robots for search and rescue missions, monitoring an orchard etc.

Bibliography

[1] Angelo Morro, Antonio Sgorbissa, and Renato Zaccaria. Path following for unicycle robots with an arbitrary path curvature. *IEEE Transactions on Robotics*, 27 (5):1016–1023, 2011.

[2] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios. *arXiv preprint arXiv:1808.03841*, 2018.

[3] José de Gea Fernández, Elie Allouis, Karol Seweryn, Frank Kirchner, and Yang Gao. Manipulation and control. *Contemporary Planetary Robotics: An Approach Toward Autonomous Systems*, 34:36, 2016.

[4] PIETRO ASTOLFI. An experiment in autonomous vineyard navigation: the grape project. 2017.

[5] Peter Kenneth Ward. *Design of a biologically inspired climbing robot and an adhesion mechanism for reliable and versatile climbing in complex steel structures*. PhD thesis, 2016.

[6] Juntong Qi, Dalei Song, Hong Shang, Nianfa Wang, Chunsheng Hua, Chong Wu, Xin Qi, and Jianda Han. Search and rescue rotary-wing uav and its application to the lushan ms 7.0 earthquake. *Journal of Field Robotics*, 33(3):290–321, 2016.

[7] Christopher Gomez and Heather Purdie. Uav-based photogrammetry and geocomputing for hazards and disaster risk monitoring–a review. *Geoenvironmental Disasters*, 3(1):23, 2016.

[8] Phuong DH Nguyen, Carmine T Recchiuto, and Antonio Sgorbissa. Real-time path generation for multicopters in environments with obstacles. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 1582–1588. IEEE, 2016.

[9] Alexey S Matveev, Hamid Teimoori, and Andrey V Savkin. A method for guidance and control of an autonomous vehicle in problems of border patrolling and obstacle avoidance. *Automatica*, 47(3):515–524, 2011.

[10] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53(3):263, 2008.

[11] Alejandro R Mosteo, Eduardo Montijano, and Danilo Tardioli. Optimal role and position assignment in multi-robot freely reachable formations. *Automatica*, 81: 305–313, 2017.

[12] Dave W Oyler, Pierre T Kabamba, and Anouck R Girard. Pursuit–evasion games in the presence of obstacles. *Automatica*, 65:1–11, 2016.

[13] Alireza Doosthoseini and Christopher Nielsen. Coordinated path following for unicycles: A nested invariant sets approach. *Automatica*, 60:17–29, 2015.

[14] Yiannis Kantaros and Michael M Zavlanos. Distributed communication-aware coverage control by mobile sensor networks. *Automatica*, 63:209–220, 2016.

[15] Xiangpeng Li, Hao Yang, Jianjun Wang, and Dong Sun. Design of a robust unified controller for cell manipulation with a robot-aided optical tweezers system. *Automatica*, 55:279–286, 2015.

[16] Xiangpeng Li, Dong Sun, and Jie Yang. A bounded controller for multirobot navigation while maintaining network connectivity in the presence of obstacles. *Automatica*, 49(1):285–292, 2013.

[17] Alexey S Matveev, MC Hoy, and Andrey V Savkin. A method for reactive navigation of nonholonomic under-actuated robots in maze-like environments. *Automatica*, 49(5):1268–1274, 2013.

[18] Andrey V Savkin and Chao Wang. Seeking a path through the crowd: Robot navigation in unknown dynamic environments with moving obstacles based on an integrated environment representation. *Robotics and Autonomous Systems*, 62 (10):1568–1580, 2014.

[19] LU Xiao-Qing, WANG Yao-Nan, and MAO Jian-Xu. Nonlinear control for multi-agent formations with delays in noisy environments. *Acta Automatica Sinica*, 40(12):2959–2967, 2014.

[20] Youfeng Su. Leader-following rendezvous with connectivity preservation and disturbance rejection via internal model approach. *Automatica*, 57:203–212, 2015.

[21] Xinmiao Sun and Christos G Cassandras. Optimal dynamic formation control of multi-agent systems in constrained environments. *Automatica*, 73:169–179, 2016.

