# Development of a Balancing Robot as an Indoor Service Agent

Shreyas Srivatchan

CMU-RI-TR-20-57

*Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.*


The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Noember 2020

**Thesis Committee:**
Ralph Hollis, Chair
Jean Oh
Roberto Shu

# Abstract

This work presents a robotic system that can navigate human environments, respond to speech commands and perform simple tasks. To achieve this, a ballbot-type robot that balances and navigates on a single spherical wheel is used. This system possesses many advantages such as omni-directional motion and agile mobility, which makes it apt to function in human environments.

In order to achieve smooth and agile navigation, prior work which formulates the system as 'differentially flat' is used. This formulation provides necessary lean angle trajectories used by the robot to move from one point to another. Leveraging this, a navigation framework using the concept of 'Velocity-Obstacles' is proposed to dynamically avoid human obstacles. An efficient and lightweight human tracking framework using 'SSD-mobilenets' and 'Simple Online Realtime Tracking (SORT)' is also developed to facilitate efficient obstacle avoidance. Finally speech capability is integrated into the system using Amazon-Alexa's extensive skills and capabilities.

To demonstrate various components of the system, two experiments are performed. In the first, the robot actively guides a user to necessary product locations in a mock grocery store. In the second, the robot performs 'go, look and tell tasks' in a household environment while avoiding moving human obstacles. Specifically, the robot goes to a particular location in the house, detects items and reports them back to the user.

## Acknowledgments

The Robotics Institute is a wonderful place to learn and grow for any student. I would like to thank all the people who have contributed over the years towards making the institute such an outstanding place. I am fortunate to have been a part of it.

I would like to express my deepest gratitude towards my advisor Professor Ralph Hollis without whom this work would not have been possible. Your constant support, guidance and mentorship have been cornerstones towards my growth as a researcher. The atmosphere you have created and the machines you have developed over the years motivate me every time I enter the lab.

This work was possible because of the continuous support of many people. I would like to especially thank Roberto Shu, my lab-mate who has been a friend and a mentor. From getting me started to providing technical direction, you have been a great support system throughout. I also thank all the alumni of the Microdynamic Systems Lab who have made it one of the best places for robotics. I would also like to take this opportunity to thank my other committee member Professor Jean Oh for taking the time to evaluate this work.

The friends I developed at CMU have played a significant role at various stages of my work. I thank Raghav and Krishna not only for being great friends but also for taking active interest in my work and providing valuable suggestions. I would especially like to thank Siva without whom it would have been impossible to conduct experiments during the lockdown. It is rare to find a person willing to stay up all night for hours at stretch to help conduct experiments.

My entire graduate experience would not have been possible without my amazing brother who has been a pillar of support and an inspiration throughout my life. Also, I take this opportunity to thank the two most important people in my life - my parents for their unconditional love and unwavering support. I would also like to thank the members of my extended family for always being by my side. Finally, I thank the almighty for this wonderful opportunity.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

The significance of mobile robots in human lives is increasing every day. We see them used in many applications such as inventory management, service applications and agriculture. One particular class of robots are those that exist in human environments and accomplish their task by interacting with human beings. In addition to being safe and agile, they need to possess efficient human interaction capabilities to function effectively. This thesis focuses on developing such a system that possesses the qualities of agile mobility, robust navigation and efficient human interaction. In this work, we achieve this goal by using a ballbot-type robot [3], which possesses unique characteristics required to function in human environments.

Ballbots are a class of dynamically stable robots that balance and navigate on a single spherical wheel. This mechanism provides them with increased agility, which make them apt to function in human environments. However, being dynamically stable, they also present significant challenges in areas of motion planning, control and human robot interaction. Figure 1.1 shows an image of two ballbot-type robots balancing using a single spherical wheel.

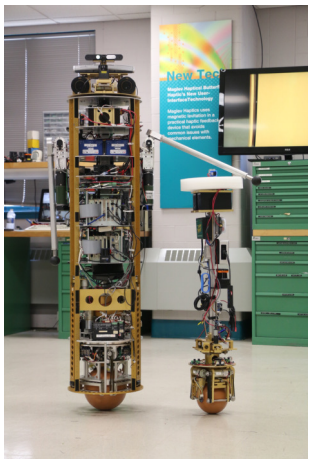


Figure 1.1: The person-sized CMU ballbot next to the a smaller ballbot, both balancing and station keeping

## 1.2 Thesis Objectives

The primary objective of this thesis is to develop a system that navigates autonomously in human environments, interacts effectively with humans, and performs desired tasks. We achieve this goal by developing and integrating efficient technologies into the robot. In this section, a brief overview of the system along with the goals of this work are given whereas subsequent chapters demonstrate each component in detail.

### 1.2.1 Mobile Robot Platform

The mobile robot platform used in this thesis is a ballbot-type robot. Ballbots are dynamically stable and possess many desirable qualities. They can be tall enough to interact with people at the eye level and slender enough to navigate in human crowds. Also, they can move omnidirectionally; a capability that is highly desirable for navigating in cluttered spaces. Perhaps the most desirable quality of ballbots is their intrinsic omnidirectional compliance. By virtue of their balancing controllers, ballbots can be easily moved by the force of a single finger, yet cannot be upset by a strong push. A detailed description of ballbots is given in [4] where many of their capabilities are described. Recently, a fleet of three smaller ballbots called shmoobots were developed (named after cartoonist Al Capp's famous Shmoo creation).[1] These robots are cost effective and have a superior mechanical design. Throughout this thesis, we will use use the shmoobot shown in Fig. 1.2 in order to develop the required system. Further details about the platform are explained in chapter 3.

### 1.2.2 Motion Planning and Navigation

The shmoobot is a dynamically stable system that performs work constantly to stay balanced on its spherical wheel. Due to this unique mechanism, standard trajectory generation approaches suitable for statically stable robots do not apply. To address this challenge, Michael Shomin and his colleagues modeled the system as 'differentially flat' [1] ; a formulation which simplified its trajectory generation process [5]. Using this formulation, they describe methods to generate trajectories and plan paths to accomplish autonomous navigation. Along with point-to-point motion planning, the authors also demonstrate a mechanism for collision avoidance where a new path is generated whenever an obstacle appears in the shmoobot's current path. This approach, where dynamically moving obstacles are assumed as static obstacles in the re-planning stage introduces many inconsistencies in the system. Here, the shmoobot is forced to change its trajectory even when an obstacle is not on a collision path. Also, the newly generated path does not guarantee collision avoidance with the obstacle. In addition to unwanted path modifications, this behaviour also leads to constant changes in the planned trajectory, which can result in loss of smoothness during navigation.

One of the main goals of this thesis, is to address this challenge by developing an obstacle avoidance mechanism which models human beings as dynamically moving agents. In this

[1] Shmoo: `http://lil-abner.com/the-shmoo`

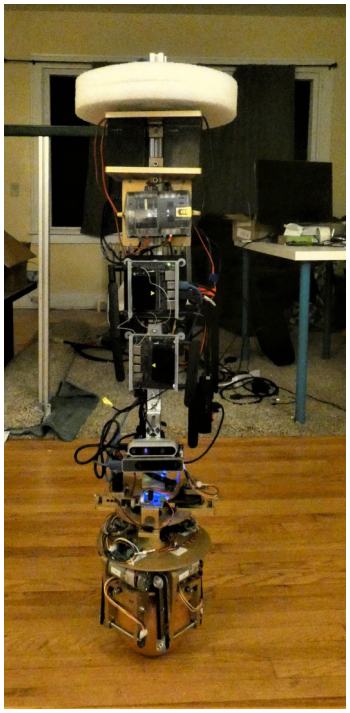


Figure 1.2: The shmoobot, balancing and station keeping on a single spherical wheel

approach, the velocity of a human obstacle is taken into consideration resulting in improved collision avoidance behaviours.

### 1.2.3 Perception

A primary requirement of the shmoobot system is the knowledge of humans in the environment. This is necessary for the shmoobot to plan paths and execute collision avoidance maneuvers. To provide this information, a human tracking system that provides fast and reliable measurements is developed. Chapter 4 discusses this system in detail.

Another important component of the robot's perception system is Simultaneous Localisation And Mapping (SLAM). To achieve this, the open- source package, Real Time Appearance Based Mapping (RTAB-Map) [6] is integrated with the shmoobot. The SLAM algorithm uses information from a depth camera and a visual-inertial odometry sensor to derive accurate information about the environment.

One of the goals of this work is to develop the shmoobot system in a cost effective manner. In that regard, the perception framework is designed to use only cameras as sensors. This is a low-cost alternative to laser range finders that previously existed on the shmoobot. An additional modification is to introduce low-cost, low-power embedded devices to carry out necessary computations. A detailed description of this setup is given in chapter 3.

### 1.2.4 Human Robot Interaction

This work also aims to design a robotic system that can effectively interact with human beings. An intuitive mode through which such an interaction can be achieved is human speech. Recent advances in speech technology and its vast adoption through devices such as Amazon's Echo Dot and Google Home makes this an exciting prospect. In this work, this capability is extended to the shmoobot by utilising Amazon's Echo Dot and its extensive capabilities. Specifically, *Alexa speech skills* are developed that allow effective communication between human users and the shmoobot. Chapter 6 discusses these skills in detail and describe their implementation for two specific applications.

## 1.3 Approach

In order to achieve the mentioned goals of this work, we develop three important components and integrate them with the shmoobot.

1. **Human identification and tracking** : We develop a lightweight human identification and tracking system that operates efficiently using low-cost, low- power embedded devices. This system provides position and velocity measurements of human beings, which are necessary to perform obstacle avoidance maneuvers.
2. **Motion Planning for dynamic obstacle avoidance** : Leveraging the concept of Velocity Obstacles [7], we augment the existing 'differential-flatness' based motion planner by developing an algorithm to avoid dynamic human obstacles.
3. **Alexa skills for human speech communication** : Using Amazon's Alexa capabilities, we develop speech skills to facilitate humans to communicate with the shmoobot and provide commands to perform necessary tasks.

## 1.4 Experiments

To demonstrate the functioning of the entire system, we design two applications where the shmoobot provides valuable service.

1. **Aiding human users in a retail store** : We develop a robotic system that can aid human customers in a retail store. In order to demonstrate this application, we perform an experiment in a mock retail store where the shmoobot acts as a service agent, helping humans find desired product locations.
2. **Go Look and Tell** : In this application, the shmoobot performs 'go look and tell' tasks in a household environment. Specifically, when a speech command is given, the shmoobot navigates to the requested location (kitchen, bedroom, etc.) and records all the objects it can find. After successfully recording the items, it navigates back to the user and reports the items it identified. While performing this task, it avoids any static or dynamic human obstacles.

## 1.5 Contributions of this work

In this work, we make the following contributions:

(a) We develop a lightweight perception system to identify and track human beings.

(b) We propose and implement a dynamic obstacle avoidance algorithm suitable for the shmoobot.

(c) We develop speech skills for two real world applications using Amazon's Alexa capabilities.

(d) We design a cost efficient hardware setup by using low cost embedded devices and camera sensors.

(e) We integrate the RTAB-Map open-source package with the system to achieve vision based Simultaneous Localisation and Mapping.

(f) We demonstrate an application where the shmoobot helps a user find desired products in a retail store.

(g) We demonstrate the shmoobot functioning in a household environment where it performs 'go, look and tell' tasks.

## 1.6 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 presents an overview of relevant research in fields pertaining to this thesis. Chapter 3 provides a description of the various subsystems of the shmoobot. It also discusses the hardware setup and the system's architecture. Chapter 4 discusses the human identification and tracking system in detail. The system is efficient and provides fast measurements suitable for dynamic obstacle avoidance. Chapter 5 presents the 'Artificial Obstacle based approach' to collision avoidance. It discusses the mathematical formulation of this method and provides a detailed description of the algorithms required for its implementation. Chapter 6 explores two real world applications of the shmoobot in human environments. Additionally, it discusses the Aamazon Alexa speech interaction skills developed for these applications. Finally, chapter 7 presents conclusion remarks of the thesis and discusses research areas for future work.

# Chapter 2

# Related Work

Development of interactive mobile robots is a popular field in robotics. As a result, there is vast literature that covers multiple aspects of their development. In this chapter, we present relevant literature in areas pertaining to this thesis.

Section 2.1 presents existing literature on service robots. This thesis derives inspiration from such robots and intends to develop a system with similar capabilities. While most service robots are statically stable, this work aims to develop a system that is dynamically stable, thereby enhancing the human experience.

Section 2.2 discusses popular implementations of human tracking systems that are commonly used. Most of these algorithms rely on multiple sensors to obtain required measurements. However, the human tracking system used in this work relies on a single RGBD camera. Additionally, this section highlights advantages of adopting such a system.

Section 2.3 introduces the concept of Velocity Obstacles for dynamic obstacle avoidance. It also highlights many other popular methods in literature. Also, it emphasises the drawbacks of these methods and establishes the requirement for an avoidance mechanism suitable for the shmoobot.

Finally, section 2.4 highlights the Human Robot Interaction method used in this work. Specifically, it introduces the use of Amazon's Alexa capabilities as an efficient mechanism for human speech interaction.

