論文 / 著書情報
Article / Book Information

題目(和文)	在宅酸素療法患者を支援する追従ロボットのテザー制御
Title(English)	Tether Control of Leader Following Robot to Support Home Oxygen Therapy Patients
著者(和文)	アラン ベン
Author(English)	Ben Allan
出典(和文)	学位:博士（工学）, 学位授与機関:東京工業大学, 報告番号:甲第9932号, 授与年月日:2015年6月30日, 学位の種別:課程博士, 審査員:鈴森　康一,遠藤　玄,小田　光茂,大熊　政明,松永　三郎,齋藤　滋規
Citation(English)	Degree:, Conferring organization: Tokyo Institute of Technology, Report number:甲第9932号, Conferred date:2015/6/30, Degree Type:Course doctor, Examiner:,,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Tether Control of Leader Following Robot to Support Home Oxygen Therapy Patients

Thesis by

Ben Allan

In Partial Fulfilment of the Requirements

for the Degree of

Doctor of Engineering

Thesis Advisors:

Koichi Suzumori

Gen Endo

Department of Mechanical and Aerospace Engineering

Tokyo Institute of Technology

Tokyo, Japan

May 2015

Abstract

This thesis aimed to investigate tether control in a mobile robot designed to support Home Oxygen Therapy patients.

Chapter 1 gives the background of the research. Chronic Obstructive Pulmonary Disease (COPD) is a common respiratory condition where airflow through the lungs is restricted, with patients experiencing coughing, wheezing, and shortness of breath. Home Oxygen Therapy (H.O.T.) is a medical treatment frequently prescribed for COPD in which the patients are supplied concentrated oxygen from an oxygen tank or concentrator. An assistive robot can improve the quality of life of H.O.T. patients by carrying the H.O.T. equipment, thus reducing their physical burden and increasing their freedom of movement. A survey of existing academic research and commercial systems showed that a suitable solution did not exist, and consequently there is a need for a robust robot system which can follow the user while also being simple to operate and low-cost. To meet these requirements, this research considers a differentially steered mobile robot with a tether interface, which is evaluated by Home Oxygen Therapy patients as much as possible.

In *Chapter 2*, three leader following control methods are presented with theory and simulated trajectories: *Pseudo-Joystick*, *Follow the Leader*, and *Follow the Leader with Constant Distance*. *Normal path deviation* was also introduced as a metric to compare the accuracy of leader following. An investigation into the effects of control parameters found that *Pseudo-Joystick* control was negatively affected by longer tether lengths, while *Follow the Leader* was unaffected.

Chapter 3 describes experiments with human leaders and a hardware prototype. Motion capture experiments were used to measure and compare the performance of *Pseudo-*

Joystick control and *Follow the Leader with Constant Distance* control under controlled conditions with healthy users. Following on from these, the robot's performance with real Home Oxygen Therapy patients was evaluated; the results of a leader following experiment and a questionnaire survey are presented and analysed. From the practical experiments it was shown that the *Follow the Leader with Constant Distance* algorithm was capable of following the user more accurately than *Pseudo-Joystick*, but both algorithms gave reasonable following performance. The questionnaire survey of Home Oxygen Therapy users identified that overall they found *Follow the Leader with Constant Distance* to be better and found *Pseudo-Joystick* control to be more uncomfortable.

Chapter 4 introduces several additional follower modes designed to improve follower performance in certain situations or address issues with previous control methods. Side following and front following modes were developed to allow the robot to remain in the user's field of vision while following their trajectory. Simulation and motion capture results were presented for two types of side following control: *Side Joystick* mode and *Side Tracking* mode; the latter was shown to have improved performance. Experiments with *Front joystick* mode showed that it could be operated in open spaces, but steering was relatively difficult for the user on more complex courses. *Brake* mode was introduced to improve the safety and usability of the robot, especially in busy environments. The problem of tether snagging was also briefly investigated, showing how a tracked trajectory is distorted if the tether hits an obstacle.

Chapter 5 describes leader following experiments conducted with a Home Oxygen Therapy patient in an outdoor environment. Two participants, one healthy user and one Home Oxygen Therapy user, were asked to walk various routes around their local area (mainly around a park and the nearby train station), while using an assistive device to carry their oxygen tank. Three different devices were compared: a conventional oxygen cart (unpowered); a commercially available cart with powered wheels; and a robot follower. The user's heart rate and oxygen saturation (SpO2) were measured, and these values were then used to compare the effect of each device on the user. The number of participants in this experiment was too few to draw statistically robust conclusions about the robot's performance (a minimum of eight participants is typically required for this). However, the results of the experiment implied that the robot performed comparably to the conventional cart, and sometimes better. This represents an interesting preliminary result which can be used to justify further experiments with a larger number of Home Oxygen Therapy patients in future.

Chapter 6 introduces an airport carrier robot as a further application for the leader following control discussed in previous chapters, demonstrating the generality of the research. The implementation of the control is described along with the results of testing in an outdoor environment.

Chapter 7 summarizes the contribution of the work in this thesis, gives final remarks and ideas about possible future work. In particular, this chapter highlights the work required to progress this area of research from its current state to a finished commercial product ready for use by Home Oxygen Therapy patients. Additional feedback and impressions from working with Home Oxygen Therapy patients are also discussed.

Acknowledgements

I would like to thank my advisor Associate Professor Gen Endo for his guidance, support and kindness. His selfless devotion to his students is truly admirable, and my research would not have been possible without him.

I am also grateful to Professor Koichi Suzumori for his guidance, and to all the members of the Suzumori Laboratory for welcoming me into their lab.

I must also thank both of my previous academic advisors, Professor Shigeo Hirose and Associate Professor Edwardo Fukushima. They provided me with a great deal of inspiration during the early stages of my research.

For hosting my initial academic exchange to Japan in 2006, and for making me feel welcome ever since, I would like to thank Professor Sadayuki Ujihashi. He, along with the students of his laboratory, were an important part of convincing me to return to Japan to study.

Finally, and most importantly, I would like to thank my family. They have always provided immeasurable support, understanding, and encouragement; any success I might have achieved is due to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

1.1.1 COPD

Chronic Obstructive Pulmonary Disease (COPD) is a common respiratory condition where airflow through the lungs is restricted, often involving permanent lung damage, with patients experiencing coughing, wheezing, and shortness of breath. Figure 1.1 shows how the bronchioles are narrowed in a person with COPD, inhibiting their ability to breathe. COPD is an umbrella term, including emphysema and chronic bronchitis, and is usually caused by tobacco smoking (though it can also be caused by exposure to other airborne irritants or pollutants). The World Health Organization reports that COPD is responsible for over 3 million deaths each year, shown in Figure 1.2, making it the fourth most common cause of death globally [1]. The effect on quality of life can be significant: those with severe shortness of breath may be unable to move around without aid, they may be unable to participate in physical activities, and they may suffer from anxiety and depression as a result [2], [3].

1.1.2 Home Oxygen Therapy

Home Oxygen Therapy (H.O.T.) is a form of *Non-Invasive Ventilation* [5] which involves the administration of concentrated oxygen for extended periods (over 15 hours per day). Home Oxygen Therapy can benefit patients with COPD[6] as it aims to further improve

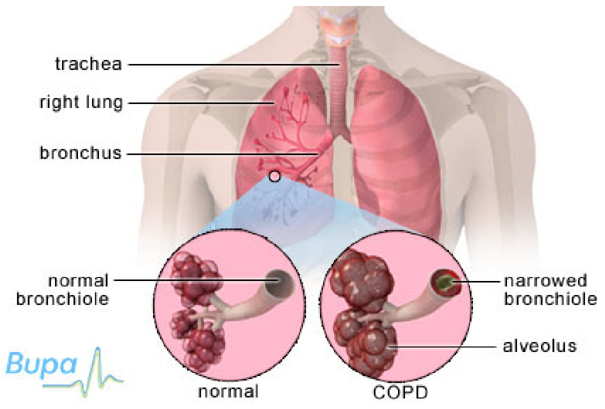


Figure 1.1: A person with COPD has narrowed bronchioles, making it difficult to breathe[4].

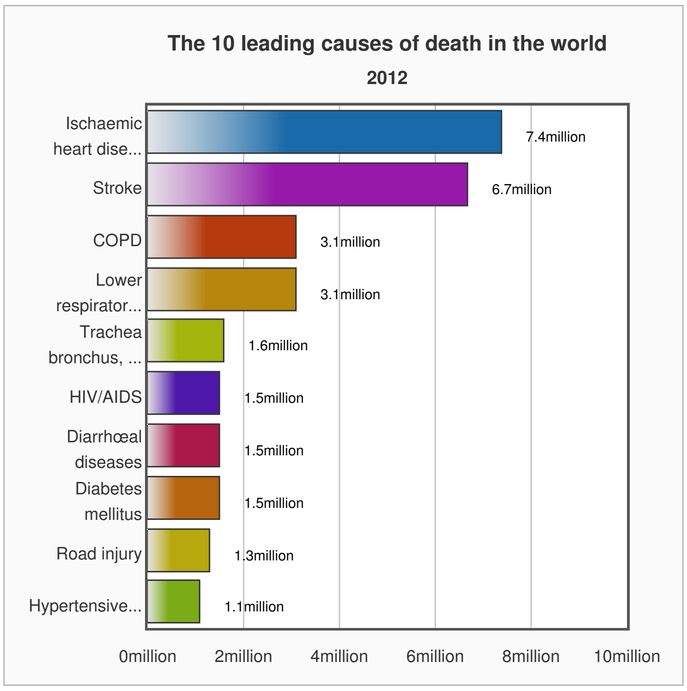
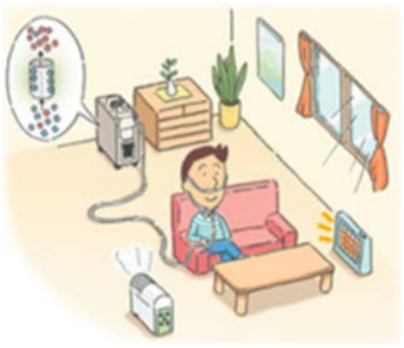


Figure 1.2: Leading causes of death worldwide, as reported by the World Health Organization[1]. COPD causes over 3 million deaths each year.

the patients' freedom and quality of life by allowing treatment outside of hospital[7], and previous research has shown a positive correlation between average daily distance walked and health related quality of life [8].

There are currently around 150,000 people using H.O.T. in Japan, and this number is expected to increase as Japan's population ages in the future. Oxygen is delivered through a mask worn on the face or nose, through a cannula, from a supply which may consists of either a canister of pressurized oxygen, a liquid oxygen tank, or a small oxygen concentrator device. This equipment typically weighs around 4 kg, and when the user leaves the house they can use a small handcart to transport it. Figure 1.3 illustrates the application of H.O.T. indoors and outdoors. Some examples of typical carts and devices used for H.O.T. are shown in Figure 1.4. Despite the benefits of H.O.T., it still imposes considerable restrictions on the users' movement and quality of life, since they must expend valuable effort to carry or pull the H.O.T. equipment.


(a) Indoors with oxygen concentrator

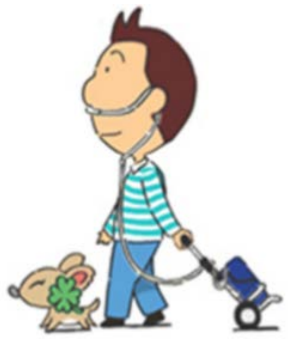

(b) Outdoors with cart and oxygen tank.

Figure 1.3: Home Oxygen Therapy.
Image source: Teijin Pharma.

1.1.3 Support Robot for H.O.T. Patients

Since H.O.T. requires the use of a cannula to supply oxygen, the system is inherently tethered and this represents a good opportunity to use a tethered robot follower. It is

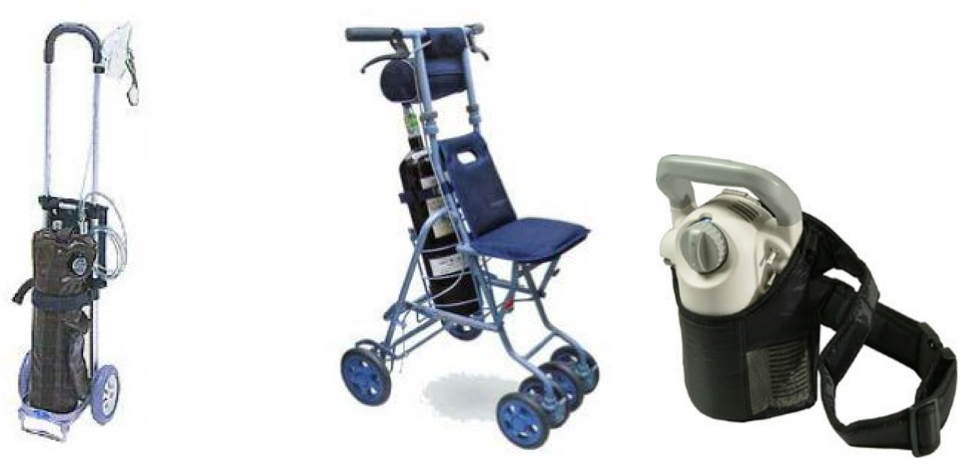

(a) Two-wheeled cart **(b)** Four-wheeled cart **(c)** Carry pouch

Figure 1.4: Home Oxygen Therapy devices.

hoped that a follower robot can improve the quality of life of H.O.T. patients by carrying the H.O.T. equipment, thus reducing their physical burden and increasing their freedom of movement. A typical role for this robot might be assisting an elderly person as they conduct daily activities such as shopping, visiting the doctor and so forth. Leader following is therefore an important area of study, since we can overcome the problem of a hazardous unknown environment by offloading the responsibility for navigation onto another actor with better knowledge. The role of a leader following algorithm is then to ensure that a follower robot can reliably and accurately move with a leader. In many cases, this means we have a follower robot being led by a human leader.

Although the main focus of this research is Home Oxygen Therapy, it may have wider applications. Japan has an aging population, and it is therefore likely that many of the future users of assistive robots will be elderly. Furthermore, the current trends in robotics are for robots to move further from the laboratory into real world environments, and for robots to be operated more and more by non-experts.

When developing a new device or new software, it is vital to consider the circumstances, and wishes of the target user. For a user to operate a device correctly, safely, and easily, it is important that the user of a product can form a suitable ‘mental model’[9]. In other words: the user must understand how the device is controlled not just in terms of what buttons to press etc., but what the underlying effect is. If a device is a ‘black box’ whose model of operation cannot easily be determined, then it is unlikely to be easy to use. It is therefore important that an assistive device can be controlled intuitively, and ideally the target users will also have significant input at the design stage.

Table 1.1 lists some basic requirements which a robot must meet if it is to be used to support Home Oxygen Therapy patients.

Table 1.1: Requirements for H.O.T. Support Robot

Item	Requirement
Operating time	> 1 hour
Mass	< 10 kg
Footprint size	$<500\times500$ mm
	(should fit in front seat of car)
Payload mass	> 4 kg
Payload size	$350\times120\times120$ mm (H×L×W)
Speed	> 0.5 m/s
Tether length range	0.2 m–1.0 m (or greater)

1.2 Survey of Related Work

The focus of this thesis is a tethered robot which can support Home Oxygen Therapy patients. As such, this chapter presents related research on leader following, tethered robots, and examines several potential solutions existing in academia and industry.

1.2.1 Leader Following

1.2.1.1 General Leader Following in Robotics

Leader following has been widely studied in robotics to date. Shao et al. has demonstrated leader following control in systems with multiple robots, with both centralized and decentralized control [10], [11]. Carpin and Parker have investigated leader following in a heterogeneous system: where the individual robots or vehicles are non-identical and have different degrees of sensing and mobility[12].

In terms of controlling a robot to follow the trajectory of a leader, Samson and Ait-Abderrahim investigated the controllability and stability when controlling a two-wheeled robot cart to follow a reference path[13]. They showed that a robot cart can stably track a virtual reference cart (both position and orientation), as long as the reference cart is constantly moving[14]. Further study of global, time-varying feedbacks and their stability and convergence has yielded several useful control outcomes for non-holonomic wheeled carts[15].

Due to the target application, this study will focus on human following, rather than vehicle-vehicle following, or convoy control, etc.

1.2.1.2 Human Following

Ohya and Nagumo developed an escort robot which could move beside [16], [17] a human user. The system, shown in Figure 1.5, used a camera to track a light emitting device attached to the user's waist [18]. The camera measured the apparent distance between two LEDs, and used this to estimate the distance between the robot and the user, and the orientation of the LEDs was used to estimate the direction. Although this escort robot was successful in basic tracking and human following, no indication of the tracking accuracy was given, and the system was also rather bulky making it somewhat unsuitable for personal use.

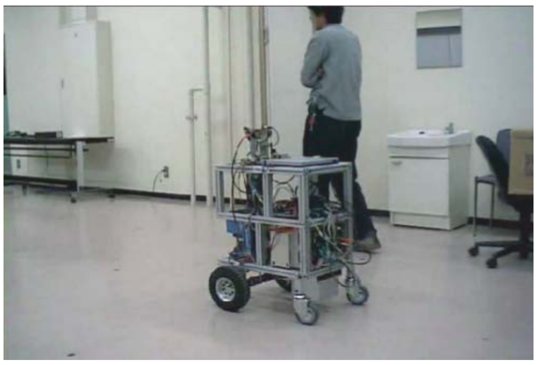


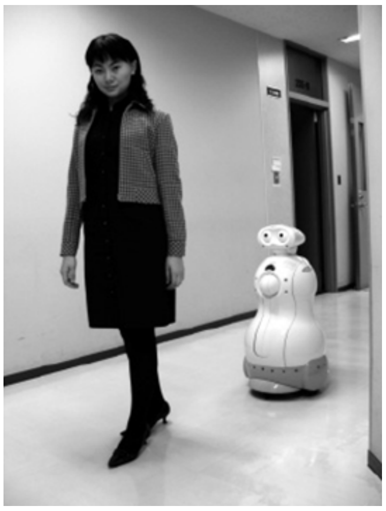
Figure 1.5: Escort robot using camera and light emitting device, developed by Ohya and Nagumo[16]

In addition to recent academic research on leader following, a number of 'porter' style robots have been developed in the Japanese industry (Figure 1.6). The *Roboporter*, manufactured by Yasukawa[19], and the *Porter Robot* manufactured by Panasonic[20] serve a very similar purpose: carrying an item of luggage while following behind a human leader. These robots make use of a variety of sensors to achieve the following operation: laser range finders, stereo cameras, and ultrasonic sensors for obstacle detection. The *ApriAttenda* robot developed by Toshiba used a primarily vision-based sensing system (supported by ultrasonic sensors) to follow a person, and it included methods to recognise particular individuals and resume following after losing visual contact [21].

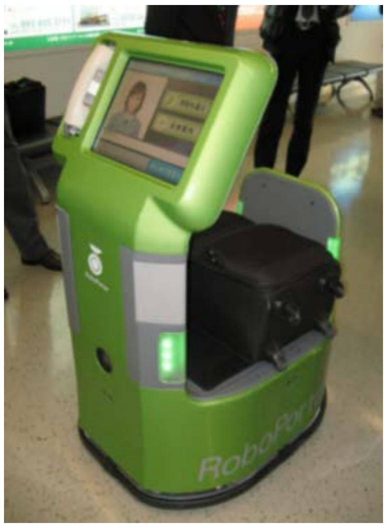
Figure 1.7 shows follower robots developed by companies whose primary focus is leisure/sports. Segway Japan developed a human following robot with simulator, based on one of their Segway mobile robots [22], [23]. In the case of the Segway follower, target detection was mainly achieved using a stereo camera system, supported by a laser range finder.

Figure 1.7b shows the *X9 Follow*: a high-end golf cart produced by Stewart Golf[24]. The X9 Follow uses a pair of bluetooth antennas in the robot, while the user carries a

(a) Panasonic porter robot[20]



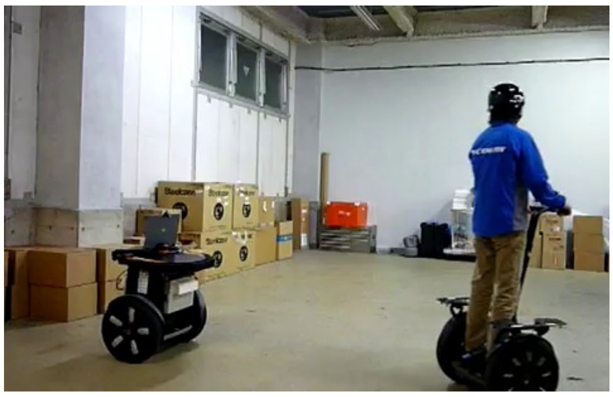
(b) Toshiba ApriAttenda[21]



(c) Yasukawa Roboporter[19]

Figure 1.6: Porter robots developed in industry.

remote control (containing a bluetooth transmitter). The remote control allows the user to switch between two different following modes: direct control (where the user steers the cart with buttons), and automatic following mode (where the user only needs to carry the remote control). Interestingly, when marketing this product the manufacturer describes the operation as being similar to a “digital elastic band”: even with a wireless system, the intuitive nature of a tether is used to explain the operating procedure. The X9 Follow costs several thousand dollars, making it prohibitively expensive for most people.


(a) Segway human following robot project[23]

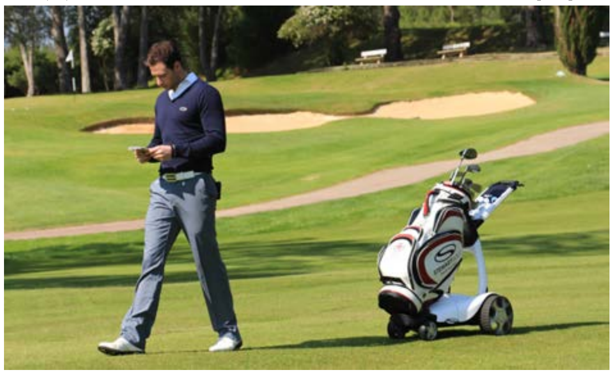

(b) X9 Follow golf cart with person following[24]

Figure 1.7: Follower robots in the sport/leisure field.

A common issue with the human following robots discussed so far is the amount of sensors required, and the associated cost. It is difficult to employ these systems in a low-cost device, designed for use by people with limited income.

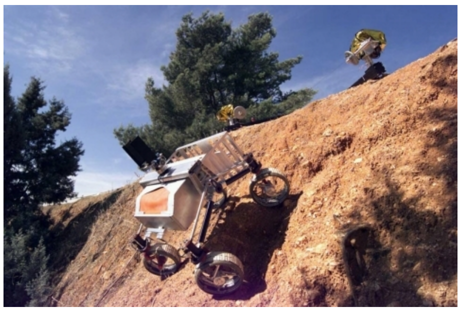
1.2.2 Tethers

1.2.2.1 Advantages of Tethers

Despite the relative popularity of wireless communication and sensing technologies, tethers — flexible cord-like members with tensile strength but low (or zero) compressive strength — still offer a number of advantages. In addition to position tracking, the use of a physical tether also allows mechanical support (important in cooperative robot systems such as [25]), and the possibility to share communication and power between robots. Figure 1.8 shows a small selection of robots which have successfully used tethers in adverse environments, demonstrating their reliability and robustness. Dante II made use of a tether to rappel down the sides of volcanic chasms for data collection missions[26]. In the case of the Souryu in-rubble inspection robot (Figure 1.8c), designed for search and rescue operations in disaster stricken areas, the original wireless design[27] was replaced with a tethered design: Souryu IV[28]. The addition of a tether also improved communication signal strength and allowed video data from an on-board camera to be relayed to the operator.

Further evidence of the usefulness of tethers can be seen in their widespread use across a variety of industries and disciplines. In aerospace, tethers have been proposed to support and control spacecraft[29], while in marine engineering tethered ROVs are commonly used for underwater survey and maintenance tasks. Outside robotics, industrial forestry machines make use of steel ropes to support themselves on steep inclines, off-road driving enthusiasts use winches to pull themselves up slopes, and weather balloons are anchored by tether.

(a) Dante II using tether-based rappelling locomotion[26].


(b) Cliffbot cooperative multi-vehicle system with active tether control[25].

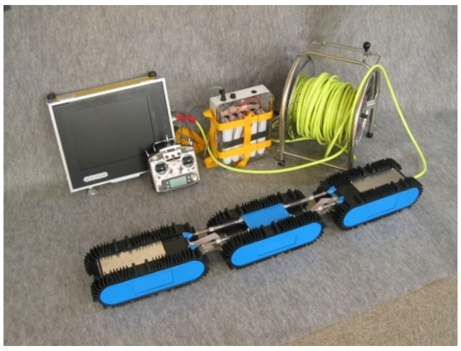

(c) Souryu-IV inspection robot[28].

Figure 1.8: Robots where tethers are used for their robustness.

1.2.2.2 Tethered Follower Robots

Na et al. developed the *Navi-Guider* in 2009[30]. The Navi-Guider system is composed of two interfaces: an interface between the user and the guiding system (which should be intuitive for non-expert users); and a second interface between the guiding system and the robot. The implementation uses a tether attached to a reel, along with sensors to measure the length and angle of the tether (potentiometer and encoder) [31]. In this sense, the Navi-Guider is similar to the *Hyper-Tether* system proposed by Fukushima et al.[32], however Navi-Guider also includes some additional ultrasonic sensors for obstacle detection and rudimentary avoidance.

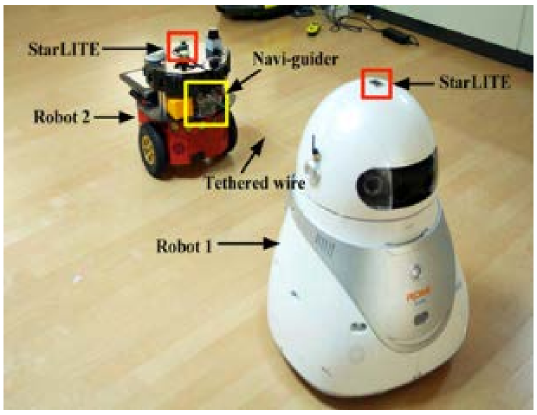
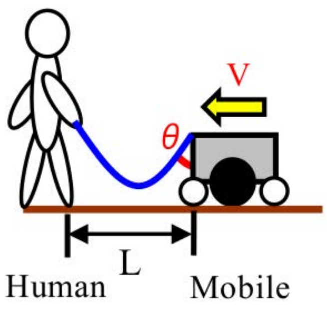


Figure 1.9: Navi-Guider tether interface for robot following, developed at ETRI[30].

Kim et al. developed a prototype tethered follower using a slightly different sensor system[33]. A spring and linear potentiometer was attached to the base of the tether. This implementation has the advantage of being relatively simple: the user pulls the tether, the spring is compressed, and the potentiometer measures the spring displacement. However, this greatly limits the range of movement of the tether, and furthermore the spring will change the feeling of the tether as the user applies force. The effect of the spring stiffness on the user's impressions were not studied in this case.

Researchers at Waseda University have proposed a novel approach to tethered leader following using a slack tether[34]. In the tethered robots discussed previously, the tether has been kept under tension to ensure the end point is known precisely. In the case of the robot developed by Ogawa et al. shown in Figure 1.10, the tether is slack, allowed to hang in a natural catenary curve[35], [36]. The drop angle at the beginning of the catenary curve is measured, and since the length of tether is fixed, the catenary angle will be proportional to the distance between the tether end points. The robot's forward speed is controlled based on the catenary angle, so when the user moves forward, the catenary angle increases, and the robot's speed also increases.



(a)

(b)

Figure 1.10: Slack tether mobile follower robot developed at Waseda University[35]. (a) To measure the distance to the user, the catenary drop angle is measured; (b) User operating the robot.

Unfortunately, this approach has some significant drawbacks. Firstly, the requirement for the tether to be a fixed length may constrain the distance between the user and robot, reducing freedom of movement in some situations. Secondly, the tether is vulnerable to vibration effects which in turn affect the catenary angle, and thus the robot's speed. Accommodating this in the robot's control is difficult, and therefore an alternative mechanical solution has been considered, where weight is added to the tether. Clearly, this added weight makes the system more cumbersome overall.

1.2.3 Comparison of Existing Solutions

A summary of the sensors used in the surveyed devices is compiled in Table 1.2. Table 1.3 evaluates the previously discussed solutions, when considered for use with Home Oxygen Therapy patients. Cost is a major limitation among the industrial porter robots; since they include expensive components such as laser range finders their final price is likely to be many thousands of dollars. These robots are also not well-suited to use in outdoor environments. The *X9 Follow* golf cart performs well outdoors, but only in relatively open environments, and its size and weight make it impractical in this case. Among the devices developed in academia, the *Navi-guider* and the spring sensor tether system both offer a promising solution as they are low-weight, fairly low-cost and the tether provides an intuitive interface. However, neither of these devices has been tested with H.O.T. patients, and in particular it is unknown how suitable the tether control algorithms employed are. These academic robots are also based on platforms which are not especially suitable for outdoor use, and most previous testing has been conducted in indoor laboratory environments.

The final row of Table 1.3 includes the required (or desired) level of features for a new Home Oxygen Therapy support robot: the focus of the research in this thesis. Such a robot must be suitable for use indoors and outdoors, with low-weight, and lower cost than alternative devices. Additionally, the robot must be more robust, and easier to use than previous devices, and its performance must be confirmed by testing with the target users.

Table 1.2: Comparison of existing devices' sensors

Device	Sensing Components						
	Stereo camera	Laser range finder	Bluetooth	Ultrasonic sensor	Tether angle sensor	Tether length sensor	Tether force sensor
Toshiba ApriAttenda[21]	✓	✓		✓			
Panasonic porter[20]	✓	✓		✓			
Yaskawa roboporter[19]		✓		✓			
X9 Follow[24]			✓				
Escort robot[16]	1 camera			✓			
Navi-guider[30]				✓	✓	✓	
Spring-sensor[33]					✓		✓
Slack tether follower[35]					✓		
H.O.T. support robot[‡]					✓	✓	

[‡] *H.O.T. support robot* represents the target specification for the research in this thesis.

Table 1.3: Comparison of existing devices' suitability for Home Oxygen Therapy support

Device	Evaluation Criteria						
	Weight	Cost*	Robustness	Ease of use	Outdoor use§	Indoor use	User testing†
Toshiba ApriAttenda[21]	★★	★	★	★★★	★	★★★	★
Panasonic porter[20]	★	★	★	★★	★	★★★	★
Yaskawa roboporter[19]	★	★	★★	★★	★	★★★	★
X9 Follow[24]	★	★	★	★★★	★★★	★	★
Escort robot[16]	★★	★★	★★	★★	★	★★★	★
Navi-guider[30]	★★★★	★★	★★★	★★★	★★	★★★	★
Spring-sensor[33]	★★★★	★★★	★★	★★★	★★	★★★	★
Slack tether follower[35]	★★	★★★	★	★★	★★	★★★	★
H.O.T. support robot‡	★★★★	★★★	★★★★	★★★★	★★★	★★★	★★★★

* Where exact costs are unavailable for a device, the cost has been estimated based on the known components and specification.

§ *Outdoor use* is qualified as operation in streets, around shops, parks etc. Off-road use is not included.

† *User testing* refers to evaluation with Home Oxygen Therapy users.

‡ *H.O.T. support robot* represents the target specification for the research in this thesis.

1.3 Scope of This Research

The aim of this thesis is to investigate tether control in a mobile robot designed to support Home Oxygen Therapy patients. A mobile, differentially steered robot platform is considered to give a simple, low-cost, controllable base. A tether interface is used due to the intuitive operation and robustness compared to previous devices (as discussed in Section 1.2.3). Different algorithms for tether control will be studied and compared using simulation, then experiments in a controlled environment in order to analyse their technical performance and their effect on the user. Practical considerations for implementing the control will also be considered. As far as possible, the evaluation will be carried out in cooperation with Home Oxygen Therapy patients (as the real target users of the robot).

While applications other than Home Oxygen Therapy are technically outside the scope, some brief discussion of these may be included where appropriate. As a particular exception, in order to demonstrate the generality of the control, the scope is extended to an additional application: the airport carrier robot described in Chapter 6.

1.4 Structure of Thesis

The approach used in this research was to analyse each control method with simulations, then experiments in a controlled environment before moving on to experiments with real target users. The structure is outlined below:

Chapter 1 gives the background of the research. The condition COPD and the treatment Home Oxygen Therapy are briefly described, along with the requirements for an assistive robot. A brief summary of related research is presented, describing some of the currently available follower robots with comments about their suitability for this application. The role of tethers in robotics is discussed with reference to examples in academic research, and the relative advantages of tethers. The Scope of the research in this thesis is also outlined.

Chapter 2 describes the differential drive robot model, and the tether and winch models used throughout this research. Two major control methods are presented with theory and simulated trajectories: *Pseudo-Joystick* and *Follow the Leader*. *Normal path deviation* is introduced as a metric to compare the accuracy of leader following.

Chapter 3 describes motion capture experiments used to measure and compare the performance of Pseudo-Joystick control and Follow the Leader control under controlled conditions. The second half of this chapter reports and discusses the results of a leader following experiment with H.O.T. patients and a questionnaire survey to gather user feedback.

Chapter 4 introduces several additional follower modes designed to improve follower performance in certain situations or address issues with previous control methods. Side following mode is proposed to allow the robot to move beside the user and remain in their field of vision. Simulation results are presented for two types of side following control: *Side Joystick* mode and *Side Tracking* mode. *Brake* mode is introduced to improve the safety and usability of the robot, especially in busy environments. The problem of tether snagging is also briefly investigated, showing how a tracked trajectory is distorted if the tether hits an obstacle.

Chapter 5 describes leader following experiments conducted with a Home Oxygen Therapy patient in an outdoor environment. The user was asked to walk various routes around their local area (mainly around a park and the nearby train station), while using an assistive device to carry their oxygen tank. Three different devices are compared: a conventional oxygen cart (unpowered); a commercially available cart with powered wheels; and a robot follower. The user's heart rate and oxygen saturation (SpO2) are measured, and these values are then used to compare the effect of each device on the user.

Chapter 6 introduces an airport carrier robot as a suitable application for the leader following control discussed in previous chapters. The design of a prototype platform is described, along with implementation of the control. The speed command relationship is investigated through calibration. Basic operation of the platform is confirmed.

Chapter 7 gives conclusions and final remarks about the research, along with some ideas about possible future work.

1.5 Chapter Summary

Home Oxygen Therapy is a medical treatment for severe lung diseases such as Chronic Obstructive Pulmonary Disease (COPD). This chapter described the nature of Home Oxygen Therapy and the problems faced by its users, notably that the user must carry the H.O.T. equipment around, restricting their freedom. A brief survey of leader following systems was then presented, including human following 'porter' robots developed in industry and several academic follower systems, including those using tethers. There is evidence that tethers are intuitive to users, making a tethered robot easy to operate. After evaluating the previously developed systems, it was found that existing devices were unsuitable for Home Oxygen Therapy due to high cost, lack of robustness and lack of testing with the target users. Accordingly, this thesis will study a tethered leader following robot to meet the needs of Home Oxygen Therapy patients, tested and evaluated by the users themselves. This robot must be low-cost, easy-to-operate, robust, and must be able to operate indoors and outdoors.

Chapter 2

Characterization of Leader Following Control

2.1 Introduction

The *Hyper-Tether* concept was proposed by Fukushima and Hirose [32] as a means of using tethers to facilitate cooperation between robots, vehicles and even humans. Figure 2.1 shows the general concept from their proposal. The simplest implementation of this system involved attaching a winch to the follower robot, connecting to the leader by tether. The winch could reel in/out so that the tether was always under tension, and it would also measure the tether length and angle. Such a tether system is capable of providing several useful functions:

1. Power supply through the tether
2. Mechanical support
3. Communication
4. Position tracking and control

Leader following (which is possible given the position tracking of the tether tip) was one aspect of *Hyper-Tether*, and to this end the following algorithms have been proposed [37], [38]:

- *Pseudo-Joystick* control
- *Follow the Leader* control

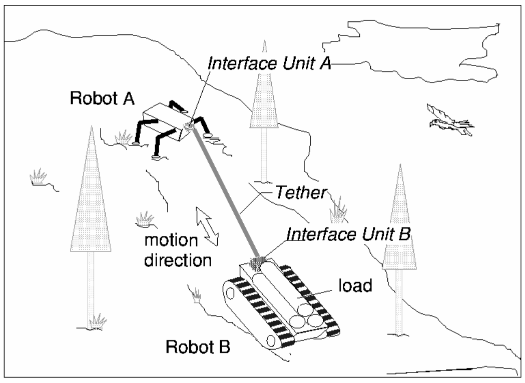


Figure 2.1: Hyper-tether concept proposed by Fukushima and Hirose

Previous research has explored these algorithms and some potential applications [39], [40]; this chapter aims to analyse the behaviour of *Pseudo-Joystick* and *Follow the Leader* and determine the effect of various parameters affecting the control. This will provide a useful basis for implementation in leader following robots, but more importantly this work will be expanded to develop new leader following algorithms in Chapter 4.

This chapter describes the control theory for both algorithms, along with the simplified robot and tether model used. The development of a leader following simulator is described, with the algorithms implemented in Python, and the simulation environment provided by the robotics simulator package V-REP. Simulation results for various input paths are presented to demonstrate basic leader following behaviour, and compare the different algorithms. Lastly, Section 2.5, 'Characterization of Parameters', describes a study of how several of the parameters present in control algorithms affect the robot's following performance. Several metrics of following performance are introduced at that point to allow us to analyse and compare the results.

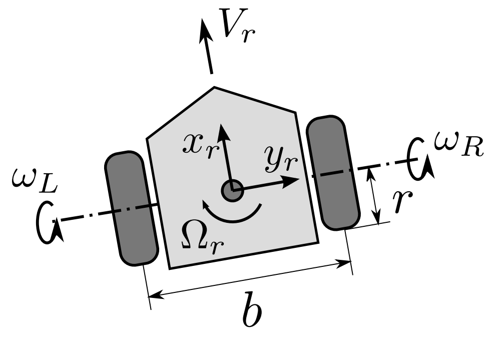
2.2 Mobile Robot Model

2.2.1 Two-wheeled Robot

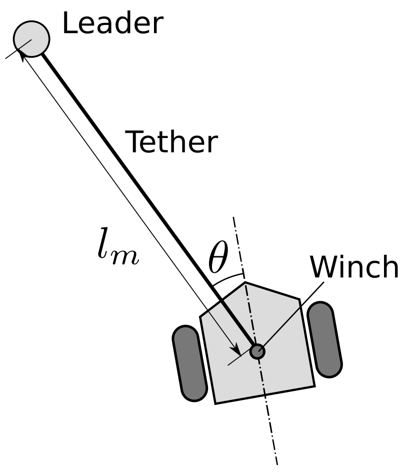
In principle, leader following can be applied to a wide range of robots and vehicles, with a variety of kinematic constraints. For this research, a two-wheeled differentially steered robot was selected because of the following advantages:

Simplicity of Implementation: Prototypes can be easily developed to provide hardware validation of control algorithms.