[22] YANG Tian-Tian, Liu Zhi-Yuan, CHEN Hong, and Pei Run. Formation control and obstacle avoidance for multiple mobile robots. *Acta Automatica Sinica*, 34 (5):588–593, 2008.

[23] Michael Malisoff, Robert Sizemore, and Fumin Zhang. Adaptive planar curve tracking control and robustness analysis under state constraints and unknown curvature. *Automatica*, 75:133–143, 2017.

[24] Maciej Marcin Michałek. A highly scalable path-following controller for n-trailers with off-axle hitching. *Control Engineering Practice*, 29:61–73, 2014.

[25] Antonio Sgorbissa and Renato Zaccaria. 3d path following with no bounds on the path curvature through surface intersection. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4029–4035. IEEE, 2010.

[26] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[27] Johann Borenstein and Yoram Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, 7 (3):278–288, 1991.

[28] Andrey V Savkin and Chao Wang. A simple biologically inspired algorithm for collision-free navigation of a unicycle-like robot in dynamic environments with moving obstacles. *Robotica*, 31(06):993–1001, 2013.

[29] Phuong D. H. Nguyen, Carmine T. Recchiuto, and Antonio Sgorbissa. Real-time path generation and obstacle avoidance for multirotors: A novel approach. *Journal of Intelligent & Robotic Systems*, 89(1):27–49, Jan 2018. ISSN 1573-0409.

[30] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[31] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised*, volume 118. Springer, 2017.

[32] Sajad Saeedi, Michael Trentini, Mae Seto, and Howard Li. Multiple-robot simultaneous localization and mapping: A review. *Journal of Field Robotics*, 33(1): 3–46, 2016.

[33] Sajad Saeedi, Michael Trentini, Mae Seto, and Howard Li. Multiple-robot simultaneous localization and mapping: A review. *Journal of Field Robotics*, 33(1): 3–46. doi: 10.1002/rob.21620. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21620.

[34] Séverin Lemaignan, Mathieu Warnier, E Akin Sisbot, Aurélie Clodic, and Rachid Alami. Artificial cognition for social human–robot interaction: An implementation. *Artificial Intelligence*, 247:45–69, 2017.

[35] Gideon Stein, Erez Dagan, Ofer Mano, and Amnon Shashua. Collision warning system, May 23 2017. US Patent 9,656,607.

[36] Daniel E Maurino, James Reason, Neil Johnston, and Rob B Lee. *Beyond aviation human factors: Safety in high technology systems*. Routledge, 2017.

[37] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[38] Amir Mukhtar, Likun Xia, and Tong Boon Tang. Vehicle detection techniques for collision avoidance systems: A review. *IEEE Transactions on Intelligent Transportation Systems*, 16(5):2318–2338, 2015.

[39] Santosh A Hiremath, Gerie WAM Van Der Heijden, Frits K Van Evert, Alfred Stein, and Cajo JF Ter Braak. Laser range finder model for autonomous navigation of a robot in a maize field using a particle filter. *Computers and Electronics in Agriculture*, 100:41–50, 2014.

[40] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. Multi-robot task allocation: A review of the state-of-the-art. In *Cooperative Robots and Sensor Networks 2015*, pages 31–51. Springer, 2015.

[41] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12):399, 2013.

[42] Youmin Zhang and Hasan Mehrjerdi. A survey on multiple unmanned vehicles formation control and coordination: Normal and fault situations. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 1087–1096. IEEE, 2013.

[43] Luigi Alfredo Grieco, Alessandro Rizzo, Simona Colucci, Sabrina Sicari, Giuseppe Piro, Donato Di Paola, and Gennaro Boggia. Iot-aided robotics applications: Technological implications, target domains and open issues. *Computer Communications*, 54:32–47, 2014.

[44] Cyril Robin and Simon Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, 40(4):729–760, 2016.

[45] Pooyan Fazli, Alireza Davoodi, and Alan K Mackworth. Multi-robot repeated area coverage. *Autonomous robots*, 34(4):251–276, 2013.

[46] SoftBank Robotics. http://doc.aldebaran.com.