## 2.1 Service robots

A primary area of robotics research is the development of service robots that actively guide humans and function in unscripted environments. One of the early successful expeditions towards this goal was the development of museum tour guide robots, Rhino [8] and Minerva [9]. Here, the robots interacted with humans and guided them to various places in the museum. These expeditions laid the foundation for many other studies which aimed at developing interactive service robots. More recently, there has been considerable interest from the industry towards the development of such robots. Large retail stores aim to utilise their capabilities to manage inventory, guide customers and provide other valuable services. A few examples are Marty, a

grocery store service robot and Tally, an inventory management robot.[1] These systems have evolved from the museum tour guide robots to have more advanced navigation and perception capabilities. However, all these systems are statically stable, a quality that prevents them from functioning efficiently in human environments. Additionally, they tend to be bulky, which limits their agility. They also require large spaces to operate, which reduces their capability to function in environments with narrow paths and constrained passages.

Similar to the systems described above, this work also attempts to develop a robot that can provide service to human beings. However, it differs in the mobile robot platform that is used. Specifically, we use the dynamically stable shmoobot as it possess significant advantages while operating in human environments. It is agile and can navigate in cluttered spaces owing to its small footprint. Also, its unique design enables it to navigate in a smooth and graceful manner which makes humans feel comfortable in their presence [4].

## 2.2 Human identification and tracking

Human identification and tracking is a vital component of interactive service robots. Most systems achieve this by utilising a sensor stack consisting of laser range finders and depth cameras. One research which effectively uses this idea is the SPENCER project [10]. Here, a pre-trained feature based tracker is used on 2D-Lidar data to obtain initial measurements [11]. This is combined with information from RGBD cameras to estimate humans in close ranges. These measurements are further refined using a robust upper body tracker and a Kalman filter. A detailed description about this mechanism can be found in [10] and [12]. Another popular open source implementation is the ROS-People package[2] which utilises similar components to track humans.

Most tracking methods including those mentioned above utilise more than one sensor to track human beings. Also, in the sensor fusion, they rely heavily on measurements from laser range finders, a costly sensor which we have eliminated in our design. Additionally, having many sensors increases the demand on computational resources, a component that is limited to our system. In this project, we perform the task of tracking humans by utilising a single RGBD camera and a lightweight perception framework. To achieve this, the Simple Online Realtime Tracking (SORT) [13] algorithm, coupled with the efficient SSD-Mobilenet object detection mechanism [14] is used. This setup provides an optimal trade-off between tracking efficiency and computational performance. Chapter 4 discusses this framework in detail and proposes extensions to obtain reliable 3D-measurements of humans in the environment.

## 2.3 Motion planning for dynamic obstacle avoidance

Dynamic obstacle avoidance is a complicated and difficult problem in robotics. There is abundant literature where many approaches and algorithms have been proposed as solutions. Often,

[1] https://www.forbes.com/sites/chriswestfall/2019/04/09/walmart-announces-new-workforce-addition-robots-robotics/#33ab0db716ff

[2] http://wiki.ros.org/people

these algorithms are designed for specific robots, and ensure that avoidance maneuvers that comply with the mechanical aspects of a particular robot are performed. For example, the popular Dynamic Window Approach [15] proposes a solution where planning is performed in the velocity space, followed by an optimisation mechanism that is suitable for differentially driven robots. Other popular open source software such as Timed Elastic Bands[3] also provide trajectories that perform avoidance maneuvers well suited for differential drive robots. For omni-directional robots, the Velocity Obstacle (VO) method [16] and its modifications such as Optimal Reciprocal Collision Avoidance (ORCA) [2] and Extended Velocity Obstacle [17] are used for collision avoidance. Though these methods provide satisfactory results, their avoidance mechanisms are designed for robots driven by velocity/motion commands. This makes them unsuitable for the shmoobot which follows a different navigation mechanism where short segments of optimised polynomial trajectories are constructed in each planning cycle [18]. In this work, we extend the VO algorithm and propose a dynamic obstacle avoidance method suitable for shmoobots.

## 2.4 Human Robot Interaction methods

A variety of interfaces have been proposed in literature for effective human robot interaction. A widely studied area is the use of gestures for non-verbal communication. For example, authors in [19] studied the use of a vision based interface to control a manipulator. Gestures such as 'stop' and 'follow' are read using a camera interface and tasks are performed based on them. Human speech interaction is also a well researched area in robotics. In prior work by Mubin et al. [20], the authors attempt to design a language that humans can use to effectively communicate with the robot. Other service robots such as Pepper[4] use a combination of speech and a display interface to interact with humans. Though the discussed methods have proven to be somewhat effective, we posit that a speech only interface, that utilises natural human language will be more intuitive. Also, using a single interaction modality makes the communication seamless and efficient. In this regard, we integrate the Amazon Echo Dot with the shmoobot and develop necessary 'speech skills' to perform specific tasks. Chapter 6 discusses these skills and associated interaction models in detail.

[3] https://github.com/rst-tu-dortmund/teb_local_planner
[4] https://www.softbankrobotics.com/emea/en/pepper

# Chapter 3

# The Shmoobot system

This chapter gives a description of various components used in the shmoobot system. Initially, an overview of the shmoobot describing its mechanism is given. This is followed by an explanation of the SLAM and Navigation system. Finally, a description of the shmoobot's architecture is provided.

## 3.1 Shmoobot Mechanism

The shmoobot used in this work is a robot that balances and navigates using a single spherical wheel. It is about one meter tall and moves the ball using its 'Inverse Mouse Ball Drive' mechanism (IMBD) [3]. Essentially, a set of rollers (shown in figure 3.1) squeeze the ball in such a way that desired movement is achieved. Through this mechanism, the ball can be moved in any direction giving the shmoobot the ability to navigate omnidirectionally.

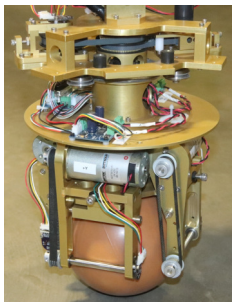


Figure 3.1: Inverse mouse ball drive mechanism

The critical balancing behavior is obtained by assessing the shmoobot's lean angle. In each cycle of the control loop, lean angles from an inertial measurement unit are assessed based on which commands are sent to the motors that control the IMBD mechanism.

The shmoobot also has the ability to yaw its body with respect to the IMBD, which is achieved using another motor-bearing setup. Though the robot does not need to yaw for navigation, this capability is sometimes helpful to orient sensors in a required direction.

### 3.1.1 Controller Design

The shmoobot uses an outer loop PD controller for trajectory following and an inner loop PID controller for balancing. This method is advantageous as limits on the lean angle can be placed ensuring that the robot does not fall over. The essential balancing behaviour of the robot is maintained by the inner loop controller and lean angles provided by the planner are achieved using the outer loop controller. This design has been extensively discussed in [21] and [22] and its advantages have been described. Fig. 3.2 shows the controller design of the shmoobot.

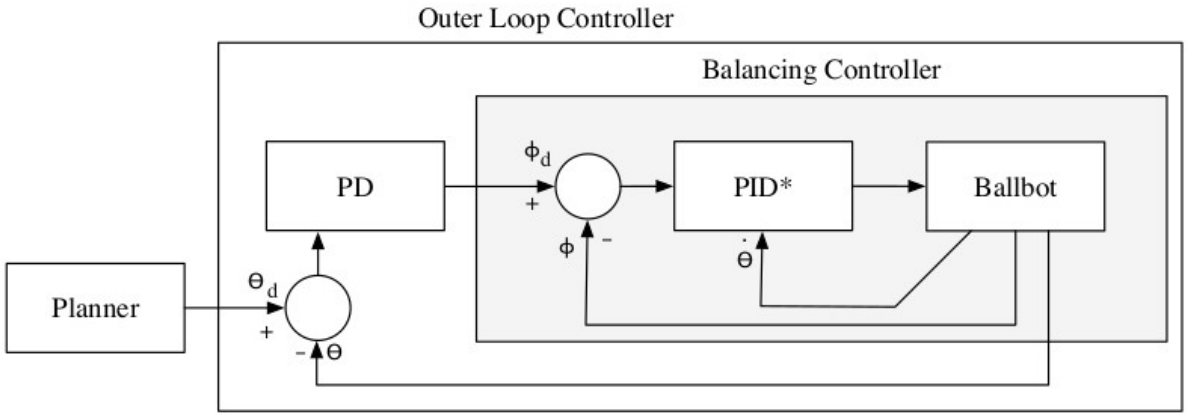



Figure 3.2: Shmoobot controller diagram [1]

## 3.2 Localisation and Mapping

The shmoobot relies on localisation and mapping capabilities for autonomous navigation. This is achieved by using the open-source software, Real-Time Appearance-Based Mapping [23] (RTAB-Map), which is a Graph-Based SLAM algorithm having dense mapping and loop closure capabilities. Additionally, the algorithm uses a memory management approach to optimise the speed and performance of its core processes. As a result, it guarantees real time performance and functions efficiently even on systems with limited computational resources. This makes the algorithm suitable for the shmoobot which uses inexpensive computers with limited resources.

The primary sensor used for the SLAM algorithm is a depth camera which provides RGBD images. Additionally, odometry information is provided using an accurate visual inertial odometry sensor. The RTAB-Map SLAM algorithm and the camera setup are well suited for the shmoobot since the system is designed to operate in closed indoor environments with many visual features.

#### Occupancy grid mapping

An essential component used for navigation is the representation of the environment as a 2D occupancy grid. This requirement is fulfilled by converting the 3D depth images from the camera into corresponding 2D laser scan representations using the *'depthimage_to_laserscan'* ROS

package. The depth images and their corresponding scan representation are used by the system to perform efficient localisation and occupancy grid mapping. The SLAM process is described in Fig. 3.3, and an instance of an occupancy grid mapping session is shown in Fig. 3.4.

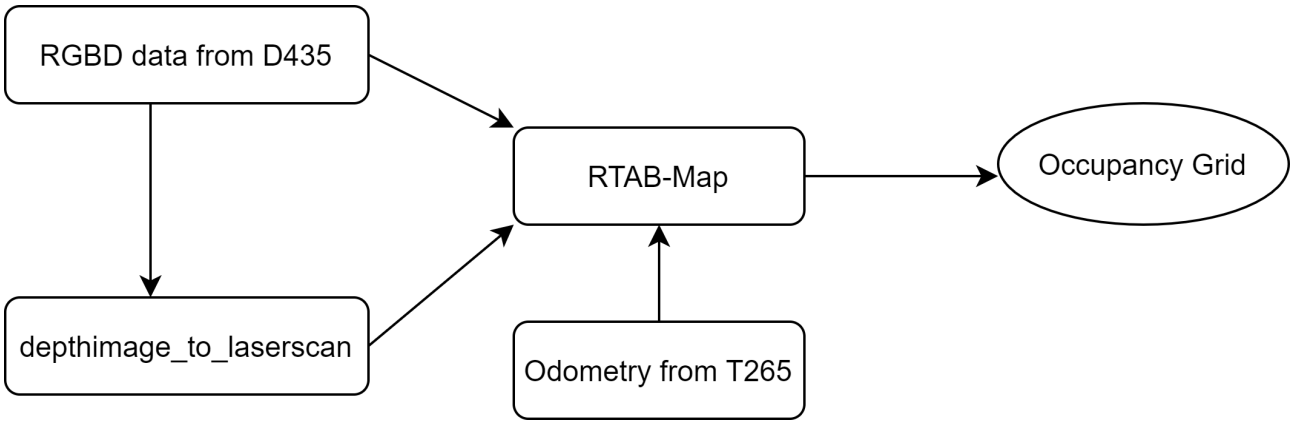
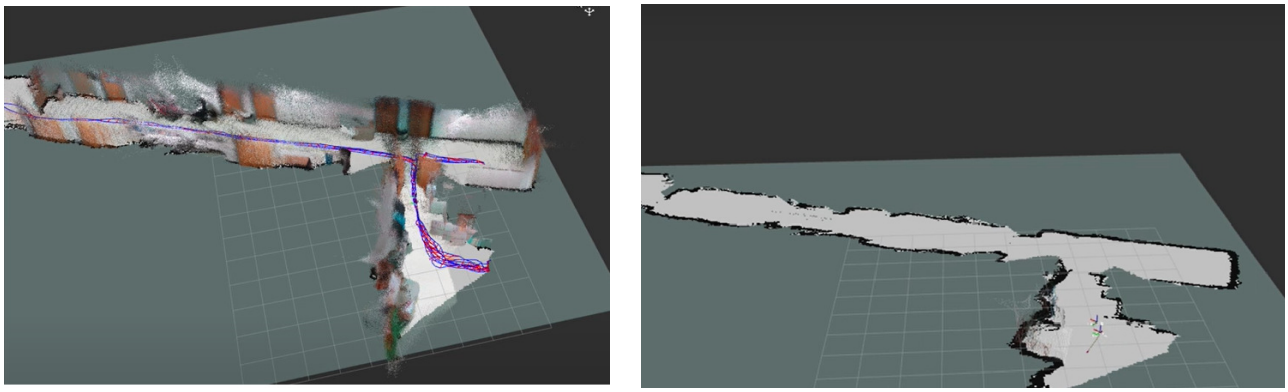



Figure 3.3: Simultaneous Localisation and Mapping process



(a) (b)

Figure 3.4: (a) shows an instance of the mapping session of Smith Hall at Carnegie Mellon University and (b) shows the constructed 2D occupancy grid. The occupancy grid has a resolution of 0.05 metres/cell.