Generality: With relatively few kinematic constraints, the platform can be considered applicable in a wide range of applications.



(a) Two-wheeled robot model.



(b) Robot with tether and winch. In this case, the winch is shown at the center of the robot, but the winch may also be positioned offset from the center.

Figure 2.2: Robot and winch models

The mobile robot model used for this research is shown in Figure 2.2a. The geometric parameters with the greatest influence on the kinematics are distance between wheels, or wheel track, b, and the wheel radius r. Left and right wheel rotational velocities are denoted ω_L and ω_R respectively. V_r represents the robot's forward/backward speed, and Ω_r is the angular velocity about the robot's center. Positions located in the robot's frame of reference are denoted (x_r, y_r), where x_r is in the direction of travel and y_r in the lateral direction (i.e. along the wheel axis). The robot has two controllable degrees of freedom in the rotational velocity of the left and right wheels (ω_L, ω_R), and three total degrees of freedom: the (x, y) planar positions along with the orientation angle, denoted Γ. The system is therefore non-holonomic. In hardware prototypes the platform may also feature free casters on the front and possibly rear of the robot.

2.2.2 Tether and Winch

Figure 2.2b shows the tether model used in this research. l_m is the tether length, measured from the winch to the tether tip. θ is the tether angle, measured between the winch and the robot body in the range $-\pi < \theta < +\pi$.

2.3 Control Theory

2.3.1 Pseudo-Joystick Mode

The simplest control method for a follower robot comprises using the tether length and direction as steering input commands: *Pseudo-Joystick control* (see Figure 2.3).

Using coordinates relative to the robot's reference frame, and given the measured tether length l_m, the desired tether length l_d and the measured tether angle θ, we can use the following control laws for the robot translational velocity V_r, angular velocity Ω_r and target direction ϕ:

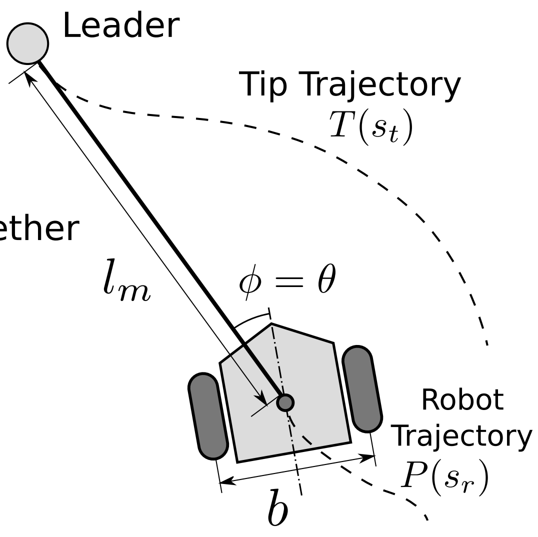


Figure 2.3: *Pseudo-Joystick* control.

$$V_r = K_p(l_m - l_d) \tag{2.1}$$

$$\Omega_r = -\frac{2V_r}{b}\sin\phi \tag{2.2}$$

$$\phi = \theta \tag{2.3}$$

Where K_p is a proportional velocity gain, and b is the axle track. Since the target angle is set equal to the tether angle in Equation (2.3), the robot will always try to move in the current direction of the tether. The desired angular velocities for the left and right wheels (ω_L, ω_R) are found using the non-holonomic kinematic equation (2.4):

$$\boldsymbol{V} = \boldsymbol{A}^{-1}\boldsymbol{q} \tag{2.4}$$

$$where:$$

$$\boldsymbol{V} = \begin{bmatrix} \omega_L & \omega_R \end{bmatrix}^T \tag{2.5}$$

$$\boldsymbol{A} = \begin{bmatrix} r/2 & r/2 \\ -r/b & r/b \end{bmatrix} \tag{2.6}$$

$$\boldsymbol{q} = \begin{bmatrix} V_r & \Omega_r \end{bmatrix}^T \tag{2.7}$$

In Equation (2.6), matrix A transforms the wheel angular velocities to body velocities, where r is the wheel radius.

2.3.2 Follow the Leader Mode

2.3.2.1 Leader Tracking

If the robot can determine its own posture relative to an inertial reference frame $\Sigma_g X_g Y_g$, more sophisticated tracking of the leader position can be achieved by recording the position of the tether tip over time. This is shown in Figure 2.4). We define the trajectory of the tether tip as $\boldsymbol{T}(s_t)$ and the trajectory of the follower robot as $\boldsymbol{P}(s_r)$, where s_t and s_r represent the distance travelled along each respective trajectory. We can calculate the target angle using the following steps:

Step 1: Estimate the follower robot posture by dead reckoning.
i) Estimate the translational and angular velocity at time t:

$$\boldsymbol{q} = \boldsymbol{A}\boldsymbol{V} \tag{2.8}$$

ii) Calculate Γ, the orientation of the follower in inertial reference frame $\Sigma_g X_g Y_g$:

$$\Gamma(t + \Delta t) = \Gamma(t) + \Omega_r(t)\Delta t \tag{2.9}$$

iii) Calculate the distance travelled by the tether tip Δs_r:

$$\Delta s_r = V_r \Delta t \tag{2.10}$$

iv) Update the position of the follower $\boldsymbol{P}(s_r + \Delta s_r)$:

$$\boldsymbol{P}(s_r + \Delta s_r) = \boldsymbol{P}(s_r) + \boldsymbol{E}^{k\Theta} \begin{bmatrix} \Delta s_r \\ 0 \end{bmatrix} \tag{2.11}$$

$$where:$$

$$\Theta = \Gamma + \theta \tag{2.12}$$

$$\boldsymbol{E}^{k\Theta} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \tag{2.13}$$

Θ is the tether angle in the inertial reference frame $\Sigma_g X_g Y_g$.

Step 2: Compute the position of the tether tip $\boldsymbol{T}(s_t + \Delta s_t)$.

$$\boldsymbol{T}(s_t + \Delta s_t) = \boldsymbol{P}(s_r + \Delta s_r) + \boldsymbol{E}^{k\Theta} \begin{bmatrix} l_m \\ 0 \end{bmatrix} \tag{2.14}$$

Step 3: Calculate the distance travelled by the tether tip.

$$\Delta s_t = \|\boldsymbol{T}(s_t + \Delta s_t) - \boldsymbol{T}(s_t)\| \tag{2.15}$$

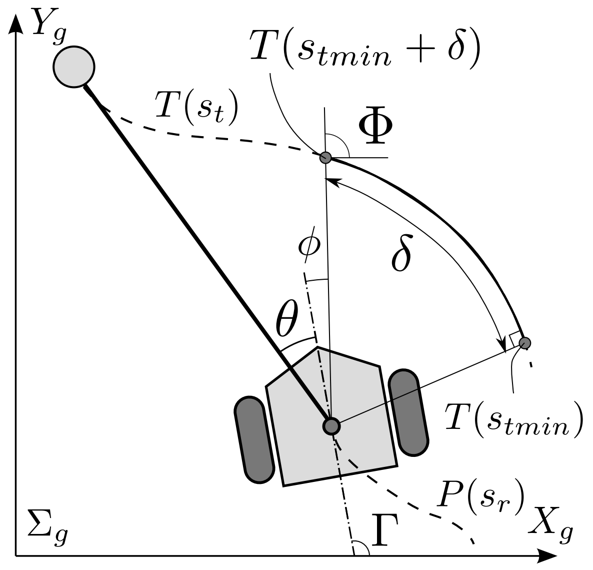


Figure 2.4: *Follow the Leader* control.

2.3.2.2 Convergence on Leader Path

At this point, the leader trajectory is known, so it is possible to select some forward point on this trajectory and command the robot to steer towards it.

Step 4: Determine the target angle ϕ.

Find $\vec{T}(s_t min)$: the point on the tether tip trajectory where the distance ρ between the follower position $\vec{P}(s_r + \Delta s_r)$ and the tip trajectory $\vec{T}(s_t)$ is minimised. We then add a lookahead distance δ along $\vec{T}(s_t)$ to find the target point $\vec{T}(s_t + \delta)$. The follower angle ϕ then ensures that $\vec{P}(s_r)$ converges to $\vec{T}(s_t)$.

$$\phi = \Phi - \Gamma \tag{2.16}$$

The inclusion of the lookahead distance δ parameter allows the robot to converge on the leader trajectory without excessive weaving (zig-zagging). The value of δ must be selected and tuned for each implementation, but reasonable following performance can typically be obtained when $b < \delta < 2b$ (where b is the wheel track of the robot). Control gains K_V and K_Ω are then introduced to produce a desired robot velocity V_r and angular velocity Ω_r, given a desired tether length l_d:

$$V_r = K_r(l_m - l_d) \tag{2.17}$$

$$\Omega_r = K_\Omega \phi \tag{2.18}$$

Finally, V_r and Ω_r can be transformed into desired wheel velocities (ω_L, ω_R) using the inverse matrix $\boldsymbol{A}^{-1}$ (2.6).

2.3.3 Follow the Leader with Constant Distance Mode

Constant Distance mode is a variant of the *Follow the Leader* mode previously introduced. Since *Follow the Leader* mode prioritises keeping the robot close to the leader trajectory at all times, it does not place significant emphasis on the length of the tether, and as such, the length of the tether may vary significantly during operation. There are situations where controlling the length of the tether may be more important than following the leader path precisely (in fact, controlling the tether length is essential in the target application which will be introduced in Chapter 3). For *Constant Distance* control, while the robot must accurately converge on the leader trajectory, we choose the target point based so that the tether length should remain (relatively) stable (Figure 2.5).

The initial algorithm for *Constant Distance* control are the same as steps 1 to 3 of *Follow the Leader* control, described in Section 2.3.2.1 above. Step 4 onwards uses a different method, described below:

Step 4: Determine the target angle ϕ.

We find a point on the leader trajectory a fixed distance of l_d from the leader (Figure 2.5),

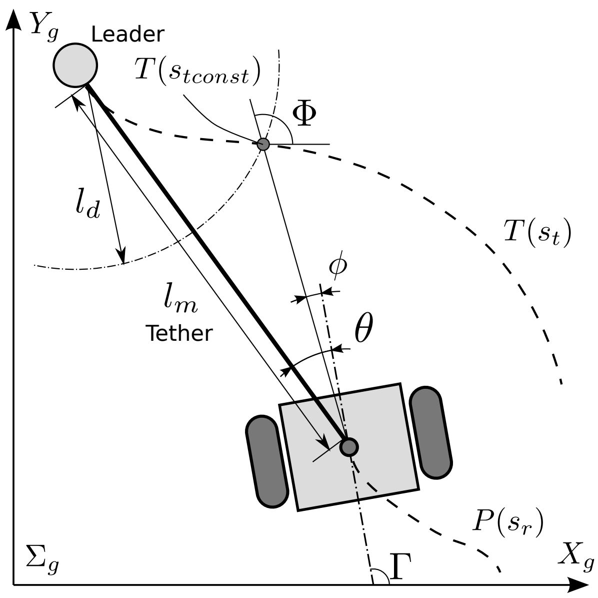


Figure 2.5: *Follow the Leader with Constant Distance* control.

$\boldsymbol{T}(s_t const)$, and calculate the angle from the robot to this point:

$$\phi = \arctan\left(\frac{\boldsymbol{T}(s_t const)_y - \boldsymbol{P}(s_r)_y}{\boldsymbol{T}(s_t const)_x - \boldsymbol{P}(s_r)_x}\right) - \Gamma \tag{2.19}$$

We can then modify the control equations (2.1), (2.2) so that the magnitude of the robot's velocity is decreased proportional to the target angle ϕ, introducing control gains K_a and K_b:

$$V_r = K_p(l_m - l_d)(1 - K_a\|\phi\|) \tag{2.20}$$

$$\Omega_r = -K_b\phi \tag{2.21}$$

As before, V_r and Ω_r can be transformed into desired wheel velocities (ω_L, ω_R) using the inverse matrix $\boldsymbol{A}^{-1}$ (2.6).

2.4 Numerical Simulation

2.4.1 Simulation Environment

To investigate the performance of the follower robot, a dynamic simulation was developed with the open source *V-REP* software package, using the *Bullet* physics engine[41]. V-REP was selected because it allows different experimental conditions to be modelled in a relatively short time, and it has been widely used for a range of robotics application [42]. For this research, we modelled a two-wheeled, differentially steered follower robot (similar to Figure 2.2a) with a frictionless caster at the front and at the rear. The wheel track b was 280 mm; the wheel radius r was 125 mm, and the distance between front and rear casters was 500 mm.

The leader was modelled as dummy point moving along a predetermined path at a fixed speed of 0.5 m/s. A model tether connected the robot to the leader, and a sensor provided length, l_m, and angle, θ, data to be used in the control algorithm. Figure 2.6 shows the simulation environment.

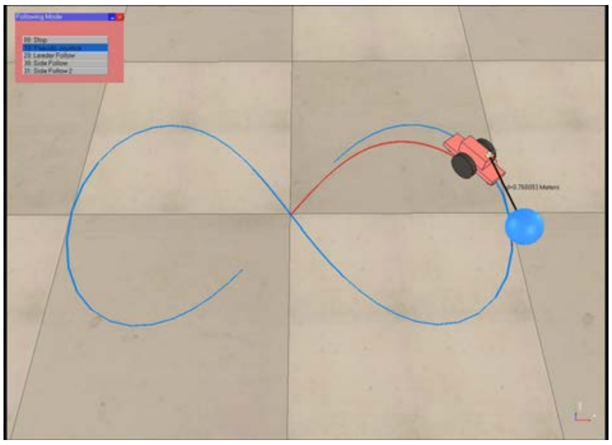


Figure 2.6: V-REP simulation environment.

In order to characterize and compare the control methods detailed in the previous section, the algorithms for *Pseudo-Joystick*, *Follow the Leader* and *Follow the Leader with*

Constant Distance were implemented in Python and linked to the V-REP simulation. The robot was first set to use *Pseudo-Joystick* control to follow the dummy leader as it moved along the pre-set path, and the resulting trajectories were plotted. The simulation was then repeated using *Follow the Leader with Constant Distance* control with the same leader path. It should be noted that the algorithms for *Follow the Leader* and *Follow the Leader with Constant Distance* differ slightly, but *Follow the Leader with Constant Distance* was used for comparison in the bulk of these analyses since it is more suitable for use in Home Oxygen Therapy (where the cannula length must be maintained).

The results of simulations with two different leader paths are presented in the following section:

- Smooth 'figure-8' path with total length around 11 m and loop diameter 2 m
- Square 'S' path with total length around 12 m side length 2 m

These were selected because their relative simplicity facilitated visual analysis, while they also offered a reasonable combination of left and right turns, both sharp and smooth.

2.4.2 Following Smooth Path

The results for the smooth, figure-8 path are presented in Figure 2.7. The robot and leader trajectories are plotted for both *Pseudo-Joystick* and *Follow the Leader with Constant Distance* control. Additionally, Figure 2.7c shows the *normal path deviation* from the leader path to the robot path. The calculation of *normal path deviation* is introduced in Section 2.5.1.3 and explained in detail in Appendix A.

2.4.3 Following Square Path

The results for the square path are presented in Figure 2.8.

2.4.4 Discussion

From the leader and robot trajectories in Figures 2.7a and 2.8a, we can confirm that the robot using *Pseudo-Joystick* control exhibits basic following behaviour. As expected, the robot's trajectory is close to the leader's, but deviation occurs when the leader path turns with a tight radius. We see that the robot tends to 'cut corners': when the leader turns and moves to the right, the robot will immediately begin steering towards the right and therefore will miss part of the leader trajectory (but still always following the general trajectory). This suggests that *Pseudo-Joystick* may be unsuitable in environments where the user needs to make many tight-radius turns, or if there are many obstacles to walk around.

Nevertheless, *Pseudo-Joystick* control has several important advantages:

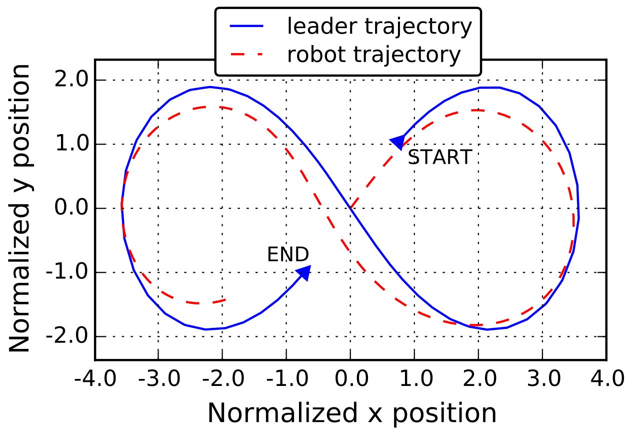
Simplicity of implementation: The algorithm does not require complex calculations or a large amount of memory. Crucially, this makes it particularly suitable to embedded microcontroller applications.

Stability: The algorithm is stable, and robust to all steering inputs.

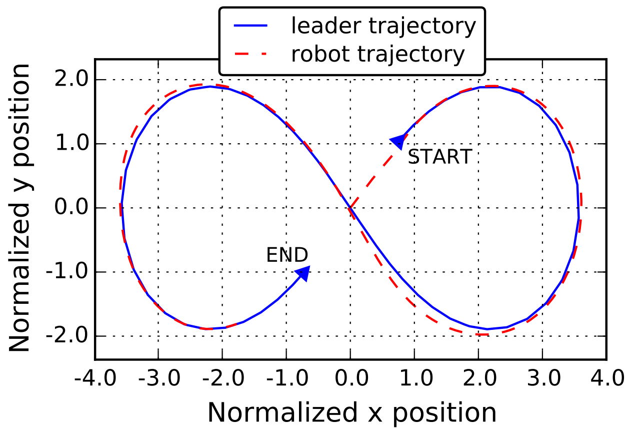
Intuitive operation: The use of the tether as a direct steering input means the operation of the robot is highly intuitive to users. A new user can learn the operation quickly, and will not be surprised by the behaviour.

When using *Follow the Leader with Constant Distance* control, the robot's trajectory follows the leader trajectory much more closely (Figures 2.7b and 2.8b), though we still see some small deviation from the leader path when the robot moves around curves (Figures 2.7c and 2.8c). It is possible to reduce this deviation by adjusting some of the control parameters as discussed in the following Section 2.5.

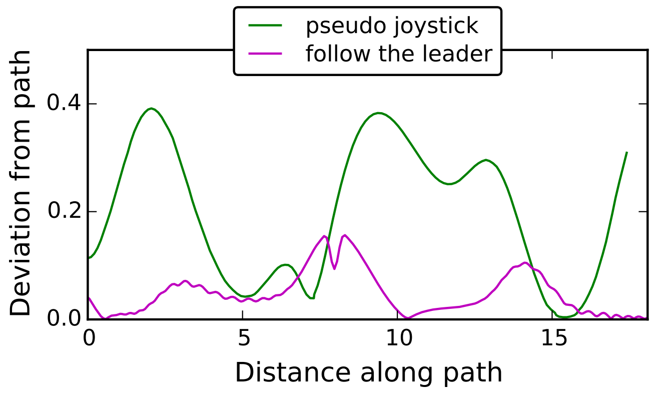
The principle advantage of *Follow the Leader with Constant Distance* control is the increased accuracy of leader tracking and corresponding lower deviation from the target path. On the other hand, its implementation introduces additional computational load and memory requirements, both of which may limit its application in embedded micro-controllers.



(a) Trajectory using Pseudo-Joystick control

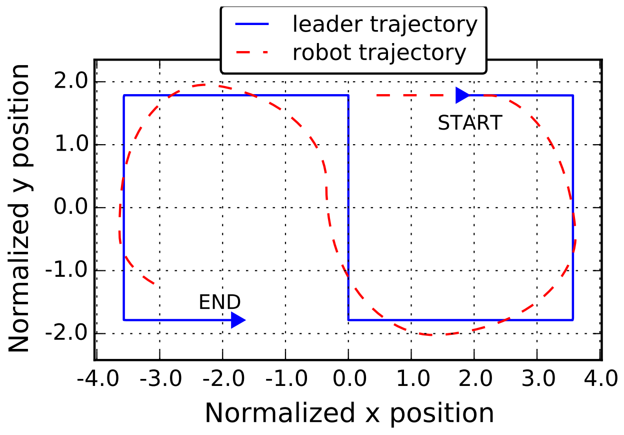


(b) Trajectory using Follow the Leader with Constant Distance control

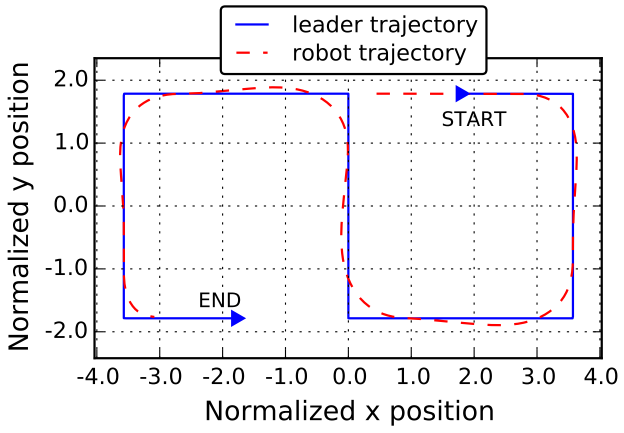


(c) Comparison of normal path deviation

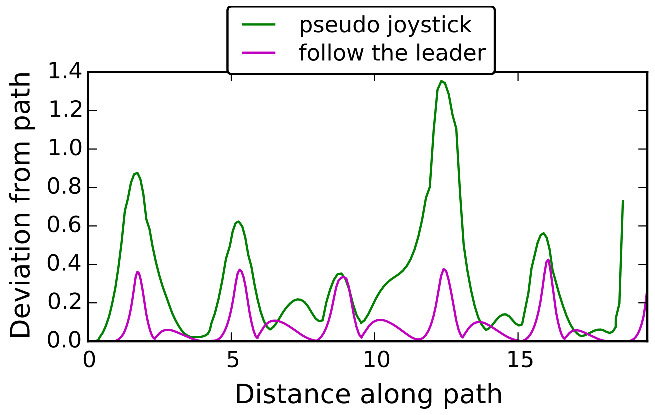
Figure 2.7: Follower performance for smooth, figure-8 path. Distances have been normalized according to the vehicle wheel track *b*.



(a) Trajectory using Pseudo-Joystick control



(b) Trajectory using Follow the Leader with Constant Distance control



(c) Comparison of normal path deviation

Figure 2.8: Follower performance for square path. Distances have been normalized according to the vehicle wheel track b.

2.4.5 Stability of Control

Due to the numerous parameters in the control algorithms (particularly the target lookup search in the case of *Follow the Leader* control), it is difficult to prove the stability using analytical methods. Furthermore, when we consider the practical operating environment of the robot, there are always disturbances caused by uneven or loose floors, and these are hard to predict and quantify. The practical use of analytical proof of stability might therefore be of limited use in terms of robot development in this case.

As an alternative, numerical methods were used to confirm the stability of the control over a bounded range of tether inputs. The bounds were selected to represent realistic limits compared to the vehicle size and the tether length and angle were varied between these bounds ($-\pi < \theta_m < +\pi$; $0 < l_m < 10$). The time taken for the control to converge was recorded, and provided the control converged in a finite time, we can reasonably assume that the control is stable in this range. Figure 2.9 shows the convergence results for *Pseudo Joystick* control, while Figure 2.10 shows the convergence results for *Follow the leader* control.

This confirmed that the control was stable for all realistic steering inputs.

2.4.6 Effect of Vibration

The tether used in the robot is relatively light, so the principle effect of tether vibration is small. However we should also consider the possible effect of vibration due to the relatively heavy oxygen tank being mounted on the robot body.

For an initial analysis, we can model the system as a two degree of freedom spring-mass system (ignoring damping at first). This first model is shown in Figure 2.11. The robot has casters on the front and rear; this prevents rotational excitement and so we can reduce our scope to just the vertical axis.

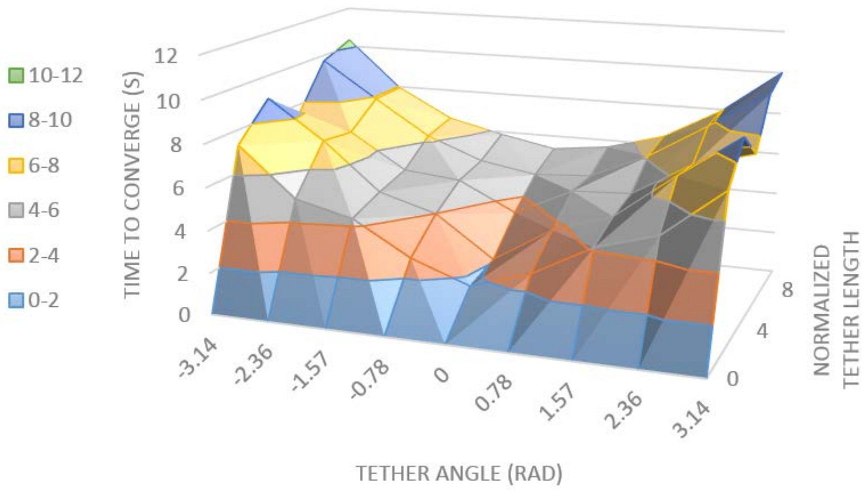


Figure 2.9: Stability of *Pseudo Joystick* control. The control inputs were varied across reasonable bounds, and the time taken for the control to converge was recorded.

Equation 2.22 shows the equations of motion for the system.

$$\begin{bmatrix} m_1\ddot{x}_1 \\ m_2\ddot{x}_2 \end{bmatrix} = \begin{bmatrix} -k_1 - k_2 & k_2 \\ k_2 & -k_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2.22}$$

k_1 represents the spring rate of the wheel tyres, and k_2 is the corresponding spring rate for the oxygen tank mounts. m_1 is the mass of the robot body and wheels, and m_2 is the mass of the oxygen tank. The major source of vibration is through the wheels of the robot, either by a rough ground surface, or by step displacements when travelling over bumps.

When we introduce a step displacement, clearly the system will oscillate as shown in Figure 2.12a. Obviously, the model so far does not represent the real machine: neoprene on the tyres provides damping, and the oxygen tank mounting also provides some damping. Adding the neoprene damping to the model provides the more reasonable damped response seen in Figure 2.12b.

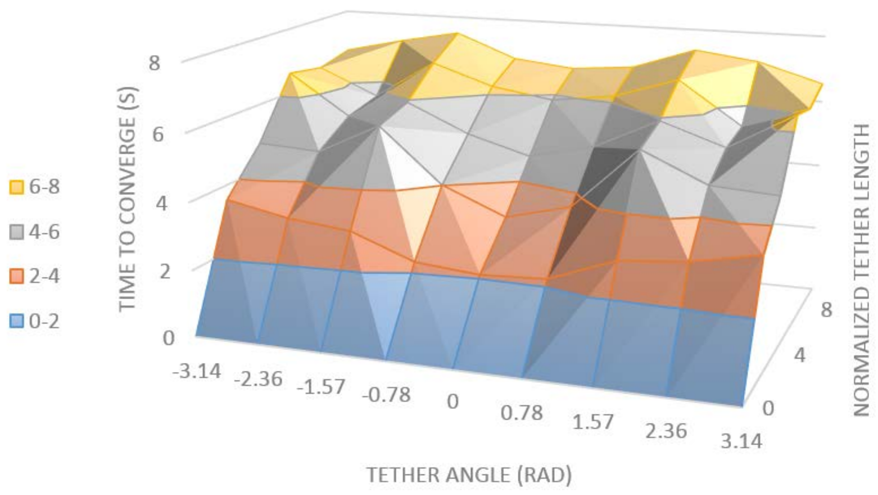
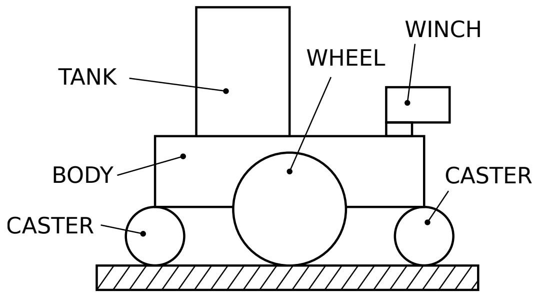
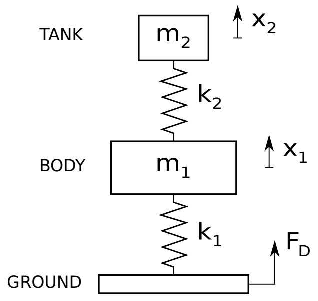


Figure 2.10: Stability of *Follow the leader* control.



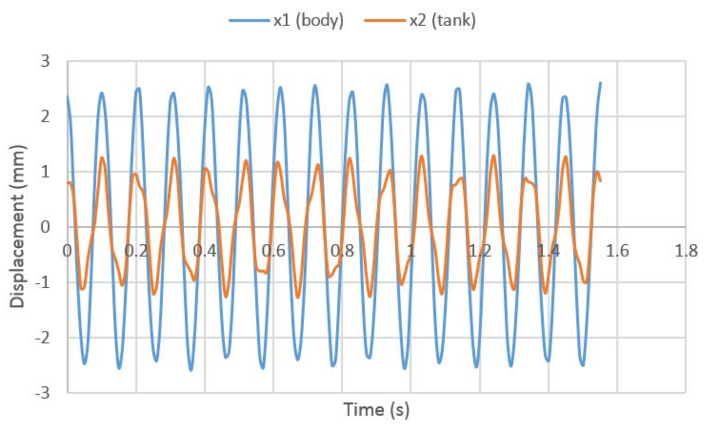
(a) Robot vibration model.



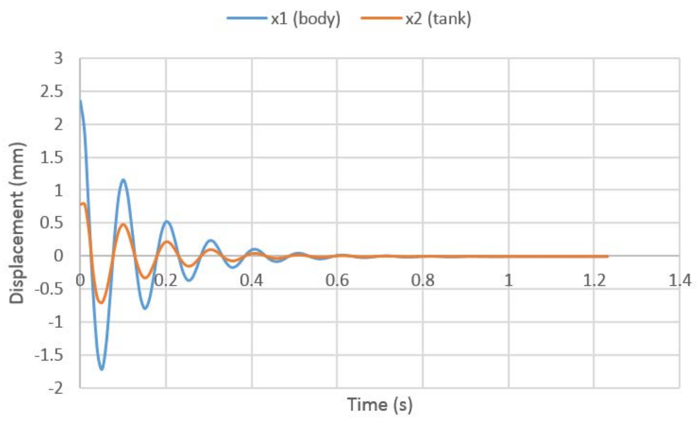
(b) Spring-mass model.

Figure 2.11: Spring-mass model robot carrying oxygen tank.

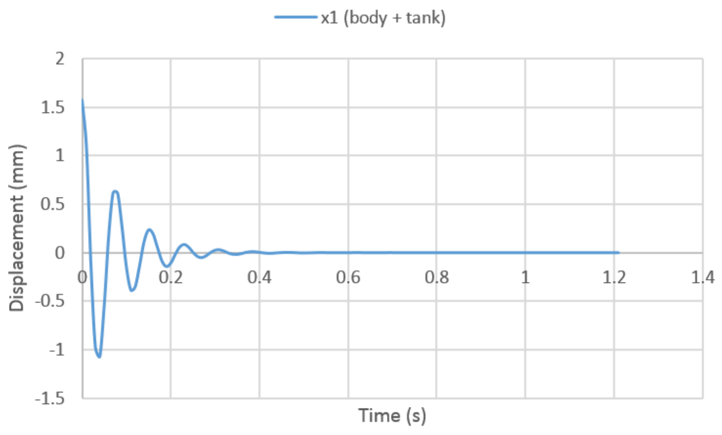
As a final consideration, it is also possible to reduce the vibration further if we can fix the oxygen tank to the robot body, thus combining them into a single effective mass (Figure 2.12c). This limits the maximum displacement of the robot body to around 1 to 1.5 mm, which is rapidly damped. We can therefore assume that the vibration will have a reasonably small effect on the tether tension. However, this may also introduce unwanted mechanical shock to the oxygen tank so the tank safety specification should be considered.



(a) Lack of damping causes continuous oscillation.



(b) Including damping from neoprene tyres in the model.



(c) Fixing the tank to the robot body so they move as one mass.

Figure 2.12: Vibration response of robot carrying oxygen tank to a step input.

2.5 Characterization of Parameters

The control laws for *Pseudo-Joystick* and *Follow the Leader* introduced a number of parameters which affect the performance of the robot follower. This section describes simulations conducted to characterize the effect on follower behaviour of the following parameters:

- Desired length of tether l_d
- Lookahead distance δ (*Follow the Leader* only)
- Recorded path step size Δs_{min} (*Follow the Leader* only)

Each of the above parameters was varied and the simulation repeated with other simulation conditions held constant. The simulation environment was the same as for previous experiments (described in Section 2.4.1). The leader was set to move around a 'figure 8' path (loop diameter 4 m, path length around 20 m) at a constant speed of 0.5 m/s.

For each simulation, the following three output metrics were calculated:

- Measured length of tether l_m
- Robot velocity V_r
- Normal path deviation

The metrics are described in more detail in Section 2.5.1.

2.5.1 Metrics to Analyze Following Behaviour

2.5.1.1 Length of Tether Profile

The length of the tether will always show some variation during leader following. This may not be a problem in most cases, provided the robot's winch has an ample length of tether. However, if the physical tether is limited in length, or if there is another physical constraint on the system (e.g. if there is also an electrical cable or a fuel line connected between the robot and the leader), then it is important to limit the variation in the tether length. For these reasons, for each simulation the variation of tether length was plotted against the distance travelled by the robot.

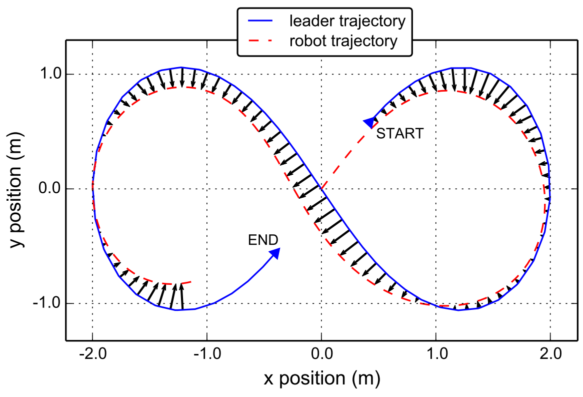
2.5.1.2 Robot Velocity Profile

The forward velocity of the robot, V_r, is plotted against the distance travelled by the robot. If the velocity profile shows sharp, rapid bursts of acceleration then the energy efficiency of the robot is likely to be reduced, and the user may also perceive the robot's motion as 'jerky'.

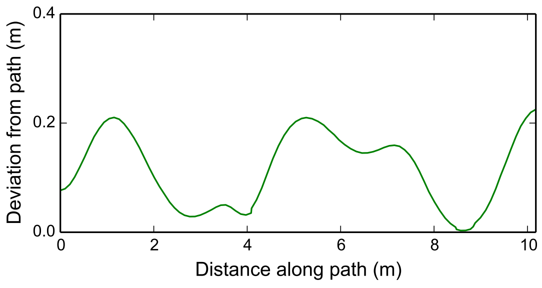
2.5.1.3 Normal Deviation Profile

Clearly, one of the main metrics for follower performance is how closely the robot can follow the leader path. To quantitatively measure the accuracy of the follower trajectory compared to the leader, we introduce a metric named *normal path deviation.* To calculate *normal path deviation* we divide the leader path into small segments, and calculate the normal (perpendicular) distance to the robot trajectory for each segment. An example of *normal path deviation* is shown in Figure 2.13.

We define *normal path deviation* from the leader path to the follower path over the valid comparison range: the uninterrupted section where both paths 'overlap' (see Figure 2.14a). Although path deviation is qualitatively obvious when comparing two trajectory plots, obtaining a quantitative measurement is a rather complicated. The algorithm, to cope with two arbitrary paths must handle intersections, duplicate points, re-segmentation and selection of the comparison range. An overview of this algorithm is shown in Figure 2.14b, while a more detailed explanation is included in Appendix A.

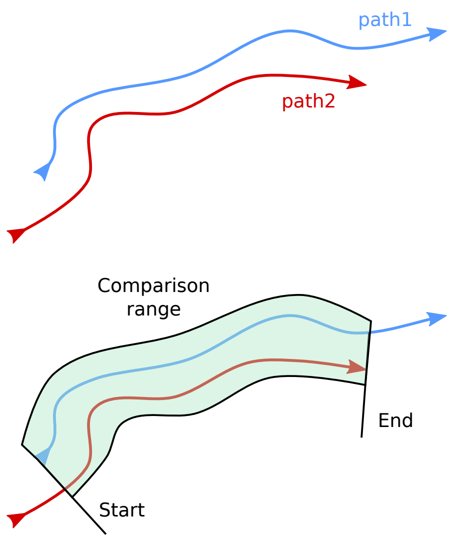


(a) Example trajectory showing normal path deviation. Normal vectors are shown as black arrows.

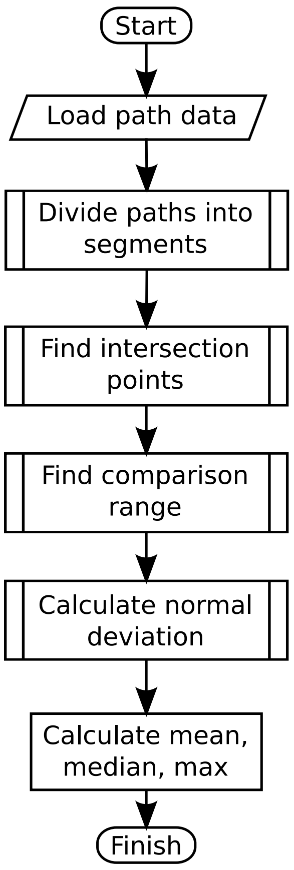


(b) Normal path deviation profile.

Figure 2.13: Example plots for normal path deviation.