[47] M. Hassan Tanveer, Carmine T. Recchiuto, and Antonio Sgorbissa. Coordinated behaviour with a Pepper Humanoid robot to estimate the distance of other robot using Inverse Perspective Mapping. In *IEEE International Conference on Automation and Robotics (ICAROB)*, 2018.

[48] Lianyang Ma, Xiaokang Yang, and Dacheng Tao. Person re-identification over camera networks using multi-task distance metric learning. *IEEE Transactions on Image Processing*, 23(8):3656–3670, 2014.

[49] Corrado Possieri and Andrew R Teel. Lq optimal control for a class of hybrid systems. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 604–609. IEEE, 2016.

[50] Michael Hoy, Alexey S Matveev, and Andrey V Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33(03):463–497, 2015.

[51] Javier Minguez, Florant Lamiraux, and Jean-Paul Laumond. Motion planning and obstacle avoidance. In *Springer handbook of robotics*, pages 1177–1202. Springer, 2016.

[52] Andrey V Savkin, Alexey S Matveev, Michael Hoy, and Chao Wang. *Safe robot navigation among moving and steady obstacles*. Butterworth-Heinemann, 2015.

[53] Emanuele Garone, Stefano Di Cairano, Ilya Kolmanovsky, et al. Reference and command governors for systems with constraints: A survey on theory and applications. *Automatica*, 75:306–328, 2017.

[54] Gökhan M Atınç, Dušan M Stipanović, and Petros G Voulgaris. Supervised coverage control of multi-agent systems. *Automatica*, 50(11):2936–2942, 2014.

[55] Kun Zhang, Jonathan Sprinkle, and Ricardo G Sanfelice. A hybrid model predictive controller for path planning and path following. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, pages 139–148. ACM, 2015.

[56] Sven Brüggemann, Corrado Possieri, Jorge I Poveda, and Andrew R Teel. Robust constrained model predictive control with persistent model adaptation. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 2364–2369. IEEE, 2016.

[57] Lorenzo Fagiano and Andrew R Teel. Model predictive control with generalized terminal state constraint. *IFAC Proceedings Volumes*, 45(17):299–304, 2012.

[58] XI Yu-Geng, LI De-Wei, and LIN Shu. Model predictive control—status and challenges. *Acta Automatica Sinica*, 39(3):222–236, 2013.

[59] Daniel Liberzon. *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2012.

[60] Gilberto Pin, Davide M Raimondo, Lalo Magni, and Thomas Parisini. Robust model predictive control of nonlinear systems with bounded and state-dependent uncertainties. *IEEE Transactions on automatic control*, 54(7):1681–1687, 2009.

[61] Graham C Goodwin, Richard H Middleton, María M Seron, and B Campos. Application of nonlinear model predictive control to an industrial induction heating furnace. *Annual Reviews in Control*, 37(2):271–277, 2013.

[62] Md Sohel Rana, Hemanshu R Pota, and Ian R Petersen. The design of model predictive control for an afm and its impact on piezo nonlinearities. *European Journal of Control*, 20(4):188–198, 2014.

[63] D Kouzoupis, A Zanelli, H Peyrl, and HJ Ferreau. Towards proper assessment of qp algorithms for embedded model predictive control. In *Control Conference (ECC), 2015 European*, pages 2609–2616. IEEE, 2015.

[64] Wongun Kim, Dongwook Kim, Kyongsu Yi, and H Jin Kim. Development of a path-tracking control system based on model predictive control using infrastructure sensors. *Vehicle System Dynamics*, 50(6):1001–1023, 2012.

[65] Tim Mercy, Ruben Van Parys, and Goele Pipeleers. Spline-based motion planning for autonomous guided vehicles in a dynamic environment. *IEEE Transactions on Control Systems Technology*, 2017.

[66] NPWB DUNBAR and RYAN FRANZ. Online control customization via optimization, based control. *Software-Enabled Control Inf. Technol. Dyn. Syst*, page 149, 2003.

[67] S Bououden, Mohammed Chadli, and Hamid Reza Karimi. An ant colony optimization-based fuzzy predictive control approach for nonlinear processes. *Information Sciences*, 299:143–158, 2015.

[68] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using the relative velocity paradigm. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 560–565. IEEE, 1993.