## 3.3 Navigation

The shmoobot navigates autonomously using a differential flatness based motion planner. Here, a short description of the flatness formulation and the corresponding motion planner are provided.

### 3.3.1 Differential Flatness

Differential Flatness [5] is a property of some systems that can reduce the complexity of their trajectory generation process. In this formulation, 'flat output' variables are formulated as functions of state variables and their derivatives. Using this formulation, the trajectory generation process is simpler as trajectories for the flat output variables yield trajectories for state variables and their control inputs. In [18], Shomin and his colleagues use this technique for the shmoobot to create feasible trajectories for autonomous navigation.

#### Trajectory generation and Optimisation

The shmoobot is a system accelerated by its lean angle $\phi$. Also, the state variable required for motion planning is the ball angle $\theta$ as it represents the global position of the robot. To generate a feasible point-to-point trajectory, corresponding trajectories for $\theta$ and $\phi$ are required. Trajectories for these variables are obtained by optimising the robot's differentially flat trajectory.

The flat output trajectory obtained for the shmoobot is a nonic polynomial with many constraints. These constraints represent initial and final configurations of the robot. Once these constraints are set, a feasible trajectory for the robot is obtained by solving a quadratic program that optimises the output trajectory. A detailed mathematical analysis of this process is provided in [1].

### 3.3.2 Ballbot navigation via differential flatness

The motion planner for the shmoobot consists of a global and a local planner. Once a goal is received, a 2D search algorithm provides the path from the robot's initial position to the desired goal. The 2D path is used as a prior to obtain a global flat trajectory after solving the optimisation problem mentioned in the previous section. This is received by the local planner which provides trajectories to the controller at specific time intervals after correcting for localisation errors. In order to achieve this, the local planner first obtains the goal for the next time period ($t_{replan}$) by sampling the global trajectory at a preset time interval ($t_{next}$). Additionally, the planner also receives the updated position of the shmoobot from the SLAM system. Using this information, it creates an optimised trajectory from the shmoobot's current position to the sampled goal for the present time interval. This process continues until the shmoobot reaches its desired location. Fig. 3.5 shows the architecture of the motion planning system on the shmoobot. A detailed explanation of this method can be found in [24] and [1].

The entire planner is implemented on the system by integrating it with the ROS-Navstack.[1] This integration enables the motion planner to use *costmaps* which is a grid representation of

[1] `http://wiki.ros.org/navigation`

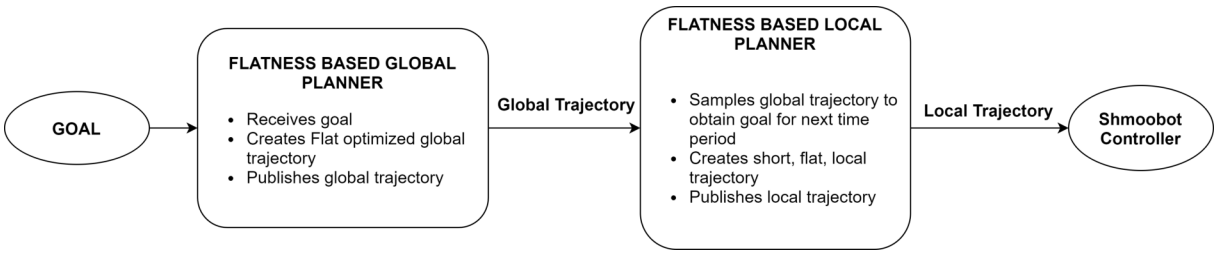



Figure 3.5: Motion Planning architecture

the surrounding environment. Through this, goals can be easily published to the planner by providing their corresponding location on the costmap.

The pseudo code for the global and local planner are provided in Algorithms 1 and 2. The dynamic obstacle avoidance mechanism in Chapter 4 provides further details on the global planner and explains how avoidance maneuvers are performed using information from the global trajectory.

**Algorithm 1:** Pseudo code for global planner

**Input:** Goal location on costmap
Obtain current robot's position from SLAM
Create optimised global trajectory
Publish global trajectory

**Algorithm 2:** Pseudo code for local Planner

**Input:** Global trajectory
**while** *goal not reached* **do**
    Obtain current robot location from SLAM
    Obtain goal for next time period $t_{next}$ by sampling global trajectory
    Obtain flat local trajectory to goal from current robot position
    Publish local trajectory to controller
    Wait for time $t_{replan}$
**end**

## 3.4 System Architecture

The entire system uses ROS [25] for its communication. ROS is a middleware which facilitates communication between different pieces of computer software. As a result, all software created for this thesis is ROS compatible.

The systems in the shmoobot can be categorised into two divisions. The low-level system consists of the tasks responsible for the balancing of the shmoobot and the high-level system consists of those that include SLAM, motion planning, human tracking and speech.

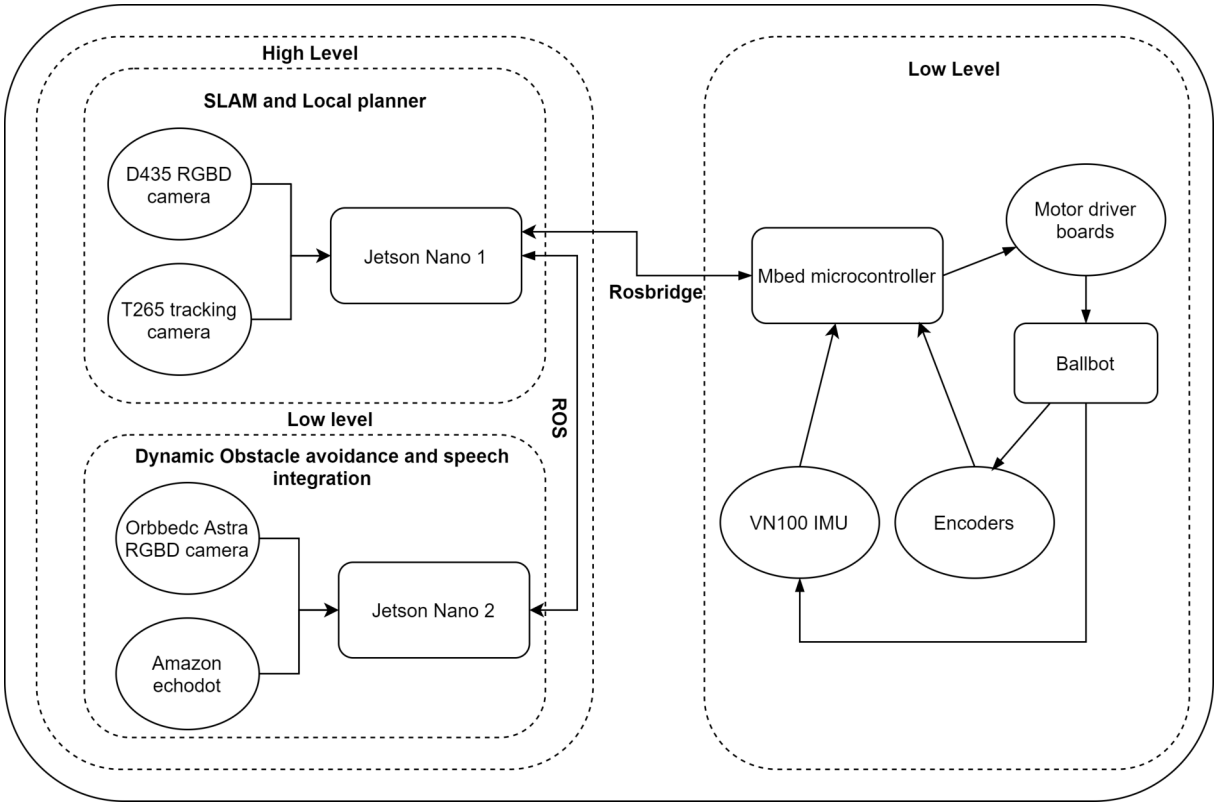



Figure 3.6: Shmoobot System Architecture

All the software for the low-level balancing controller runs on the NXP LPC1768 mbed microcontroller. The microcontroller communicates with a VectorNav VN-100 Inertial Measurement Unit (IMU) to asses the robot's lean angle. Based on this, it sends commands to the motor drivers that control the IMBD mechanism to achieve the balancing behaviour. The control loop that is responsible to achieve this behavior functions at 500 Hz.

All the other software is distributed between the two on-board Jetson Nano computers. These devices are low cost, yet suitable for the shmoobot to function efficiently. The first computer performs the primary task of establishing ROS and initiating communication with the low-level system. Also, it provides commands to the NXP LPC1768 microcontroller to start/stop the robot or to set it in navigation mode (outer loop control). Another significant function of this computer is to house the RTAB-Map SLAM algorithm and the differential flatness based motion planner. In order to achieve this, it communicates with an Intel RealSense D435 RGBD camera and an Intel RealSense T265 tracking camera through USB 3.0 connections.

The second computer performs the human tracking and dynamic obstacle avoidance for which it communicates with an Orbbec-Astra RGBD camera. Additionally, it also communicates with an Amazon Echo Dot device for human speech integration. Fig. 3.6 represents the architecture of the shmoobot system and Fig. 3.7 describes the various hardware components that are used.

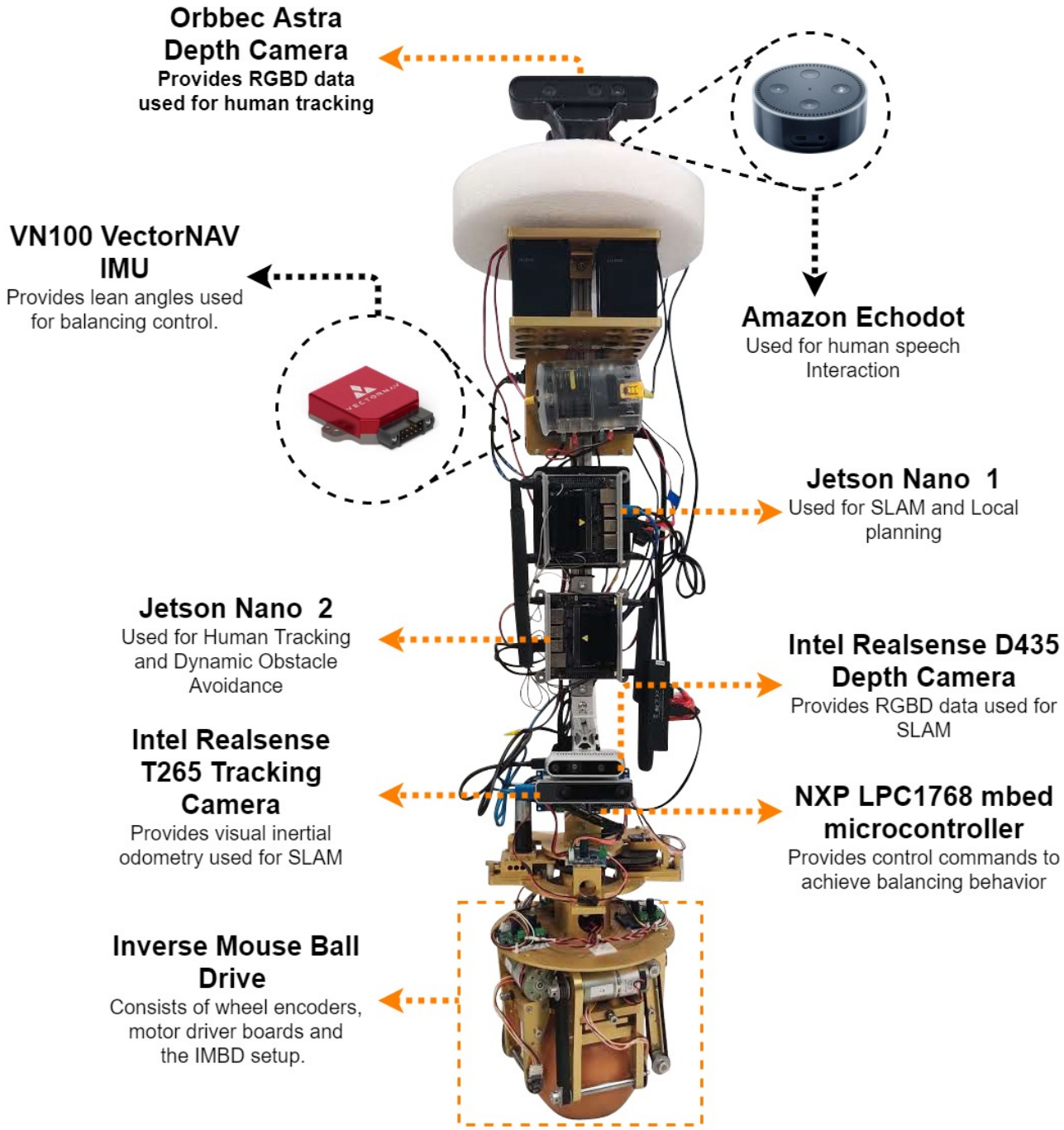



Figure 3.7: The shmoobot system and its hardware components. Components within the black dotted circle are behind the robot.