(a)


Start
Load path data
Divide paths into segments
Find intersection points
Find comparison range
Calculate normal deviation
Calculate mean, median, max
Finish

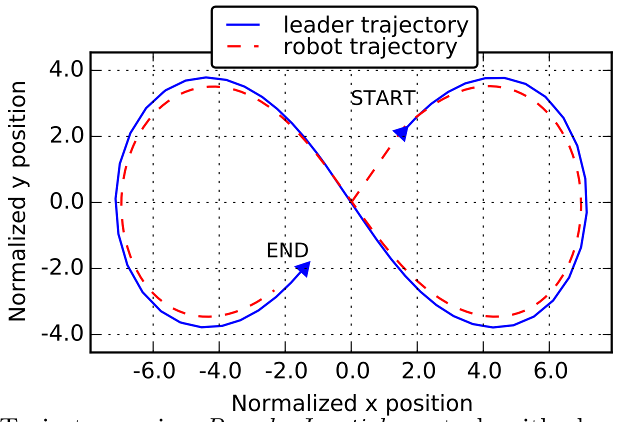
(b)

Figure 2.14: (a) Path comparison range. (b) Normal path deviation algorithm. Full details of the algorithm are explained in Appendix A.

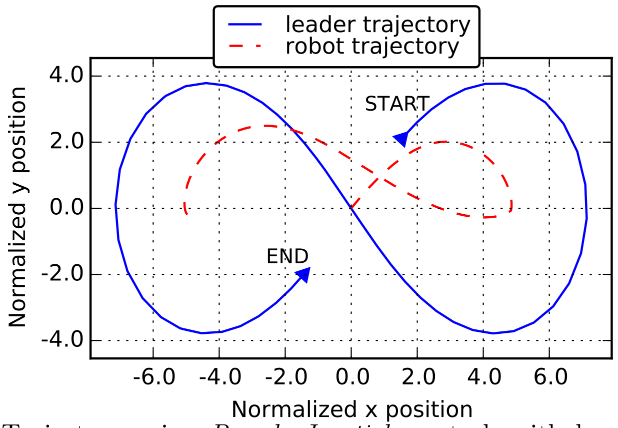
2.5.2 Effect of Length of Tether l_d on Pseudo-Joystick Control

The resulting trajectories for non-dimensional tether lengths 1.0 and 4.0 (0.5 m and 2.0 m in real measurements) are shown in Figure 2.15a and 2.15b respectively. Clearly, the accuracy of the following performance is greatly affected by the length of the tether. When the length is relatively short (1.0), the robot's trajectory follows the leader's fairly closely. However, with a longer tether, the robot is effectively oversteered and misses large sections of the leader path. There is also a large discrepancy between the distance travelled by the leader (around 40) and the robot (around 20); this may cause problems if additional control algorithms process the paths using these distances as a reference index.

Figure 2.16 shows the calculated metrics measured tether length l_m, robot velocity V_r, and normal path deviation. In all cases, the tether length is relatively stable, due to the velocity control law (2.1) (Figure 2.16a). In Figure 2.16b, we see that the robot velocity V_r varies more greatly as the desired tether length l_d is increased, and becomes quite erratic at the longest length, where the robot stops completely several times. As expected, the deviation from the leader path increases at longer tether lengths (Figure 2.16c). Overall, these results demonstrate that the performance of *Pseudo-Joystick* control is significantly better at shorter tether lengths.



(a) Trajectory using *Pseudo-Joystick* control, with short tether length ($l_d = 1.0b$)



(b) Trajectory using *Pseudo-Joystick* control, with long tether length ($l_d = 4.0b$)

Figure 2.15: Follower performance with short and long tether lengths. With *Pseudo-Joystick* control, the discrepancy between the leader and robot paths increases significantly as the tether length increases. Distances have been normalized according to the vehicle wheel track b.

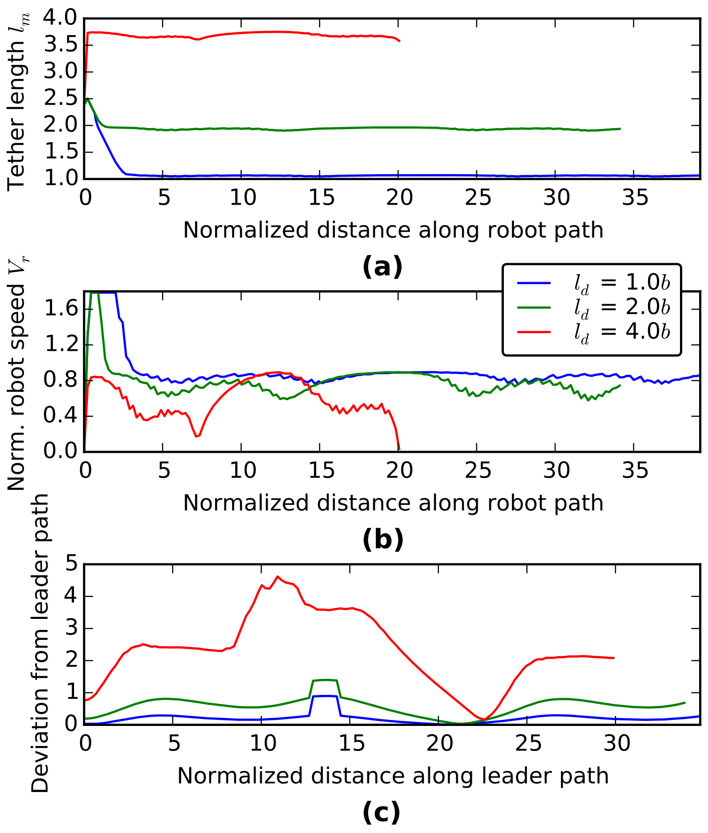


Figure 2.16: Effect of length of tether on follower performance. (a) Measured tether length l_m; (b) Robot velocity V_r; (c) Normal path deviation. As the desired tether length is increased, the velocity becomes more erratic and the deviation increases, so the follower performance is worse.

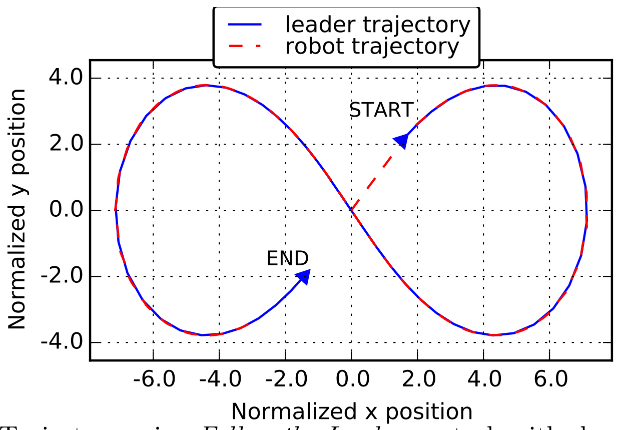
2.5.3 Effect of Length of Tether l_d on Follow the Leader Control

Since it records the history of the leader position, *Follow the Leader* control is mainly unaffected by variations in the length of the tether. Simulations confirmed that the results were similar with short and long tether lengths.

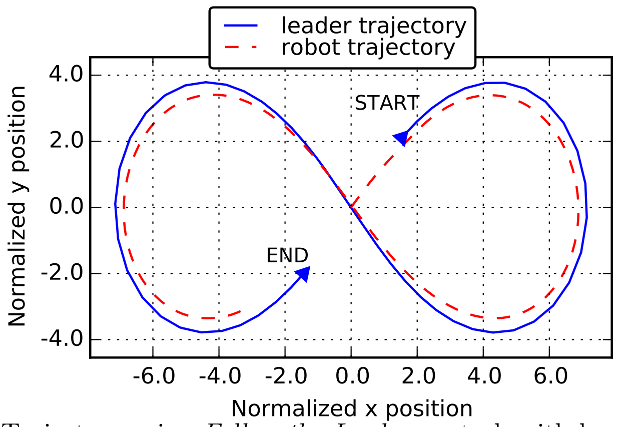
2.5.4 Effect of Lookahead Distance δ on Follow the Leader Control

The resulting trajectories for non-dimensional lookahead distances 0.2 and 2.0 are shown in Figure 2.17a and 2.17b respectively. The calculated metrics are presented in Figure 2.18. When the lookahead distance, δ, is shorter, the robot can follow the leader path more accurately. However, the robot also tends to weave (zig-zag) back and forth across the path, as evidenced by the small oscillations on the $\delta = 0.2$ curves on Figures 2.18a and b. With a moderate lookahead distance ($\delta = 1.0$) the accuracy of the trajectory is relatively high, and there is no weaving behaviour. When the lookahead distance is increased further ($\delta = 2.0$) the accuracy of the trajectory is adversely affected (Figure 2.18c).

When selecting the value of the parameter δ, it is necessary to balance the desired accuracy of trajectory with the elimination of weaving behaviour.



(a) Trajectory using *Follow the Leader* control, with short lookahead distance ($d = 0.2$)



(b) Trajectory using *Follow the Leader* control, with long lookahead distance ($d = 2.0$)

Figure 2.17: Follower performance with short and long lookahead distances. Distances have been normalized according to the vehicle wheel track b.

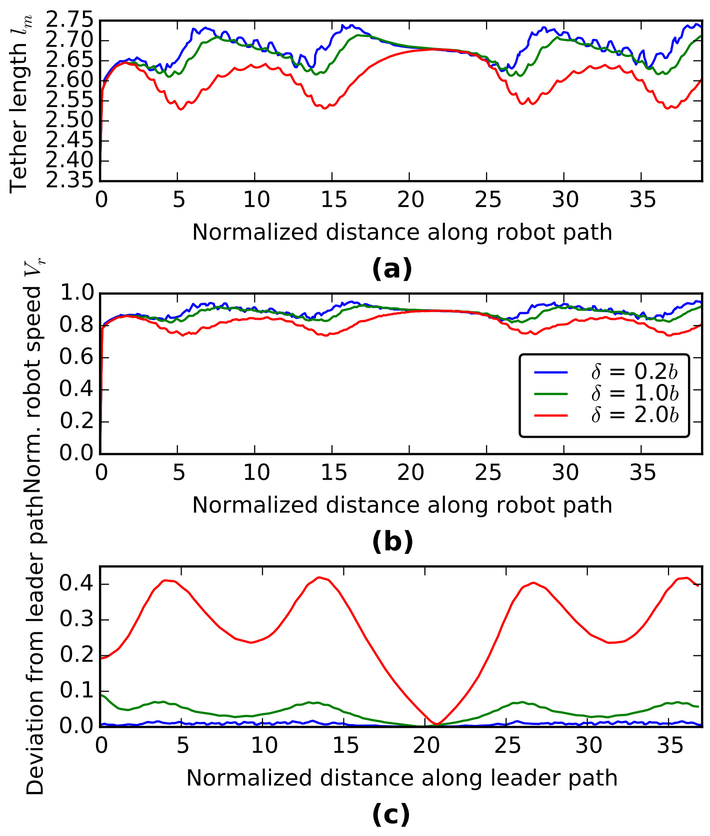


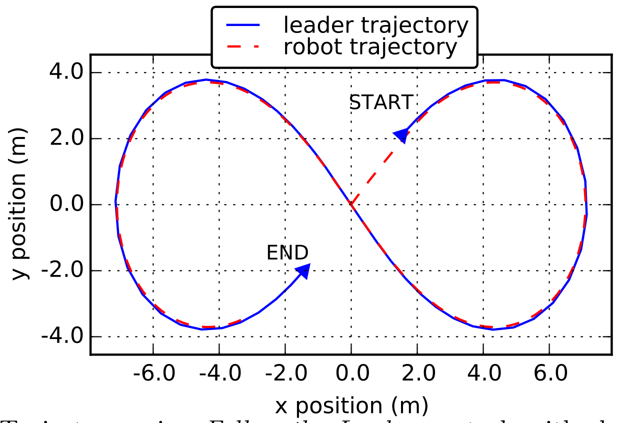
Figure 2.18: Effect of lookahead distance on follower performance. (a) Measured tether length l_m; (b) Robot velocity V_r; (c) Normal path deviation.

2.5.5 Effect of Step Size Δs_{min} on Follow the Leader Control

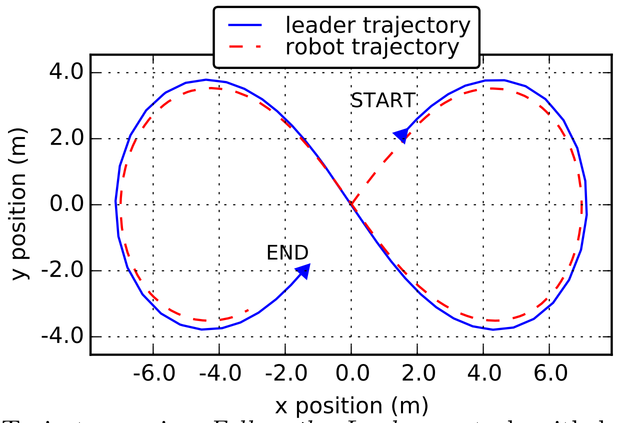
The resulting trajectories for non-dimensional step size 0.02 and 0.8 are shown in Figure 2.19a and 2.19b respectively. The step size effectively determines the resolution with which the leader path is tracked: when the step size is small, the leader path is tracked accurately and the follower robot's trajectory closely follows the leader's (Figure 2.19a). Conversely, when the step size is larger the tracked path has lower resolution and the robot's trajectory is seen to deviate further from the leader's (Figure 2.19b). Further increases in the step size result in further loss of resolution from the tracked path and the robot will 'cut corners'.

Figure 2.20 shows the calculated metrics measured tether length l_m, robot velocity V_r, and normal path deviation. The effect of step size Δs_{min} on tether length is negligible at smaller step sizes, however with increasing step size the leader path will eventually be reduced in resolution to the point where corner cutting occurs and the tether length variation also increases (Figure 2.20a). Figure 2.20b shows the effect on robot velocity V_r is negligible. The deviation profiles in Figure 2.20c suggest that normal path deviation increases with Δs_{min}; there is a significantly greater deviation at the highest step size tested ($\Delta s_{min} = 0.8$).

Based on the simulation results, we should select a smaller step size to ensure the follower trajectory accurately matches the leader. However, there is also a lower limit on step size, determined by the robot hardware and software implementation. A lower step size will require a greater amount of memory to store the recorded leader path, and this may be restricted if the hardware uses a microcontroller with limited memory. The computational load may also be increased (depending on the nature of the algorithm implementation) with a smaller step size. The simulations in this thesis typically use a step size of $\Delta s_{min} = 0.02m$.



(a) Trajectory using *Follow the Leader* control, with short step size ($\Delta s_{min} = 0.02$)



(b) Trajectory using *Follow the Leader* control, with long step size ($\Delta s_{min} = 0.8$)

Figure 2.19: Follower performance with short and long step sizes. Distances have been normalized according to the vehicle wheel track b.

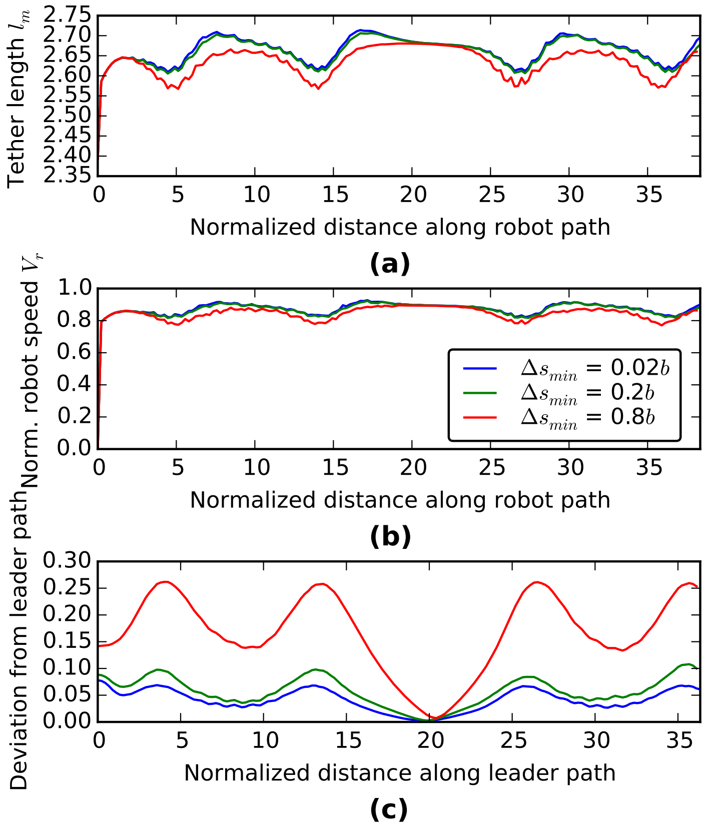


Figure 2.20: Effect of recorded path step size on follower performance. (a) Measured tether length l_m; (b) Robot velocity V_r; (c) Normal path deviation.

2.6 Chapter Summary

This chapter described a differential drive robot model, with a tether and winch. Two major control methods were presented with theory and simulated trajectories: *Pseudo-Joystick* and *Follow the Leader*. *Normal path deviation* was introduced as a metric to compare the accuracy of leader following. An investigation into the effects of control parameters found that *Pseudo-Joystick* control was negatively affected by longer tether lengths, while *Follow the Leader* was unaffected.

Chapter 3

Motion Capture Experiments

3.1 Introduction

Motion capture experiments were conducted in order to evaluate the performance of the previous leader following algorithms in a controlled environment. This chapter describes the experimental setup and results from those experiments. Furthermore, it was informative to compare the motion capture results to the performance in a less controlled environment. For this reason, study with Home Oxygen Therapy patients was analysed to obtain the leader and robot trajectories. Selected relevant results from a user questionnaire are also presented to compare the *Pseudo-Joystick* and *Follow the Leader with Constant Distance* methods.

3.2 Mobile Robot Platform

The leader following experiments were conducted using the mobile robot shown in Figure 3.1 (developed by Endo et al. [40], [43]). Table 3.1 lists the specification of the robot. The chassis has four wheels in a rhomboid configuration: two large diameter active wheels on the left and right, and two passive casters wheels on the front and rear. An inclined parallel bogie linkage connects the front caster to the main chassis and allows the robot to traverse vertical steps up to 80 mm (the height of a typical street curb in Japan) with minimal driving torque. Each active wheel is powered by a 20 W in-wheel motor with a gearhead and an additional bevel gear giving a total reduction of 86.4:1.

The tether base is attached to a compact, lightweight winch reel which can pivot freely around the yaw axis, and a constant force spring keeps the tether under tension. The tether length and angle are measured by rotary potentiometers. The control algorithms were implemented in *C* and compiled for the SH2 microprocessor on-board the robot.

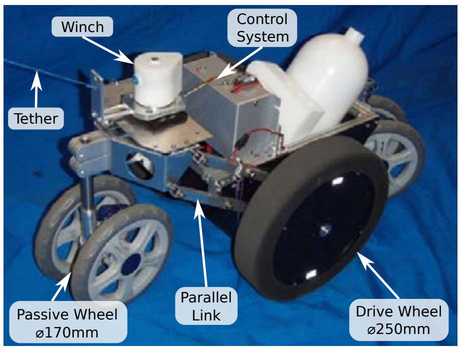


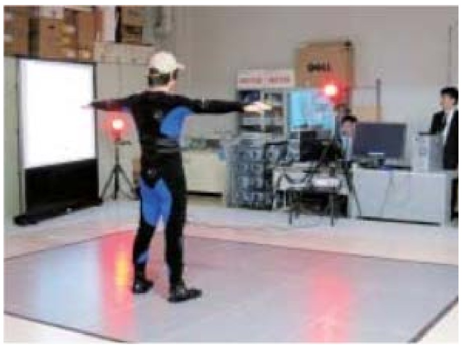
Figure 3.1: Oxydog robot: two large wheels are powered by in-wheel motors, parallel bogey linkage allows step climbing.

Table 3.1: Follower Robot Specification

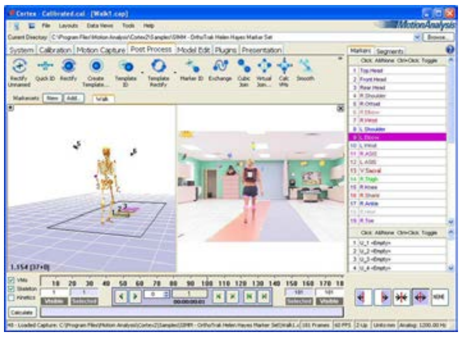
Dimensions L×W×H	670×330×350 mm
Wheel radius	125 mm
Wheel track	280 mm
Mass	7.5 kg
Max. Velocity	1.0 m/s
Max. Step Height	90 mm
Operating Time	180 min
Payload	2.5 kg

3.3 Motion Capture System Description

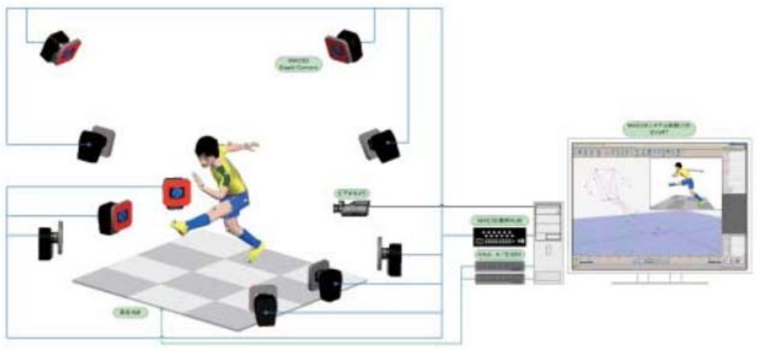
The experiments were conducted in a specialised motion capture room fitted with a commercial capture system made by *Motion Analysis*[44]. The system, shown in Figure 3.2, used 10 digital cameras to measure the position of reflective markers with an accuracy of ±1 mm (sampling rate: 200 Hz).



(a)



(b)



(c)

Figure 3.2: Motion capture system. (a) subject wearing suit with reflective markers; (b) *Cortex* software interface; (c) layout of cameras in motion capture room.

3.3.1 Preliminary Experiments

Several short preliminary experiments were conducted in order to become familiar with the equipment and the tracking, recording and analysis processes. Several problems were identified at that time, along with appropriate solutions:

Marker size and cleanliness: A small number of the available markers had become dirty and the cameras were sometimes unable to detect them. Old, dirty markers were avoided in future experiments.

Marker detection and occlusion: Occasionally markers became occluded by one of the subject's garments. This was remedied by ensuring clothing was securely fitted.

Marker identification: The recording software sometimes mixed up the markers so that, although they were all tracked, the model skeleton was wrongly configured. In future experiments this was solved by calibrating the model to include the full range of motion of the joints.

3.4 Motion Capture Experiments

3.4.1 Experiment Setup

Reflective markers were placed at various points on the user and the robot as shown in Figure 3.3, especially measuring the position of the user's waist and the robot's center. Calibration, data collection and post-processing was performed using *Cortex* analysis software. Figure 3.4 shows the experimental setup and the resulting motion capture model.

In the experiment, the robot followed a healthy human leader around a series of simple obstacles. Three obstacles were placed in a straight line at 1.5 m intervals (similar to Figure 3.7), and the user was instructed to walk from one end to the other and back while weaving in-and-out of the obstacles (a total distance walked of around 10 to 12 m). Prior to each experiment, the robot was set to use either *Pseudo Joystick* or *Follow the Leader with Constant Distance* control, and the experiment was then repeated using the second control method. Five different subjects participated in using the robot for these tests.

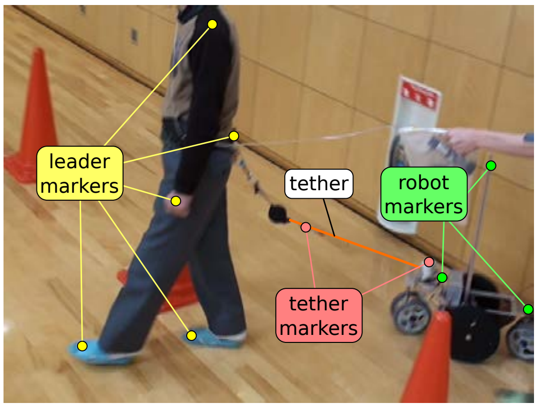


Figure 3.3: Position of reflective markers in motion capture experiment.

3.4.2 Results

Figure 3.5 shows the results from one of the subjects, and the results for all test subjects are summarized in Table 3.2. As before, the robot follows the leader's path more closely with *Follow the Leader with Constant Distance* (Figure 3.5b) than with *Pseudo Joystick* control (Figure 3.5a) in most cases. Though *Pseudo Joystick* follows the general trajectory, Figure 3.5c shows it had significantly greater deviation from the leader path, and this gave rise to a risk of collision with the obstacles. While *Follow the Leader with Constant Distance* control had low deviation and less risk of collision, there was still some error in its trajectory. This was likely due to slip between the drive wheels and the floor surface having a detrimental impact on the robot's ability to turn, and the possible offset between the actual measured tether tip and the tracked marker on the subject's waist.

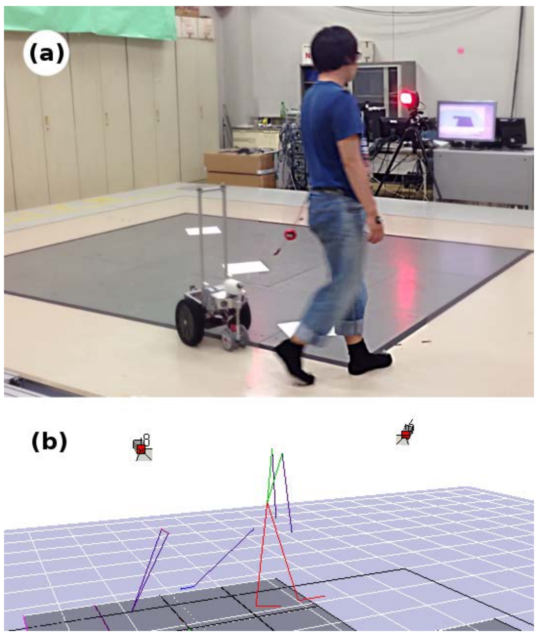
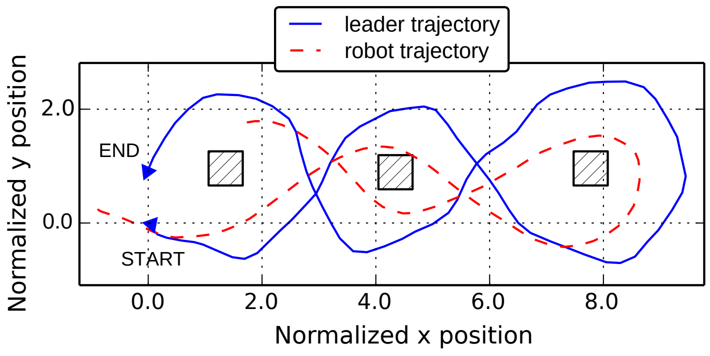


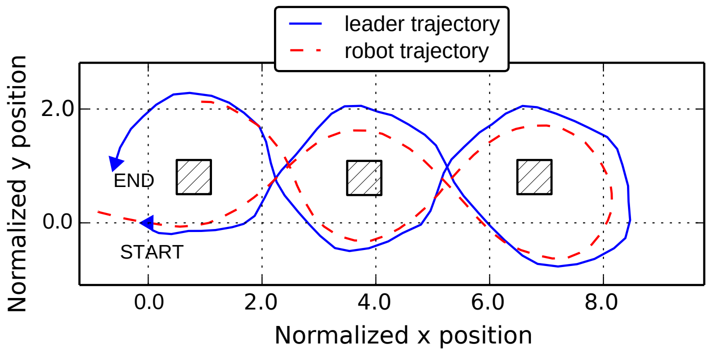
Figure 3.4: Motion capture experiment. (a) motion capture room; (b) the user and robot models constructed from motion capture data using *Cortex* software.

Table 3.2: Follower performance in motion capture experiment (Values normalized according to vehicle wheel track b)

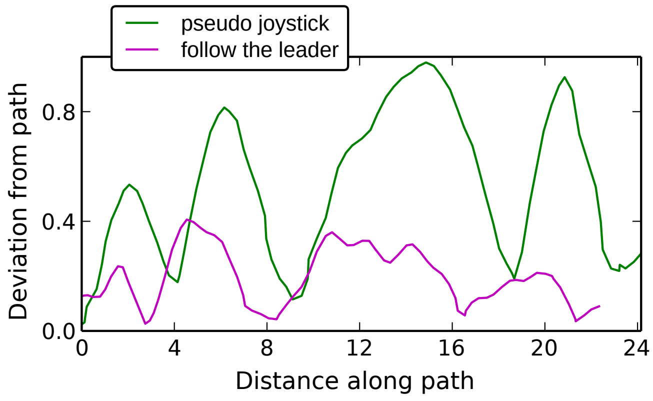
	Normal deviation from leader path			
	Pseudo-Joystick		*Follow the Leader with Constant Distance*	
	Mean	Std. deviation	Mean	Std. deviation
Subject 1	0.334	0.190	0.306	0.170
Subject 2	0.308	0.146	0.336	0.130
Subject 3	0.486	0.270	0.188	0.118
Subject 4	0.534	0.300	0.268	0.190
Subject 5	0.452	0.260	0.286	0.218
Average	0.423	0.233	0.277	0.165



(a) Trajectory using *Pseudo-Joystick* control



(b) Trajectory using *Follow the Leader with Constant Distance* control



(c) Comparison of normal path deviation

Figure 3.5: Motion capture experiment results (for one subject).

3.5 Comparison with User Experiments

After confirming that the robot could perform adequately in a controlled environment, the next step was to compare the motion capture results against performance in a less controlled environment. This section describes analysis of data collected during a study by Endo et al. *Study on a practical robotic follower to support home oxygen therapy patients - Questionnaire-based concept evaluation by the patients*[45]. In this application, the follower robot is being used as an assistive device for patients using Home Oxygen Therapy. The experiments were conducted under medical supervision, in collaboration with *Meeting for the Pulmonary Rehabilitation Studies in Hokushin* and its adjunct group *Hokushin Flying Disc Club.*

3.5.1 OxyDog

OxyDog is a follower robot designed to improve the quality of life of H.O.T. patients by carrying the H.O.T. equipment, thus reducing their physical burden and increasing their freedom of movement. The ultimate goal of this concept is shown in Figure 3.6. Details of the OxyDog specification were described in Section 3.2. H.O.T. uses a cannula of limited length between the oxygen supply on the robot and the user, so to avoid stressing this cannula it is necessary to keep the distance between the robot and the user constant (or close to constant).

3.5.2 Experiment Description

The follower experiment was very similar to the motion capture experiment described in Section 3.4.1: each participant was instructed to walk from one end to the other and back while weaving in-and-out of obstacles placed at 1.5 m intervals (see Figure 3.7). At first, the purpose of the research was explained to all participants, including a description of

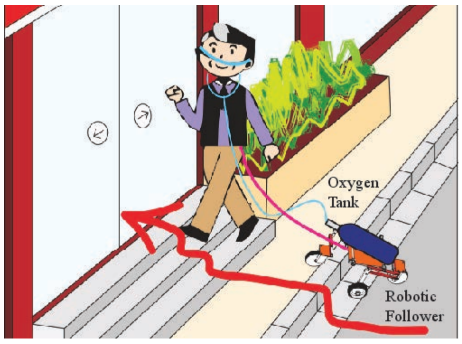


Figure 3.6: Goal of Oxydog robot. The robot carries the user's oxygen tank and follows them around as they go about their daily life.

the robot and how to operate it. For each volunteer, the robot tether was attached to a waist belt, and the robot was randomly assigned to use either control 'method A' (*Pseudo-Joystick* control) or 'method B' (*Follow the Leader with Constant Distance* control). As in the previous experiment, the user was asked to walk in and out of cones placed at 1.5 m intervals (see Figure 3.7), while the robot followed them (a total distance walked of around 16 to 18 m). The subjects were instructed that, when using *Pseudo-Joystick* control they should occasionally look back to check the robot's position; these additional instructions were necessary since early tests had shown that without glancing back it was almost impossible to avoid obstacles. A researcher walked closely behind the patient to provide assistance in case of any unexpected problems, and medical staff were present to supervise (Figure 3.8 shows patients operating the robot).

After completing the walking activity with both control methods, each patient was asked to answer a short questionnaire. If a patient felt unable to answer all of the questions for any reason, they were able to skip questions (always under medical supervision).

3.5.3 Video Analysis Method

Due to the limited space available in the testing area, and the risk of burdening patients by attaching sensors, it was not possible to use motion capture or other sensing equipment to

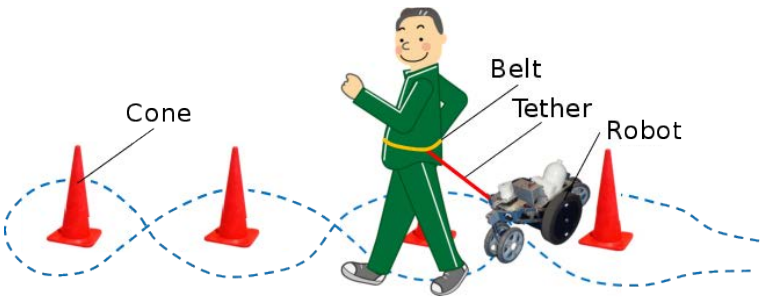


Figure 3.7: Walking task: the user should walk in and out of the cones. The tether is attached to a belt on the waist.

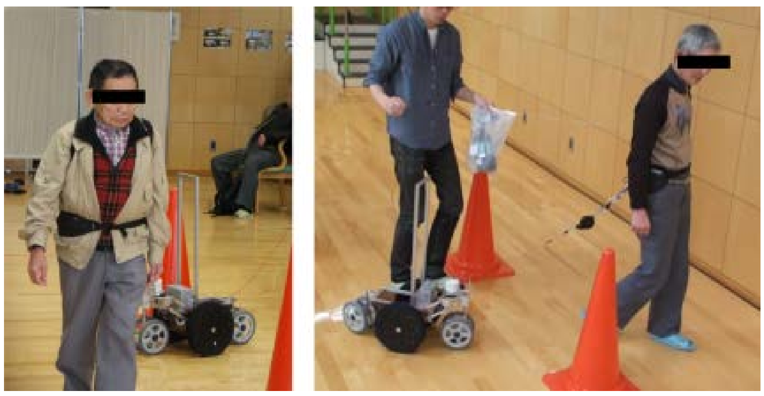


Figure 3.8: Home Oxygen Therapy users participating in experiment *(photo used with subjects' permission)*.

record the position of the user and the robot in real-time. For this reason, the experiments were recorded with a video camera, and this video data was later analysed to determine the trajectory data. The video was analysed, frame by frame, to record the position of the user's feet when they struck the floor. This position was then compared to a known map of the experiment floor to measure the position, and the positions of the left and right feet were averaged to approximate the user's center of gravity (motion capture experiments have confirmed that this gives a reasonable approximation of the user's center of gravity). A similar procedure was used for the robot's wheels. Though coarse, this procedure allowed rough trajectory tracking without overly burdening patients; the accuracy of this method to be around ±40 mm.

Figure 3.9 shows the floor map used in the video analysis process. Figure 3.10 shows a sample of the floor map after being marked with the trajectories. Due to the time-consuming nature of this process, it was only possible to analyse the data of five subjects (selected at random).

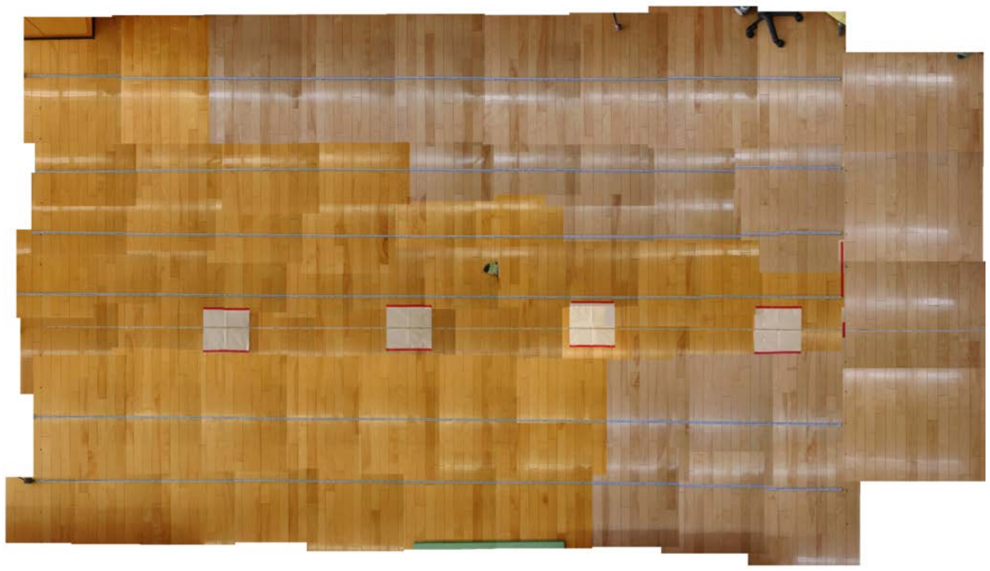


Figure 3.9: Floor map used for video analysis. The video footage was compared to the floor map to locate the coordinates of the user and robot at each time step.

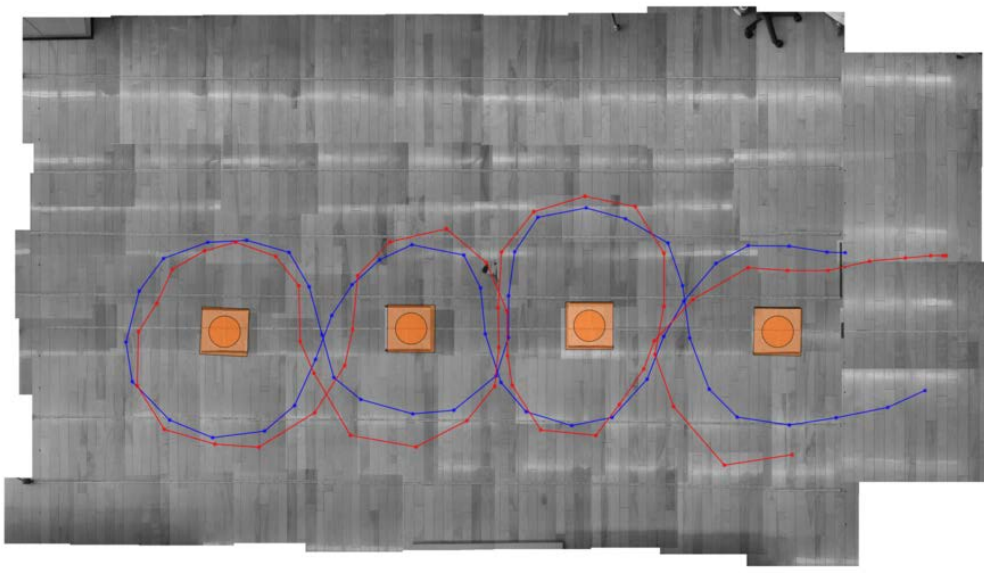


Figure 3.10: Leader trajectory (blue) and robot trajectory (red) marked on floor map. The leader position is marked as the center point between the user's feet during contact with the floor. The robot position was marked as the center point between the two wheel contact points.