[69] Yoshiaki Kuwata, Michael T Wolf, Dimitri Zarzhitsky, and Terrance L Huntsberger. Safe maritime autonomous navigation with colregs, using velocity obstacles. *IEEE Journal of Oceanic Engineering*, 39(1):110–119, 2014.

[70] Frédéric Large, Christian Laugier, and Zvi Shiller. Navigation among moving obstacles using the nlvo: Principles and applications to intelligent vehicles. *Autonomous Robots*, 19(2):159–171, 2005.

[71] Oscar Montiel, Ulises Orozco-Rosas, and Roberto Sepúlveda. Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. *Expert Systems with Applications*, 42(12):5177–5191, 2015.

[72] Muhannad Mujahed, Dirk Fischer, and Bärbel Mertsching. Admissible gap navigation: A new collision avoidance approach. *Robotics and Autonomous Systems*, 103:93–110, may 2018. doi: 10.1016/j.robot.2018.02.008. URL `https://doi.org/10.1016/j.robot.2018.02.008`.

[73] Zhihao Xu, Robin Hess, and Klaus Schilling. Constraints of potential field for obstacle avoidance on car-like mobile robots. *IFAC Proceedings Volumes*, 45(4):169–175, 2012. doi: 10.3182/20120403-3-de-3010.00077. URL `https://doi.org/10.3182/20120403-3-de-3010.00077`.

[74] M.H. Korayem and S.R. Nekoo. The SDRE control of mobile base cooperative manipulators: Collision free path planning and moving obstacle avoidance. *Robotics and Autonomous Systems*, 86:86–105, dec 2016. doi: 10.1016/j.robot.2016.09.003. URL https://doi.org/10.1016/j.robot.2016.09.003.

[75] Antonio Sgorbissa. Integrated robot planning, obstacle avoidance, and path following in 2d and 3d: ground, aerial, and underwater vehicles. pages –, 2017. doi: 10.13140/rg.2.2.14838.80969.

[76] Rami Al-Jarrah, Mohammad Al-Jarrah, and Hubert Roth. A novel edge detection algorithm for mobile robot path planning. *Journal of Robotics*, 2018, 2018.

[77] R Deepu, B Honnaraju, and S Murali. Path generation for robot navigation using a single camera. *Procedia Computer Science*, 46:1425–1432, 2015.

[78] Riccardo Falconi, Lorenzo Sabattini, Cristian Secchi, Cesare Fantuzzi, and Claudio Melchiorri. Edge-weighted consensus-based formation control strategy with collision avoidance. *Robotica*, 33(2):332–347, 2015.

[79] Navid Dadkhah and Berenice Mettler. Survey of motion planning literature in the presence of uncertainty: Considerations for uav guidance. *Journal of Intelligent & Robotic Systems*, 65(1-4):233–246, 2012.

[80] Volkan Sezer and Metin Gokasan. A novel obstacle avoidance algorithm:“follow the gap method”. *Robotics and Autonomous Systems*, 60(9):1123–1134, 2012.

[81] Chia Hung Hsu and Chia-Feng Juang. Evolutionary robot wall-following control using type-2 fuzzy controller with species-de-activated continuous aco. *IEEE Trans. Fuzzy Systems*, 21(1):100–112, 2013.

[82] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1): 23–33, 1997.

[83] Kai O Arras, Jan Persson, Nicola Tomatis, and Roland Siegwart. Real-time obstacle avoidance for polygonal robots with a reduced dynamic window. In *Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on*, volume 3, pages 3050–3055. IEEE, 2002.

[84] Petter Ogren and Naomi Ehrich Leonard. A convergent dynamic window approach to obstacle avoidance. *IEEE Transactions on Robotics*, 21(2):188–195, 2005.

[85] Bruno Damas and José Santos-Victor. Avoiding moving obstacles: the forbidden velocity map. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4393–4398. IEEE, 2009.

[86] Lionel Lapierre, Rene Zapata, and Pascal Lepinay. Simulatneous path following and obstacle avoidance control of a unicycle-type robot. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2617–2622. IEEE, 2007.