# Chapter 4

# Human Tracking

This chapter gives a description of the human tracking system used on the shmoobot. It relies on the efficient Simple Online And Realtime Tracking [13] (SORT) algorithm and the SSD-Mobilenet [14] object detection mechanism to perform efficient tracking. Additionally, a Kalman filter is used to obtain reliable 3D position and velocity measurements of humans in the environment. The system is lightweight and renders fast measurements on the Jetson Nano device used on the shmoobot.

## 4.1 Simple Online And Realtime Tracking

Simple Online And Realtime Tracking [13] is an efficient algorithm used for the task of multiple object tracking. In order to perform successful tracking, the task is formulated as a data association problem, where the goal is to associate object detections across frames.

When the camera renders a frame, an efficient detection mechanism is used to obtain the bounding boxes of required objects. This is followed by a data association process which takes place in two phases. First, a Kalman filter [26] is used to estimate the positions of tracked objects based on their measurements in the previous frame. Next, these estimates along with the obtained bounding box measurements are fed to a data association algorithm. This algorithm provides an optimal mapping for each detected object in the current frame to an object in the previous frame. After data association, a filtration process is used to obtain the final tracked measurements. A detailed explanation of this process can be found in [13].

#### Kalman Filter design

The Kalman filter used in the SORT algorithm is based on a constant velocity model that approximates displacements of detections across frames. To estimate this, the state of each obeject is represented as

$$\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]$$

where $u$ represents the horizontal pixel location of the centre of the object, $v$ represents the vertical pixel location of the centre of the object, while $s$ and $r$ represent the scale and aspect ratio (a constant value) of the object's bounding box respectively.

**Data Association**

The data association process provides a mapping from current frame detections to previous frame detections. To achieve this, the Hungarian Algorithm [27], which provides a solution in polynomial time is used. The Hungarian Algorithm requires the association problem to be depicted by a cost matrix [27]. This matrix represents the similarity between each pair of detections in the current and previous frames. In order to obtain similarity values, the 'Intersection-Over-Union (IOU)' metric between bounding boxes is calculated.[1] Once the cost matrix is populated with required IOU values, it is fed to the Hungarian Algorithm, which provides an optimal association for each detected object in the current frame.

## 4.2 Object Detection

The SORT algorithm requires object detections to be made in each frame. In the original implementation of SORT [13], the authors use Faster Region CNN (Faster R-CNN) [28] as the mechanism for object detection. Though this network is fast, it isn't fast enough to provide detections at a satisfactory rate on the Jetson Nano device. Therefore, in this work, we use SSD-Mobilenet v2 [14], a much faster object detection mechanism. An important characteristic of SSD-Mobilenets is that they provide the best trade-off between accuracy and speed. This attribute makes them extremely suitable to be deployed on the Jetson Nano device used on the shmoobot. Additionally, we use the TensorRT[2] version of the SSD Mobilenet, an optimisation that ensures high-performance inference on NVIDIA's graphics processing units (GPUs). This modification helps the tracking to function at a high rate of 15 HZ.

## 4.3 Obtaining 3D pose estimations

The SORT algorithm tracks object detections across frames and provides their 2D locations. In order to obtain 3D poses, the output from the SORT algorithm along with associated depth information is fed to a 3D Kalman filter. This filter is also based on a constant velocity model where the state is represented as

$$\mathbf{x_{3d}} = [x, y, z, \dot{x}, \dot{y}, \dot{z}].$$

Here, $x, y$ and $z$ represents the $3D$ position coordinates of detected objects with respect to the robot and $\dot{x}, \dot{y}$ and $\dot{z}$ denote their velocities. In each iteration, this filter is used to obtain the final 3D position and velocity measurements of human beings in the environment.

## 4.4 Process

The entire framework of the human tracking system is represented by Fig. 4.1. As described, the raw RGB frame from the camera is used by the SSD Mobilenet to provide bounding boxes of

[1]The formula for IOU between two bounding boxes is $\frac{area\ of\ overlap}{area\ of\ union}$. A value of 1 indicates a perfect match.

[2]`https://developer.nvidia.com/tensorrt`

detected humans. These detections are fed to the SORT tracker which renders their tracked 2D measurements. The 2D measurements, along with associated depth information is fed to a 3D Kalman filter which provides the updated position and velocity measurements of humans in the environment.

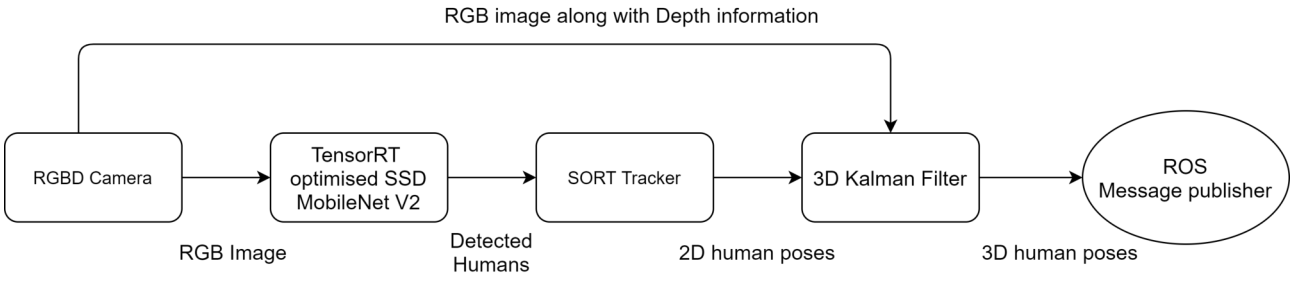
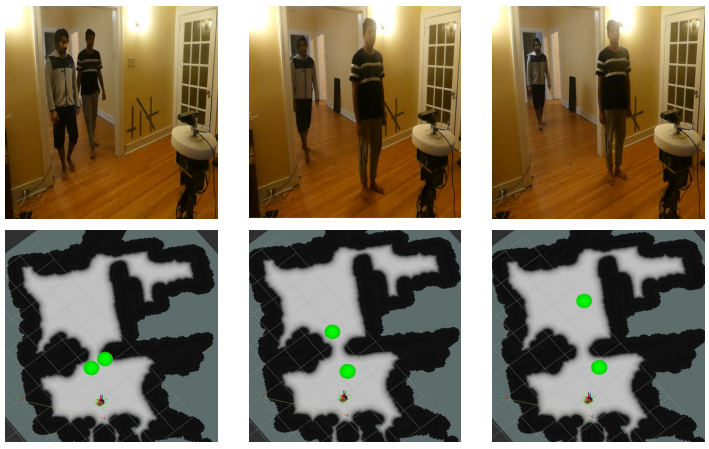



Figure 4.1: Human tracking system

## 4.5 Results

This section shows the results obtained from the human tracking system described above. As shown in Fig. 4.2, the robot tracks the people in its surroundings using the RGBD-camera kept on its top. In addition to the position of the tracked humans, their velocity information is also published as ROS messages.



(a) (b) (c)

Figure 4.2: Images on the top show the actual position of the humans relative to the robot. The bottom images show their tracked state on the rviz visualisation tool. In all the bottom images, the shmoobot is represented by a cylindrical marker whereas the tracked humans are represented by green spherical markers.

## 4.6 Remarks

This chapter presented a mechanism to track human beings in the environment. It is shown to provide fast and reliable measurements on the Jetson Nano embedded devices used on the system. More importantly, it provides measurements at a rate fast enough for the shmoobot to perform dynamic obstacle avoidance.

Though the method discusses human tracking, the underlying framework can be used to track objects of any type. This is possible due to the use of an object detection framework in the tracking process. For example, the system can be easily extended to track pets which can provide a better understanding of the environment.

The core tracking methodology used in this framework is the SORT algorithm. Although it is lightweight, its effectiveness considerably depends on the efficiency of the object tracking mechanism. Specifically, if the detection process fails to provide bounding boxes in some frames, the tracking algorithm will fail to make correct data associations. This limitation can be overcome by using an association metric where the appearance of tracked humans is learnt by the system [29]. Though this method will require extra computational resources, it will reduce the reliance of the SORT algorithm on the object detection process.

# Chapter 5

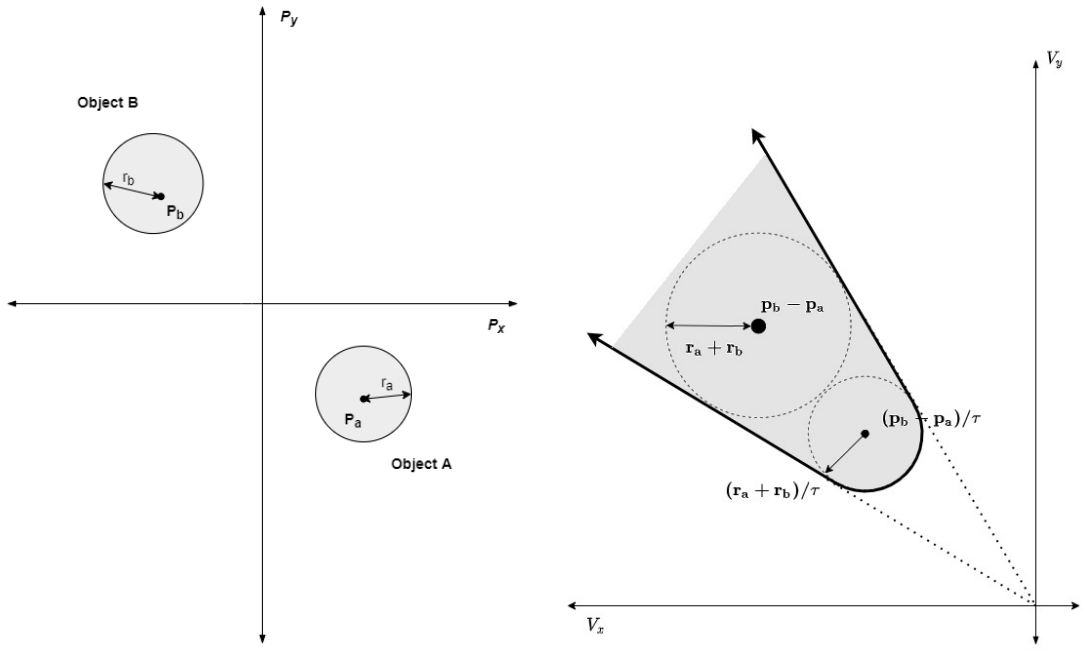
# Collision Avoidance using Artificial Obstacles

This chapter presents a method to perform dynamic obstacle avoidance on the shmoobot. In order to achieve this, we extend the Velocity Obstacle [16] method and develop a planning mechanism that functions seamlessly with the existing differential flatness based motion planner. Essentially, by creating *artificial obstacles* on the costmap, we force the shmoobot to navigate paths that are collision free. In addition to performing collision avoidance, this method also ensures that the smoothness and efficiency provided by the existing motion planner are maintained.

We first provide an overview of the VO method. This is followed by a description of the artificial obstacle based approach to collision avoidance. Finally we present implementation details and experimental results by deploying the method on the shmoobot.

## 5.1 Velocity Obstacle method

For two objects $A$ and $B$, the velocity obstacle $VO^{\tau}_{A|B}$ represents the set of relative velocities of A with respect to B that will result in a collision between $A$ and $B$ at some moment before time $\tau$. $VO^{\tau}_{A|B}$ can be read as the velocity obstacle of object A induced by object B. Specifically, if $v_a$ is the velocity of object $A$ and $v_b$ is the velocity of object $B$, then, a collision between the two bodies will occur at some moment before time $\tau$ if $v_a - v_b \in VO^{\tau}_{A|B}$. Figure 5.1 geometrically shows the velocity obstacle of two objects. It can be seen depicted by a truncated cone with its apex at the origin and legs tangent to a disc that is centred at $p_b - p_a$ and having a radius of $r_a + r_b$. Here, $p_b$ and $p_a$ represents the position of the objects and $r_a$ and $r_b$ represent their respective radii. The amount of truncation of the cone depends on the value of $\tau$. Essentially, if the relative velocity of the two objects falls within the shaded region depicted in Fig. 5.1, they will end up colliding with each other at some time before $\tau$. A formal introduction to the concept along with detailed descriptions can be found in [16] and [2]. In this work, we use the VO method to predict collisions with obstacles and to plan evasive maneuvers.




(a) (b)

Figure 5.1: (a) shows the configuration of two objects. (b) shows the corresponding velocity obstacle $VO^{\tau}_{A|B}$ region represented by the shaded region [2].

## 5.2 Artificial obstacle based approach

In this approach, the dynamic obstacle avoidance problem is divided into two categories; collision detection and collision avoidance. For collision detection, the straightforward Velocity Obstacle method is applied. Whereas, for collision avoidance, the algorithm calculates 2D-positions of predicted collisions and considers them as obstacles for the next planning cycle. When the shmoobot plans to navigate to a goal, it follows a path that avoids these 'artificial obstacles', thereby preventing collisions with actual moving human obstacles. Fig. 5.2 shows an example of the created artificial obstacles on the costmap.