3.5.4 Results

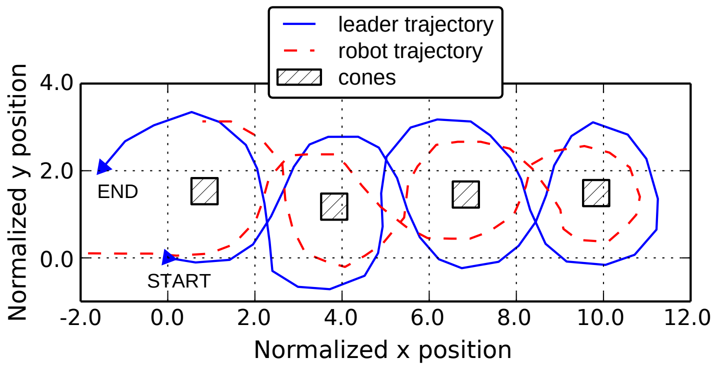
Figure 3.11 shows that the robot was able to follow the patient successfully around the cones using both control methods, and that as expected, Pseudo Joystick control (Figure 3.11a) shows slightly greater deviation than *Follow the Leader with Constant Distance* control (Figure 3.11b). The results for the five subjects analysed are summarized in Table 3.3.

Table 3.3: Follower performance in experiment with Home Oxygen Therapy patients (Values normalized according to vehicle wheel track b)

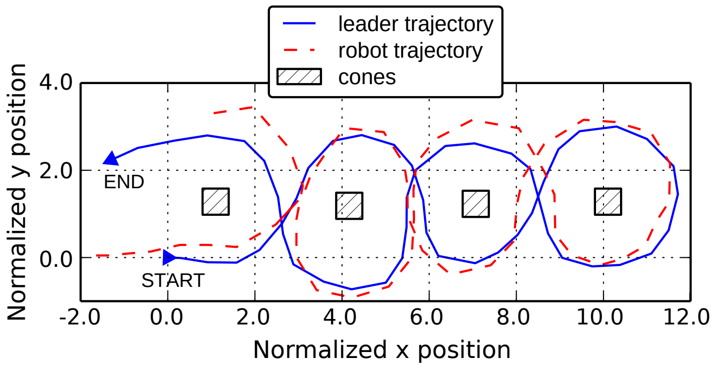
	Normal deviation from leader path			
	Pseudo-Joystick		*Follow the Leader with Constant Distance*	
	Mean	Std. deviation	Mean	Std. deviation
Subject 1	0.306	0.220	0.388	0.290
Subject 2	0.408	0.262	0.238	0.190
Subject 3	0.356	0.220	0.310	0.174
Subject 4	0.358	0.232	0.204	0.156
Subject 5	0.342	0.266	0.254	0.178
Average	0.354	0.240	0.279	0.198

3.6 Questionnaire Survey Results

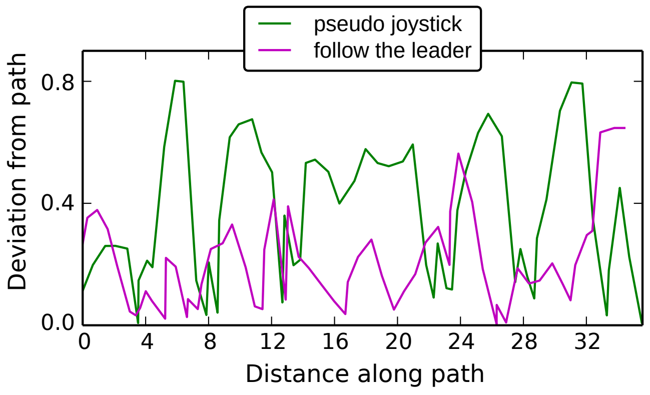
The questionnaire was completed by 14 volunteers after operating the robot; there were 12 men and 2 women, with an average age of 71.6 years. In addition to asking about the patient's basic information, lifestyle, and use of H.O.T., the questionnaire also asked questions about the control of the robot, as presented in Figure 3.12.



(a) Trajectory using *Pseudo-Joystick* control



(b) Trajectory using *Follow the Leader with Constant Distance* control



(c) Comparison of normal path deviation

Figure 3.11: Results of experiment with Home Oxygen Therapy patients (for one subject).

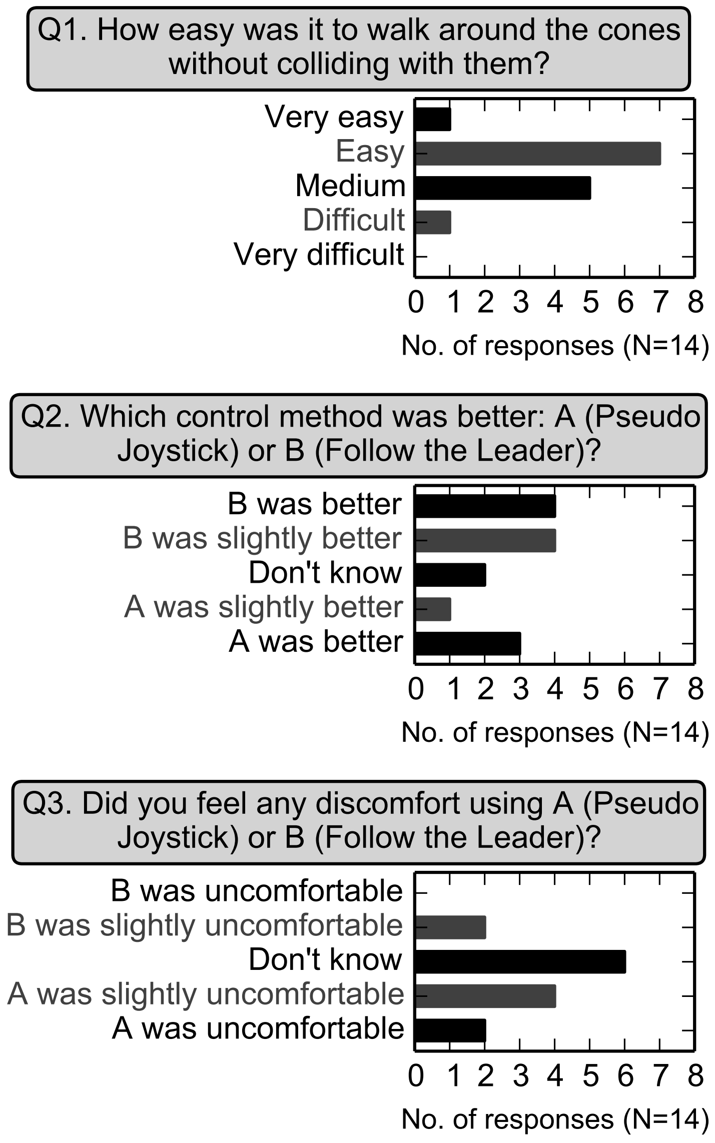


Figure 3.12: Questionnaire survey results. In these questions 'Follow the Leader' refers to *Follow the Leader with Constant Distance* control.

3.7 Discussion

Since both control methods have been shown to give reasonable following performance in the cones task, it is next necessary to examine the questionnaire responses to determine the patients' evaluation and preferences.

The responses to Question 1, 'How easy was it to walk around the cones without colliding with them?' established a baseline for the effectiveness of the robot in this task (Figure 3.12a). Most of the patients responded 'Easy' or 'Very Easy', with only one responding 'Difficult'. This is important as the cones walking task is an approximation of some of the daily activities that real H.O.T. users undertake, such as walking to the shops while avoiding other people, and any assistive device should be able to complete this activity without causing difficulty.

Question 2, 'Which control method was better: A (Pseudo Joystick) or B (Follow the Leader)?', gives a qualitative comparison of the control methods (Figure 3.12b). The results are mixed but show a slight preference for *Follow the Leader with Constant Distance* (8 positive responses) over *Pseudo-Joystick* (4 positive responses). The preference for *Follow the Leader with Constant Distance* may be due to the relative comfort: there is no need to glance backwards at the robot when using it. The fact that other users preferred *Pseudo-Joystick* control may be explained by the better responsiveness: with *Pseudo-Joystick* control, the robot will respond almost immediately to a steering input, while Follow the Leader inherently involves a delayed steering response since it records the history of the leader's position. Thus some users will find *Pseudo-Joystick* more intuitive in this sense. In addition to evaluating the technical efficacy of control methods it is essential to also consider the users' preferences; and the fact that different users prefer different control methods may suggest that user-switchable control could improve the robot's usability.

Question 3, 'Did you feel any discomfort using A (Pseudo Joystick) or B (Follow the

Leader)?', was asked to identify further usability problems (Figure 3.12c). Among the responses there was a clear trend that *Pseudo-Joystick* was more uncomfortable to use (6 'uncomfortable' responses) than *Follow the Leader with Constant Distance* (2 'uncomfortable' responses). *Pseudo-Joystick* may be more awkward to use since it requires the user to occasionally glance backwards, and while this is an easy task for a young, healthy user, it is important to note that it places relatively more physical strain on an elderly person (particularly a person using H.O.T.). These results guide further design revisions as avoiding discomfort is of paramount importance in this application: as the goal is to increase the users' freedom and well-being, the assistive robot must avoid causing any unnecessary distress which could have a negative effect on breathing and overall health. The responses collected so far indicate that *Follow the Leader with Constant Distance* is likely to be a better choice for H.O.T. users.

3.8 Chapter Summary

The first half of this chapter described motion capture experiments used to measure and compare the performance of *Pseudo-Joystick* control and *Follow the Leader with Constant Distance* control under controlled conditions. The second half of this chapter reported and discussed the results of a leader following experiment with H.O.T. patients and a questionnaire survey to gather user feedback.

From the practical experiments it was shown that the *Follow the Leader with Constant Distance* algorithm was capable of following the user more accurately than Pseudo-Joystick, but both algorithms gave reasonable following performance in the walking-around-cones task. The questionnaire survey of H.O.T. users identified that overall they found *Follow the Leader with Constant Distance* to be better and found *Pseudo-Joystick* control to be more uncomfortable. *Pseudo-Joystick* control is likely to be more intuitive to some users because of its immediate response to user commands, but the need to look back and check the robot's position can introduce some discomfort.

Chapter 4

Additional Follower Modes

4.1 Introduction

The previous chapters have investigated *Pseudo-Joystick* control mode and *Follow the Leader with Constant Distance* control mode, analyzing leader following behaviour in simulations and experiments with real users. Based on the outcomes of that research, especially feedback from users, several limitations of the previous control were found. This chapter describes new follower modes, each designed to address a particular problem in leader following.

The sections in this chapter describe the modes listed below, including the motivation for their development, details of the control algorithm and validation by simulation or experiment:

- Side following mode
  - Side joystick mode
  - Side tracking mode
- Front joystick mode
- Brake mode
- Tether snagging avoidance mode

4.2 Side Following

4.2.1 Introduction

During experiments with Home Oxygen Therapy patients, the users raised concerns about the relative position of the robot. It was suggested that the users felt uneasy with the robot following behind them, and a questionnaire survey confirmed this. The question shown in Figure 4.1, *Were you bothered by the robot following you from behind?* was asked to 14 users. The responses tended towards the negative: 5 people selected the moderate response *Normal*, while a further 5 people indicated that they were *Bothered* or *Highly Bothered*. Addressing this issue should help to improve the users' comfort while operating the robot, and in the case of Home Oxygen Therapy patients it should also help to improve health.

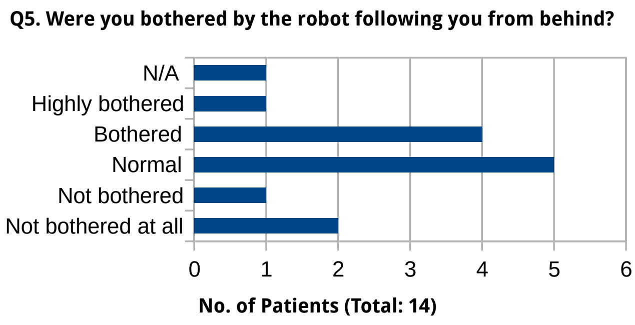


Figure 4.1: Questionnaire result regarding robot position.

The concept of side following is to position the robot so that it is beside the user, and inside their field of vision. Even if the position is slightly behind the user, this may be permissible provided it is still inside the peripheral vision. Consider a person walking with a well-trained dog on a leash, as shown in Figure 4.2. The dog may walk a little in front of the user, or a little behind, but in general the position remains close to the peripheral vision.

Figure 4.2: Dog walking position. Typically, a dog walks beside the owner, inside the field of vision.

Similarly, side following mode should allow the robot to follow beside the user, eliminating (or reducing) the need to turn around to check the robot's position. Two different control modes for side following are described in this section, *Side Joystick* mode and *Side Tracking* mode.

4.2.2 Side Joystick Control

A general outline of *Side Joystick* mode is shown in Figure 4.3. The concept is to use the tether length and angle as relatively direct steering inputs (similar to *Pseudo-Joystick* described in Section 2.3.1). A desired length l_H (this is effectively the desired side offset) is subtracted from the measured tether length, l_m, and multiplied by angular gain K_Ω to give the desired angular velocity of the robot, Ω_r (4.1). In a similar fashion, the desired angle θ_d is subtracted from the measured angle, θ_m, and multiplied by velocity gain K_V to give the desired angular velocity of the robot, Ω_r (4.2).

$$V_r = -K_V(\theta_m - \theta_d) \tag{4.1}$$

$$\Omega_r = K_\Omega(l_m - l_H) \tag{4.2}$$

For typical side following, the target angle θ_d is set to $\pm\frac{\pi}{2}$ radians. Figure 4.4 shows a

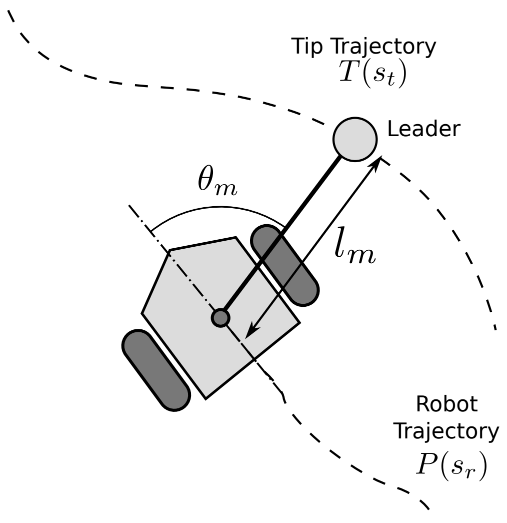


Figure 4.3: Side Joystick control mode

block diagram representation of how side joystick transforms the sensor input into output wheel velocities.

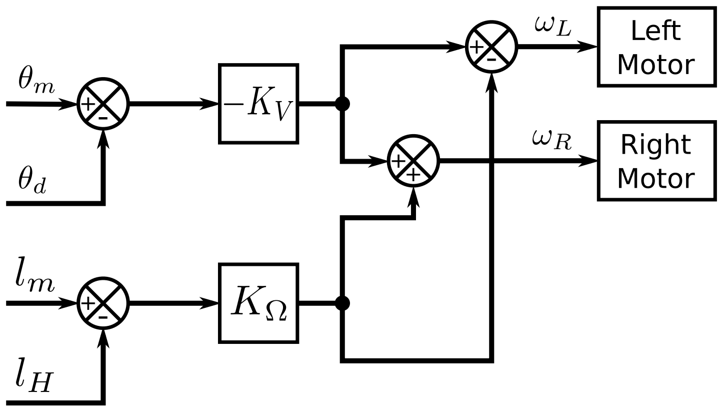


Figure 4.4: Side joystick control block diagram.

4.2.3 Simulation: Side Joystick

Side Joystick mode was implemented in Python, and tested in the same V-REP simulation environment previously described in Section 2.4.1. The results are shown in Figure 4.5. The simulation confirms that the general behaviour is correct: the robot stays beside the leader and adjusts its position accordingly when the tether position changes. However, the follower path is a little distorted in the region of the internal right-hand turn; clearly the algorithm does not handle inside turns sufficiently well. Outside turns are handled much better.

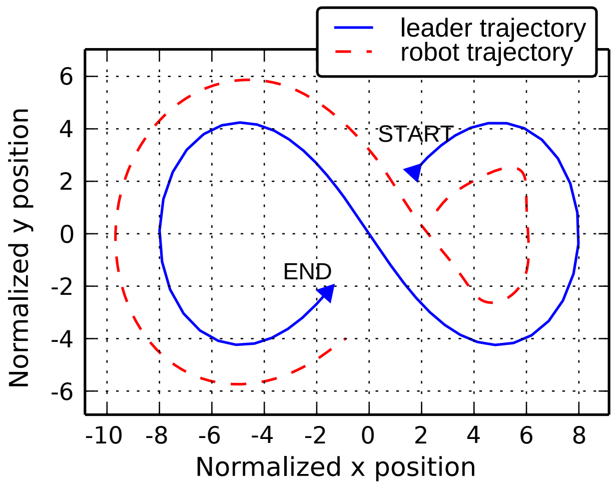
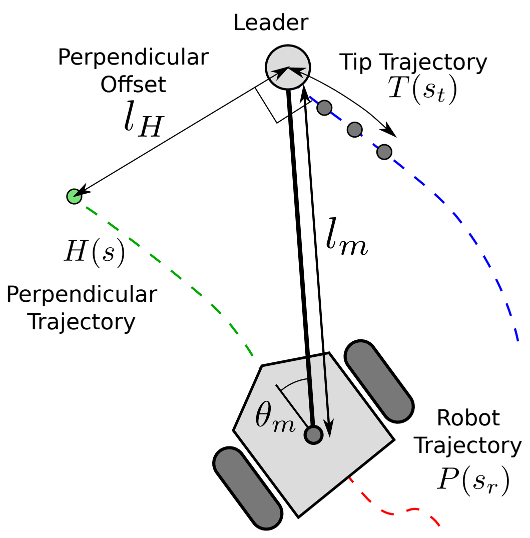


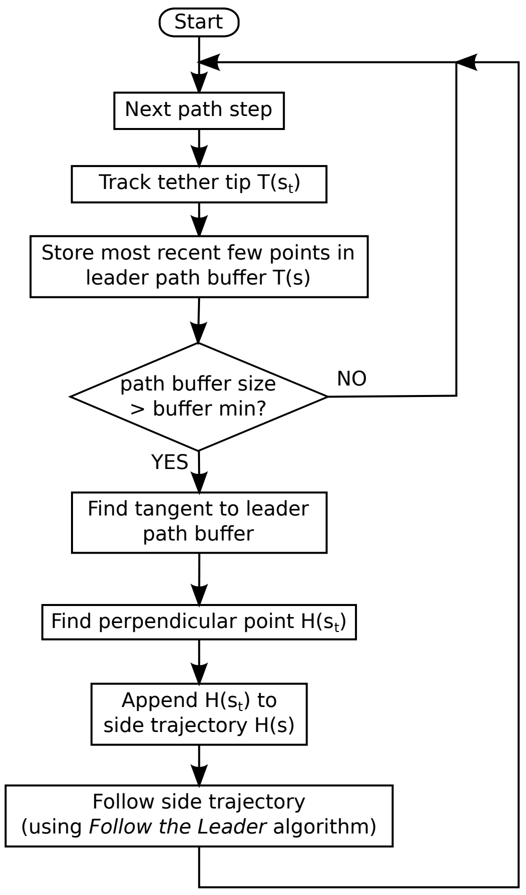
Figure 4.5: Trajectory using *Side Joystick* control, with normalized side offset $l_H = 2.0$

4.2.4 Side Tracking Control

Side Tracking control was designed to improve on the side following performance of *Side Joystick* mode, by recording the history of the leader position and transforming it to a suitable side offset target. A general outline of *Side Tracking* mode is shown in Figure 4.6a, and flow diagram of the algorithm is presented in Figure 4.6b.



(a) Diagram.


Start
Next path step
Track tether tip T(s$_t$)
Store most recent few points in leader path buffer T(s)
path buffer size > buffer min?
NO
YES
Find tangent to leader path buffer
Find perpendicular point H(s$_t$)
Append H(s$_t$) to side trajectory H(s)
Follow side trajectory (using *Follow the Leader* algorithm)

(b) Algorithm.

Figure 4.6: Side Tracking control mode.

In this case, the leader position $T(s_t)$ is recorded and added to a leader path buffer $T(s)$. The length of buffer $T(s)$ is relatively short: it is only necessary to store the previous few points (enough to allow the calculation of a tangent vector). The next step is to calculate the tangent vector to the leader path $T(s)$, as close to the tip point $T(s_t)$ as reasonably possible. The tangent vector is then rotated by π radians, and scaled by the desired side offset value l_H. This new side offset point is then added to a target trajectory $H(s)$, which should be correctly offset from the leader trajectory. The robot can then look up a target point on $H(s)$ using the same method as *Follow the Leader* mode (described in Section 2.3.2.2).

4.2.5 Simulation: Side Tracking

The improved performance can be seen in Figure 4.7: the robot follows beside the leader path with more constant offset, and handles both inside and outside turns well. However, the cost of this improved performance is increased computational load. Determining the tangent can be performed fairly fast, but scaling the perpendicular vector requires an unavoidable hypotenuse calculation ($c = \sqrt{a^2 + b^2}$). Since these are likely to be too slow on embedded microprocessors, it is necessary to sacrifice some accuracy by using a 'fast' simplified version of the sum[46].

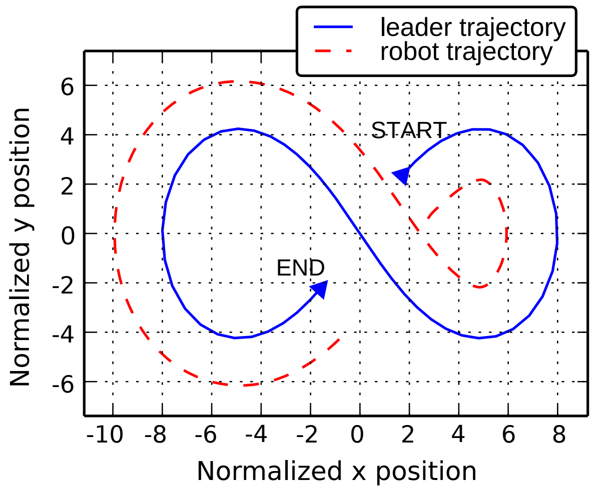


Figure 4.7: Trajectory using *Side Tracking* control, with normalized side offset $l_H = 2.0$

4.2.6 Motion Capture Experiments

Practical experiments were carried out to measure the side following performance using the same motion capture system discussed in Chapter 3.

For the first experiment, the user was instructed to hold the tether and walk forward in a straight line for several metres. The resulting trajectories are plotted in Figure 4.8.

The robot movies beside the user, keeping a relatively straight line with some noticeable weaving.

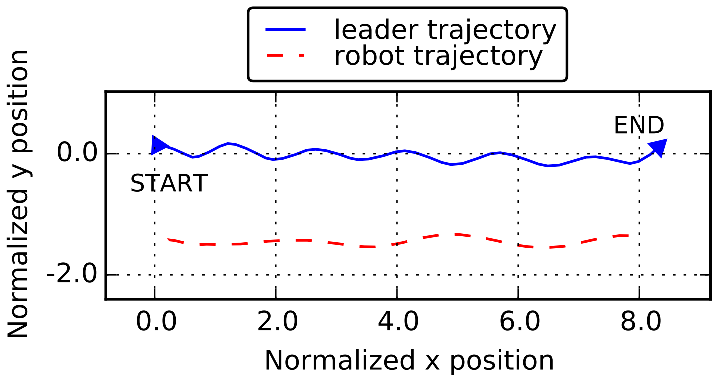


Figure 4.8: Straight line trajectory using *Side joystick* control, with normalized side offset $l_H = 1.5$. The robot follows beside the user, however some slight weaving behaviour can be observed.

After the straight line experiment, the user was instructed to walk in a figure-8 pattern while the robot follows along beside (thus completing a right-hand turn with the robot on the inside, and a left-hand turn with the robot on the outside). The trajectory in Figure 4.9 highlights the difficulty of the inside turn for side following: the robot has to perform a small-radius turn and move slowly (or wait) while the user walks around.

In terms of the user experience it was found that the robot was fairly easy to operate, and the robot stayed close to the user's field of vision most of the time (falling out of view sometimes).

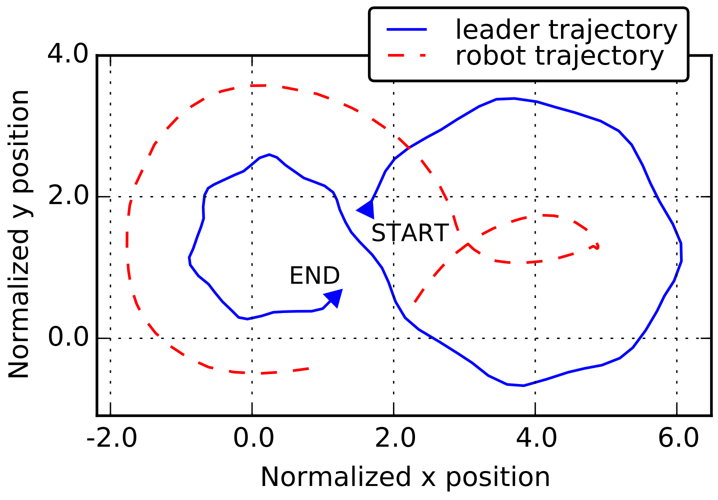


Figure 4.9: Figure-8 trajectory using *Side joystick* control, with normalized side offset $l_H = 1.5$. The robot follows beside the user, however there is some weaving behaviour as the robot does not maintain an even side offset.

4.3 Front Following

4.3.1 Introduction

Side following mode was developed to try to keep the robot in the user's field of view during operation, and therefore potentially improve the user's comfort. However, a disadvantage of side following mode is the decreased maneuverability around obstacles. Positioning the robot next to the user makes it more difficult to pass smoothly through small gaps, for example through doorways.

Front joystick mode is proposed to potentially combine some of the benefits of side following and rear following:

- The user can see the robot and its movements very clearly
- The user and robot can fit through doorways more easily

4.3.2 Control

The operating principle of *Front joystick* mode is demonstrated in Figure 4.10. Basically, the positions of the robot and the user have been switched compared to the original leader following design: the robot is in front of the user and moves forwards (away from the user). The user holds the tether in their hand (as before), and approaches the robot. The measured tether length l_m gets shorter, and this causes the robot to increase its speed, moving away from the user. As the user moves their hand from side to side the measured tether angle θ_m is changed, and the robot steers to the left or right accordingly. An important difference compared to previously examined control modes is that *the user must actively move the tether tip to steer the robot correctly along the desired trajectory*, and in practice this may be relatively difficult compared to previous modes.

The control laws in (4.3) and (4.5) convert the tether inputs into linear and rotational velocity commands.

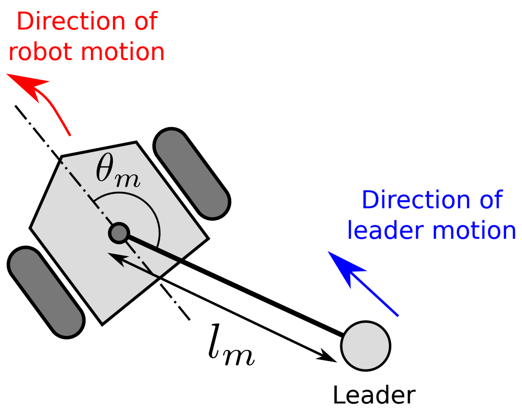


Figure 4.10: Front Joystick control mode. The robot moves in front of the user, who adjusts the tether angle to steer. The user requires some effort to control the trajectory of the robot.

$$V_r = -K_V(l_m - l_d) \tag{4.3}$$

$$\phi_f = \theta_m - \theta_d \tag{4.4}$$

$$\Omega_r = \begin{cases} 0 & \text{if } -\frac{\pi}{2} \leq \phi_f \leq +\frac{\pi}{2} \\ K_\Omega \phi & \text{if } \phi_f < -\frac{\pi}{2} \text{or } \phi_f > +\frac{\pi}{2} \end{cases} \tag{4.5}$$

4.3.3 Motion Capture Experiments

As for previous control modes, motion capture experiments were used to examine the performance of *front joystick* mode in the hardware prototype.

In the first experiment, the user was instructed to walk forwards in a straight line while controlling the robot. As shown by the trajectories in Figure 4.11, the robot moves in front of the user as expected. However Figure 4.11 also shows a slight drift from the intended straight line course. This occured each time the experiment was repeated, and was a result of the inherent difficulty the user experienced controlling the robot in this manner. The general trajectory could be controlled, but not precisely.

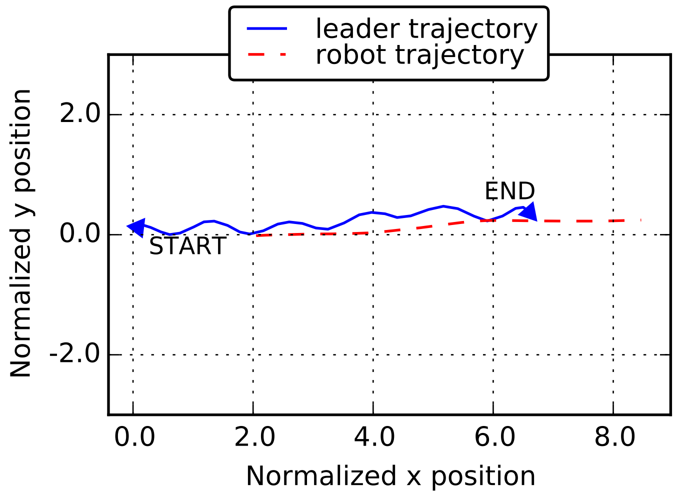


Figure 4.11: Straight line trajectory using *Front joystick* control. Note that even though the desired trajectory is simple, the user must make several steering corrections to keep the robot on course.

Figure 4.12 shows the resulting tracjectories when the user attempted to control the robot around a figure-8 course. Here we can see that the general behaviour is correct, and the robot successfully moves around the figure-8, but we also see that considerable ‘oversteer’ is required to steer the robot around the relatively tight corners.

Overall, hardware experiments found that *front joystick* mode is relatively easy to operate in open situations, but gets more difficult (requiring more effort and skill from the user) in tight maneuvering.

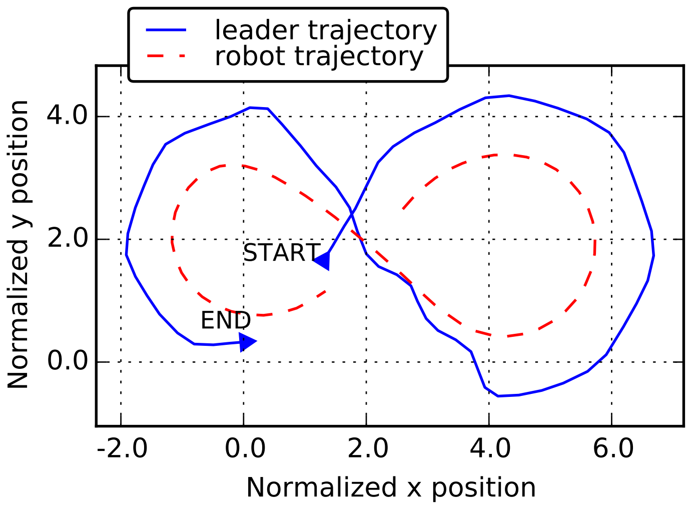


Figure 4.12: Figure-8 trajectory using *Front joystick* control. Note that in order to effectively steer the robot around the course, the user must 'oversteer' considerably by moving wide around the corners.

4.4 Brake Mode

4.4.1 Introduction

When testing robots with real users in 'daily life' environments such as outdoor streets, shops, etc. there are a huge number of potential hazards and obstacles which may affect the user's trajectory. As a simple example, the user may stop abruptly when coming to the edge of a road. At these times, obviously the robot should stop following the user. However, ceasing following may not be enough: in earlier experiments, the robot was observed to creep very slightly forward, or to rotate as the tether moved in the user's hand. Although it may not stray from the leader path, these kinds of creeping motions or twitchy turning have the effect of upsetting, confusing or startling the user. In the worst case, some unnecessary robot movement may prompt the user to change their position in a dangerous manner, for example stepping onto the road.

The motivation for *Brake* mode is clear, and the requirements are also straightforward: the robot should stop when close to the user.

4.4.2 Brake Mode Control

Brake mode operates by checking the tether sensor data once per time step, and if the robot is close to the user, then *Brake* mode is executed. The selection algorithm is shown in Figure 4.13. The criterion 'close to the user' is determined by checking the length of the tether, and setting a threshold value. Below this threshold, the robot enters *Brake* mode, the target velocity is set to zero, and (in the case of the hardware prototype) the motor speed control gains are increased to help maintain position on slopes.

When the user starts walking again, the robot leaves *Brake* mode and returns to the follower mode it had been using previously.

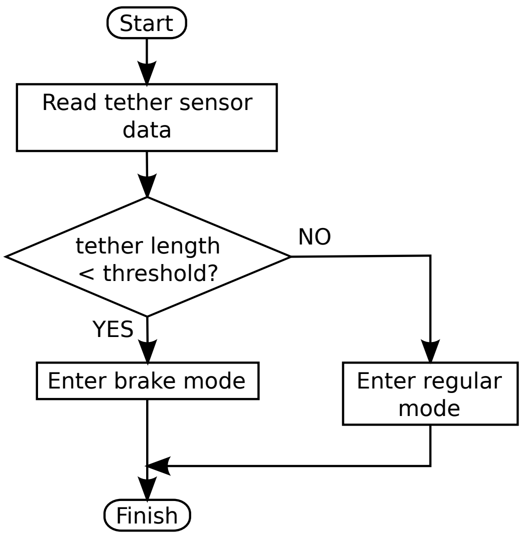


Figure 4.13: Brake mode selection algorithm. The algorithm is executed once at the beginning of each control time step, after reading the tether sensors.

4.4.3 Testing with Hardware Prototype

Brake mode was tested during a number of user experiments in a busy neighbourhood in Osaka, Japan. It was used in conjunction with *Pseudo-Joystick* and *Follow the Leader* control, to ensure that the robot behaved appropriately around the user (an elderly user of Home Oxygen Therapy). The terrain included slopes, curbs and light cobbles, and the route included several road crossings with moving obstacles such as people, cars and bicycles.

As shown in the example in Figure 4.14, the robot stopped and braked whenever the user stopped. It also successfully resumed motion when the user started walking again.

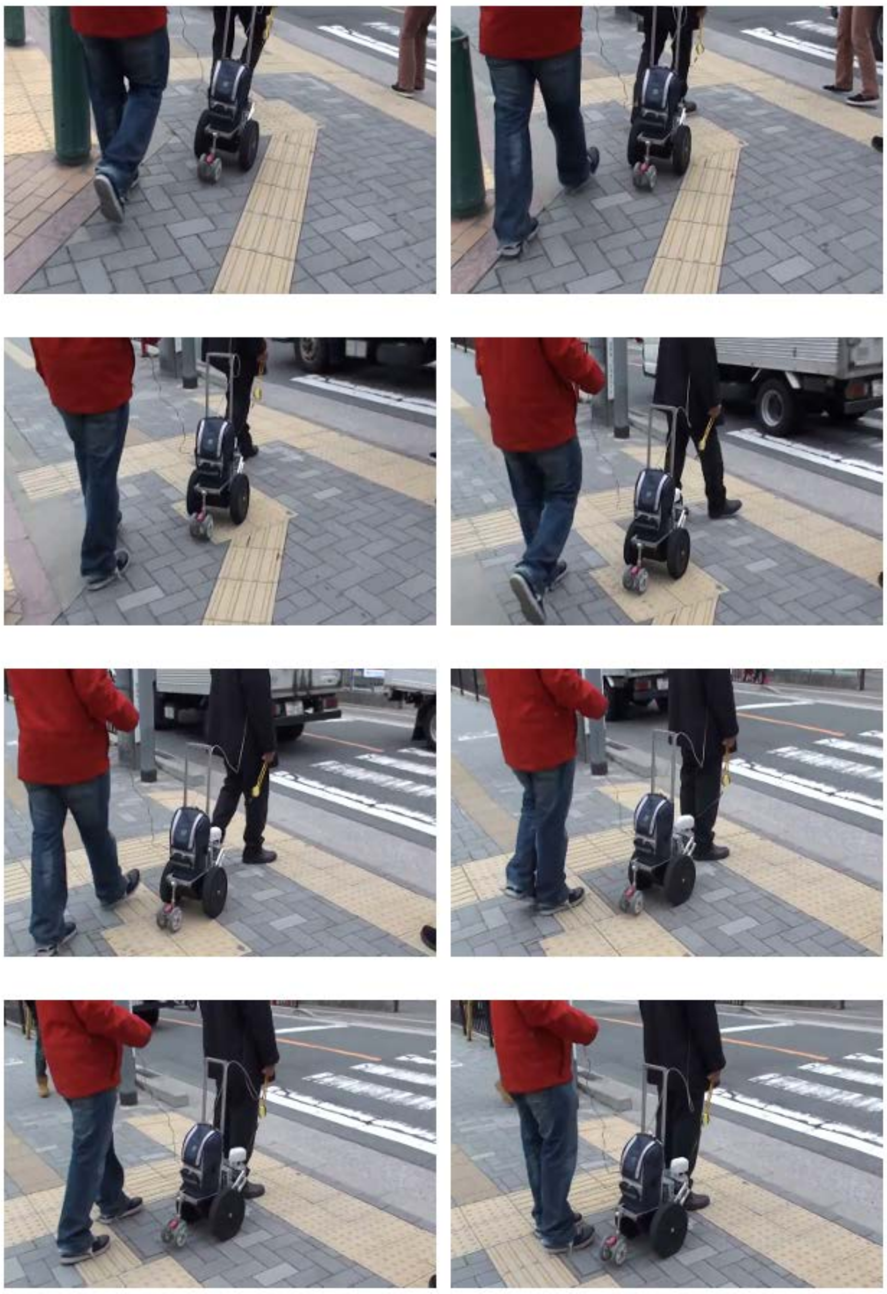
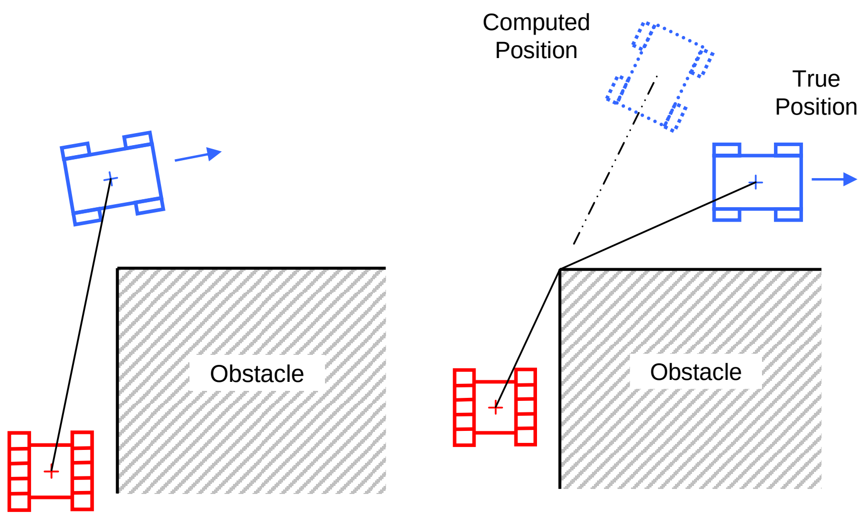


Figure 4.14: Brake mode being tested in a busy street environment by a Home Oxygen Therapy user.