[87] Michele Aicardi, Giuseppe Casalino, Antonio Bicchi, and Aldo Balestrino. Closed loop steering of unicycle like vehicles via lyapunov techniques. *IEEE Robotics & Automation Magazine*, 2(1):27–35, 1995.

[88] Danny Soetanto, Lionel Lapierre, and Antonio Pascoal. Adaptive, non-singular path-following control of dynamic wheeled robots. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 2, pages 1765–1770. IEEE, 2003.

[89] A Medina-Santiago, JL Camas-Anzueto, Juan Alejandro Vazquez-Feijoo, HR Hernández-de León, and R Mota-Grajales. Neural control system in obstacle avoidance in mobile robots using ultrasonic sensors. *Journal of applied research and technology*, 12(1):104–110, 2014.

[90] S. Wen, W. Zheng, J. Zhu, X. Li, and S. Chen. Elman fuzzy adaptive control for obstacle avoidance of mobile robots using hybrid force/position incorporation. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):603–608, July 2012. ISSN 1094-6977. doi: 10.1109/TSMCC.2011.2157682.

[91] Hui Miao and Yu-Chu Tian. Dynamic robot path planning using an enhanced simulated annealing approach. *Applied Mathematics and Computation*, 222:420–437, oct 2013. doi: 10.1016/j.amc.2013.07.022. URL `https://doi.org/10.1016/j.amc.2013.07.022`.

[92] Afshin Mohammadi, Meysam Rahimi, and Amir Abolfazl Suratgar. A new path planning and obstacle avoidance algorithm in dynamic environment. In *2014 22nd Iranian Conference on Electrical Engineering (ICEE)*. IEEE, may 2014.

[93] Thulasi Mylvaganam and Mario Sassano. Autonomous collision avoidance for wheeled mobile robots using a differential game approach. *European Journal of Control*, 40:53–61, mar 2018. doi: 10.1016/j.ejcon.2017.11.005. URL https://doi.org/10.1016/j.ejcon.2017.11.005.

[94] Yanyan Dai, YoonGu Kim, SungGil Wee, DongHa Lee, and SukGyu Lee. A switching formation strategy for obstacle avoidance of a multi-robot system based on robot priority model. *ISA transactions*, 56:123–134, 2015.

[95] Lin Hsien-I. 2d-span resampling of bi-RRT in dynamic path planning. *International Journal of Automation and Smart Technology*, 4(4):39–48, mar 2015. doi: 10.5875/ausmt.v5i1.837. URL https://doi.org/10.5875/ausmt.v5i1.837.

[96] Chang-bae Moon and Woojin Chung. Kinodynamic planner dual-tree rrt (dt-rrt) for two-wheeled mobile robots using the rapidly exploring random tree. *IEEE Transactions on industrial electronics*, 62(2):1080–1090, 2015.

[97] Farah Kamil, Tang Sai Hong, Weria Khaksar, Mohammed Yasser Moghrabiah, Norzima Zulkifli, and Siti Azfanizam Ahmad. New robot navigation algorithm for arbitrary unknown dynamic environments based on future prediction and priority behavior. *Expert Systems with Applications*, 86:274–291, 2017.

[98] Farhad Bayat, Sepideh Najafinia, and Morteza Aliyari. Mobile robots path planning: Electrostatic potential field approach. *Expert Systems with Applications*, 100:68–78, 2018.

[99] Andreas Birk and Stefano Carpin. Rescue robotics—a crucial milestone on the road to autonomous systems. *Advanced Robotics*, 20(5):595–605, 2006.

[100] Nicholas R Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1 (1):7–38, 1998.

[101] Tamio Arai, Enrico Pagello, and Lynne E Parker. Advances in multi-robot systems. *IEEE Transactions on robotics and automation*, 18(5):655–661, 2002.

[102] Brian P Gerkey, Richard T Vaughan, Kasper Stoy, Andrew Howard, Gaurav S Sukhatme, and Maja J Mataric. Most valuable player: A robot device server for distributed control. In *Intelligent Robots and Systems, 2001. Proceedings.*

2001 IEEE/RSJ International Conference on, volume 3, pages 1226–1231. IEEE, 2001.

[103] Alonzo Kelly, Anthony Stentz, Omead Amidi, Mike Bode, David Bradley, Antonio Diaz-Calderon, Mike Happold, Herman Herman, Robert Mandelbaum, Tom Pilarski, et al. Toward reliable off road autonomous vehicles operating in challenging environments. *The International Journal of Robotics Research*, 25(5-6): 449–483, 2006.