### 5.2.1 Artificial Obstacle generation

To generate the artificial obstacles, we initially obtain the set of achievable robot velocities represented as $RV$. For the shmoobot, this is a set of velocities with same magnitude but different directions ranging from $0$ to $2\pi$. Mathematically, let $s_{avg}$ represent the average speed of the ballbot. Then, in polar coordinates,

$$RV = \{s_{avg}\angle\theta_1, s_{avg}\angle\theta_2 ... s_{avg}\angle\theta_n | \theta_{1-n} \in [0, 2\pi]\}. \tag{5.1}$$

Now, we create an invalid set $IV$ which represents the set of robot velocities that will lead to a collision. For each robot velocity $v_a$ in $RV$, if $v_a - v_b$ belongs in $VO_{A|B}$, then $v_a$ is invalid and will lead to a collision. Therefore, $v_a$ is added to $IV$. Here, $v_b$ represents the obstacle's velocity.

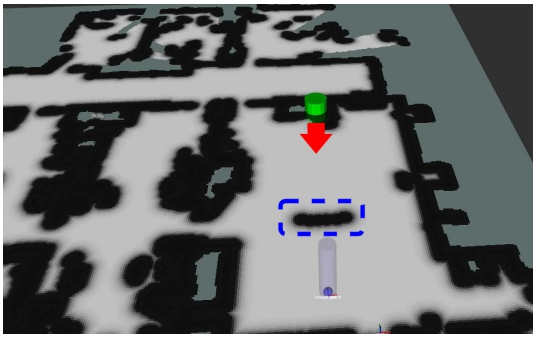


Figure 5.2: The region surrounded by blue dotted lines show the artificial obstacle region that is created. The red arrow indicates the direction of the obstacle's velocity, which is represented by a green cylinder

Note that to find if a velocity is invalid, we check if it belongs to the entire Velocity Obstacle region $VO_{A|B}$ and not the timed Velocity Obstacle region $VO^{\tau}_{A|B}$. Geometrically, this region falls within the entire cone that has its apex at the origin. It is represented by the shaded region in Fig. 5.4. Thus, mathematically, $IV$ is represented as,

$$IV = \{v \,|v \in RV,\ v - v_b \in VO_{A|B}\}. \tag{5.2}$$

Once we have $IV$, for each element, we find corresponding 2D locations on the costmap where the collision would have occurred if the shmoobot had assumed that velocity. These locations will then be assigned an obstacle's cost on the costmap. Fig. 5.3 describes the process to generate the artificial obstacles.

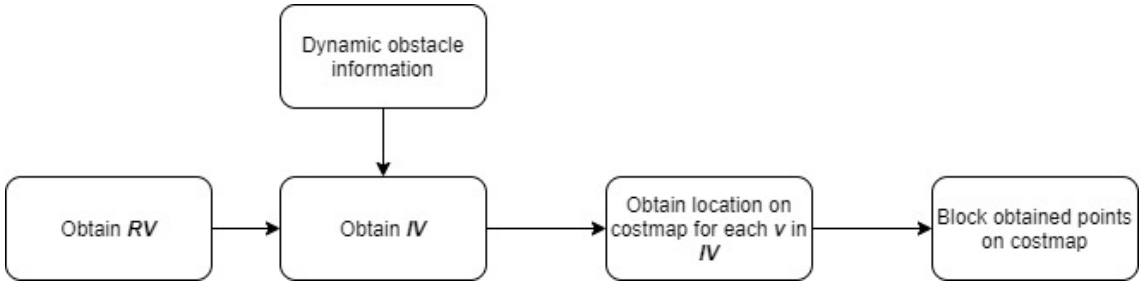



Figure 5.3: Method to create Artificial obstacles

**Multiple obstacles**

The algorithm can easily be extended to avoid multiple obstacles. When we check if a velocity in $RV$ is valid, we perform validations with obtained VO regions of each obstacle in the environment. Correspondingly, the invalid velocity set $IV$ is calculated for each obstacle using which we obtain the artificial obstacle set for all the obstacles in the environment. The algorithm's complexity is O(n) where n is the number of dynamic obstacles.

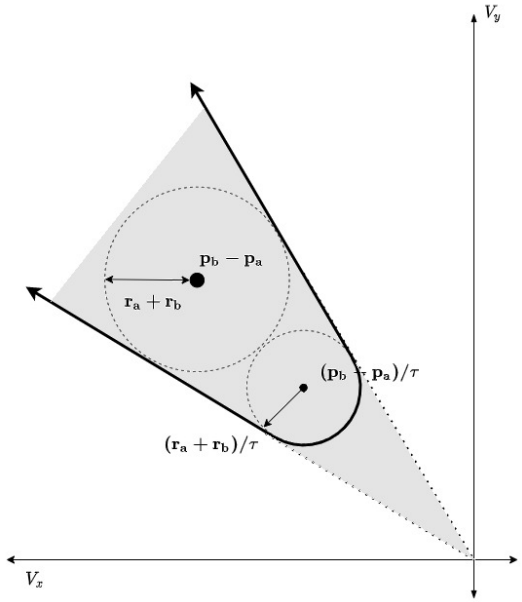



Figure 5.4: The region within the truncated cone (denoted by bold lines) represents $VO^{\tau}_{A|B}$ whereas the entire shaded region represents $VO_{A|B}$.

**Obtaining invalid positions on the map**

In order to obtain collision locations, we formulate the robot and the obstacle as circular rigid bodies moving with constant velocities. By solving motion equations, we obtain the time taken for them to collide. Using this, we can get their corresponding collision locations. Here, we show a brief derivation of this calculation. A detailed explanation can be found through the link in the footnote.[1]

Let us consider two circular rigid bodies as shown in Fig. 5.5. Object A with center at $c_1$ has a radius of $r_1$ and moves with a velocity of $v_1$. Object B with center at $c_2$ has a radius of $r_2$ and moves with a velocity of $v_2$. Let the positions of the objects when collision occurs be $c_1'$ and $c_2'$. We need to find the time taken for the bodies to collide. Therefore, we can write.

$$c_1 + tv_1 = c_1' \tag{5.3}$$
$$c_2 + tv_2 = c_2'. \tag{5.4}$$

Also, the distance between $c_1'$ and $c_2'$ at the time of collision is $(r_1 + r_2)$. This can be written as

$$|c_2' - c_1'| = (r_1 + r_2). \tag{5.5}$$

Squaring both sides, we get,

$$|c_2' - c_1'|^2 = (r_1 + r_2)^2. \tag{5.6}$$

[1] `https://www.cc.gatech.edu/~jarek/graphics/material/collisionsDeshpandeKharsikarPrabhu.pdf`

Substituting the above equation with equation 5.3 and 5.4, we get

$$((c_2 + tv_2) - (c_1 + tv_1))^2 = (r_1 + r_2)^2. \tag{5.7}$$

Rearranging these terms, we get the quadratic equation,

$$(v_2 - v_1)^2.t^2 - 2(c_2 - c_1)(v_2 - v_1)t + ((c_2 - c_1)^2 - (r_1 + r_2)^2) = 0. \tag{5.8}$$

This is of the form $at^2 + bt + c = 0$, where

$$a = (v_2 - v_1)^2; b = 2(c_2 - c_1).(v_2 - v_1); c = ((c_2 - c_1)^2 - (r_1 + r_2)^2).$$

By solving for $t$, we get the time of collision. This can be obtained using the following rules:

1. Find the Discriminant $\sqrt{b^2 - 4ac}$. If this is less than 0, ignore the root. Else if this is greater than 0, solve for $t$ and obtain the value.
2. if two values for $t$ are obtained after step 1, then the min value of both the roots is considered as the solution.

Once we have the value of time taken for the robot to collide, its corresponding position can be easily obtained. If Object A is the robot, equation 5.3 is used to find $c_1'$ which represents the position of the robot at collision time. Now, $c_1'$ will be a location that is assigned an obstacle cost on the robot's costmap.

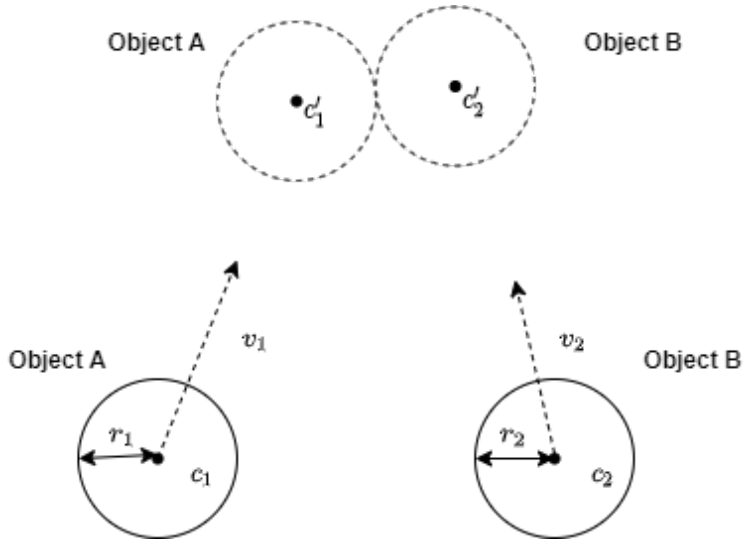
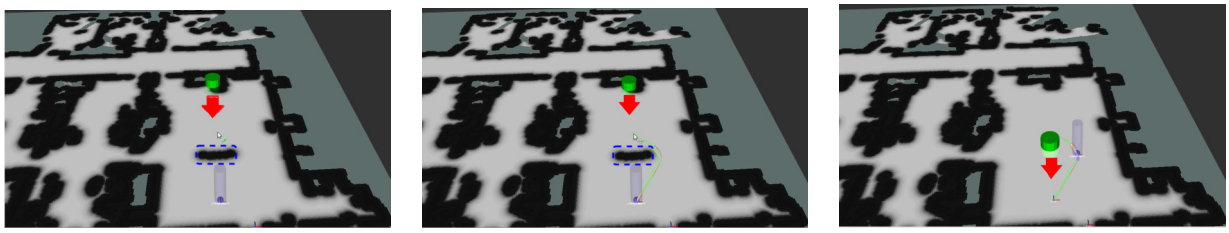



Figure 5.5: Collision between two circular rigid bodies.

The explanation above describes the method to obtain corresponding collision locations for one element in $IV$. By iterating through the entire set, we can obtain the artificial obstacles for all the elements.

This set is the dark region enclosed by the dotted lines in Fig. 5.2. Once this is obtained, the robot plans a path around this region resulting in collision a free path. This behaviour can be seen in Fig. 5.6.



(a) (b) (c)

Figure 5.6: (a) shows the creation of the artificial obstacle region. (b) shows the robot planning a path around the artificial obstacle region and (c) shows the robot navigating in a manner where it avoids collision with the actual obstacle. The blue dotted lines in (a) and (b) represents the created artificial obstacles and the red arrow indicates the obstacle's velocity direction. The red and green lines in (b) and (c) represents that planned path and the trajectory followed respectively.

### 5.2.2 Implementation

In order to function on the shmoobot, the artificial obstacle method is designed to operate as a separate process running in parallel with the flatness based motion planner. To perform collision avoidance, the algorithm communicates with the planner for two purposes. First, it derives information about the shmoobot's speed and velocity from the published global trajectory. Second, it indicates to the global planner when an alternative path has to be chosen. All the remaining processes of the obsacle avoidance algorithm function independently of the motion planner.

A description of the flatness formulation and trajectory generation process was given in Chapter 2. Also, an overview of the shmoobot's motion planner was provided. Here, we expand on this description and provide a detailed explanation of the global planner. This is followed by an explanation of the mechanism used to obtain velocity information from the global trajectory. Finally, implementation details of the obstacle avoidance algorithm along with the combined architecture of the entire process are provided.

#### Differential Flatness based global planner

**Algorithm 3:** Pseudo code for flatness based global planner

**Input:** Goal location on costmap

1 Obtain current robot's position from SLAM
2 Obtain robot plan using a 2D planner
3 Sample plan to obtain waypoints
4 Assign time for waypoints based on chosen $s_{avg}$
5 Obtain optimised flat global trajectory
6 Publish global trajectory

The pseudo code for shmoobot's global planner is provided in Algorithm 3. The planner initially obtains the location of the goal on the costmap. In step 2, it uses this information and creates a path using a 2D search algorithm. In steps 3 and 4, the path is sampled and time values

are assigned between consecutive waypoints. These values represent the shmoobot's navigation duration between the waypoints. In step 5, the algorithm uses this information and creates an optimised flat trajectory. This trajectory contains information about the state of the shmoobot planned for the entire duration of the path. Also, for any instant of time over the course of the path, the global trajectory can be used to obtain the corresponding 2D position that is planned. Finally, in step 6, this trajectory is published as a ROS message. The published global trajectory is used for local planning and dynamic obstacle avoidance. The local planner uses the global trajectory as a prior to provide short trajectory segments in each planning cycle and the obstacle avoidance algorithm uses the trajectory to obtain velocity information for collision checking.

### Obtaining velocity information

An essential requirement for the artificial obstacle method is the knowledge of shmoobot's *current velocity* and *planned speed*. The current velocity refers to the velocity at which the shmoobot is navigating for the present time period. This value is required to check for potential collisions. Whereas, the planned speed is the average speed that the shmoobot is going to assume for future planning cycles. This value is required to generate artificial obstacles on the costmap. Steps to obtain these values are described below.