4.5 Tether Snagging Detection

4.5.1 Introduction

Tether snagging occurs when an obstacle lies in between the tether tip and follower in such a way that the tether is bent. In this condition, position tracking becomes difficult since the leader's location cannot be determined (Figure 4.15). Typically, this could happen when the leader moves around a corner: prior to the corner the tether is free (Figure 4.15a), during cornering the tether is snagged and an error is introduced (Figure 4.15b).



(a) Tether is free **(b)** Tether is snagged

Figure 4.15: Tether snagging. When the tether contacts an obstacle, it becomes 'snagged' and the computed position of the leader vehicle becomes incorrect.

In complex environments with rough terrain or unknown obstacles, tether snagging is very likely to be problematic to leader following. In narrow spaces such as a corridor in a building, tether snagging may also occur when moving around corners.

This section investigates the effect of snagging on path tracking for sharp and round obstacles, and presents a simple algorithm to detect when snagging occurs *for limited cases*.

The objective of this study was to characterise snagging behaviour and find the limits of when it occurred and could be avoided. To achieve this, the case of leader following through a corridor was considered, with the aim of finding a 'safe' following distance (gap between leader and follower) where snagging would not occur.

4.5.2 Simulation Environment Parameters

A kinematic simulation was used to investigate tether snagging. Simulation parameters are listed in Table 4.1. The test environment was designed in the shape of a corridor of width w with a single 90°corner. A relatively narrow corridor represents a fairly difficult route for a tethered follower, while the sharp corner was expected to provide the clearest disturbance to the tether and therefore be the easiest to detect. Vehicle size was set to match that of the hardware prototype, developed later.

Table 4.1: Kinematic Simulation Parameters

Parameter	Value
Vehicle length	0.4 m
Corridor width, w	1.0 m
Target following speed, v_F	0.4 m/s
Target following gap	2–5 m
Follower algorithm step size	5 mm
Simulation time step size	0.005 s

The test environment is shown in Figure 4.16. The leader vehicle moves along the centre of the corridor at constant speed, and moves around the corridor in a curve of radius $r_{path} = 0.5\ w$. The simulation was repeated for a range of follower gaps, i.e. the distance between the leader and follower vehicles along the trajectory was varied from 1 m to 5 m.

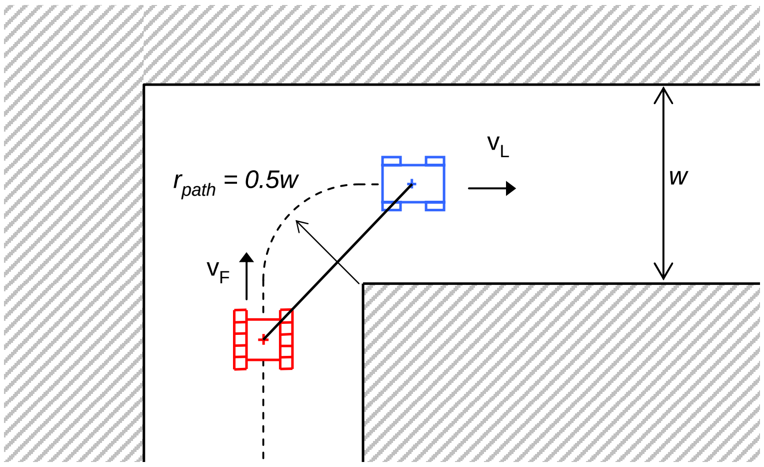


Figure 4.16: Simulation corridor environment. Leader travels down the centre of the corridor, and moves around the corner in a curve of radius $r_{path} = 0.5\ w$.

4.5.3 Effect of Tether Snagging on Path Tracking

Figure 4.17 shows the trajectories resulting from a leader following when tether snagging occurs. The leader path is clearly marked on the figure; this shows the trajectory which would be measured if there was no obstacle and no snagging occurred. It is interesting to note that the follower correctly measures the leader position up to a point just *beyond* the corner; the follower will not collide with the corner itself. Clearly, deviation from the target course occurs after the corner and increases significantly as the gap between the vehicles is increased. For a corridor width $w = 1.0$ m, it can be seen that tether snagging does not occur with a gap of 1 m, and in this case the gap is small enough to allow accurate leader following.

4.5.4 Normalised Limits of Snagging and Collision

To better characterise the limits of when snagging occurs, additional simulations were carried out for following gaps in the range 1.75 m to 2.75 m. The results are presented in Figure 4.18. From this it can be seen that the first gap at which snagging occurs in this

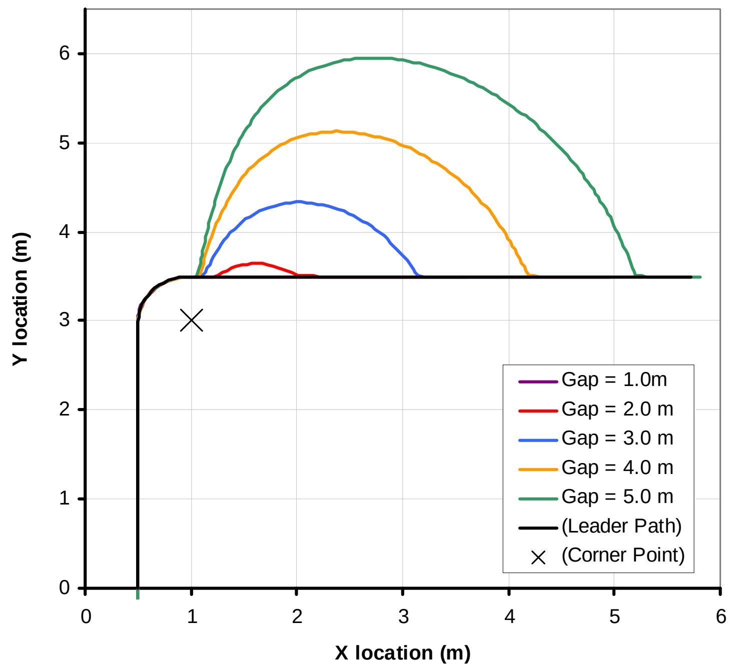


Figure 4.17: Trajectories for leader-following when tether snagging occurs. Gap between leader and follower was adjusted from 1 m to 5 m.

case is approximately 1.75 m, causing a very slight deviation in the measured trajectory. The criteria for avoiding a collision can also be predicted from Figure 4.18. It can be seen that a gap of 2.75 m will cause a collision with the environment wall for any case. For a gap of 2.5 m however, the target path itself does not intersect with the wall, and so the possibility of a collision depends on the vehicle dimensions. For these tests, the vehicle dimensions were 0.4 m $\times$ 0.25 m and the limit for collisions occurs for a gap greater than 2.25 m.

The snagging and collision limit values are relevant for the particular corridor environment considered, which had a width w of 1 m. However the results can be normalised to provide generalised limits. Figure 4.19 shows the parameters affecting collision, where the vehicle dimension is considered as a bounding circle enclosing the vehicle. The bounding circle is an approximation, but it allows all vehicle shapes to be considered in a conservative

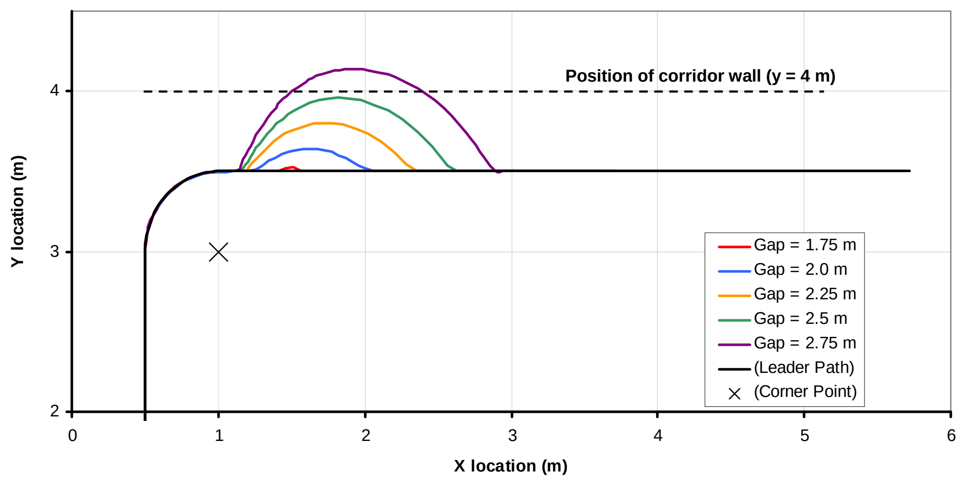


Figure 4.18: Path deviation and collision during tether snagging. Gap between leader and follower was adjusted from 1.75 m to 2.75 m. The position of the corridor wall (dashed line) is marked to show where a collision would occur.

fashion, where the worst case collision limit is found.

The relationship between following gap g_F and proximity to collision d_{col}, determined by simulation, is graphed in Figure 4.20, with values normalised according to corridor width, w. From this data, it is possible to extract a formula relating d_{col} to g_F (Equation (4.6)) and therefore find a limiting condition to avoid a collision in a narrow corridor (Equation (4.8)):

$$g_F = -0.664d_{col}^2 - 1.4d_{col} + 2.565 \tag{4.6}$$

Condition for no collision, all values normalised according to corridor width:

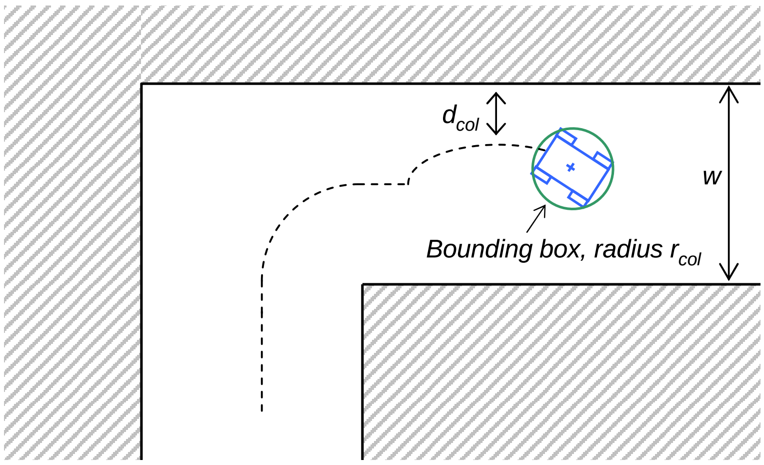


Figure 4.19: Collision parameters.

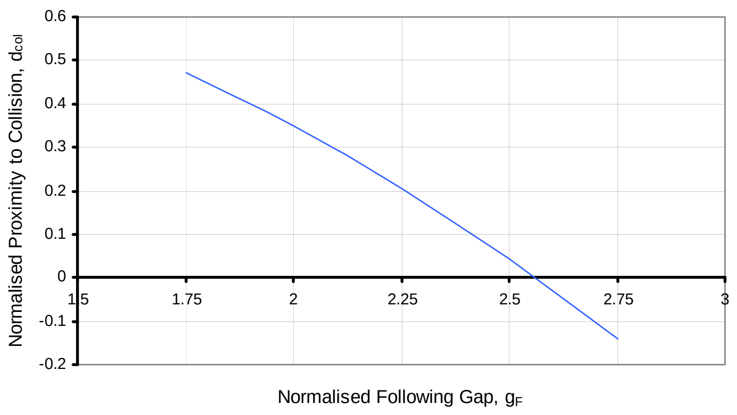


Figure 4.20: Relationship between following gap and proximity to collision. Values are normalised according to corridor width, w

$$d_{col} \geq r_{col} \tag{4.7}$$

$$g_F \leq -0.664 r_{col}^2 - 1.4 r_{col} + 2.565 \tag{4.8}$$

The limit to avoid tether snagging entirely in a corridor environment is given by Equation (4.9):

$$g_F < 1.75w \tag{4.9}$$

Since the problem of tether snagging is geometric, the limits calculated above should be scalable to a system of any size with the same geometry.

4.5.5 Snagging Detection and Compensation

While it may be possible to detect tether snagging situations by using external sensors similar to those used for obstacle avoidance (e.g. ultrasonic sensors, computer vision systems, LIDAR), it is interesting to investigate how much information we can measure from the tether alone. For this reason, the tether length and angle data was analysed to determine if the action of snagging caused some detectable anomaly. To detect the point at which the snagging occurs, the rate of change along the target path of three variables was recorded:

$$\text{Rate of tether angle change} = \frac{\Delta\theta}{\Delta s_L} \tag{4.10}$$

$$\text{Rate of tether length change} = \frac{\Delta L}{\Delta s_L} \tag{4.11}$$

$$\text{Rate of path angle change} = \frac{\Delta\beta_{path}}{\Delta s_L} \tag{4.12}$$

The change in path angle $\frac{\Delta\beta_{path}}{\Delta s_L}$ was found by comparing each point on the leader path to its neighbouring points, then calculating the angle between them. i.e. Very sharp turns will result in a high angle. It was predicted that the snagging may have caused a large, sharp change in the angle of the path.

The simulated tether data was analysed to determine if the methods outlined above could be used to detect tether snagging. Figure 4.21a shows the variation of tether length and angle against s_L with a following gap, g_F, of 5 m. For clarity, values for a 'virtual tether' have also been plotted - the virtual tether shows the measurement that would occur if the obstacles were not present. Although a change in gradient can be seen around the snag points (located at $s_L = 7.91$ m and 14.95 m), it is not clearly defined. Figure 4.21b shows the change in angle between the waypoints on the leader path. In this case there are clear maxima at the snag points, and we can use these to identify the incorrect tether data. The first peak corresponds to the point at which the tether first becomes snagged on the obstacle, the second point corresponds to the point at which the tether becomes 'unsnagged': where the tether stops contacting the obstacle.

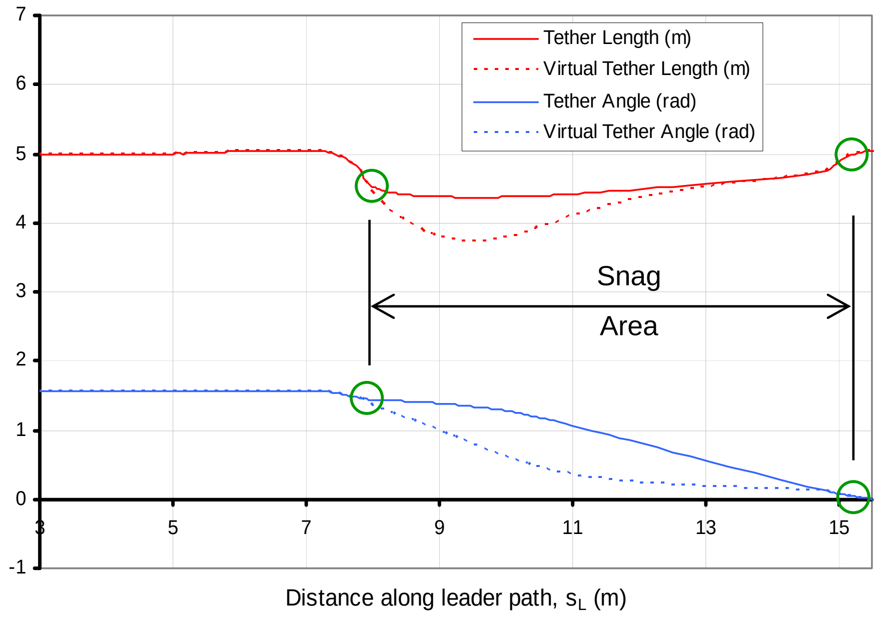
In order to develop a detection algorithm, it is desirable to find a parameter which shows a significant change at the snag points. The figures show that clear maxima of the path angle β_{path} can be observed, suggesting that it is possible to discriminate the snag points from the rest of the path data. Based on this data, it is possible to detect the tether snagging by examining the change in path angle, and setting a threshold value:

$$\beta_{limit} = 0.4 \text{ rad} = 0.4w \text{ (normalised)} \tag{4.13}$$

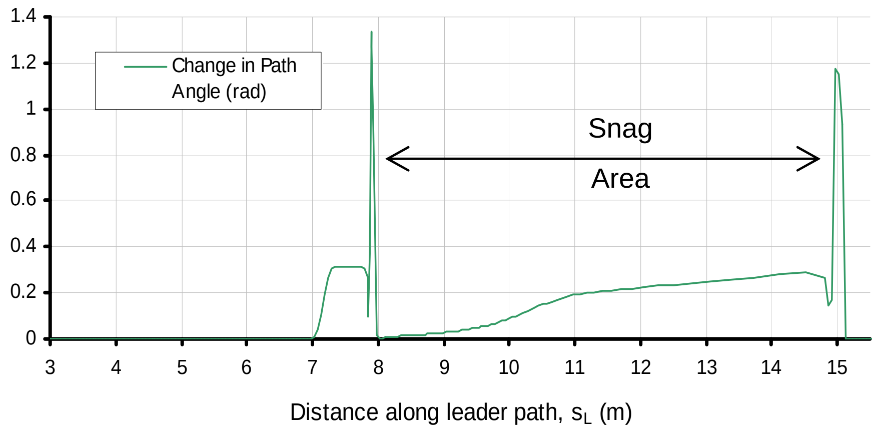
In practical terms, this limit is useful only when the leader vehicle has some restriction in its movement, i.e. when the minimum turning radius is limited. This means that tether snagging detection by change in path angle may only be used when some information about the leader is known; if the leader is not capable of making very sharp turns then it is possible to discriminate snag points. Unfortunately this method cannot be applied to a human leader, but in other applications where the leader is a buggy or car-type vehicle it may be applicable.

4.6 Chapter Summary

This chapter introduced several additional follower modes designed to improve follower performance in certain situations or address issues with previous control methods. Side following mode was developed to allow the robot to move beside the user and remain in their field of vision. Simulation results were presented for two types of side following control: *Side Joystick* mode and *Side Tracking* mode; the latter was shown to have improved performance. *Front joystick* mode was developed to allow the user to operate the robot in front of them, which may be more comfortable for some users and tasks. Though *Front joystick* mode was effective, practical experiments showed it was relatively difficult for the user to operate. *Brake* mode was introduced to improve the safety and usability of the robot, especially in busy environments, and it was successfully tested during several outdoor experiments with a Home Oxygen Therapy user. Finally, the problem of tether snagging was investigated and a rudimentary method of detection described.



(a) Variation of length and angle



(b) Variation of path angle

Figure 4.21: Detection of tether snagging for following gap of 5 m. Virtual tether represents the values for a tether which is not affected by obstacles. The snag points, and snag area have been indicated.

Chapter 5

Outdoor Experiments with Home Oxygen Therapy Patient

5.1 Introduction

With the aim of supporting Home Oxygen Therapy users in their daily life, it is vitally important to test the robot in conditions which closely approximate daily activities. The activity and mobility of H.O.T users varies for each individual, but in general they wish to enjoy the freedom to visit local shops, the park, the train station and other amenities, as well as exercising outside. It is often medically beneficial to perform some limited physical exercise, and a patient's doctor might recommend some daily/weekly exercise targets. This chapter describes an experiment to investigate the performance of the robot in a realistic outdoor situation: supporting the user on short trips around a local park.

When evaluating service and support robots it can be very difficult to get a quantitative measure of their usability or effectiveness. In this respect, Home Oxygen Therapy provides a very fortunate research opportunity since we can measure the user's heartrate and oxygen saturation as a quantitative estimate of a device's effectiveness. These values are affected by the user's physical exertion, and they in turn affect the user's health and safety (particularly oxygen saturation). Sensors were used to record the condition of a volunteer patient as they undertook some walking courses while the robot carried the oxygen tank. The performance of the robot would be compared to a manual oxygen tank cart (typically used by H.O.T. patients currently), and also to a commercial shopping cart with powered wheels.

To make statistically significant conclusions about the effectives of the robot, it is clear that we need to conduct experiments with a large number of participants (eight participants is a reasonable minimum for this kind of research; more is better). However, recruiting Home Oxygen Therapy patients is difficult, with many having restrictions on their physical activity as advised by their doctors, and setting up experiments requires considerable time and funding. The final goal of this research is to reach the most vulnerable users, testing the robot a with a wide and representative range of participants, but before this can be achieved it is necessary to demonstrate that the robot can perform safely. As such, this chapter describes a preliminary experiment in which the robot was tested with one Home Oxygen Therapy user, and an additional healthy participant. We cannot make statistically robust conclusions about the robot's performance from these experiments, but any implied trends or relationships are still useful in justifying and preparing for further research.

This experiment was approved by the ethical review board of the Institute of Biomaterial and Bioengineering at Tokyo Medical and Dental University (approval number: *2014-02*). Prior to the experiment, permission to use public roads was obtained from Suita police, and permission to use a municipal park was obtained from Suita City Office. The experiment and participants were covered by suitable accident and liability insurance. This experiment was financially supported by the Association for Technical Aids (ATA).

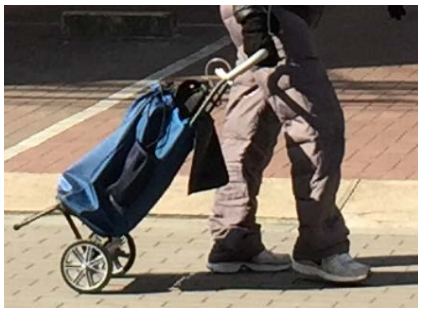
5.2 Assistive Devices Tested

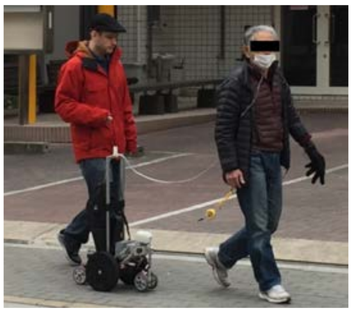
Three different devices were tested: a conventional oxygen tank cart, a commercial shopping cart with powered wheels, and the Oxydog robot (described in Chapter 3).

5.2.1 Conventional Oxygen Tank Cart

The conventional oxygen tank cart, shown in Figure 5.1a, is a widely available design often used by Home Oxygen Therapy Patients. The oxygen tank is stored in a purpose-built bag and secured to the tank's aluminium frame with velcro straps. The cart has two passive wheels and is steered manually by the user pulling the handle. Naturally, since the cart is unpowered, the user must support the weight of the cart and oxygen tank.

(a) Conventional oxygen tank cart (unpowered)


(b) Commercial powered cart


[redacted]

(c) Oxydog robot

Figure 5.1: Devices tested in outdoor experiment. In the case of the oxydog robot, a support researcher walks behind the robot holding an emergency stop switch in case of any safety problems.

5.2.2 Commercial Powered Cart

The cart shown in Figure 5.1b includes electric motors to power the wheels. The handle includes a spring sensor: when the user grasps the handle more firmly, the cart velocity is increased. A sensor detects when the cart is tilted forward greater than 15°, and turns on the drive motor. If the cart topples over or is dropped, the cart should turn off automatically.

The intended effect is for the motor to assist the user and therefore reduce the effort required to pull the cart. However, the user must still support the vertical component of the load (which increases as the cart is inclined more). It is not designed for medical use, rather the intended application is assisting elderly people when carrying shopping or other heavy loads. During the experiment the oxygen tank was placed at the bottom of the large bag.

5.2.3 OxyDog Follower Robot

The Oxydog robot was described in more detail in Chapter 3. For these experiments, the oxygen tank (inside its purpose-built bag) was placed inside the intended compartment and secured to the frame with velcro. The user held the end of the tether in their preferred hand, and the robot followed using the *Pseudo Joystick* algorithm (Section 2.3.1).

For safety, a researcher walked closely behind the robot at all times during the experiment, holding a cut-off switch which could quickly disable the robot in case of a problem.

5.3 Participants

The first participant in these experiments was a 22-year-old male volunteer, who was healthy and did not suffer from COPD or any form of lung disease (Figure 5.2a). The second participant in these experiments was a 70-year-old male volunteer; a regular Home Oxygen Therapy user (Figure 5.2b). The location of the experiment was the local area around the second participant's home, so he was familiar with the park area and the routes were similar to those he walks regularly. He was very keen to participate actively and contribute to the research.

Among Home Oxygen Therapy users, there is a wide variation in the severity of condition, physical fitness, quality of life and other factors. It is important to note that this participant was particularly healthy for a H.O.T. patient. He regularly exercised: he walked several kilometers each day, he cycled, and he visited the gym frequently. His relatively good health condition made him an ideal candidate for these outdoor experiments, as the risk to his health was limited (compared to more severe COPD patients).

(a) Subject 1 (22 years old)

(b) Subject 2 (70 years old)

Figure 5.2: Test subjects. Subject 1 is a healthy individual, subject 2 is a Home Oxygen Therapy user.

5.4 Measurements

5.4.1 Heartrate

Heartrate was measured using a wrist-mounted heartrate monitor, such as those typically used by athletes for sport training. Naturally, when the user exerts themselves physically, their heartrate will increase, and this allows us to consider heartrate as a proxy measurement of the physical effort spent on a task.

5.4.2 Oxygen Saturation (SpO2)

Peripheral capillary oxygen saturation, or SpO2, is a measure of the oxygen saturation in the blood, estimated from measurements taken at an extremity (e.g. a finger). Oxygen saturation is an important measurement in patients with COPD[47], and SpO2 can be monitored as a measure of the patient's condition during exercise. SpO2 is stated as a percentage, and typically values below 90% are undesirable for healthy people. During the experiment, SpO2 was recorded using a pulse oximeter attached to the user's finger. Additionally, a glove was worn over the user's hand to prevent cold weather affecting the measurement.

5.4.3 Position and Time

The experiment was recorded to digital video continuously, this provided a visual reference for the position along with timestamps to synchronize with the other instruments. A personal GPS unit similar to those used in sport was also used to record the position, as a backup.

5.5 Experiment Location

The location used for this experiment was Esaka Koen, a park in Suita City, Osaka, Japan. This area was chosen partly because it was close to the participant's residence, and because it gave a good representation of the daily trips H.O.T. users are likely to undertake (e.g. to local amenities such as the library). Figure 5.3 shows an aerial view of the park and the nearby amenities.

Within the park, two specific courses were identified for analysis.

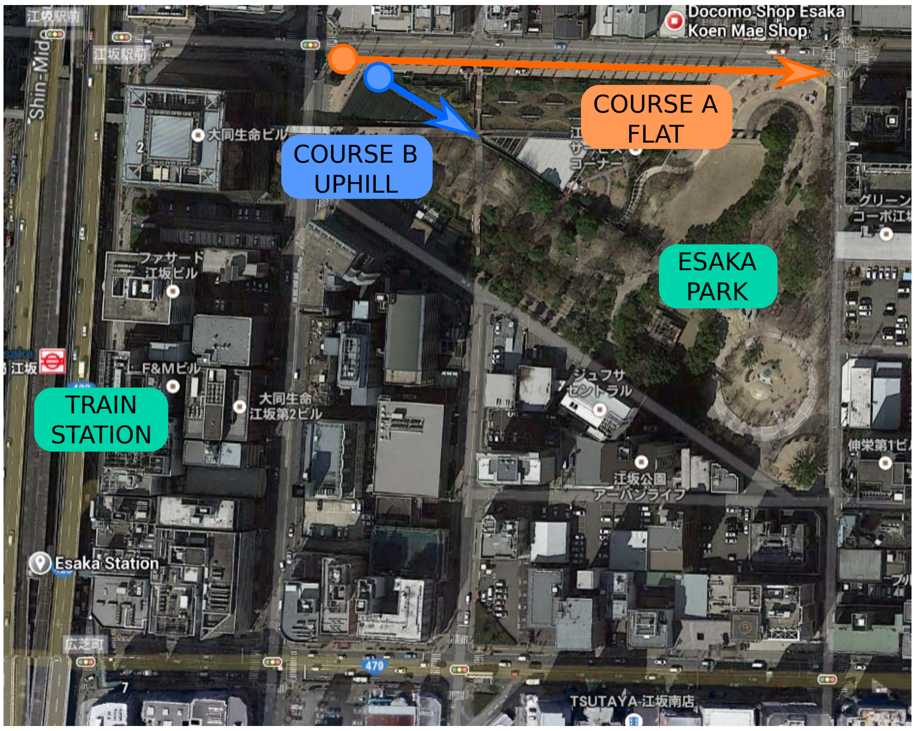


Figure 5.3: Map of the outdoor experiment location, Esaka Koen

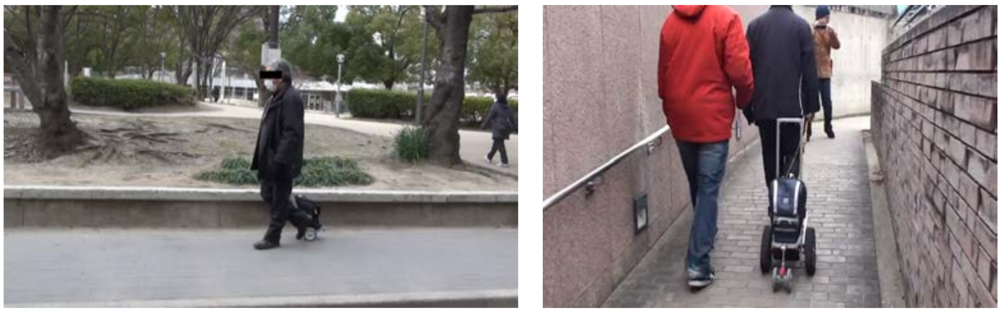
Course A Long, straight section along the north edge of the park. Very flat. 200 m long. Expected to be relatively easy to walk on. See Figure 5.4a.

Course B Short, uphill section near the library in the center of the park. 47 m long. Moderately steep. Expected to be require slight effort to walk up. See Figure 5.4b.

The slope of the uphill course was measured at eight points along the length, with the average slope found to be 7.97°(approximately 14% grade). Both courses have also been marked on the map in Figure 5.3.

5.6 Method

Prior to the experiment, the researchers met with each participant to clearly explain the task to them, and the participant was able to test using the devices so they were familiar



(a) Course A: flat ground, 200 m long

(b) Course B: uphill slope (14% grade), 47 m long

Figure 5.4: Courses for outdoor experiment

with their operation. The measurement instruments (pulse oximeter, heartrate meter, GPS) were attached to the participant. The two courses, A (flat) and B (uphill) were then traversed at a natural pace with the conventional oxygen cart. Following completion of the courses, the experiment was repeated with the commercial powered cart, and again with the robot follower. Each participant then repeated all of the above experiments in a second trial (carried out the same day, or the following day). Table 5.1 summarises all of the experiments.

During the experiment, the condition of the participant was monitored and several researchers were on hand to give assistance if required. In the case of the robot follower, one researcher also walked closely behind the robot holding an emergency stop switch in case of any safety problems. The participant was also instructed to monitor their SpO2 level and adjust their activity level to try to keep it above 90%.

After the experiment was completed, the data was downloaded from the measurement instruments for analysis.

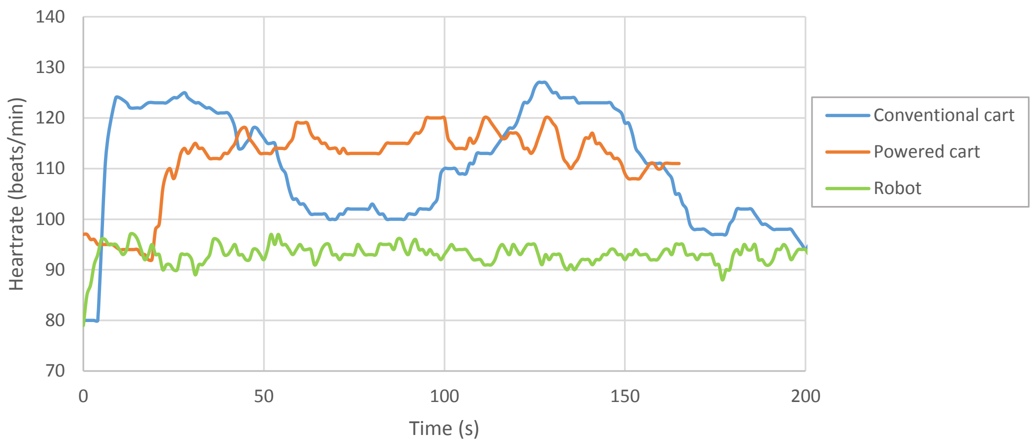
Table 5.1: List of outdoor experiments

Subject	Course	Device	Trial
1 (healthy)	A (flat)	Conventional cart	1
		Powered cart	1
		Robot	1
		Conventional cart	2
		Powered cart	2
		Robot	2
	B (uphill)	Conventional cart	1
		Powered cart	1
		Robot	1
		Conventional cart	2
		Powered cart	2
		Robot	2
2 (H.O.T. user)	A (flat)	Conventional cart	1
		Powered cart	1
		Robot	1
		Conventional cart	2
		Powered cart	2
		Robot	2
	B (uphill)	Conventional cart	1
		Powered cart	1
		Robot	1
		Conventional cart	2
		Powered cart	2
		Robot	2

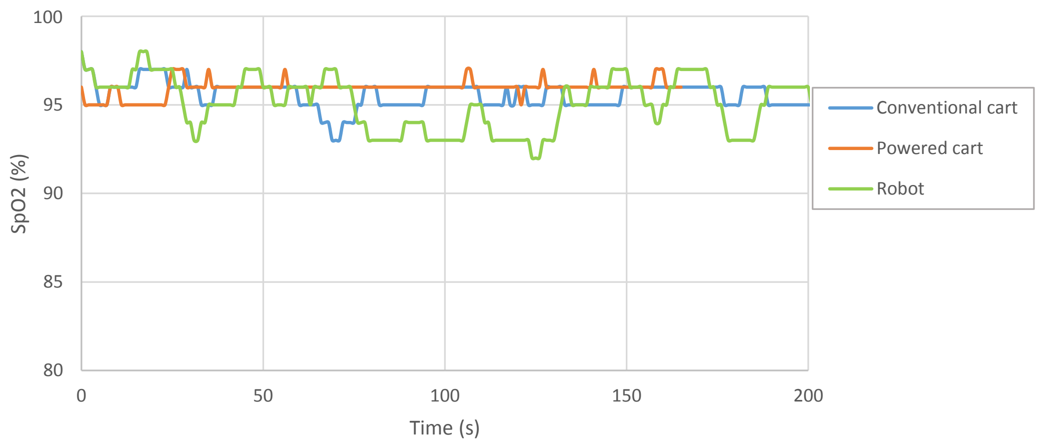
5.7 Results

5.7.1 Subject 1 (Healthy)

This section presents the heartrate and SpO2 profiles measured for subject 1. Figure 5.5 plots the results from the first trial on course A (flat) for heartrate and SpO2. The results from the second (repeated) trial on course A are plotted in Figure 5.6. The results for course B (uphill) are plotted in Figure 5.7 for the first trial, and Figure 5.8 for the second trial.

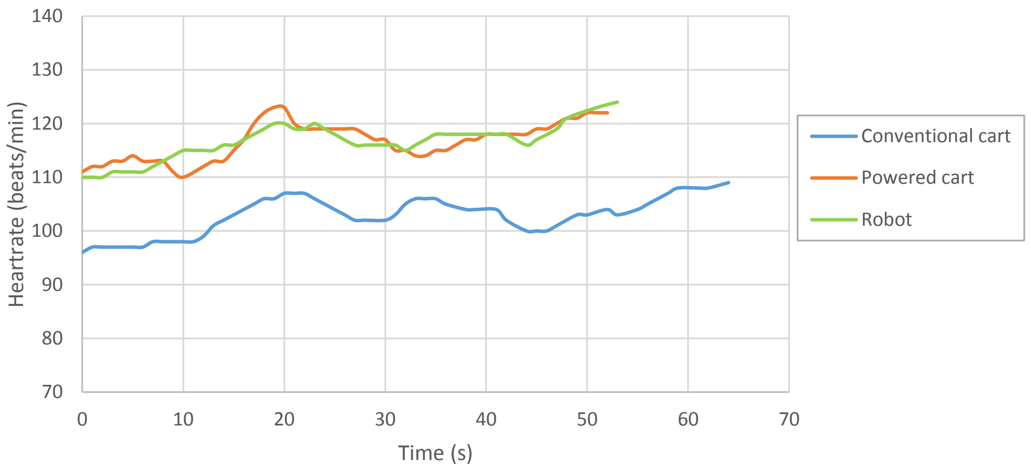


(a) Variation of heartrate during walking experiment

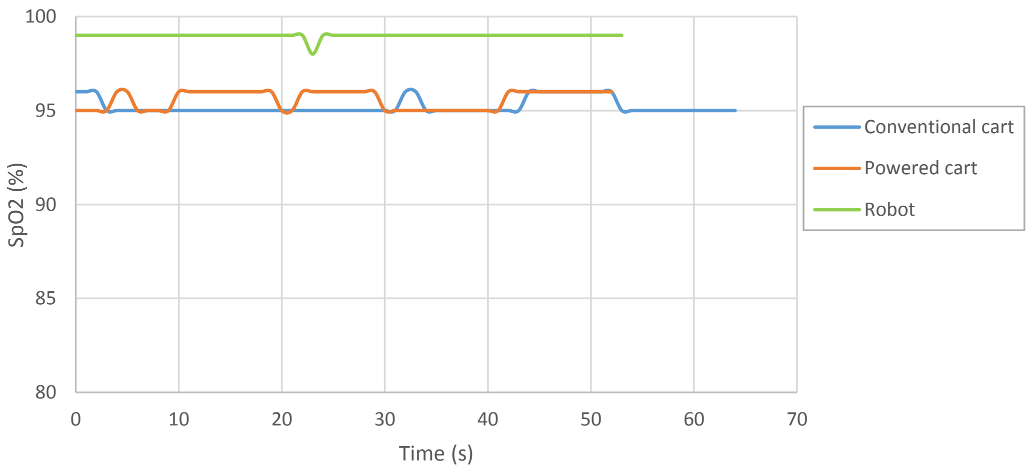


(b) Variation of oxygen saturation (SpO2) during walking experiment

Figure 5.5: Results for subject 1 (healthy) on course A (flat), first trial. Three devices were tested: conventional (unpowered) cart, cart with powered wheels, and follower robot.