[104] L Chaimowicz, A Cowley, D Gomez-Ibanez, B Grocholsky, MA Hsieh, H Hsu, JF Keller, V Kumar, R Swaminathan, and CJ Taylor. Deploying air-ground multi-robot teams in urban environments. In *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, pages 223–234. Springer, 2005.

[105] Luiz Chaimowicz, Ben Grocholsky, James F Keller, Vijay Kumar, and Camillo J Taylor. Experiments in multirobot air-ground coordination. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 4053–4058. IEEE, 2004.

[106] Edwin Olson, Johannes Strom, Rob Goeddel, Ryan Morton, Pradeep Ranganathan, and Andrew Richardson. Exploration and mapping with autonomous robot teams. *Communications of the ACM*, 56(3):62–70, 2013.

[107] Jan Carlo Barca and Y Ahmet Sekercioglu. Swarm robotics reviewed. *Robotica*, 31(3):345–359, 2013.

[108] Wei Ren. Multi-vehicle consensus with a time-varying reference state. *Systems & Control Letters*, 56(7-8):474–483, 2007.

[109] Wei Ren, Randal W Beard, and Ella M Atkins. A survey of consensus problems in multi-agent coordination. In *American Control Conference, 2005. Proceedings of the 2005*, pages 1859–1864. IEEE, 2005.

[110] Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[111] Richard M Murray. Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):571–583, 2007.

[112] Francesco Bullo, Jorge Cortes, and Sonia Martinez. *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*, volume 27. Princeton University Press, 2009.

[113] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control*, 51(3):401–420, 2006.

[114] Ioannis Rekleitis, Gregory Dudek, and Evangelos Milios. Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, 31 (1-4):7–40, 2001.

[115] Robert Zlot, Anthony Stentz, M Bernardine Dias, and Scott Thayer. Multi-robot exploration controlled by a market economy. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 3, pages 3016–3023. IEEE, 2002.

[116] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3):376–386, 2005.

[117] William Rone and Pinhas Ben-Tzvi. Mapping, localization and motion planning in mobile multi-robotic systems. *Robotica*, 31(1):1–23, 2013.

[118] Sebastian Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1(1-35):1, 2002.

[119] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99 – 134, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(98)00023-X. URL `http://www.sciencedirect.com/science/article/pii/S000437029800023X`.

[120] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987. doi: 10.1287/moor.12.3.441. URL `https://doi.org/10.1287/moor.12.3.441`.

[121] Carol Martínez, Thomas Richardson, Peter Thomas, Jonathan Luke du Bois, and Pascual Campoy. A vision-based strategy for autonomous aerial refueling tasks. *Robotics and Autonomous Systems*, 61(8):876 – 895, 2013. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2013.02.

006. URL http://www.sciencedirect.com/science/article/pii/S0921889013000420.

[122] Kai M Wurm, Christian Dornhege, Patrick Eyerich, Cyrill Stachniss, Bernhard Nebel, and Wolfram Burgard. Coordinated exploration with marsupial teams of robots using temporal symbolic planning. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5014–5019. IEEE, 2010.

[123] Marwan Hussein, Matthew Renner, Masaaki Watanabe, and Karl Iagnemma. Matching of ground-based lidar and aerial image data for mobile robot localization in densely forested environments. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1432–1437. IEEE, 2013.

[124] Nathan Michael, Shaojie Shen, Kartik Mohta, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, Kazunori Ohno, et al. Collaborative mapping of an earthquake damaged building via ground and aerial robots. In *Field and Service Robotics*, pages 33–47. Springer, 2014.

[125] Keith YK Leung, Timothy D Barfoot, and Hugh HT Liu. Decentralized localization of sparsely-communicating robot networks: A centralized-equivalent approach. *IEEE Transactions on Robotics*, 26(1):62–77, 2010.