1. **Planned Speed** : The *planned speed* is obtained from one of the preset parameters used by the shmoobot's global planner. An important process of Algorithm 3 is described in step 4, where time values are allocated between waypoints. These values are chosen in such a manner that the trajectory maintains a preset average speed (shown as $s_{avg}$ in the algorithm). This value is the *planned speed* that is used by the algorithm to create artificial obstacles.
2. **Current velocity** : The *current velocity* of the shmoobot is obtained from the global trajectory, which contains corresponding position information for the entire duration of the path. First, a time interval ($T_{traj}$) is chosen for which the velocity is calculated. Next, the global trajectory is sampled at the chosen time (present time + $T_{traj}$) to obtain the corresponding position ($p_{next}$). Also, the current position of the shmoobot ($p_{present}$) is obtained from the SLAM system. Using this, we obtain the displacement value for the present interval. By dividing this value by the time interval, i.e, ($\frac{p_{next}-p_{present}}{T_{traj}}$), we obtain the value of *current velocity* which is used by the algorithm to perform collision checks.

### Dynamic Obstacle Avoidance Algorithm

The dynamic obstacle avoidance process described in Algorithm 4 requires the global trajectory from the motion planner and the dynamic obstacle list from the human tracking system as inputs. The algorithm first obtains the current velocity of the shmoobot using the previously described steps. Next, it checks for possible collisions with all the dynamic obstacles using methods discussed in previous sections. This process is described in steps 3-10. In steps 12-16, it checks for predicted collisions after which it creates and publishes the artificial obstacle set to modify the costmap. Finally, in steps 17-19, it republishes the goal, indicating to the global planner to make an alternate path. When the planner performs this operation, a new collision free path is

generated. This process continues to happen at a chosen time interval until the goal is reached. The list below provides a description of the parameters used in Algorithm 4.

1. $T_{traj}$ : Duration over which the average velocity of the shmoobot is calculated.
2. $t_{vo}$ : Time for which the VO is calculated. It represents $\tau$ in $VO^{\tau}_{A|B}$.
3. $T_{oa}$ : Time period for the entire obstacle avoidance process.
4. $\theta_{diff}$ : Angle difference by which the interval $[0, 2\pi]$ is sampled.
5. $G$ : Global Trajectory published by the motion planner.
6. $H$ : Dynamic obstacle list published by the human tracking system.
7. $v_{curr}$ : Current velocity of the shmoobot.
8. $H_{collision}$ : List of dynamic obstacles with which collision is predicted.

**Algorithm 4:** Dynamic Obstacle avoidance

```
   Input: G, H
 1 while G consists of trajectories do
 2     Obtain current robot velocity v_curr
 3     H_collision ← empty
 4     isCollision ← false
 5     for each h_i in H do
 6         Calculate VO for time t_vo and velocity v_curr
 7         if collision predicted then
 8             isCollision ← true
 9             add h_i to H_collision
10         end
11     end
12     if isCollision is true then
13         stop robot navigation
14         obtain artificial obstacles for all h_i in H_collision
15         publish artificial obstacles
16     end
17     if robot stationary and goal not reached then
18         publish goal to global planner
19     end
20     Wait for time T_oa
21 end
```

A fundamental process of the avoidance algorithm is described by steps 13-15 where the artificial obstacles are obtained. Algorithm 5 expands on these steps and describes this process in detail. Here, the algorithm iterates through each obstacle in the collision set $H_{collision}$ and creates a map containing the distance value for each invalid velocity. When two distance values are obtained for one invalid velocity, the smaller value is used. Finally, the map, containing the required artificial obstacle information is published.

**Algorithm 5:** Obtaining artificial obstacles

**Input:** $v_{avg}$, $H_{collision}$

```
1  obstacleMap ← empty
2  for i = 0; i ≤ 2π; i = i + θ_diff do
3      for each h_i in H_collision do
4          obtain distance d_i to collision
5          if i is not a key in obstacleMap then
6              obstacleMap[i] = d_i
7          else
8              if obstacleMap[i] < d_i then
9                  obstacleMap[i] = d_i
10             end
11         end
12     end
13 end
14 Publish obstacleMap
```

An important implementation detail of Algorithm 4 is described in step 14 where the robot is commanded to stop if a collision is predicted. Though this behaviour is not necessary to perform obstacle avoidance, it possesses two significant advantages. The first is behavioral in nature. Specifically, when obstacles are blocking all valid paths, the robot will wait in its position which is a desirable behavior. Once a path becomes valid, the robot will continue navigating towards it's goal. The next advantage is practical in nature. When the robot is commanded to change its path while navigating to a location, the planner does not always provide a trajectory that results in an immediate change in direction. Sometimes, an indirect path which has a smoother lean angle trajectory is provided. This indirect path sometimes may not be a collision free. These scenarios happen especially when the planned collision avoiding path narrowly misses a dynamic obstacle. However, when starting from rest, the robot immediately assumes the newly generated path which ensures that narrow collisions are also avoided.

An essential component in the entire process is the creation of artificial obstacles on the costmap. In order to achieve this, layered costmaps [30] are used. Layered costmaps processes the costmap data in semantic layers which makes it easy to maintain particular types of obstacles. To add the artificial obstacle information to the costmap, we maintain a *dynamic obstacle layer* which modifies the master costmap based on the generated obstacle list in Algorithm 5.

### Combined Motion planning framework

The fundamental architecture for the flatness based motion planner was discussed in Chapter 3. Fig. 5.7 shows the combined motion planning framework describing the integration of the flatness based motion planner with the obstacle avoidance mechanism . The flow of information between various processes is also shown.

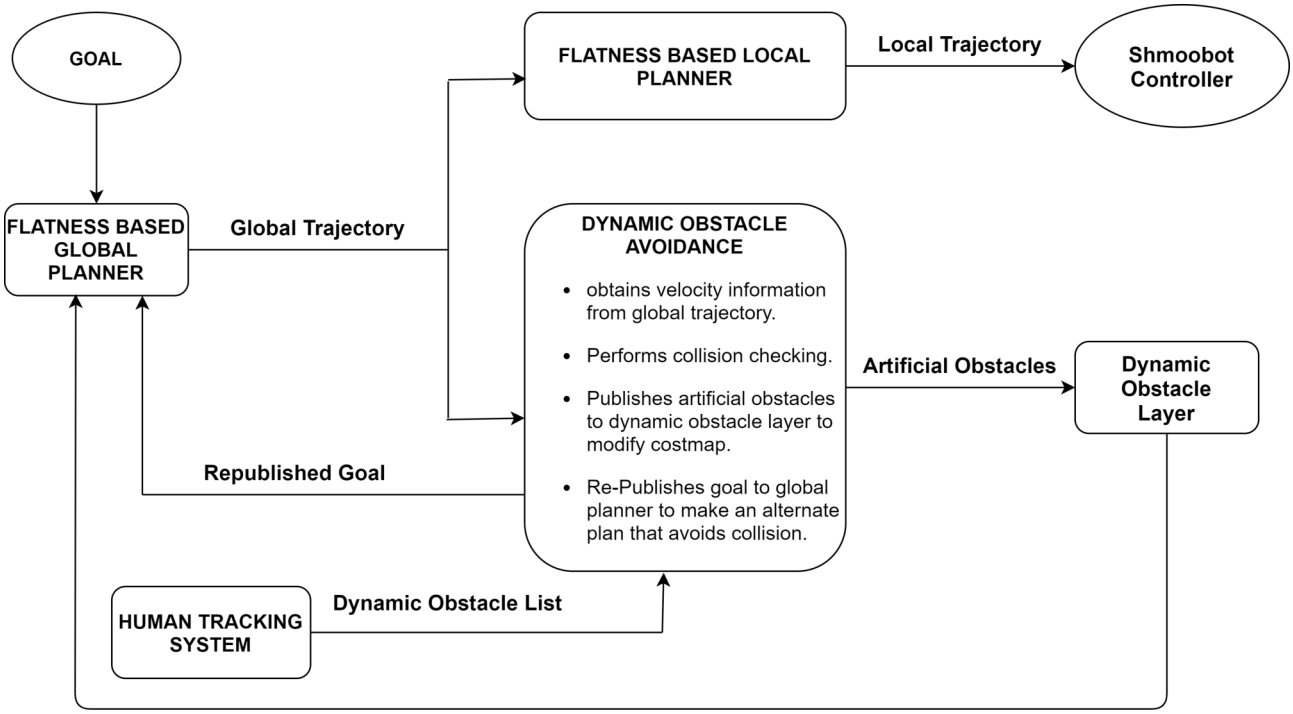



Figure 5.7: Combined Motion Planning framework

## 5.3 Experiments

Experiments were performed by integrating the algorithm with the shmoobot. Initially, a goal on the created map is published after which the robot plans a path to the goal. As it encounters an obstacle, it stops, and modifies the costmap with generated artificial obstacles. Subsequently, it plans a collision free path, and finally reaches its goal. Images in Fig. 5.8 show the various stages of the obstacle avoidance algorithm.

(a) (b) (c) (d)

Figure 5.8: The top images represent the state of the robot and the obstacle on the map whereas the bottom images show their actual positions. (a) shows the shmoobot navigating after a goal is published on the costmap. (b) shows the artificial obstacle region that is generated to avoid the encountered obstacle. (c) shows the shmoobot avoiding collision with the obstacle. (d) shows the shmoobot reaching its desired goal. In all the top images, the cylindrical marker indicates the shmoobot, whereas the green sphere indicates the human.

## 5.4 Remarks

This chapter presents a method to perform dynamic obstacle avoidance on the shmoobot. The method has been shown to avoid obstacles by planning collision free trajectories. A key aspect of this algorithm is that it directly modifies the costmap to block collision locations. This leaves the flatness based planner intact and preserves its smoothness and efficiency qualities.

The performance of the algorithm depends on the number of obstacles in the environment. Having lesser obstacles leads to a faster creation time of the artificial obstacle set. Therefore this method is well suited for small and medium crowds. However, it can be extended to manage crowded environments by treating groups of people as obstacles and estimating future collision regions for the entire group. Using such a strategy could create a more robust method capable of managing crowded environments as well.

# Chapter 6

# Applications

This chapter discusses two real world applications of the shmoobot where it functions autonomously in human environments. The first is a retail store application where it guides a customer to required product locations. In the next, the shmoobot is in a household environment where it performs 'Go, Look and Tell' tasks. For each application, speech interaction skills are developed using Amazon's Alexa capabilities. We first provide a general overview of the Alexa speech interaction model. This is followed by a description of both the applications and their corresponding speech skills. Finally, we provide results showing the usage of the shmoobot in these applications.

## 6.1 Human speech interaction using Amazon Alexa skills

The shmoobot uses Amazon's Alexa capabilities which follows a *request-response*[1] model to communicate with human users. When a user speaks to an Alexa enabled device, the speech is streamed to the Alexa service on the cloud. This service recognizes the speech, deciphers its meaning and sends structured *requests* to a hosted server. The server fulfills the request and replies with a structured *response*. An application that uses these capabilities requires the development of an *Alexa Skill* which implements the functionalities mentioned above.

Any Alexa Skill requires the development of two entities; an interaction model and an application logic. These entities perform the task of sending requests and responses respectively.

### 6.1.1 Interaction Model

The interaction model loosely defines the conversational structure between a human user and the Alexa device. It also performs the essential task of deciphering the meaning of a speech command. This is done by categorising the spoken utterance into a set of actions also known as *Intents*. For example, the utterance *"Can you help me find butter?"* can be categorised to a *find* Intent. In order to make this categorisation, the interaction model looks for predefined words or phrases that the user might say. In the above example, the interaction model may look for the phrase *help me find* to categorise the utterance into a *find* Intent. Once this categorisation is made,

[1] `https://en.wikipedia.org/wiki/Request-response`

a structured request in the form of a JSON[2] message is constructed and sent to the application logic.

### 6.1.2 Application Logic

The application logic is responsible to fulfill the user's request and provide an appropriate response. Once it receives a request of a particular Intent type from the interaction model, it performs a predefined set of actions and sends an appropriate response. For instance, in the above example, once a *find request* is received, the application logic can search for the location of butter in its database and provide a response saying *"Of course! It is on aisle 5."* A diagram representing the entire process is shown in Fig. 6.1.

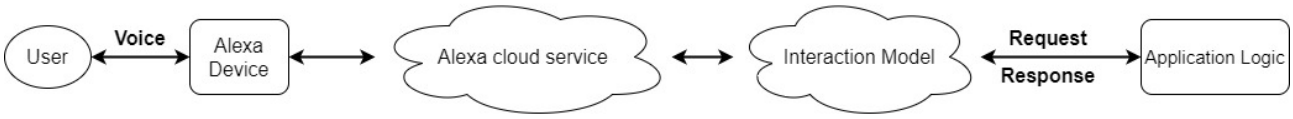



Figure 6.1: Alexa speech process

## 6.2 Applications

Here, we discuss two real world applications developed using the shmoobot. We also provide a description of the corresponding Alexa Skill developed for each application.

### 6.2.1 Retail Store Application

The retail industry has been undergoing major changes for sometime. In recent years, there have been many efforts to automate multiple aspects this industry. For example, many stores have display screens through which details about specific products can be found. More recently, robots have been used to perform passive tasks such as inventory scanning to create an accurate database of the store's products. However, we posit that robots can play a more active role in retail environments by interacting with customers and providing valuable assistance. For example, a robot can greet the customer on their arrival. On request, it can acquire the list of desired items either through a mobile phone application or human speech. The robot can then connect to the store's product location database and actively guide a customer to required items. While doing so, it can also advertise any special offers and discounts about the products. To perform such tasks, the robot would need to posses skills required to communicate with humans and navigate in their environment.