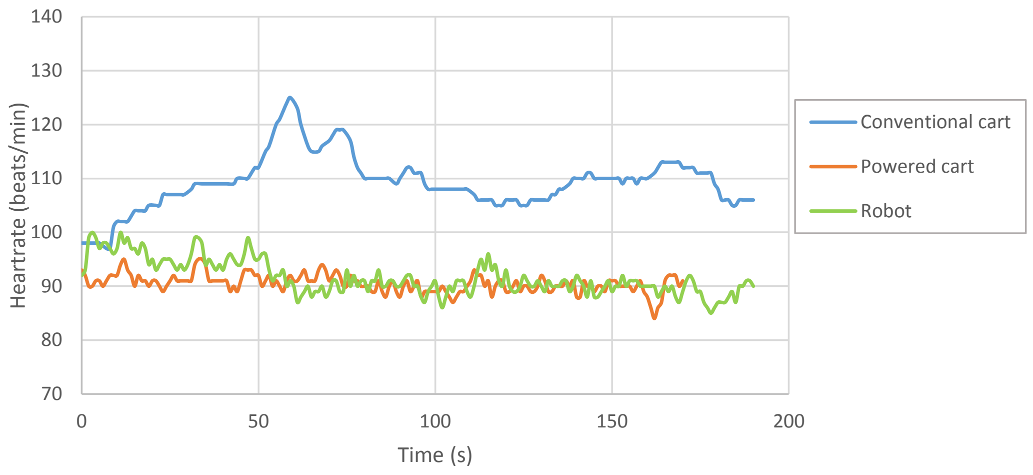


(a) Variation of heartrate during walking experiment

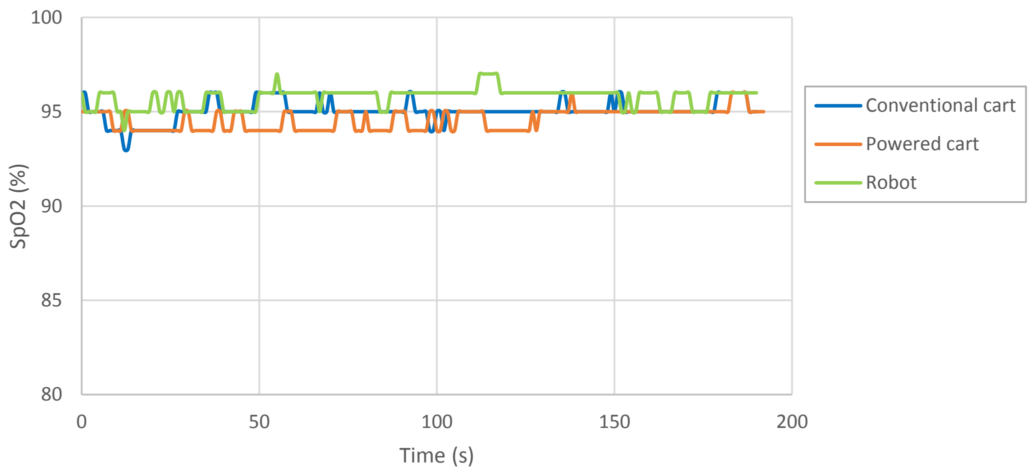


(b) Variation of oxygen saturation (SpO2) during walking experiment

Figure 5.6: Results for subject 1 (healthy) on course A (flat), second trial. Three devices were tested: conventional (unpowered) cart, cart with powered wheels, and follower robot.

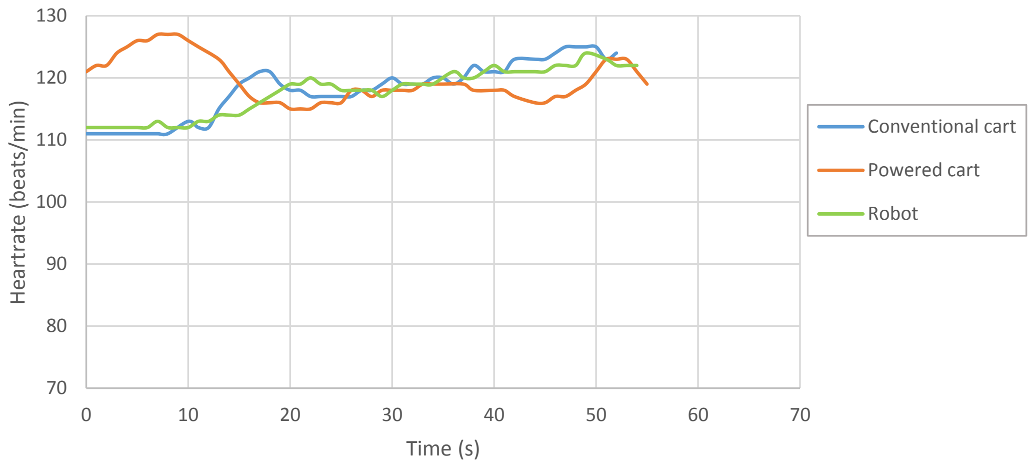


(a) Variation of heartrate during walking experiment

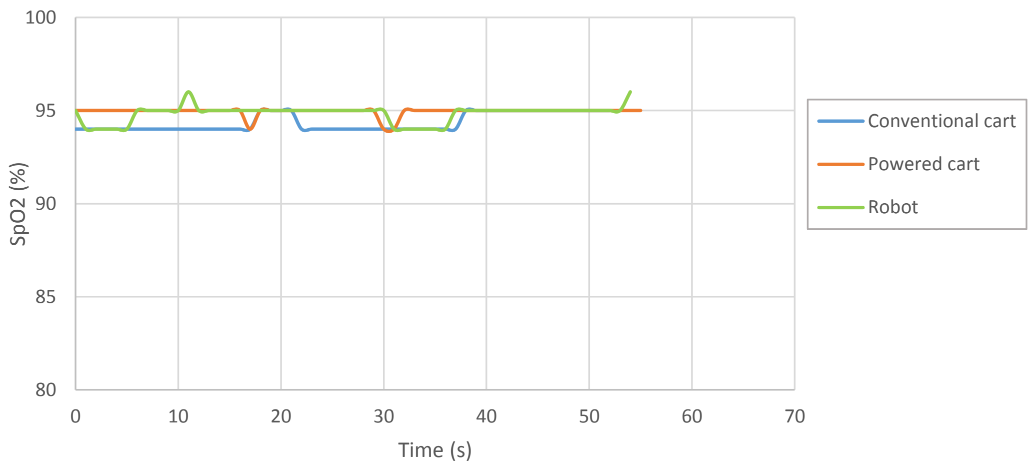


(b) Variation of oxygen saturation (SpO2) during walking experiment

Figure 5.7: Results for subject 1 (healthy) on course A (uphill), first trial. Three devices were tested: conventional (unpowered) cart, cart with powered wheels, and follower robot.



(a) Variation of heartrate during walking experiment

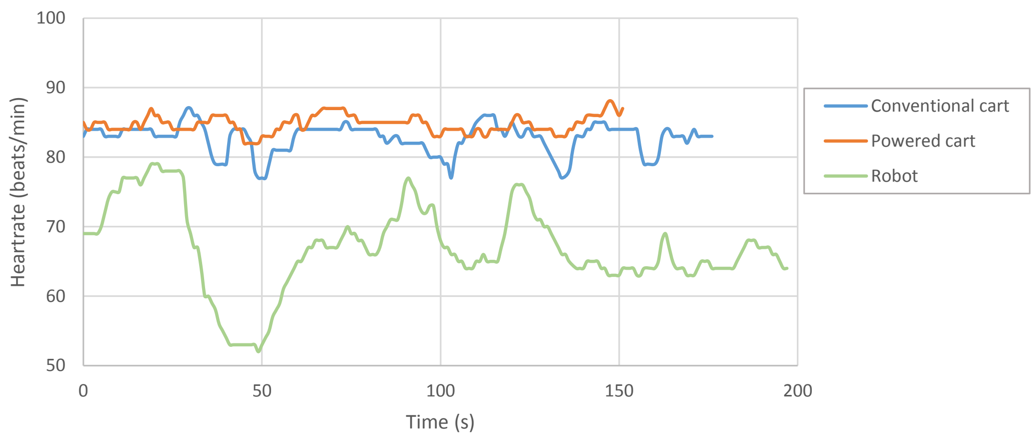


(b) Variation of oxygen saturation (SpO2) during walking experiment

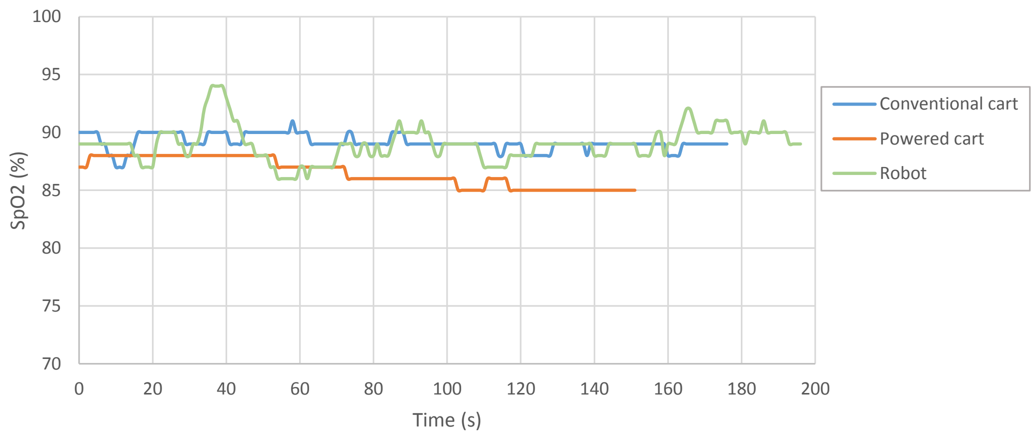
Figure 5.8: Results for subject 1 (healthy) on course A (uphill), second trial. Three devices were tested: conventional (unpowered) cart, cart with powered wheels, and follower robot.

5.7.2 Subject 2 (H.O.T. User)

This section presents the heartrate and SpO2 profiles measured for subject 2. Figure 5.9 plots the results from the first trial on course A (flat) for heartrate and SpO2. The results from the second (repeated) trial on course A are plotted in Figure 5.10. The results for course B (uphill) are plotted in Figure 5.11 for the first trial, and Figure 5.12 for the second trial.

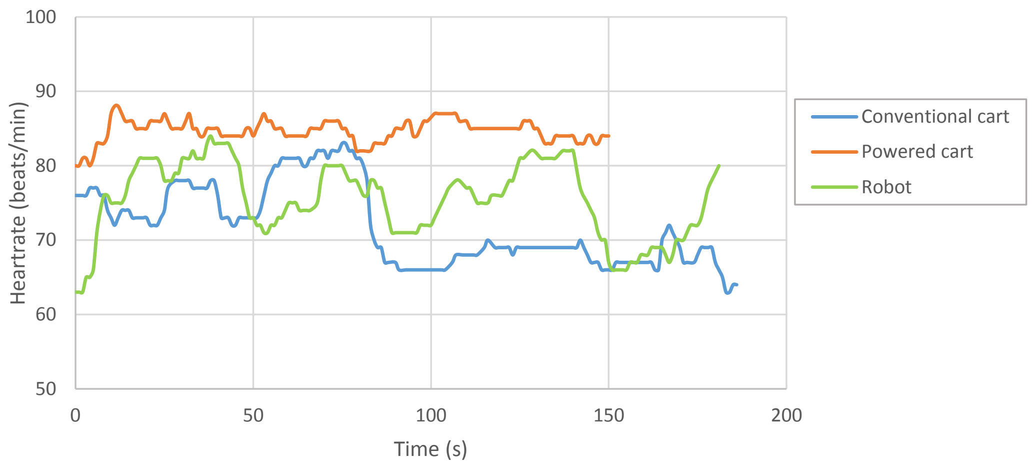


(a) Variation of heartrate during walking experiment

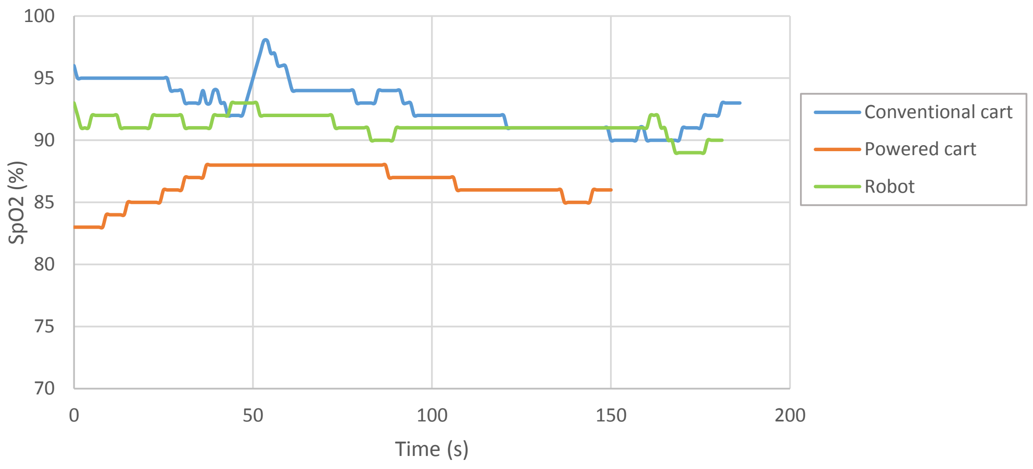


(b) Variation of oxygen saturation (SpO2) during walking experiment

Figure 5.9: Results for subject 2 (H.O.T. user) on course A (flat), first trial. Three devices were tested: conventional (unpowered) cart, cart with powered wheels, and follower robot.

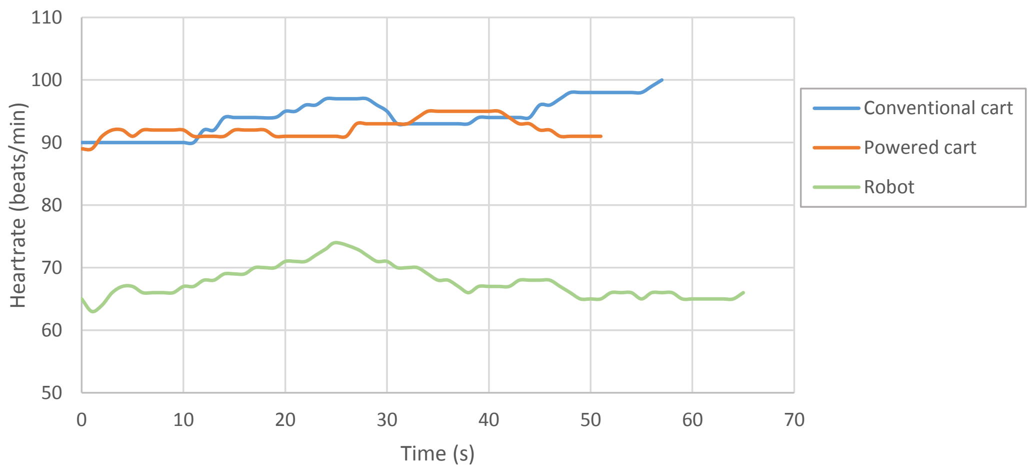


(a) Variation of heartrate during walking experiment

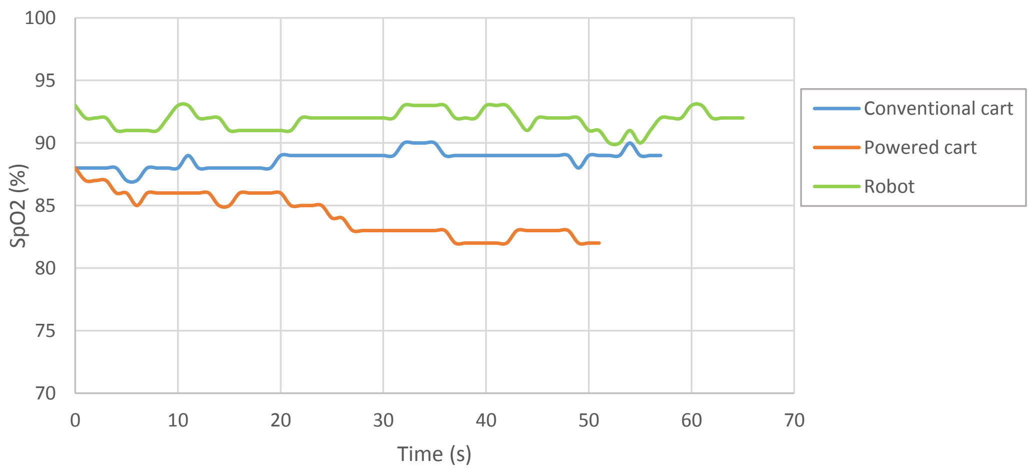


(b) Variation of oxygen saturation (SpO2) during walking experiment

Figure 5.10: Results for subject 2 (H.O.T. user) on course A (flat), second trial. Three devices were tested: conventional (unpowered) cart, cart with powered wheels, and follower robot.

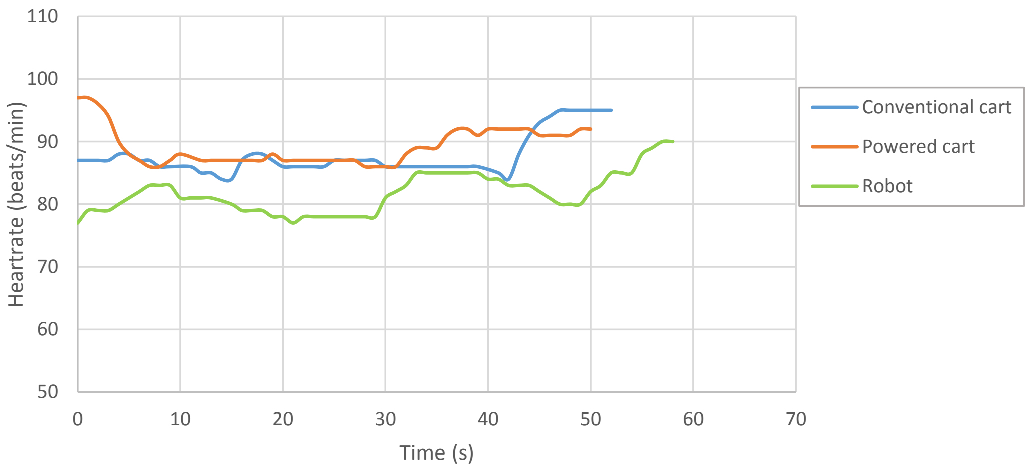


(a) Variation of heartrate during walking experiment

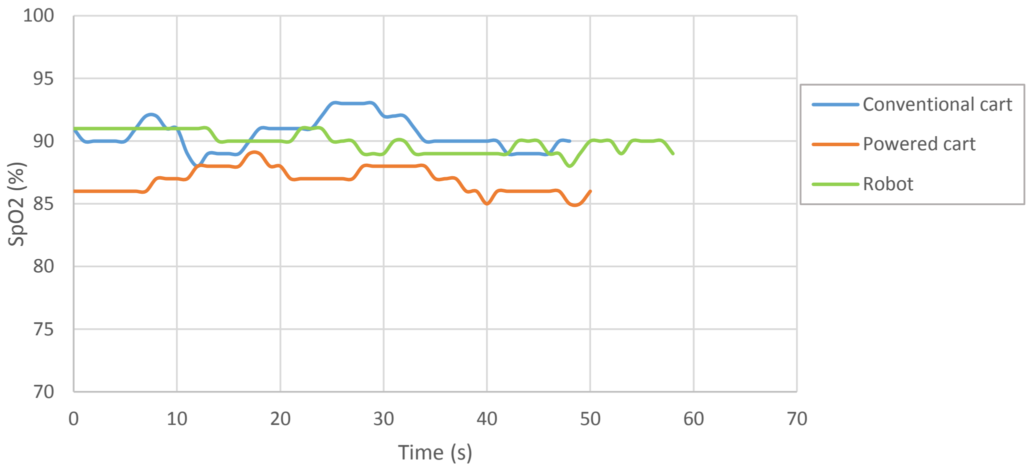


(b) Variation of oxygen saturation (SpO2) during walking experiment

Figure 5.11: Results for subject 2 (H.O.T. user) on course A (uphill), first trial. Three devices were tested: conventional (unpowered) cart, cart with powered wheels, and follower robot.



(a) Variation of heartrate during walking experiment



(b) Variation of oxygen saturation (SpO2) during walking experiment

Figure 5.12: Results for subject 2 (H.O.T. user) on course A (uphill), second trial. Three devices were tested: conventional (unpowered) cart, cart with powered wheels, and follower robot.

5.7.3 H.O.T. User's Impressions

It is also important to consider the feelings and opinions of the users, and for this reason the H.O.T. using participant (subject 2) was interviewed after the experiments. Considering the conventional cart, the participant felt that it was tiring as they always had to support the cart as they pulled it. When evaluating the commercial powered cart, they found it unattractive due to the need to constantly control the cart and pull it with one hand. Regarding the robot, they felt the following performance was better than they expected, and that walking with free hands is beneficial. However the noise of the robot was a problem: too loud for use in daily life. Lastly, they remarked that it would be better if the robot could move in front of them, where they could easily see its position.

5.8 Discussion

Sections 5.8.1 and 5.8.2 consider the results for the conventional cart, comparing the different users and terrain. Sections 5.8.3 and 5.8.4 discuss the differences between the conventional (unpowered) cart, the commercial powered cart and the support robot.

5.8.1 Subject 1 and Subject 2

The heartrate of the younger, healthier subject 1 was around 100 to 120 bpm and varied during the experiments. The oxygen concentration (SpO2) was generally above 95% and rarely decreased during the experiments; as is typical for a subject with a healthy cardiovascular system. The heartrate of the older H.O.T. user was lower, as is expected for an older person, generally in the range of 70 to 90 bpm. The oxygen concentration (SpO2) of subject 2 is lower and shows more variation than subject 1. During the experiments, the SpO2 level sometimes decreased when walking uphill (Figure 5.11b) or during sustained walking (Figure 5.10b). One of the goals of a support robot should be to reduce this variation in SpO2, allowing the user to maintain a safer, high level of oxygen saturation while enabling them to participate in more activities.

5.8.2 Course A (Flat) and Course B (Uphill)

For the healthy subject 1, it is difficult to see a clear difference between the uphill course and the flat course. In Figure 5.8, the heartrate is a little higher on the uphill course compared to the flat course (Figure 5.6), while in both cases the SpO2 remains flat around 95%. The results from subject 2 show a clearer difference. The H.O.T. user's heartrate is significantly higher on the uphill course (Figure 5.11a) than the flat course (Figure 5.9a), increasing as the subject climbs the slope. The SpO2 levels show a little variation during the uphill course (Figure 5.12b), but overall the trends are similar for both courses. These results give an indication of the relative difficulty of the different terrain and how this can affect the user's health. If the user wishes to strictly keep their SpO2 level above 90% as doctors recommend, then they are prohibited from walking up inclines such as this 14% grade, and their freedom is further limited.

5.8.3 Performance of Commercial Powered Cart

In the first trials with the H.O.T. user, the heartrate results are similar for both the conventional cart and the powered cart, while the SpO2 levels are slightly lower for the powered cart (Figures 5.9 and 5.11). This would suggest that the powered cart requires slightly more exertion (or is more uncomfortable) for the user, however the results from the second trial on flat ground disagree with this (Figure 5.10). It is difficult to explain this discrepancy, but it is possible that varying environmental factors such as temperature and weather affected the results, or the level of the user's activity differed between trials. The performance of the powered cart with subject 1 are mixed: the performance is similar to the conventional cart in some trials, worse in others. The fact that the powered cart had several trials where the heartrate was significantly higher than the conventional cart suggest it is not suitable for this application, and the user's interview comments (Section 5.7.3) further reinforce this. The performance might be explained by the cart's design: the centre of gravity, and the support load required from the user vary as the tilt angle is changed. Particularly if the powered cart is tilted over to large angle (such as might be the case when climbing a slope with the cart behind you), the user must exert more force to support it.

5.8.4 Performance of Support Robot

With the H.O.T. user, the robot appears to perform a little better than the conventional cart on the uphill course, and similarly on the flat course. Figures 5.11 and 5.12 show the user's heartrate was lower when operating the robot on the uphill course, implying less physical exertion, while the SpO2 levels were the same or higher. Heartrate and SpO2 results on the flat course varied somewhat (5.9 and 5.10), but overall the robot's performance was similar to the conventional cart.

With the healthy subject, the robot also compared favourably to the conventional cart on the uphill course (Figures 5.7 and 5.8), while results on the flat course were better in the first trial, worse in the second (Figures 5.5 and 5.6).

Crucially, in all the trials the oxygen saturation level when using the robot was similar to the conventional cart, sometimes higher. This implies that the robot can provide a level of user safety which is, at minimum, no worse than the current solution (the conventional cart). These results, combined with the user's comments in Section 5.7.3 are encouraging and show that more experiments should be conducted in future to further confirm the effectiveness of the robot. As mentioned in the introduction, any future trials should involve a larger number of participants (ideally, more than eight) to measure the significance more robustly. Additionally, future experiments should seek to improve the controls of the experiment. Factors such as temperature, weather, walking speed and fatigue may affect the data and should be controlled wherever possible. However, conducting such experiments outdoors in a real environment in a safe manner is difficult to arrange, and the user's comfort, consent and safety should always be considered before any experimental controls.

5.9 Chapter Summary

This chapter described leader following experiments conducted with a Home Oxygen Therapy patient in an outdoor environment. The participants included a Home Oxygen Therapy user and a young healthy volunteer; they were asked to walk various routes around their local area (mainly around a park and the nearby train station), while using an assistive device to carry their oxygen tank. Three different devices were compared: a conventional oxygen cart (unpowered); a commercially available cart with powered wheels; and a robot follower. The user's heart rate and oxygen saturation (SpO2) were measured, and these values were then used to compare the effect of each device on the user. While the number of participants in this experiment was too few to draw statistically robust conclusions about the robot's performance, the results of the experiment implied that the robot performed comparably to the conventional cart, and sometimes better. This represents an interesting preliminary result which can be used to justify further experiments with a larger number of Home Oxygen Therapy patients in future. The performance of the commercial powered cart was slightly worse than the conventional cart in several trials, implying that it may be unsuitable for this application.

Chapter 6

Control of Airport Carrier Robot

6.1 Introduction

The leader following control discussed in the previous chapters are intended to be general: they are not specific to a particular robot or environment. To demonstrate this generality, this chapter will introduce a different leader following application using the same tether-based control.

Currently, when airport staff move baggage around airports, they often push baggage carts around manually, which is tiring work due to the amount and weight of bags. Figure 6.1 shows two typical cart designs used in airports: the flat bottomed luggage cart in Figure 6.1a (also used by airport customers); and the high-sided cargo trolley in Figure 6.1b which is used to move goods (e.g. deliveries of cases of drinks to be sold in airport shops). While there are a number of powered vehicles for transporting baggage and goods inside airports, these may be difficult to operate, or they may be cumbersome.

The airport carrier is a mobile platform designed to transport baggage and goods around airports. The use of a tether-controlled platform can provide a safe, compact means of transporting baggage and goods around the airport, while being intuitive to operate.

(a) Airport luggage cart

(b) Airport cargo trolley

Figure 6.1: Typical carts used in airports.
Image source: alibaba.com

6.1.1 Project Background and Collaborators

Under the Japanese Government's *Strategic Innovation Program* (SIP), Tokyo Institute of Technology has launched a joint venture with Ota City with the aim of improving people's quality of work and life [48]. This involves a collaboration between the university and a number of local companies in Ota City, including local manufacturers, business enterprises[49], software companies[50] and *Japan Airport Terminal Co. Ltd.*[51] The goal is to foster collaboration and improve the design capabilities of local companies, developing innovative products and supporting them with marketing and sustainable business strategy.

The project relevant to this research aims to develop a cart robot for use in Haneda Airport, Ota City [52]. The roles of each collaborator are outlined in Table 6.1.

The work from this thesis was used to design the top level control algorithms, assist with porting the low-level software implementation, and carrying out prototype testing.

Table 6.1: Airport Carrier Project Roles

Role	Collaborators
Target user	Japan Airport Terminal Co., Ltd.
Hardware development	Yasuhisa KOKI Co., Ltd.
Part manufacturing	Ono-denki Co., Ltd.
	Mimasu Industries Co., Ltd.
Software development	Yamasyou Co., Ltd.
	Tokyo Institute of Technology
Project management	Tokyo Medical and Dental University
	Tokyo Institute of Technology

6.2 First Hardware Prototype

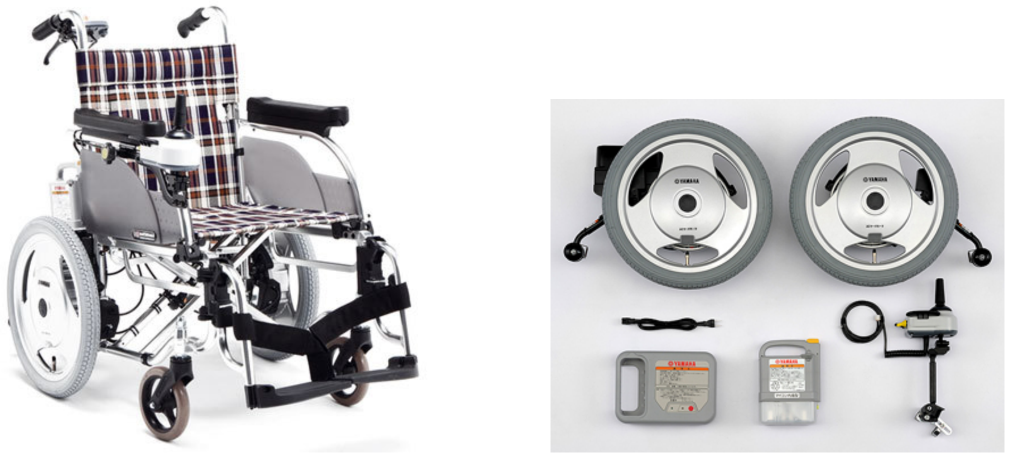
Prior to constructing a full-scale proof of concept for the mobile platform, a smaller prototype was constructed to test the wheel control and facilitate software development. The major components of the prototype are described in the following sections.

6.2.1 Powered Wheels

The prototype uses the electrically powered *Joy Unit X* wheel pack [53], manufactured by Yamaha Motors (Figure 6.2b). *Joy Unit X* wheels are commercially available as an after-market modification for wheelchairs (Figure 6.2a). The wheel diameter is 16 inches, the motor power is 120 W (per wheel), and the total weight of the pack (including battery) is 14 kg.

6.2.2 Mobile Platform

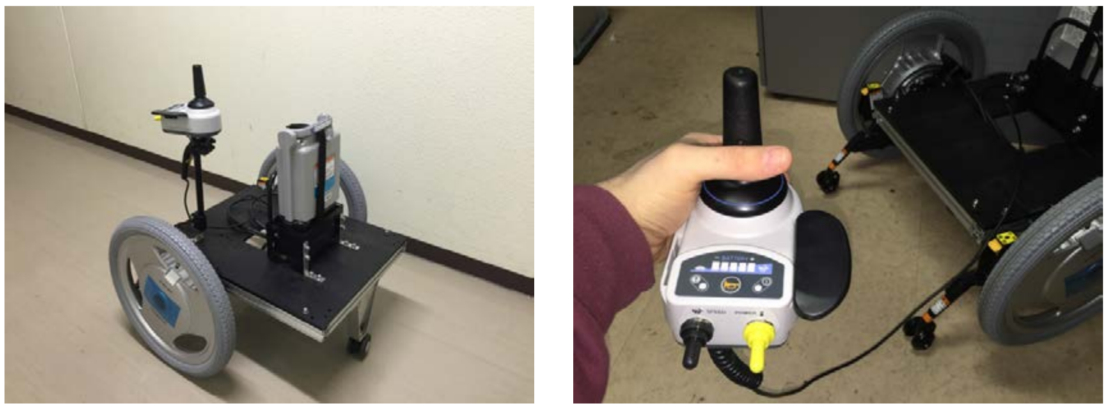
The platform consists of an acetyl resin baseplate with an extruded aluminium frame (Figure 6.3). The Yamaha wheels are mounted at either side on the rear, with a caster at the front. A tether winch (including potentiometers to measure tether angle and length) is mounted at the front of the platform. Figure 6.4 shows the tether winch in more detail,



(a) (b)

Figure 6.2: (a) Wheelchair fitted with powered wheels; (b) Yamaha Joy Unit X powered wheels.

along with the SH2 Tiny microcontroller board which runs the control software.



(a) Airport carrier prototype platform. **(b)** Hardware joystick for controlling airport carrier.

Figure 6.3: Airport carrier prototype.

6.3 Control

In their off-the-shelf state, the Yamaha powered wheels can be controlled using the hardware joystick pictured in Figure 6.3b. However, by means of an *Academic Pack* (also

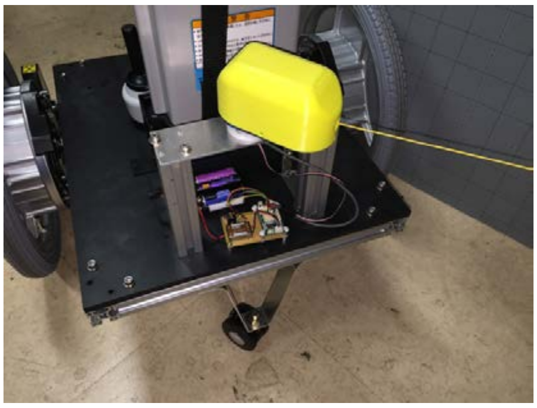


Figure 6.4: Tether winch mounted on airport carrier prototype. The winch contains potentiometers to measure the tether length and angle; these are connected to a TITech SH2 Tiny microcontroller which runs the control software.

available from Yamaha), it is also possible to control the wheel speeds through a serial interface. Figure 6.5 shows the typical operation of the academic pack: serial commands are sent from a PC, through a serial cable, to a port attached to the hardware joystick board. For this robot, the serial commands will be sent from an SH2 Tiny microprocessor instead of a PC.

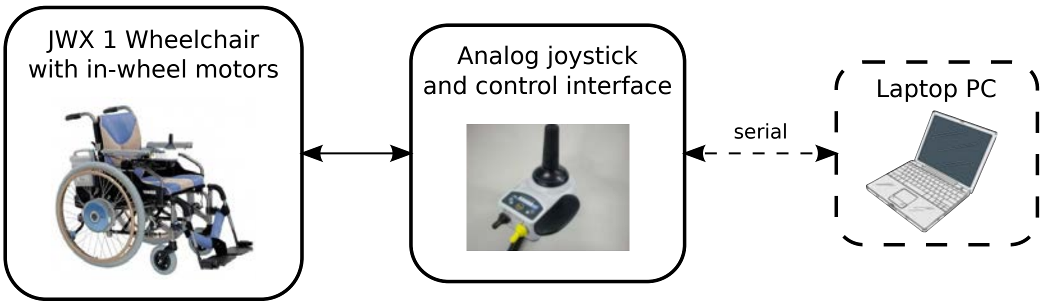

JWX 1 Wheelchair with in-wheel motors
Analog joystick and control interface
serial
Laptop PC

Figure 6.5: Use of *Academic Pack* to control Yamaha powered wheels by serial connection.

6.3.1 Calibration of Speed Commands

The Yamaha wheel pack has a specified protocol for serial communication. Unfortunately, this protocol does not allow us to specify individual wheel speeds. Instead, the desired output must be given in the following commands:

- Forward/Backward command (-100% to +100%)
- Left/Right command (-100% to +100%)

To implement control methods such as *Follow the Leader*, it is necessary to specify the wheel speeds directly. Therefore, the relationship between input command and wheel speed was investigated by varying the input commands across their range, and recording the output wheel speeds. The resulting relationship is plotted as a surface in Figure 6.6. For clarity, Figure 6.7 also plots the variation in wheel speeds with one input command fixed. Clearly, the relationship between the input commands and wheel speeds is mostly linear, but the maximum speeds are capped at around 800 to 1000 rpm (forward), and -400 to -500 (reverse). There is also a lower threshold of around 25%, below which the output speeds are zero regardless of input command.

6.4 First Prototype Testing

6.4.1 Outdoor Operation Test

To confirm that the robot could operate for extended periods and in a varied environment, the airport carrier was tested outdoors for over 30 minutes (continuous). The test environment included sharp corners, slopes, and various low obstacles (such as drain covers). Figure 6.8 shows the robot following a user around corners on a delivery access slope. The user was able to hold the tether walk comfortably without needing to look at the robot frequently.

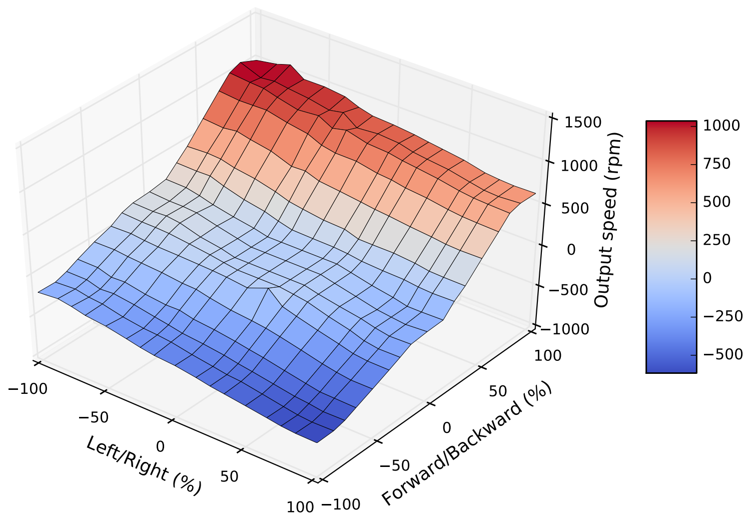
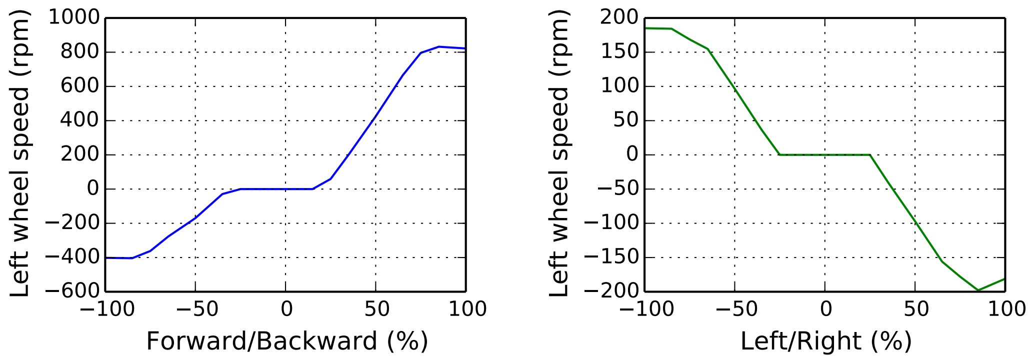


Figure 6.6: Left wheel speed output surface. Left/Right input command varied from -100% to +100%. Forward/Backward input command varied from -100% to +100%. (Right wheel output is similar, but with the Left/Right command reversed.)