[126] Heng-Da Cheng, X_ H_ Jiang, Ying Sun, and Jingli Wang. Color image segmentation: advances and prospects. *Pattern recognition*, 34(12):2259–2281, 2001.

[127] Hunjae Yoo, Ukil Yang, and Kwanghoon Sohn. Gradient-enhancing conversion for illumination-robust lane detection. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1083–1094, 2013.

[128] Abdulhakam AM Assidiq, Othman O Khalifa, Md Rafiqul Islam, and Sheroz Khan. Real time lane detection for autonomous vehicles. In *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, pages 82–88. IEEE, 2008.

[129] J Spanjaards. Real-time camera based lane detection for autonomous driving. *Eindhoven Univ. Technol., Eindhoven, The Netherlands, Internship Rep. DC, 2016*, 2016.

[130] Massimo Bertozzi and Alberto Broggi. Gold: A parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE transactions on image processing*, 7(1):62–81, 1998.

[131] Soren Kammel and Benjamin Pitzer. Lidar-based lane marker detection and mapping. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 1137–1142. IEEE, 2008.

[132] Albert S Huang, David Moore, Matthew Antone, Edwin Olson, and Seth Teller. Finding multiple lanes in urban road networks with vision and lidar. *Autonomous Robots*, 26(2-3):103–122, 2009.

[133] D. Chwa. Robust distance-based tracking control of wheeled mobile robots using vision sensors in the presence of kinematic disturbances. *IEEE Transactions on Industrial Electronics*, 63(10):6172–6183, Oct 2016. ISSN 0278-0046. doi: 10.1109/TIE.2016.2590378.

[134] Hanspeter A Mallot, Heinrich H Bülthoff, JJ Little, and Stefan Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological cybernetics*, 64(3):177–185, 1991.

[135] Robert Laganiere. Compositing a bird's eye view mosaic. *image*, 10:3, 2000.

[136] Chien-Chuan Lin and Ming-Shi Wang. A vision based top-view transformation model for a vehicle parking assistant. *Sensors*, 12(4):4431–4446, 2012.

[137] Sibel Yenikaya, Gökhan Yenikaya, and Ekrem Düven. Keeping the vehicle on the road: A survey on on-road lane detection systems. *ACM Computing Surveys (CSUR)*, 46(1):2, 2013.

[138] Chunzhao Guo, Jun-ichi Meguro, Yoshiko Kojima, and Takashi Naito. Automatic lane-level map generation for advanced driver assistance systems using low-cost sensors. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3975–3982. IEEE, 2014.

[139] Javier Civera, Andrew J Davison, and JM Martinez Montiel. Inverse depth parametrization for monocular slam. *IEEE transactions on robotics*, 24(5):932–945, 2008.

[140] Miguel Oliveira, Vitor Santos, and Angel D Sappa. Multimodal inverse perspective mapping. *Information Fusion*, 24:108–121, 2015.

[141] Xiaogang Wang. Intelligent multi-camera video surveillance: A review. *Pattern recognition letters*, 34(1):3–19, 2013.

[142] S Tuohy, D O'Cualain, E Jones, and M Glavin. Distance determination for an automobile environment using Inverse Perspective Mapping in OpenCV. In *IET Irish Signals and Systems Conference (ISSC)*, 2010.

[143] Douglas P Boyle, Hoshin V Gupta, and Soroosh Sorooshian. Toward improved calibration of hydrologic models: Combining the strengths of manual and automatic methods. *Water Resources Research*, 36(12):3663–3674, 2000.

[144] Matteo Munaro, Filippo Basso, and Emanuele Menegatti. Openptrack: Open source multi-camera calibration and people tracking for rgb-d camera networks. *Robotics and Autonomous Systems*, 75:525–538, 2016.

[145] Pedro Miraldo and Helder Araujo. Calibration of smooth camera models. *IEEE transactions on pattern analysis and machine intelligence*, 35(9):2091–2103, 2013.

[146] Daniel Moreno and Gabriel Taubin. Simple, accurate, and robust projector-camera calibration. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 464–471. IEEE, 2012.

[147] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.