The shmoobot system developed as a part of this work is ideal to perform the tasks mentioned above. It possesses necessary navigation and interaction skills enabling it to function as an effective service agent in a retail store. To explore this possibility, we have developed an experiment

[2] `https://www.json.org/json-en.html`

where the shmoobot communicates through speech and actively guides a customer to product locations.

### Overview

To demonstrate this application, the shmoobot is taken to a mock retail store containing common items such as ketchup, cereals, juices, etc. Initially, a map of the store is made using the Lidar-based Hector Slam algorithm [31] by physically moving the robot around the environment. Once the map is made, the shmoobot is taken to different products in the store and verbally commanded to save their locations. For example, the operator would say *"Store this location as butter"* and the shmoobot would save the location on the constructed map, giving a confirmation saying *"I have saved this location as butter."* The shmoobot can also be verbally commanded to store any details about the product like discounts and offers. Once this is done for various items in the store, the shmoobot is ready to assist customers to find various products. Specifically, the customer would say *"I need butter"* or *"Can you help me find butter?"* and the shmoobot would take them to butter's location. While doing so, the shmoobot will inform the user of details about the product.

### Alexa speech skill

A custom speech skill for the retail application is developed. The developed interaction model consists of six Intents that are used to fulfil a user's request. The application logic server is hosted on the Jetson Nano 2 computer on the shmoobot. It also integrates with the navigation subsystem to publish goals to the motion planner. Table 6.1 provides a list of intents that are developed for this application. Along with the intent, a sample utterance and a reply is also provided.

| **Intent** | **Sample Utterance** | **Sample Reply** |
|---|---|---|
| Welcome | "Hello there" | "Hello there, what would you like to do?" |
| Help | "Can you help me" | "Of course! How can I help you?" |
| Store | "Store this location as {item}" | "Thank you. I have stored the location of {item} |
| Remove | "remove {item} from the list" | "I have successfully deleted the location of {item}" |
| Guide | "take me to the {item} section" | "Let me take you to the location of {item}" |
| Goodbye | "Thanks a lot for your help" | "Thanks for allowing me to help. Have a nice day" |

Table 6.1: Intents for retail application

In the Intents described, the items present in the retail store are referenced through a mechanism called *slots*. This is represented by the paranthesised keyword {*item*} in table 6.1. Slots act as arguments to intents and can assume different values. In this application, the {*item*} slot assumes values of products in the retail store such as bread, butter, juices etc.

### Experiment

In this experiment, the human user is first welcomed by the shmoobot and is offered help. The user then asks for the location of pickles using the utterance *"I need pickles"* for which the

shmoobot responds by saying *"Let me take you to the location of pickles."* Correspondingly, it starts to navigate towards the location of pickles in the retail store. While navigating, the user is also informed about an offer on pickles. Specifically, the shmoobot says *"Also, I would like to let you know that pickles has a 20 percent off."* The shmoobot, then reaches the desired location after which the user picks up the item from the rack. In the same session, the user also requests help to find ketchup, salsa and juices. Fig. 6.2 shows stages of a session where the shmoobot guides the user to find juices in the retail store[3].


Figure 6.2: In (a), the user asks shmoobot's help to find juices. (b) shows the shmoobot responding with an affirmative statement. (c) shows the shmoobot providing details about the product. (d) shows user collecting the required item. In (e), the user thanks the shmoobot for assistance. (f) shows the shmoobot going back to its station and responding with an appropriate greeting.

### 6.2.2 Go Look and Tell

The development of domestic robots is a popular field of research. Many labs and industries around the world have shown considerable interest towards the development of systems that can function as human assistants in household environments. To be effective, these robots need to possess many skills such as graceful navigation and human interaction. The shmoobot system possess many of these qualities making it a suitable system to function in domestic household environments.

[3]The retail store facility used for this experiment was made available to us by Bossa Nova Robotics.

The shmoobot can perform a variety of tasks in a common household. For example, it can be used to look for a lost item (like a house key) by scanning the entire house. In another task, it could transport small items from one place to another using a tray attached to its body. The shmoobot can also perform more complex tasks with manipulators attached to it. For example, it can function as a waiter at a dinner party where it could transport food from the kitchen to various guests in the dining room. First, it could use its speech skills to take down orders from guests. Next, it could navigate to the kitchen by avoiding dynamic human obstacles on the way. Finally, using its manipulators, it can lift a tray containing required items and bring them from the kitchen to corresponding guests. In this work, we have developed one such experiment where the shmoobot performs 'Go, Look and Tell' tasks in a household environment.

### Overview

*Go*, *Look* and *Tell* refers to a set of tasks where the robot navigates to a commanded location and records all the items it can find there. Once this is completed, it navigates back, and informs the user of all the items it could find at that location.

The entire experiment is performed in a household environment with moving human obstacles. To avoid collisions with them, the shmoobot uses its obstacle avoidance algorithm. In order to record items at a particular location, it uses its on-board SSD-Mobilenet Object detection mechanism.

### Alexa speech skill

For this task, an Amazon Skill consisting of five Intents is developed. In addition to Intents for greeting a user, specific Intents responsible for 'Go', 'Look' and 'Tell' tasks are created. Also, an Intent resulting in the shmoobot performing the Go and Look tasks sequentially is created. In addition to providing responses, the application logic interacts with the navigation and the perception systems to perform the Go and Look tasks respectively. Table 6.2 provides a description of the created intents for this experiment. The parenthesised keyword $\{location\}$ is a slot, which assumes values of different locations in the household.

| **Intent** | **Sample Utterance** | **Sample Reponse** |
|---|---|---|
| Welcome | "Launch the robot" | "Hello there, how can I help you?" |
| Go | "Go to the {location}" | "Going to the {location}" |
| Look | "Take a look" | "Taking a look now." |
| Tell | "What did you see?" | "I saw one refrigerator, one table...." |
| GoLook | "Go to the {location}" take a look and come back" | "Going to the {location} |

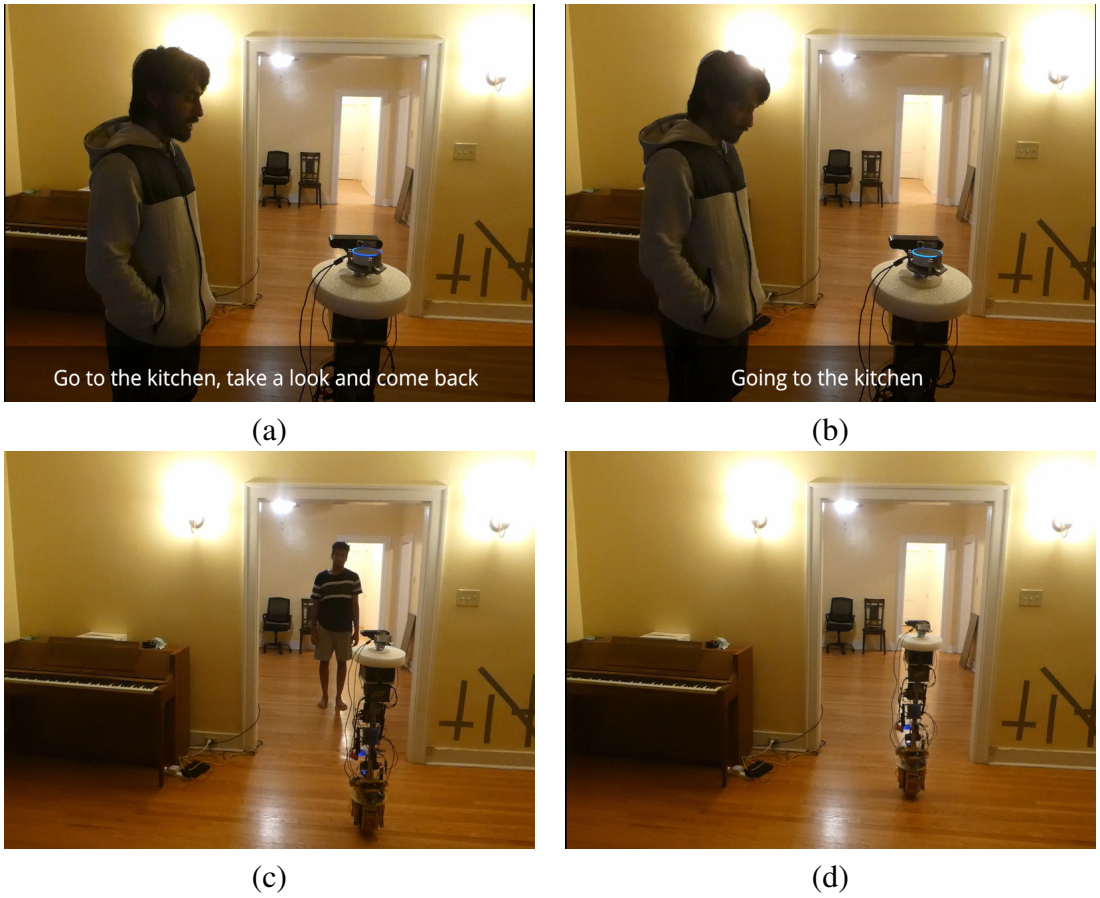
Table 6.2: Intents for Go Look and Tell

### 6.2.3 Experiment

The experiment begins with the user verbally commanding the shmoobot to go to the kitchen and return after recording items it can find. Once the corresponding utterance is interpreted by the speech model, an affirmative reply is provided, after which the shmoobot begins navigating to the kitchen. While navigating, it encounters an obstacle blocking all the valid paths. This is shown in Fig. 6.3 (c). Since there are no paths for the shmoobot to navigate, it waits in its position for the obstacle to move. Once a path becomes available, it resumes navigation towards the kitchen. Images in Fig. 6.3 describe various stages of this sequence.

The shmoobot reaches the kitchen and uses its on-board Orbbec Astra RGBD camera and its SSD-Mobilenet object recognition mechanism to make a list of all the items it can find. Once this process is completed, it returns to the user after avoiding another dynamic obstacle.

Finally, the user asks the shmoobot for the list of items in the kitchen, after which it enumerates the items it has recorded. This sequence of events is shown in Fig. 6.4.




(a) (b)

(c) (d)

Figure 6.3: (a) shows the user commanding the robot to perform the required task. (b) shows the robot responding with an appropriate reply. (c) shows the robot encountering the human obstacle and waiting for a valid path. In (d), the human moves out of the path after which the robot continues navigation.

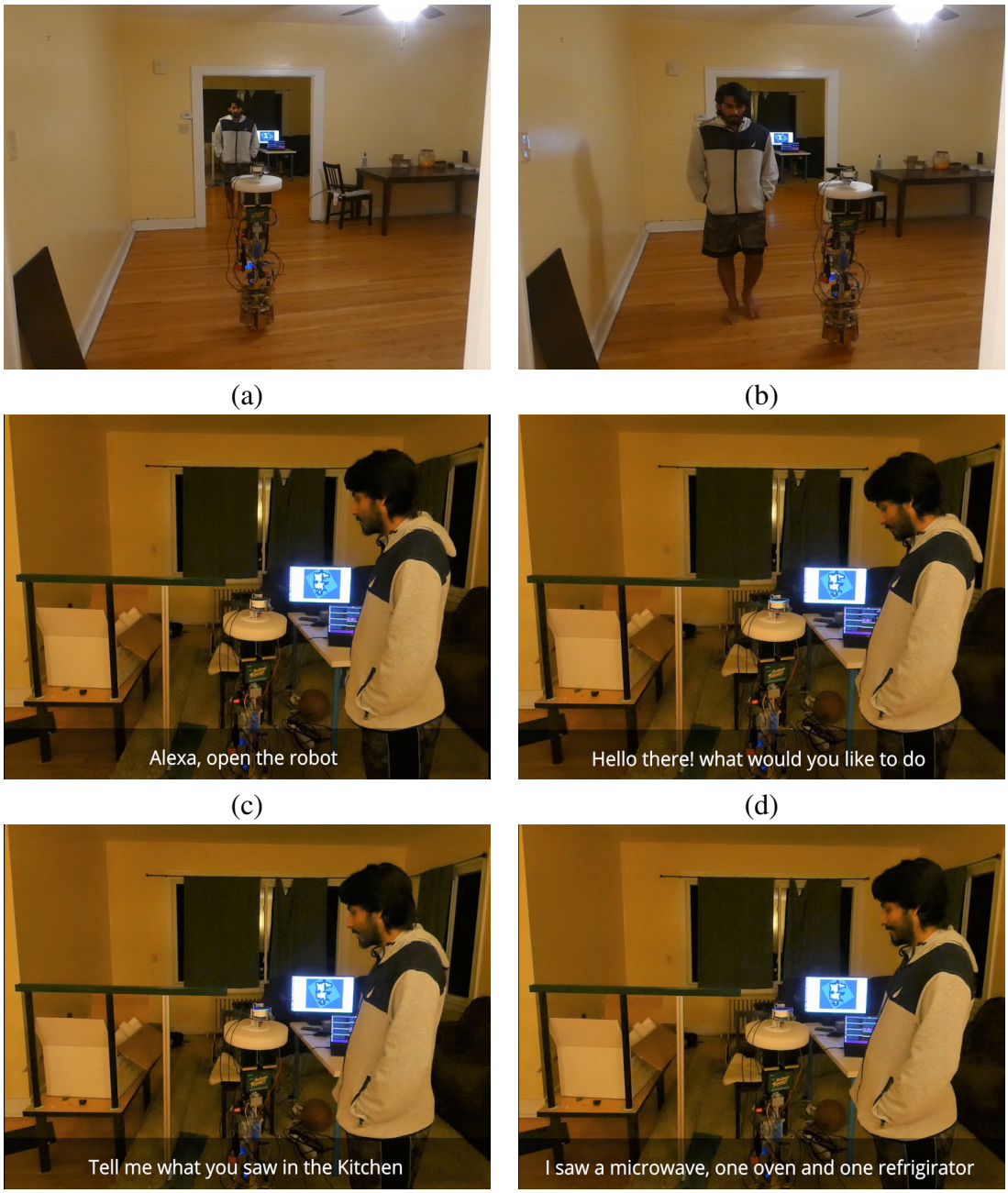



Figure 6.4: (a) shows the shmoobot returning from the kitchen after recording found items. (b) shows the shmoobot encountering a human obstacle and performing successful collision avoidance maneuvers to reach the goal. In (c), the user initiates conversation with the shmoobot. (d) shows the robot responding with an appropriate greeting. (e) shows the user asking the shmoobot to inform him about found items in the kitchen. (d) shows the shmoobot informing the user of all the items it has recorded.