(a) Left/Right command is 0% **(b)** Forward/Backward command is 0%

Figure 6.7: Left wheel speed output for fixed input commands. (Right wheel output is similar, but with the Left/Right command reversed.)

6.4.2 Slope Braking Test

Stopping on a slope can be a difficult task for the robot, particularly when moving downhill due to the additional gravitational force vector. To confirm that the robot could stop quickly enough (i.e. stop without bumping the user), the robot was operated on a steep downhill slope (Figure 6.9). The user pulled the tether and walked down the slope, stopping abruptly after several meters. Every time, the robot was able to successfully stop without bumping the user.

6.4.3 Operation Around Automatic Doors

Automatic doors are often considered a difficult obstacle for tethered robots, due to the risk of the tether being caught between the doors. The control of the airport carrier avoids this problem by trying to maintain a close distance to the user. The robot was tested moving through a double set of automatic doors as seen in Figure 6.10.

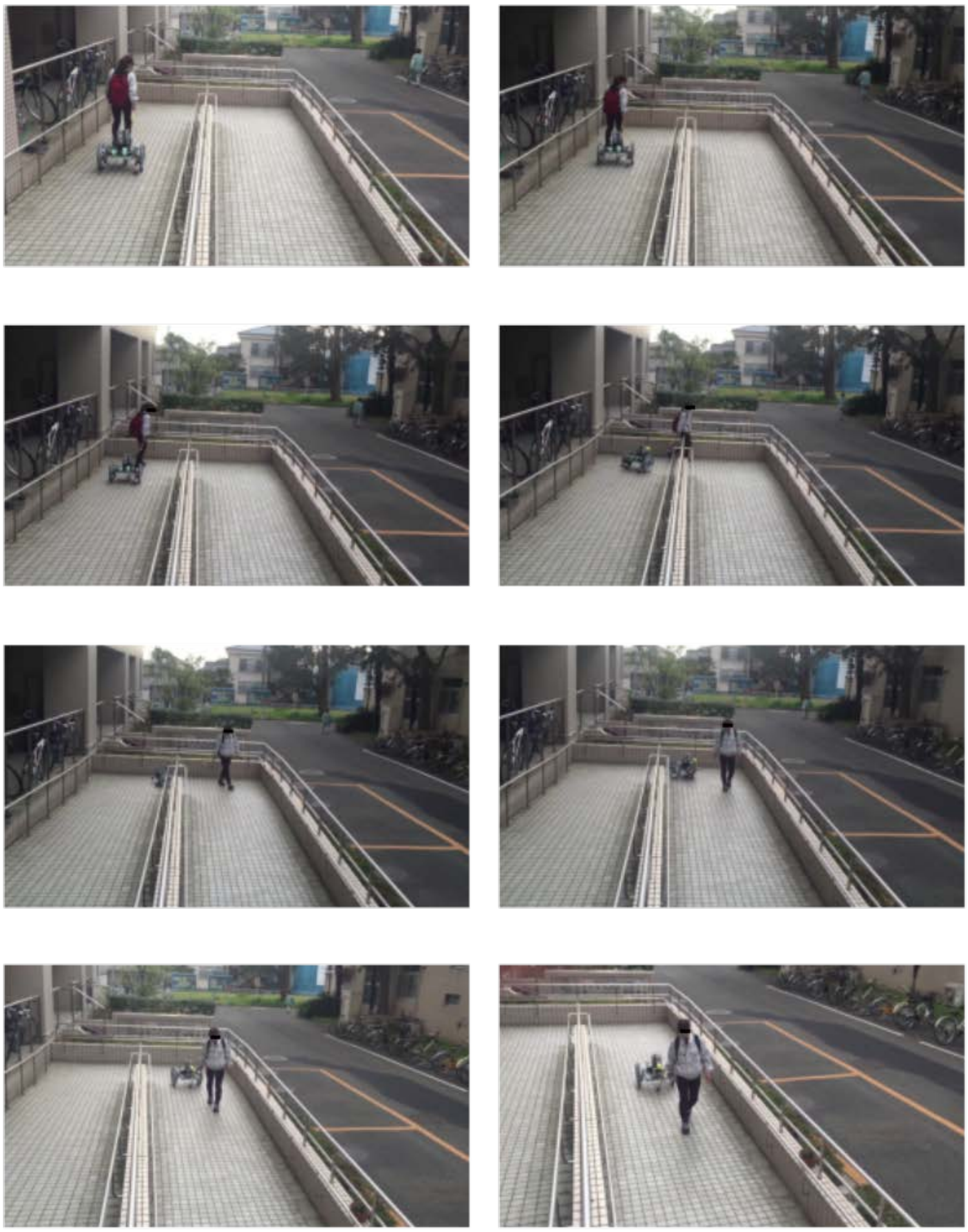


Figure 6.8: Testing airport carrier prototype outdoors with corners.

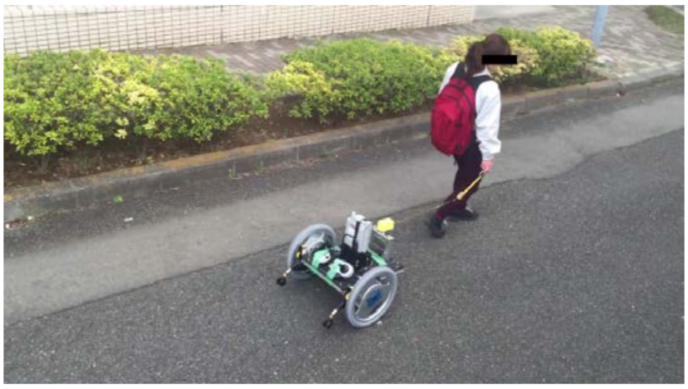

[redacted]

Figure 6.9: Testing airport carrier prototype on downhill slopes.

6.5 Final Prototype Testing

After confirming the control algorithms in the first prototype platform, the control software was implemented on the final hardware prototype. Figure 6.11 shows two identical final prototype airport carriers, one mounted to a cargo trolley, the other mounted to a baggage cart. Figure 6.12 shows the airport carrier being operated by a user pulling the tether.

As a final test, the airport carrier was operated by tether in a narrow environment (Figure 6.13). The robot successfully followed the user's position; thus the tether provided a suitable interface to control the cargo trolley.

6.6 Chapter Summary

This chapter introduced an airport carrier robot as a suitable application for the leader following control discussed in previous chapters. The design of a prototype platform was described, along with implementation of the control. The speed command relationship was investigated through calibration, and basic operation of the prototype platform was confirmed in an extended outdoor test with a varied environment.

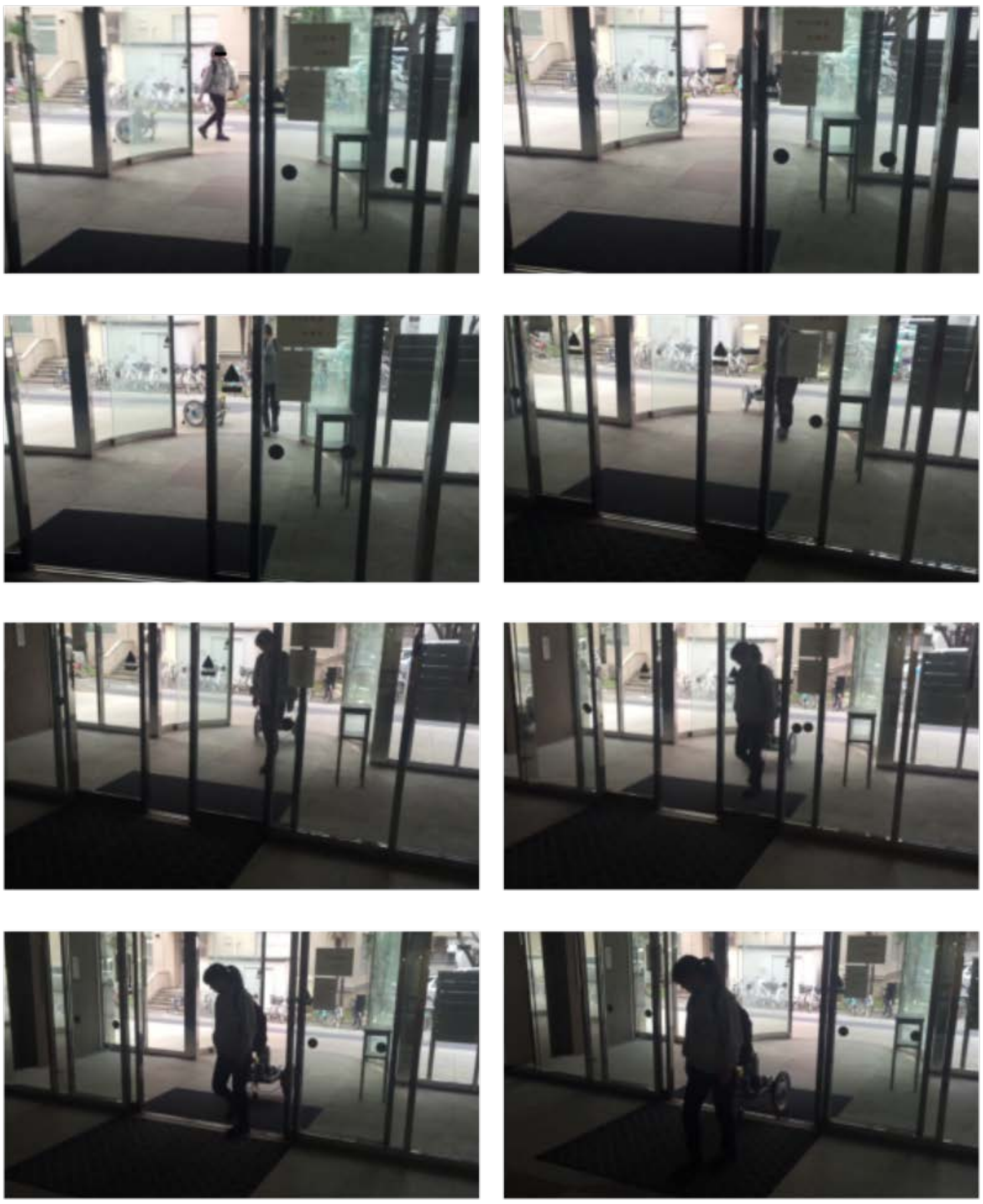


Figure 6.10: Testing airport carrier prototype with automatic doors.

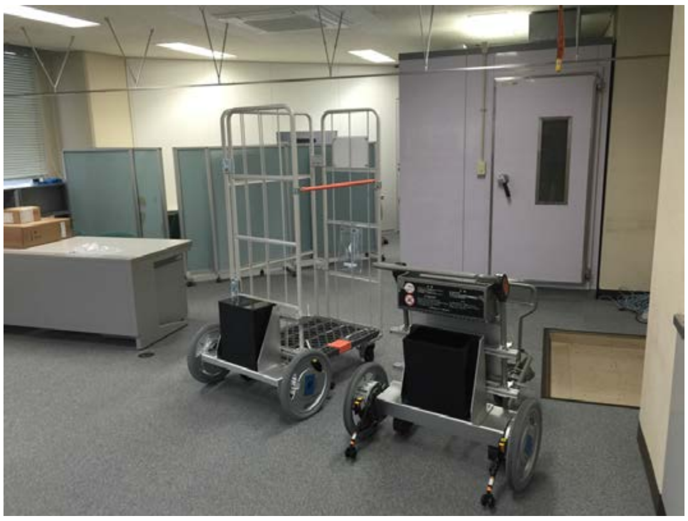


Figure 6.11: Two prototypes attached to baggage carts. Left: cargo trolley; right: baggage cart.

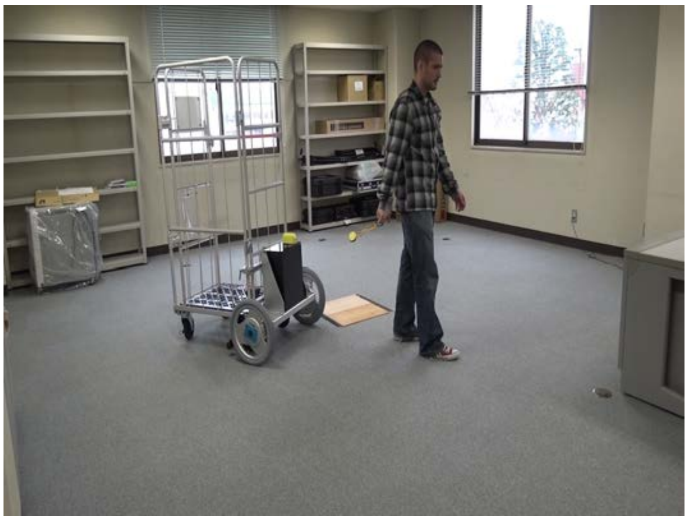


Figure 6.12: Airport carrier attached to cargo trolley

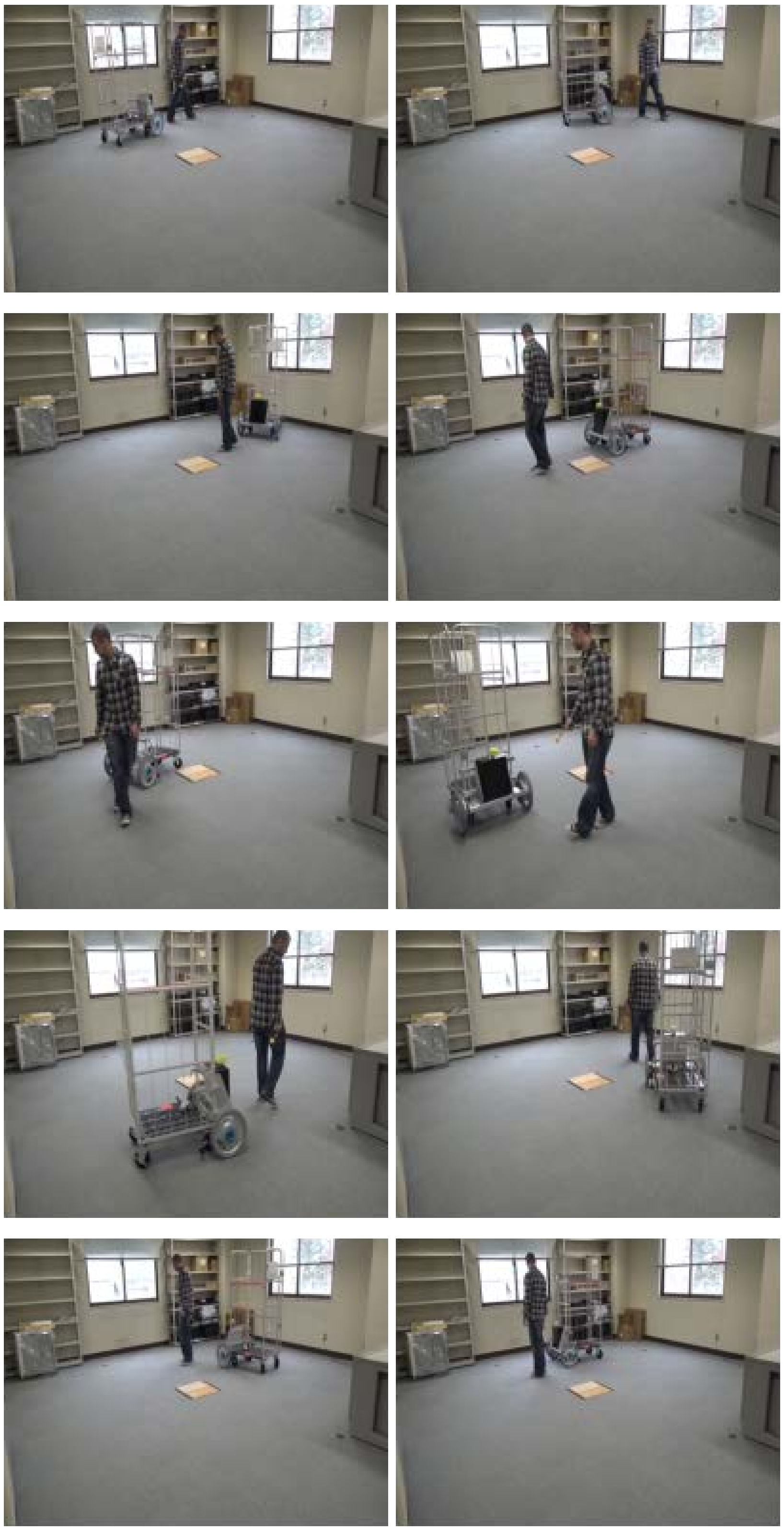


Figure 6.13: Testing airport carrier with cargo trolley. In a relatively narrow space, the cart could successfully follow the user. Steering the cart by tether was relatively simple compared to using the hardware joystick.

Chapter 7

Conclusions

This thesis aimed to investigate tether control in a mobile robot designed to support Home Oxygen Therapy patients. Initially, the nature of Home Oxygen Therapy was described, noting that the users' lives are negatively affected by the need to carry the H.O.T. equipment around. A survey of related work was conducted to evaluate the suitability of existing devices and academic research for the application of supporting H.O.T. patients. This survey showed that existing devices were unsuitable due to their high cost (especially where robots included expensive sensors such as laser range finders), or a lack of robustness, among other factors. Based on this, it was determined that a differentially steered mobile robot with a tether interface could meet the needs of Home Oxygen Therapy patients. The rest of this thesis investigated the suitability of this robot, with a focus on tether control and evaluation by real patients.

Chapter 2 presented three control methods for tethered leader following: *Pseudo-Joystick*, *Follow the Leader* and *Follow the Leader with Constant Distance*. *Normal path deviation* was introduced as a metric to compare the accuracy of leader following and numerical simulations were performed to characterize leader following behaviour for each algorithm. An investigation into the effects of control parameters found that *Pseudo-Joystick* control was negatively affected by longer tether lengths, while *Follow the Leader* was unaffected.

Following the numerical simulation, the leader following behaviour was studied using practical experiments with a hardware mobile robot prototype (Chapter 3). First, experiments were conducted with healthy users in a motion capture room. After confirming the behaviour in this controlled environment, the results of a leader following experiment with H.O.T. patients and a questionnaire survey to gather user feedback were analysed. From the practical experiments it was shown that the *Follow the Leader with Constant Distance* algorithm was capable of following the user more accurately than *Pseudo-Joystick*, but both algorithms gave reasonable following performance. The questionnaire survey of H.O.T. users identified that overall they found *Follow the Leader with Constant Distance* to be better and found *Pseudo-Joystick* control to be more uncomfortable. *Pseudo-Joystick* control is likely to be more intuitive to some users because of its immediate response to user commands, but the need to look back and check the robot's position can introduce some discomfort.

Based on user feedback gathered during earlier experiments, Chapter 4 introduced several additional follower modes designed to address some of the raised issues or improve follower performance in certain situations. Side following mode was developed to allow the robot to move beside the user and remain in their field of vision. Simulation results were presented for two types of side following control: *Side Joystick* mode and *Side Tracking* mode; the latter was shown to have improved performance. *Front joystick* mode was developed to allow the user to operate the robot in front of them, which may be more comfortable for some users and tasks. Though *Front joystick* mode was effective, practical experiments showed it was relatively difficult for the user to operate. *Brake* mode was introduced to improve the safety and usability of the robot, especially in busy environments, and it was successfully tested during several outdoor experiments with a Home Oxygen Therapy user. Additionally, the problem of tether snagging was investigated and a rudimentary method of detection described.

To further evaluate the robot's performance as an assistive device, experiments were conducted with a Home Oxygen Therapy patients in a realistic outdoor environment (Chapter 5). Two participants, a young healthy volunteer and an older H.O.T. user, were asked to walk various routes around a local park while using an assistive device to carry their oxygen tank. Three different devices were compared: a conventional oxygen cart (unpowered); a commercially available cart with powered wheels; and a robot follower. The user's heart rate and oxygen saturation (SpO2) were measured, and these values were then used to compare the effect of each device on the user. While the number of participants in this experiment was too few to draw statistically robust conclusions about the robot's performance, the results of the experiment implied that the robot performed comparably to the conventional cart, and sometimes better. This represents an interesting preliminary result which can be used to justify further experiments with a larger number of Home Oxygen Therapy patients in future. The performance of the commercial powered cart was slightly worse than the conventional cart in several trials, implying that it may be unsuitable for this application.

Chapter 6 introduced an airport carrier robot as a further application for the leader following control discussed in previous chapters, demonstrating the generality of the research. Tether control was demonstrated in two airport carrier prototypes, operating in varied environments including slopes, automatic doors, elevators, outdoor paths.

****

Overall, the research presented in this thesis has shown that a tether controlled mobile robot can be an effective device for Home Oxygen Therapy patients. The robustness of the tether interface and control was proven with several extended experiments, including in busy outdoor environments. User feedback has been gathered multiple times; this has been used to develop new control algorithms to better meet the users' needs. For example, *Side Following* and *Front Following* were developed to allow the user to see the robot's position; this was based on explicit requests from the users. This kind of feedback is vital for robot development, and this research has been very fortunate in providing opportunities to collaborate with users as enthusiastic as the Home Oxygen Therapy patients who participated. An important lesson learned over the course of this work was the tremendous variation between individuals in terms of their interaction with the robot, their walking pattern, and their comfort at operating a ‘machine’. For these reasons it is important to test with a variety of real users in a realistic environment as much as possible, since these variations cannot easily be modeled or simulated.

It is hoped that the experiments which measured the patient’s heartrate and oxygen saturation will provide an effective platform for similar experiments in future. Recruiting suitable participants for these trials is very difficult, due to the time, cost and safety requirements, and every individual patient will have a different level of health and allowable physical activity. However, gaining access to more and more users requires demonstrating the safety of the device (and experimental design) in stages. As such, the preliminary experiments in Chapter 5 should be greatly beneficial in future applications.

Despite the general effectiveness of the robot, there are still a number of flaws in the current design. On the mechanical side, the robot is too noisy (due to the gearing in the wheels) and every user complains that the noise level is too loud for everyday use. The requirement of keeping the cost low has meant there are no sensors to detect obstacles around the robot, and so when operating in busy environments there is a risk of bumping or in the worst case getting stuck. A similar problem may occur if the tether touches some obstacle (e.g. a corner of a corridor) causing it to deflect and thus introduce an error in the position tracking (as described in Section 4.5). i.e. the tether interface gives us a proxy measurement of the leader’s position, not a direct one. Up to now these problems have been mitigated very effectively by controlling the length of the tether to keep it relatively short, but a more complete, robust system should tackle the obstacle problem directly. Considering the current status of the robot control, *Side Following* and *Front Following* modes still present a risk of tangling/wrapping when used with real oxygen cannula. These algorithms may be locally robust against tangling, but it is necessary to have much stricter control of this aspect before trialling them with real patients.

This work can make an important societal impact in the future, particularly in Japan where the population is aging and the need for care and support is increasing. The final target of this research is to have a commercially available robot which can support Home Oxygen Therapy patients, but to achieve this, several more iterations of *test-feedback-improve* are necessary. The robot's hardware should be made lighter and more compact where possible, and the noise should be reduced. In terms of control, it will likely be necessary to provide a means to switch between different control modes to cope with different situations. For example, using rear following for most walking tasks, but switching to front following in crowded or difficult to navigate areas. Providing an intuitive interface for this switching which can be readily understood by patients is a challenge. More trials will need to be conducted in future, with a larger number of patients, and with stricter controls for factors which can affect the patient's activity and comfort (further collaboration with medical researchers will likely be helpful in this area). With continuing research and collaboration, it may be possible to realise a practical product within the next five years.

Appendices

Appendix A

Calculation of Normal Path Deviation

Normal path deviation was introduced in Section 2.5.1.3 as a metric for measuring follower performance, i.e. the similarity between two paths or trajectories. Since the metric has been widely used throughout the research presented in this thesis, for example to compare the performance of two different leader following algorithms, it is important to consider the algorithms used to calculate it, along with any limitations. This appendix explains the algorithms used to calculate the normal path deviation in moderate detail: flowcharts are included to show the main logic, but code listings and other implementation details are omitted.

A.1 Algorithm Overview

The approach used is to divide two paths into linear segments, and then measure the normal (perpendicular) distance from the each segment on path 1, to the corresponding segment on path 2. Time data is ignored in the analysis; the comparison only considers the two trajectories in terms of position coordinates. An outline of the main algorithm is shown in Figure A.1. This section gives an brief summary of the algorithm, with more detail for each step given in the following sections.

At first, the data from two paths is loaded. In the experiments in Chapter 3, this would typically represent a human leader trajectory (path 1) and a robot follower trajectotry (path 2), but in princpiple any two paths may be compared. The paths are then re-segmented so that all of their linear segments conform to specified minimum and maximum length criteria (Section A.2). It is then necessary to find all the points where the paths intersect, and insert new nodes at these points (Section A.3). Once the paths have been re-segmented, and the intersection points determined, the next step is to find a valid comparison range (Section A.4). Finally, the normal path deviation can be calculated (A.5), and the resulting normal deviation profile can be analysed to find the mean, median, maximum (or any other desired statistics).

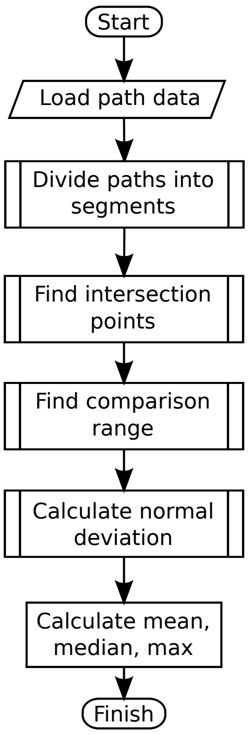

Start
Load path data
Divide paths into segments
Find intersection points
Find comparison range
Calculate normal deviation
Calculate mean, median, max
Finish

Figure A.1: Normal path deviation algorithm.

A.2 Path Segmentation

In this work, a point on the path with known position is referred to as a *node*, and the straight line connecting two nodes is referred to as a *segment* or occasionally *seg*. Nodes and segments are indexed separately, with numbers starting at 0 for the beginning of the path. A simple diagram of nodes and segments, along with their numbering, is shown in

Figure A.2.

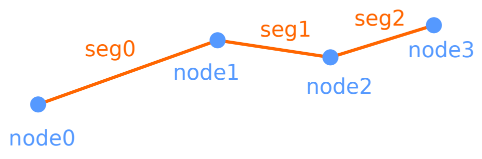


Figure A.2: Path segments and points.

Before the two paths can be compared, it is necessary to ensure the path segments are a reasonable size for comparison. So we can set minimum and maximum length criteria, then process the paths accordingly.

Figure A.3 shows the algorithm used to check for minimum segment length. Starting at the first segment, it loops over each segment in turn. Each segment is checked against the minimum length criteria, and if the segment is too small, it is merged with the next segment. This same check occurs for all segments on the path, with a slight difference for the last segment, which is merged backwards with the previous segment if necessary.

As a further advantage, the minimum segment length check will also remove any duplicate nodes on the path. Duplicate nodes are common when analysing experiment data where position has been tracked over time and the robot may be stationary for some period.

The path is then processed once more to remove segments which are larger than the maximum segment length, as shown in Figure A.4. In this case, segments which are too long are identified, then sub-divided into smaller segments which meet the maximum length criterion. Existing nodes are also shifted up to accommodate the newly inserted nodes and segments.

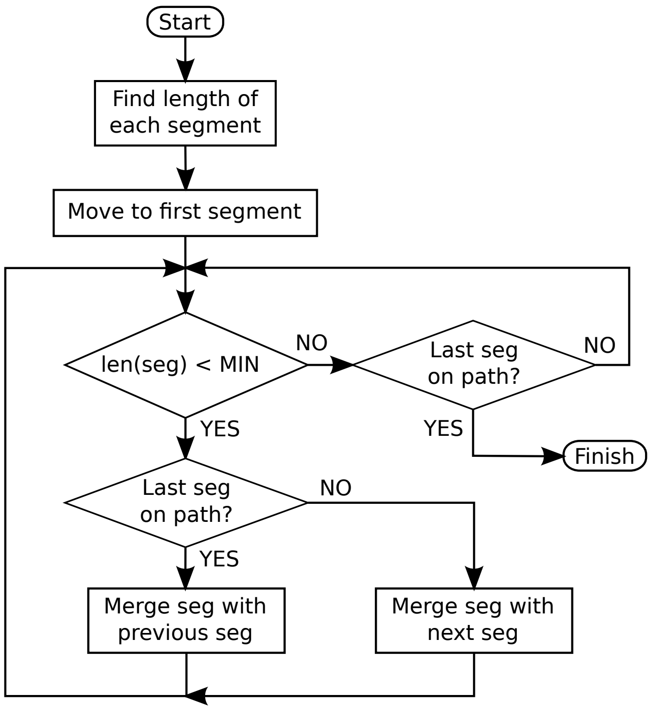


Figure A.3: Path segmentation algorithm for minimum segment length criterion.

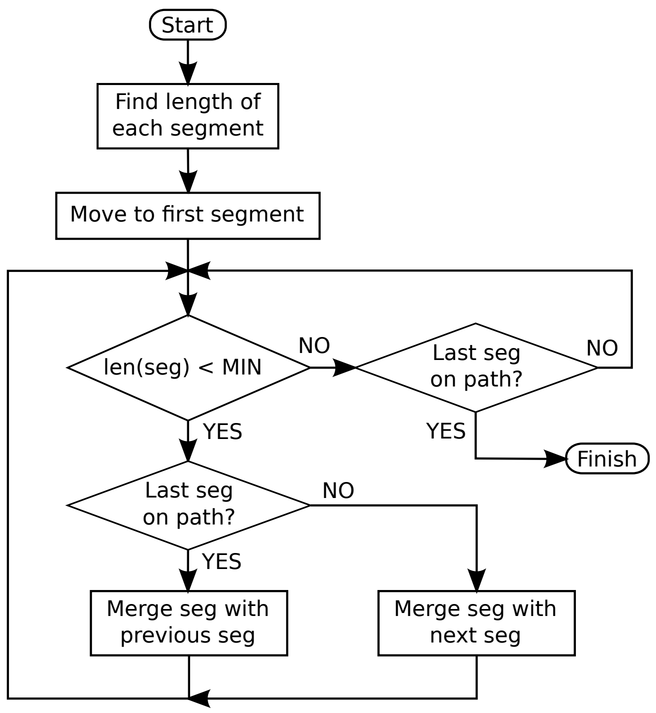


Figure A.4: Path segmentation algorithm for maximum segment length criterion.

A.3 Path Intersection

Finding the intersection points for two aribtrary paths is unfortunately rather computationally expensive. It is more or less necessary to test every pair of segments between path1 and path2, although it is also possible to use a simple bounding box check to speed up the intersection test. The intersection algorithm, shown in Figure A.5, loops over all pairs of segments, recording intersection points and inserting new nodes, as processes the paths.

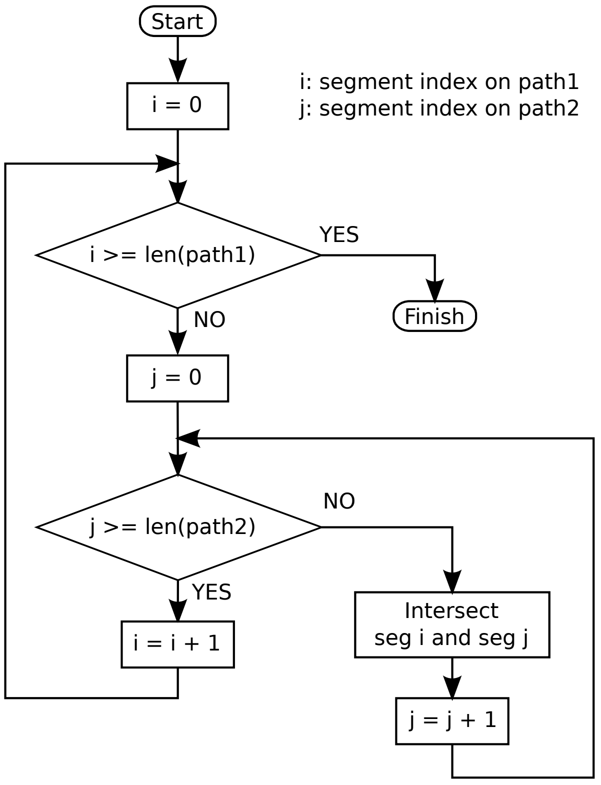


Figure A.5: Path intersection algorithm. The algorithm loops over all segements in both paths, testing each pair of segments for intersection.

The intersection test used for each pair of segments is based on two line vectors as shown in Figure A.6. m and n represent the proportional distance along vectors $\vec{AB}$ and $\vec{CD}$ respectively. From Figure A.6, we get the following system of equations (A.1),

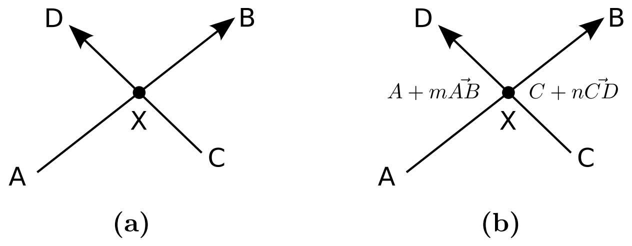


Figure A.6: Line segment intersection. (a) Two vectors, AB and CD, intersect at point X. (b) There are two equations for X: one for each line.

including four unknowns: X_x, X_y, m, n.

$$\begin{aligned}(B_x - A_x) \times m &= X_x - A_x \\ (D_x - C_x) \times n &= X_x - C_x \\ (B_y - A_y) \times m &= X_y - A_y \\ (D_y - C_y) \times n &= X_y - C_y\end{aligned} \tag{A.1}$$

We can then represent these in matrix form (A.2), and solve for the unknowns to the location of the intersection point.

$$\begin{bmatrix} B_x - A_x & 0 & -1 & 0 \\ 0 & D_x - C_x & -1 & 0 \\ B_y - A_y & 0 & 0 & -1 \\ 0 & D_y - C_y & 0 & -1 \end{bmatrix} \begin{bmatrix} m \\ n \\ X_x \\ X_y \end{bmatrix} = \begin{bmatrix} -A_x \\ -C_x \\ -A_y \\ -C_y \end{bmatrix} \tag{A.2}$$

The resulting point represents the intersection of the two lines as if they were infinite; since we have finite line segments we should, as a final step, check that the intersection occurs inside the bounds of the segments. This is achieved by checking the values of m

and n are between 0 and 1.

$$\begin{aligned} 0 \leq m \leq 1 \\ 0 \leq n \leq 1 \end{aligned} \tag{A.3}$$

A.4 Determining Comparison Range

Normal path deviation is only defined over a valid comparison range: the region where both paths overlap (Figure A.7). The algorithm for determining the comparison range is shown in Figure A.8. Beginning at the first segment of path 1, the normal deviation process (described in Section A.5) is used to find the nearest corresponding segment on path 2. This becomes the start of the comparison range. Next, beginning from the last segment of path 2, the same procedure is used to find the nearest corresponding segment on path 1, which becomes the end of the range. It is then possible to compare the two paths effectively using the normal path deviation algorithm.

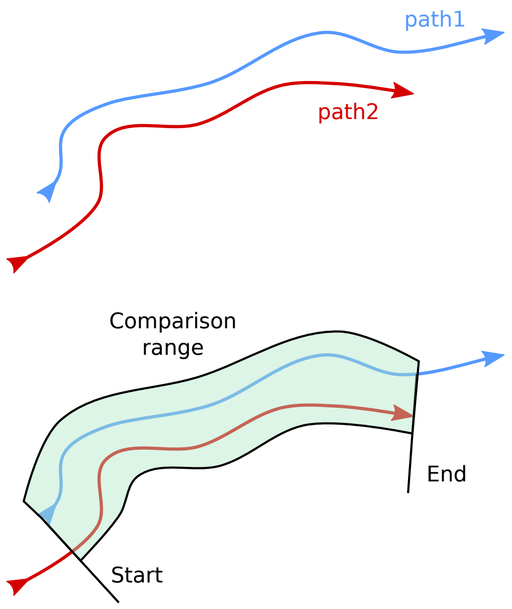


Figure A.7: Path comparison range. The valid comaparison range between two paths is the region where they overlap.

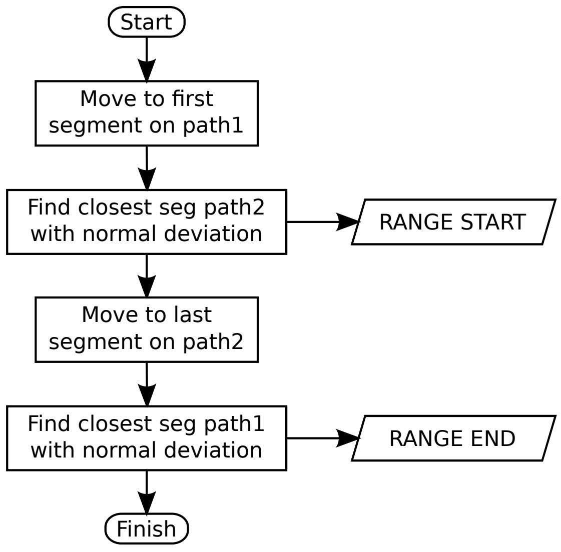

Start
Move to first segment on path1
Find closest seg path2 with normal deviation
RANGE START
Move to last segment on path2
Find closest seg path1 with normal deviation
RANGE END
Finish

Figure A.8: Path comparison range algorithm.

A.5 Normal Deviation

The final algorithm stage is shown in Figure A.9.

The midpoint of each segment on path 1 is calculated, and a normal (perpendicular) vector is extended from these midpoints. Next, the normal vectors are intersected with path 2. The distance from the segment midpoint on path 1 to the intersection on path 2 is the normal deviation for this segment. Plotting the deviation for every segment on path 1 shows the normal path deviation profile. An example trajectory with the normal vectors highlighted is plotted in Figure A.10a, and the resulting deviation profile is plotted in Figure A.10b.