# Chapter 7

# Conclusions and Future Work

This work has explored the potential of the shmoobot as a suitable robotic platform to provide service to humans. The reliance on a single spherical wheel to balance and navigate provides many benefits, especially when operating in cluttered environments. The body of work in this thesis has focused on developing and integrating effective technologies to enhance this system. This chapter enumerates the major contributions of this work and discusses future improvements that can result in an improved system.

## 7.1 Contributions

The major contributions of this work are listed below.

**Developing a low cost hardware setup for the shmoobot**

Throughout this work, we have taken a conscious effort to develop the system in a cost effective manner. This makes the technology accessible to other researchers or even industries which aim to develop on this work. A fundamental step towards this effort is the use of low cost hardware for the robot. Chapter 3 described this setup where low cost Jetson Nano embedded devices and RGBD cameras were used.

**Developing human tracking system using Jetson Nano and RGBD cameras**

The human tracking system discussed in Chapter 4 was developed to function only using information from a camera sensor. This design choice was made in keeping with the hardware setup of the shmoobot.

An important characteristic of this method is that it provides fast measurements which are required by the obstacle avoidance algorithm. To achieve this, the SORT algorithm was extended to obtain required 3D measurements. Additionally, the usage of the SSD-Mobilenet object detection mechanism significantly increased the efficiency of the system.

**Developing Dynamic Obstacle Avoidance algorithm for the shmoobot**

The dynamic obstacle avoidance algorithm discussed in Chapter 5 is a significant contribution of this work. It presented the 'Artificial Obstacle' based approach where avoidance maneuvers were performed by modelling humans as dynamically moving agents. A primary advantage of this method is that it integrates well with the differential flatness based planner on the shmoobot. Also, the system is lightweight as it fundamentally relies on the efficient Velocity Obstacle Method. Though the algorithms presented were specific to ballbot systems, the method can easily be extended to other dynamically stable robots as well.

**Demonstrating Real world Applications with the shmoobot**

In Chapter 6, two real world applications for the shmoobot system were described. Additionally, corresponding Amazon Alexa skills and interaction models were discussed. Both these applications demonstrated the shmoobot system performing simple tasks in human environments. However, the system can be developed to perform more complex tasks in a variety of applications. These experiments illustrate that shmoobot systems possess great potential for widespread usage in unscripted human environments.

**Software Implementation**

All the software code for this work was developed to be ROS compatible. The human tracking system and the dynamic obstacle algorithm were developed using C++ to function efficiently in real time. The Flask micro-web-framework [1], written in Python, was used to host the application logic server required to integrate Amazon Alexa skills with the shmoobot system.

## 7.2 Future Work

There is still significant work yet to be done to realise the full potential of ballbot-type robots. In this section, a few research ideas for potential future work are discussed.

### 7.2.1 Contact Informed Navigation

Though the proposed dynamic obstacle avoidance mechanism proved to be effective, it is not suited for densely crowded scenarios. This is because the costmap modifications which are essential to the algorithm will not be accurate leading to incorrect paths. A natural and efficient way to handle dense crowds is to use physical contact as an instrument to inform motion planning. This is possible with the shmoobot as it is a highly compliant system. Prior work by Shomin et al. [1] demonstrated the use of compliance on the ballbot to modify costmaps. This capability can be extended to motion planning scenarios to handle densely crowded environments.

[1] `https://en.wikipedia.org/wiki/Flask_(web_framework)`

### 7.2.2 Human tracking through Pan and Tilt

A primary disadvantage of the human tracking system is the use of a single fixed RGBD camera. Though cost effective, this reduces the visual range of the robot which can limit the effectiveness of the obstacle avoidance algorithm. One way to overcome this limitation is to have the camera attached to a pan/tilt setup through which the entire environment can be scanned for dynamic obstacles. Development of corresponding algorithms to facilitate this behavior will also be required. Through such a setup, omnidirectional obstacle avoidance behaviours can be achieved. Having this setup will increase the visual range of the robot while maintaining the cost by using only a single camera.

### 7.2.3 Multi Agent Ballbot Systems

The area of multi agent systems is a rapidly flourishing field in robotics. In recent times, many industries are exploring solutions to tasks like warehouse management and package delivery through multi agent systems. We have previously seen that ballbots posses significant advantages that can be useful in the applications mentioned above. For example, two or more ballbots can coordinate and carry a heavy object from one place to another in a warehouse environment. Using ballbots for such tasks will be more efficient than using differential drive robots, that are commonly used for such applications today. Hence, research in the area of 'multi agent planning and control for ballbots' would further their capabilities and provide solutions to many real world problems existing today.

### 7.2.4 Shmoobot as a mobile manipulator

While the shmoobot is a suitable system to function in human environments, its capabilities can be significantly enhanced by introducing low cost manipulators. Prior works such as [32] and [1] explore this possibility by developing suitable applications and associated technologies. In [1], the authors explore a health care application where the ballbot uses its two degree of freedom manipulators to assist a person to transition from sitting to standing. In another work [33], the authors investigate mechanics of the ballbot through which it uses its manipulators to lift heavy objects. More recently, in [34], the authors presented a mechanical design for a seven degree of freedom manipulator that can be deployed on the ballbot. With such a manipulator, they hypothesise that complex tasks such as pushing an elderly person on a wheel chair can be performed. These initial results demonstrate the potential of the ballbot as a mobile manipulator.

A similar manipulator can easily be designed for the shmoobot. With manipulators, the shmoobot can perform more complex service tasks. For example, in the retail application discussed in previous chapters, the shmoobot can pick the items from the shelves and deliver them to people. It can also function as a very effective service agent in indoor environments where it can transport items from one place to another, or perform essential pick and place tasks.

# References


[1] M. Shomin, "Navigation and physical interaction with balancing robots," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, October 2016. (document), 1.2.2, 3.2, 3.3.1, 3.3.2, 7.2.1, 7.2.4

[2] J. van den Berg, S. Guy, M. Lin, and D. Manocha, *Reciprocal n-Body Collision Avoidance*, 04 2011, vol. 70, pp. 3–19. (document), 2.3, 5.1, 5.1

[3] T. Lauwers, G. Kantor, and R. Hollis, *One is enough!*, 05 2007, vol. 28, pp. 327–336. 1.1, 3.1

[4] U. Nagarajan, G. A. Kantor, and R. Hollis, "The ballbot: An omnidirectional balancing mobile robot," in *International Journal of Robotics Research*, May 2014, p. vol. 33 pp. 917–930. 1.2.1, 2.1

[5] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of nonlinear systems: Introductory theory and examples," *International Journal of Control*, vol. 61, pp. 13–27, 06 1995. 1.2.2, 3.3.1

[6] M. Labbé and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 734–745, 2013. 1.2.3

[7] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, pp. 760–, 07 1998. 2

[8] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," in *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, ser. AAAI '98/IAAI '98. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1998, pp. 11–18. [Online]. Available: http://dl.acm.org/citation.cfm?id=295240.295249 2.1

[9] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second-generation museum tour-guide robot," vol. 3, 02 1999, pp. 1999 – 2005 vol.3. 2.1

[10] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore, H. Hung, O. Ramírez, M. Joosse, H. Khambhaita, T. Kucner, B. Leibe, A. Lilienthal, T. Linder, M. Lohse, and L. Zhang, *SPENCER: A Socially Aware Service Robot for Passenger Guidance and Help in Busy Airports*, 03 2016, vol. 113, pp. 607–622. 2.2

[11] K. Arras, S. Grzonka, M. Luber, and W. Burgard, "Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities," 05 2008, pp. 1710–1715. 2.2

[12] O. Hosseini Jafari, D. Mitzel, and B. Leibe, "Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras," 06 2014. 2.2

[13] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," *2016 IEEE International Conference on Image Processing (ICIP)*, Sep 2016. [Online]. Available: http://dx.doi.org/10.1109/ICIP.2016.7533003 2.2, 4, 4.1, 4.2

[14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 06 2018, pp. 4510–4520. 2.2, 4, 4.2

[15] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics & Automation Magazine, IEEE*, vol. 4, pp. 23 – 33, 04 1997. 2.3

[16] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, pp. 760–, 07 1998. 2.3, 5, 5.1

[17] A. Levy, C. Keitel, S. Engel, and J. McLurkin, "The extended velocity obstacle and applying ORCA in the real world," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 16–22. 2.3

[18] M. Shomin and R. Hollis, "Differentially flat trajectory generation for a dynamically stable mobile robot," 05 2013, pp. 4467–4472. 2.3, 3.3.1

[19] O. Oyedotun and A. Khashman, "Deep learning in vision-based static hand gesture recognition," *Neural Computing and Applications*, vol. 28, 04 2016. 2.4

[20] O. Mubin, J. Henderson, and C. Bartneck, "You just do not understand me! speech recognition in human robot interaction," vol. 2014, 08 2014, pp. 637–642. 2.4

[21] U. Nagarajan, G. Kantor, and R. Hollis, "Integrated planning and control for graceful navigation of shape-accelerated underactuated balancing mobile robots," 05 2012. 3.1.1

[22] U. Nagarajan, B. Kim, and R. Hollis, "Planning in high-dimensional shape space for a single-wheeled balancing mobile robot with arms," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 130–135. 3.1.1

[23] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation: LABBÉ and MICHAUD," *Journal of Field Robotics*, vol. 36, 10 2018. 3.2

[24] M. Shomin and R. Hollis, "Fast, dynamic trajectory planning for a dynamically stable mobile robot," 09 2014, pp. 3636–3641. 3.3.2

[25] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009. 3.4

[26] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960. 4.1

[27] H. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, 05 2012. 4.1

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, 06 2015. 4.2

[29] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3645–3649. 4.6

[30] D. Lu, D. Hershberger, and W. Smart, "Layered costmaps for context-sensitive navigation," *IEEE International Conference on Intelligent Robots and Systems*, pp. 709–715, 10 2014. 14

[31] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011, pp. 155–160. 6.2.1

[32] Z. Li and R. Hollis, "Toward a ballbot for physically leading people: A human-centered approach," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4827–4833. 7.2.4

[33] F. Sonnleitner, R. Shu, and R. L. Hollis, "The mechanics and control of leaning to lift heavy objects with a dynamically stable mobile robot," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9264–9270. 7.2.4

[34] R. Shu and R. Hollis, "Development of a humanoid dual arm system for a single spherical wheeled balancing mobile robot," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 2019, pp. 499–504. 7.2.4

# Appendices

## Software packages used

1. ROS Navstack : This package was used to setup the robot's odometry and its frames. The costmap functionality and 2D search was also used through this package.

   `http://wiki.ros.org/navigation`

2. RTAB-Map : This ROS package used to integrate Visual Slam into the robot.

   `http://wiki.ros.org/rtabmap_ros`

3. Librealsense : This ROS package was used to integrate the intel RealSense cameras into the system.

   `http://wiki.ros.org/librealsense`

4. Amazon Developer console: This was used to develop amazon skills for the system.

   `https://developer.amazon.com/`

5. Flask : This micro web framework was used to send responses to the ALexa server.

   `https://flask.palletsprojects.com/en/1.1.x/`

6. depthimage_to_laserscan : This ROS package was used to convert 3D depth images to 2D scan representations.

   `http://wiki.ros.org/depthimage_to_laserscan`

7. Human Tracking: ROS package for human tracking based on Simple And Online Realtime Tracking Algorithm.

   `http://clarinet.msl.ri.cmu.edu:9999/Shreyas/sort.git`

8. Collision Avoidance: ROS package for collision avoidance based on artificial obstacle method.

   `http://clarinet.msl.ri.cmu.edu:9999/Shreyas/ObstacleAvoidance.git`