A.6 Limitations

The calculation of normal path deviation works well for most cases, and was successfully calculated for the experiment and simulation data presented in this thesis. However, there

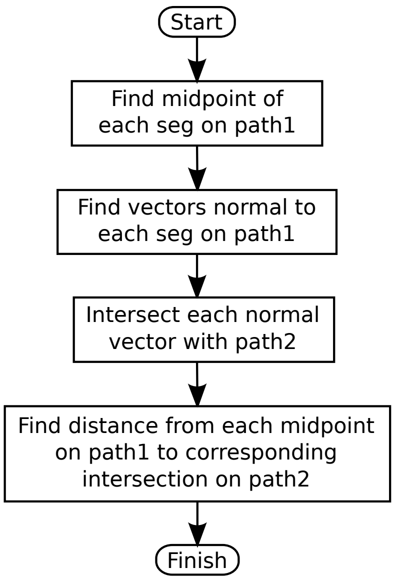
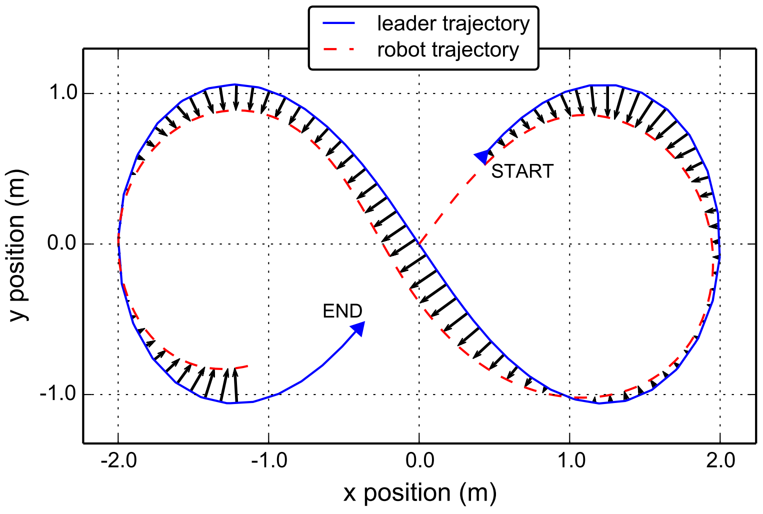

Start
Find midpoint of each seg on path1
Find vectors normal to each seg on path1
Intersect each normal vector with path2
Find distance from each midpoint on path1 to corresponding intersection on path2
Finish

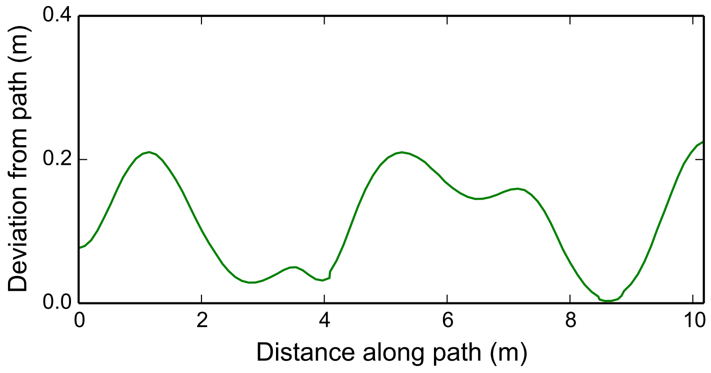
Figure A.9: Normal path deviation algorithm.

are cases where the algorithm does not perform well, and care must be taken to identify and mitigate such cases. Most importantly, the algorithm assumes that the two paths will be relatively similar; normal path deviation will be meaningless if calculated for paths which are very dissimilar. For robot leader following experiments, this is rarely a problem.

Additionally, problems can arise if the path segment size is too large. Figure A.11 shows a reasonably easy test case, where the segments in path 1 are reasonably small and the normal vectors can easily intersect with a sensible region of path 2. This can be contrasted with Figure A.12, where the segments are too large and the normal vectors occasionally do not intersect path 2 in a sensible region. This can be mitigated with a visible inspection of the deviation profile (by plotting), and adjustment of segment size if any problems are found.



(a) Example trajectory showing normal path deviation. Normal vectors are shown as black arrows.



(b) Normal path deviation profile.

Figure A.10: Example plots for normal path deviation.

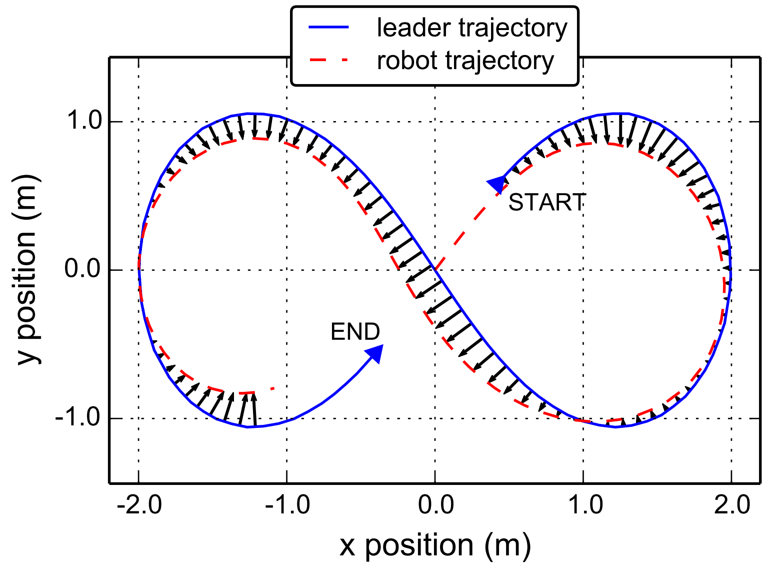


Figure A.11: Normal path deviation for a relatively easy test case. All normal vectors can be reasonably intersecting with the robot path.

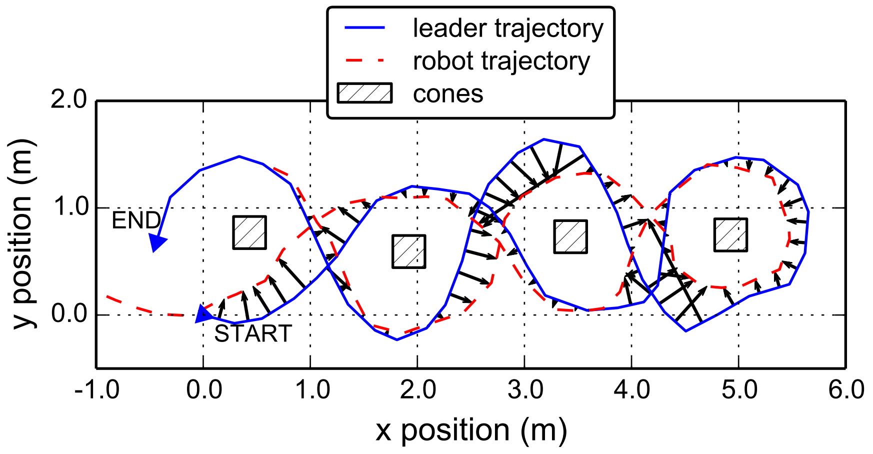


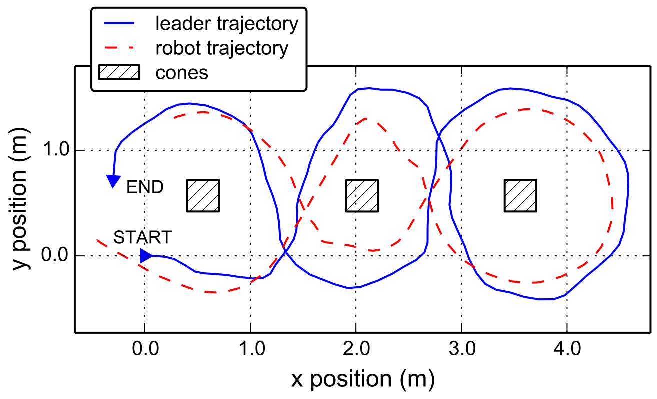
Figure A.12: Normal path deviation for a relatively difficult test case. The path segments are too large, so some normal vectors cannot intersect with the robot path.

Appendix B

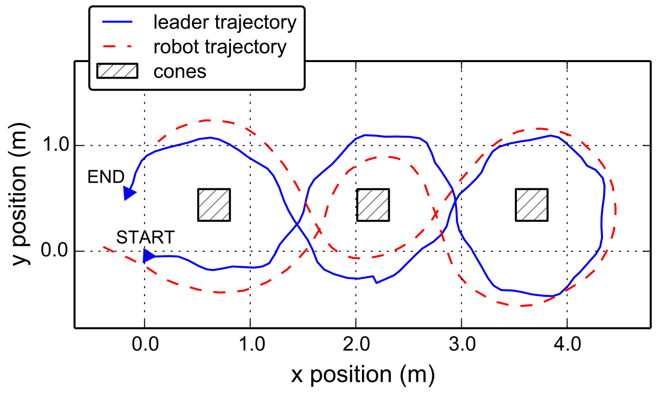
Results of Motion Capture Experiments

Selected results were presented and summarized in Chapter 3, and the full results for each subject are included in this appendix. These comprise robot and leader trajectories for each experiment, along with the calculated normal path deviation.

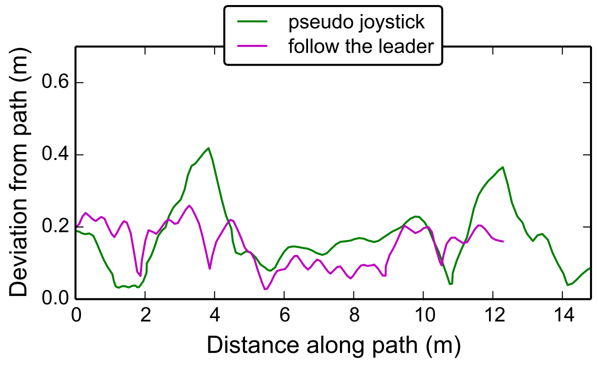
B.1 Subject 1



(a) Pseudo-joystick control.

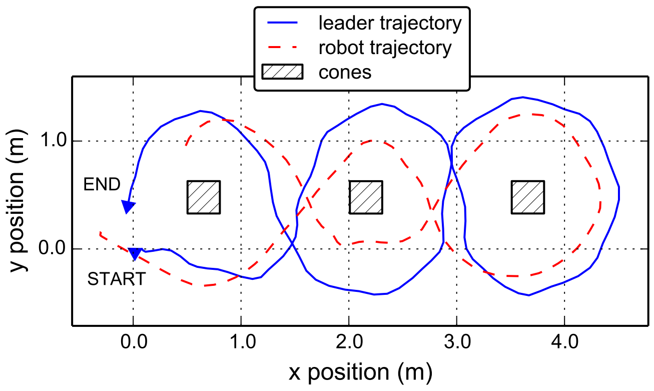


(b) Follow-the-leader with constant distance control.

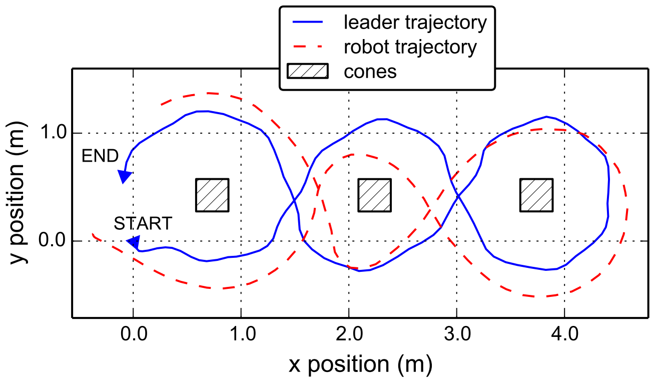


(c) Normal path deviation.

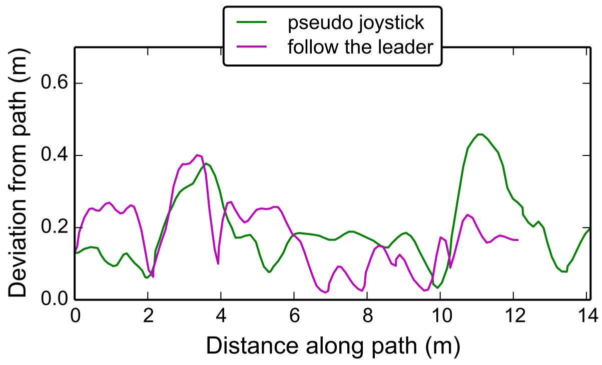
Figure B.1: Motion capture results for subject 1, test 1.



(a) Pseudo-joystick control.

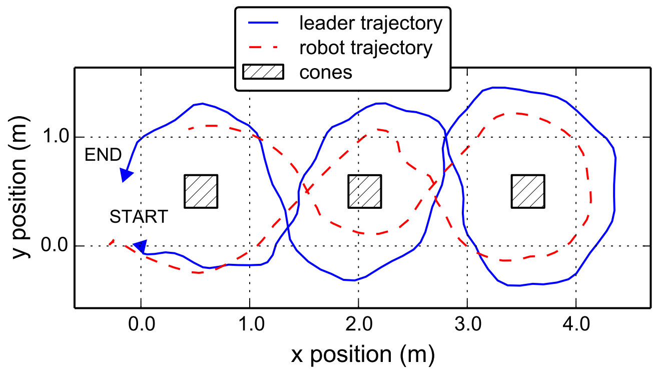


(b) Follow-the-leader with constant distance control.

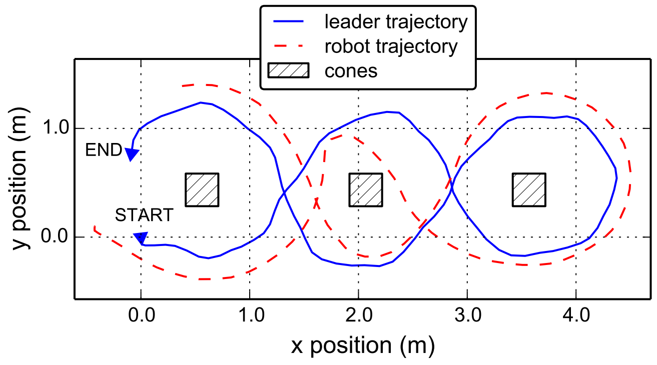


(c) Normal path deviation.

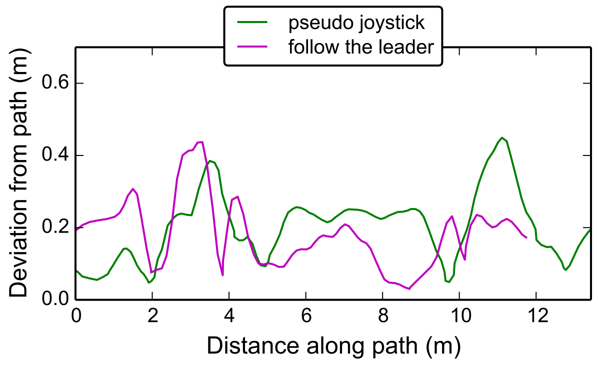
Figure B.2: Motion capture results for subject 1, test 2.



(a) Pseudo-joystick control.



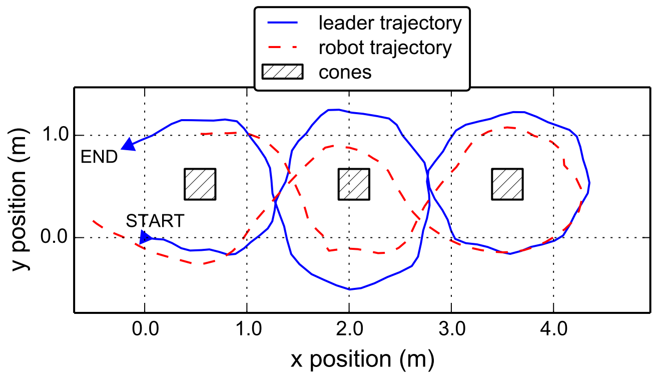
(b) Follow-the-leader with constant distance control.



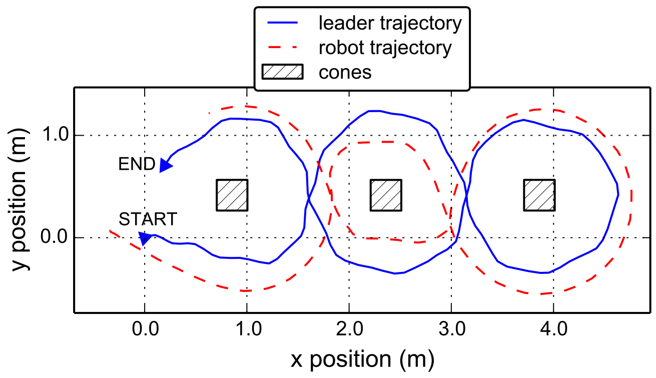
(c) Normal path deviation.

Figure B.3: Motion capture results for subject 1, test 3.

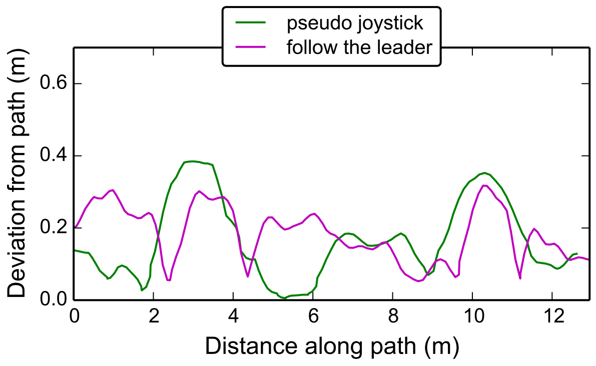
B.2 Subject 2



(a) Pseudo-joystick control.

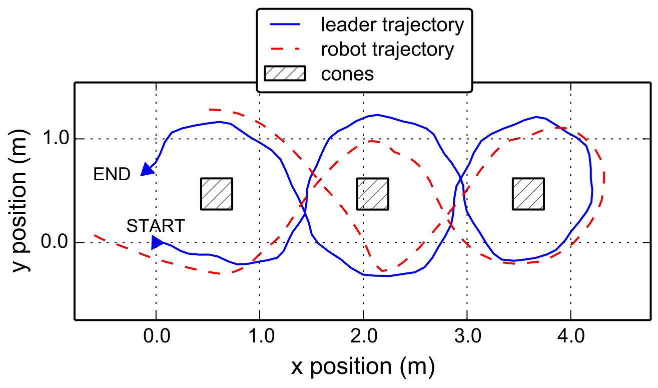


(b) Follow-the-leader with constant distance control.

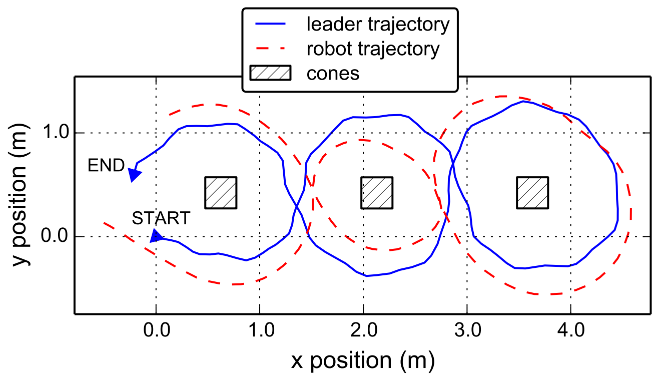


(c) Normal path deviation.

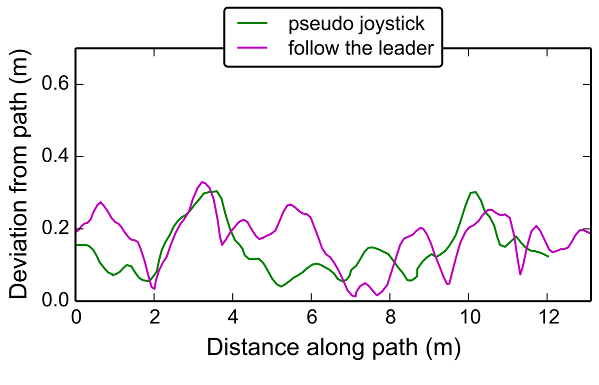
Figure B.4: Motion capture results for subject 2, test 1.



(a) Pseudo-joystick control.

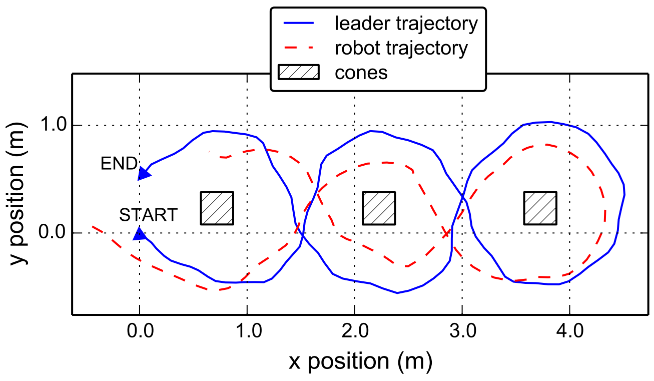


(b) Follow-the-leader with constant distance control.

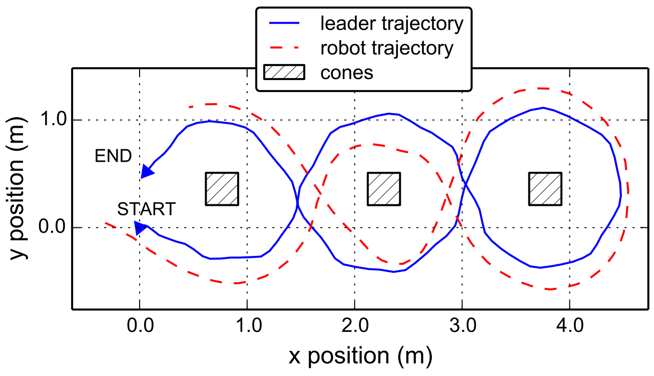


(c) Normal path deviation.

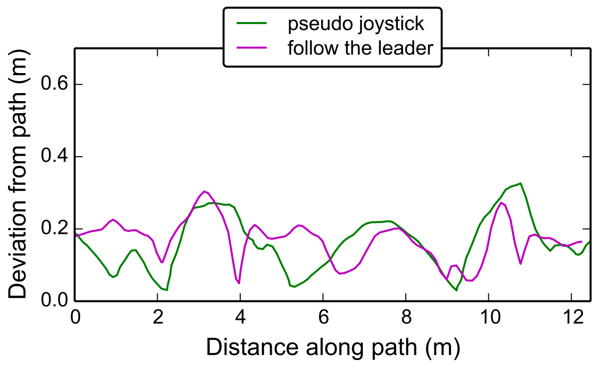
Figure B.5: Motion capture results for subject 2, test 2.



(a) Pseudo-joystick control.



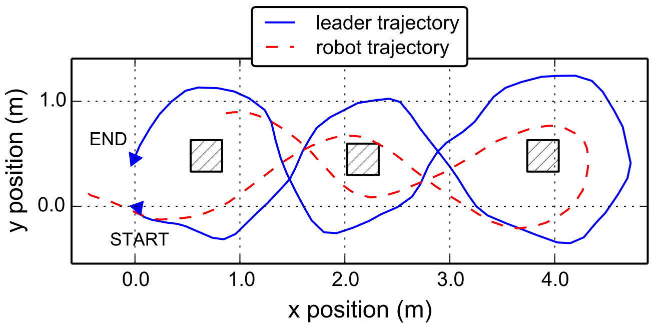
(b) Follow-the-leader with constant distance control.



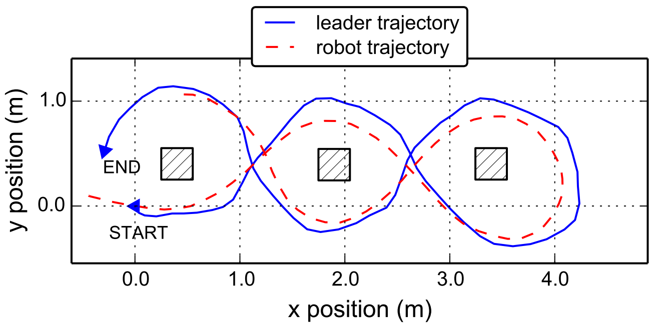
(c) Normal path deviation.

Figure B.6: Motion capture results for subject 2, test 3.

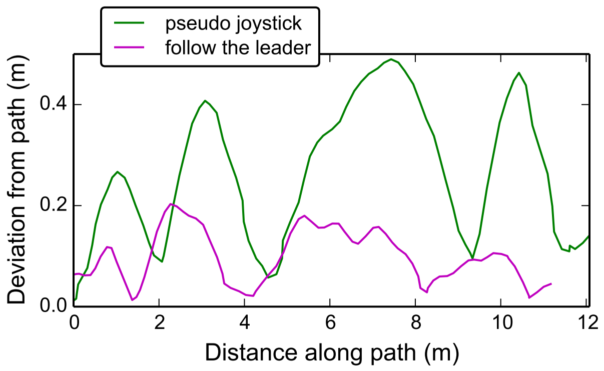
B.3 Subject 3



(a) Pseudo-joystick control.

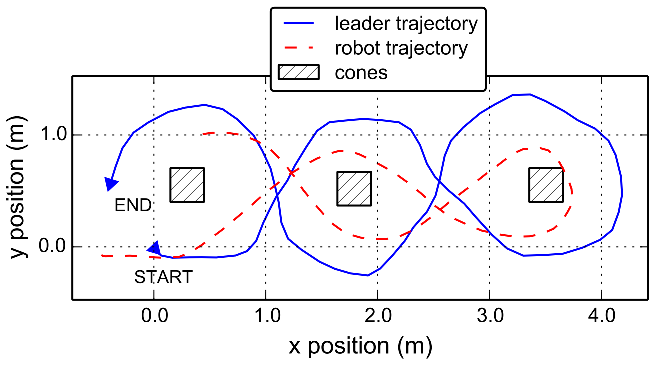


(b) Follow-the-leader with constant distance control.

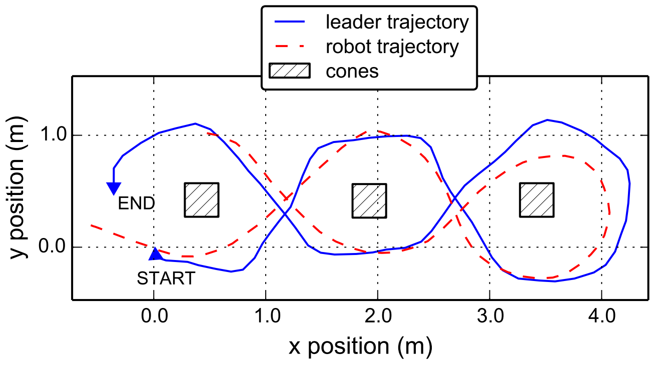


(c) Normal path deviation.

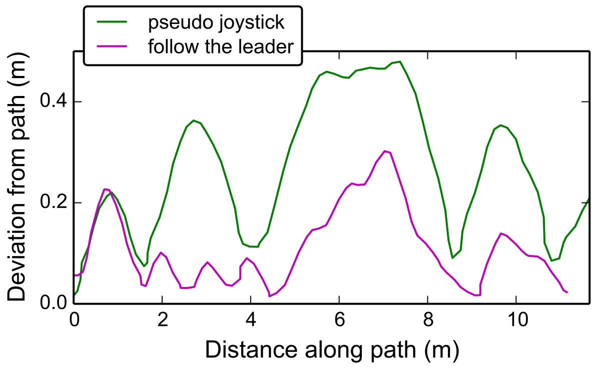
Figure B.7: Motion capture results for subject 3, test 1.



(a) Pseudo-joystick control.

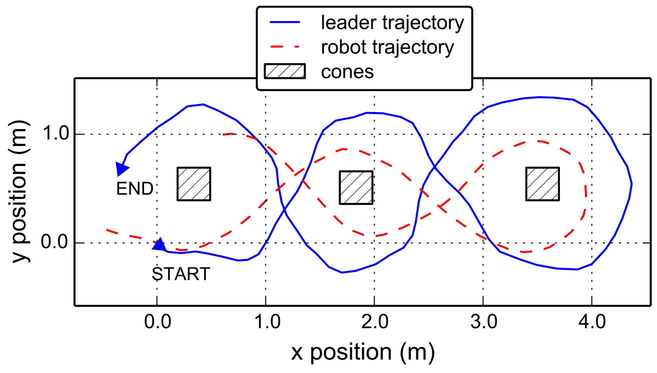


(b) Follow-the-leader with constant distance control.

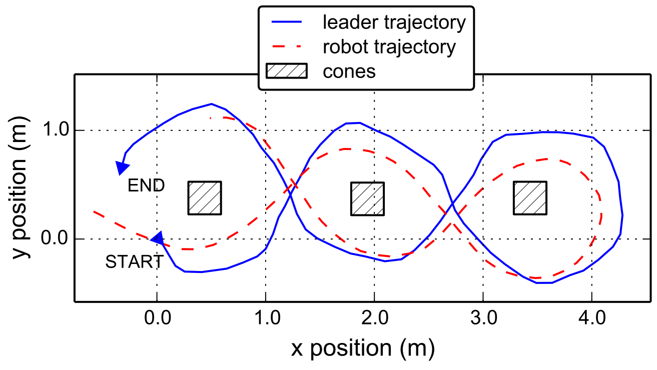


(c) Normal path deviation.

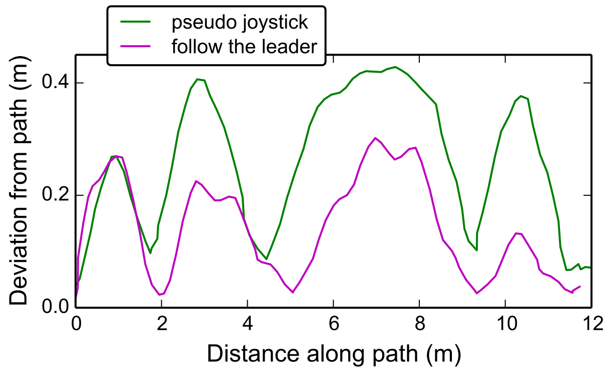
Figure B.8: Motion capture results for subject 3, test 2.



(a) Pseudo-joystick control.



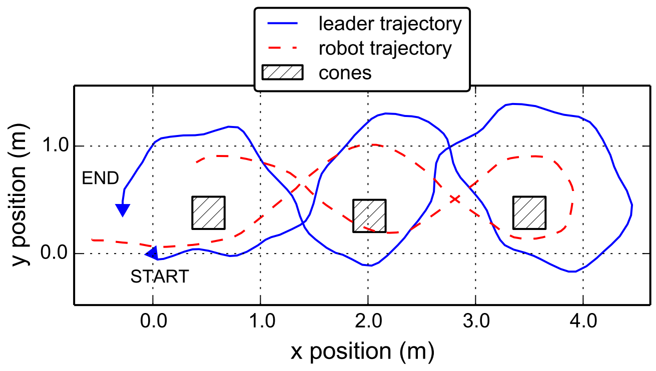
(b) Follow-the-leader with constant distance control.



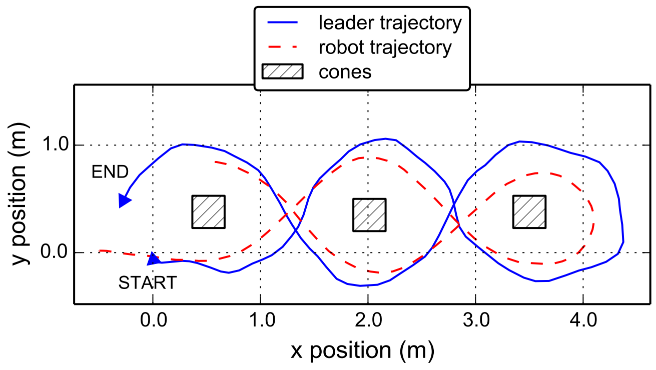
(c) Normal path deviation.

Figure B.9: Motion capture results for subject 3, test 3.

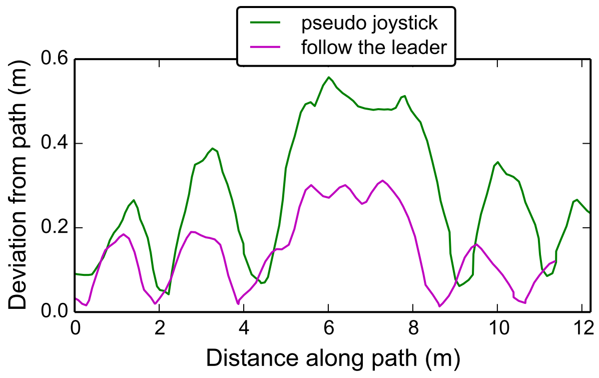
B.4 Subject 4



(a) Pseudo-joystick control.

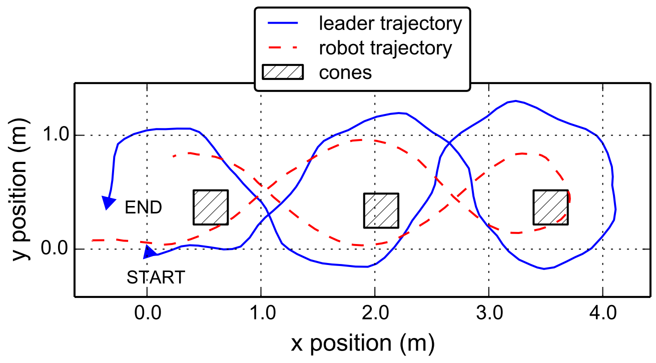


(b) Follow-the-leader with constant distance control.

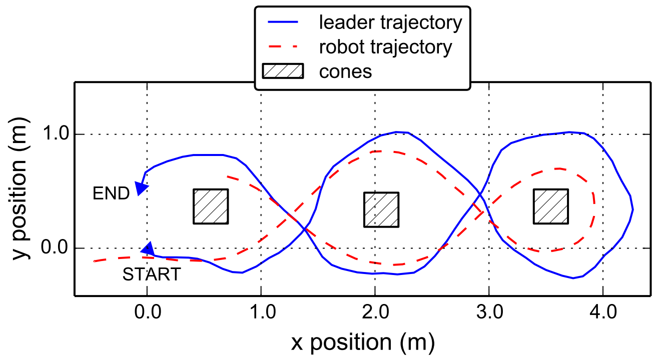


(c) Normal path deviation.

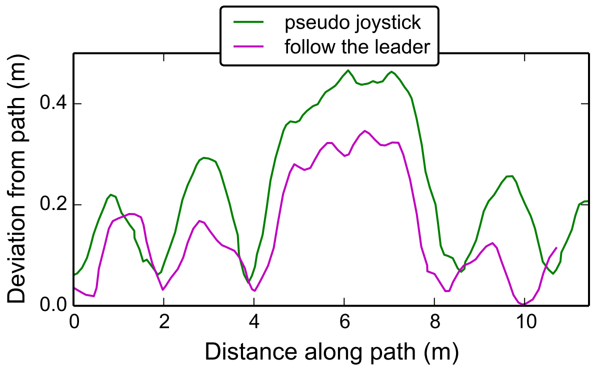
Figure B.10: Motion capture results for subject 4, test 1.



(a) Pseudo-joystick control.

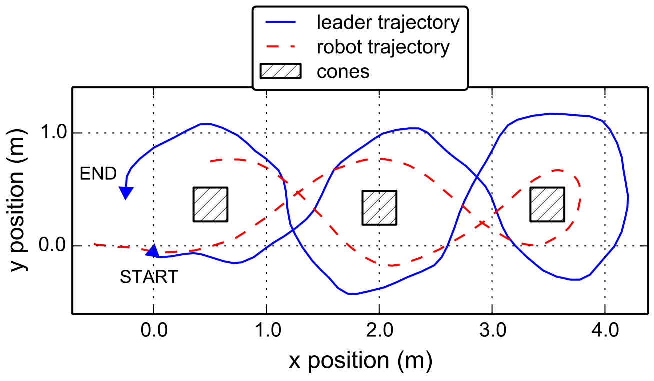


(b) Follow-the-leader with constant distance control.

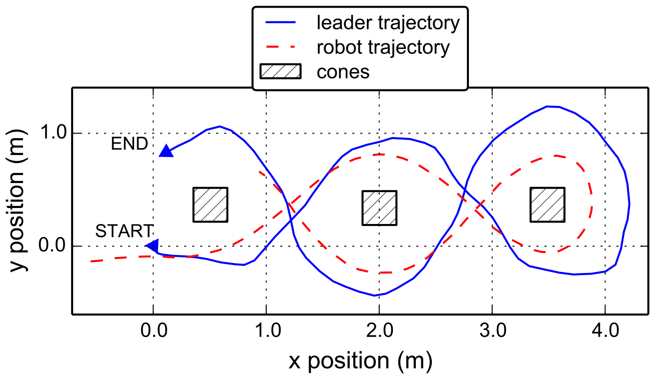


(c) Normal path deviation.

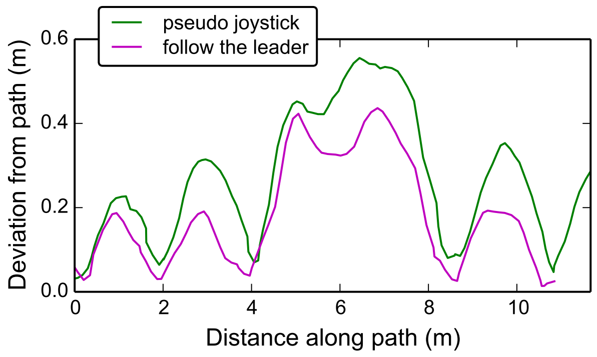
Figure B.11: Motion capture results for subject 4, test 2.



(a) Pseudo-joystick control.



(b) Follow-the-leader with constant distance control.



(c) Normal path deviation.

Figure B.12: Motion capture results for subject 4, test 3.

Appendix C

Python Implementation of Follower Algorithms

The follower algorithms presented in Chapter 2 and Chapter 4 were implemented in Python for prototyping, tuning, and analysis, before they were later ported to C for implemetation in hardware prototypes. The Python library containing the follower algorithms is listed below, as it shows the structure of the follower 'mode' model as well as relevant implementation details. Each follower mode was implemented as a separate class, with inherited behaviour where appropriate.

Some additional code, not presented here, was included in the V-REP simulation file to facilitate communication between the V-REP server and the Python implementation above (e.g. a small amount of Lua code was used to initialize the simulation, etc.). In the Python Follower Library the *utils.py* module is imported. This contained some additional helper functions for tasks such as plotting output trajectories.

```python
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

import math
import matplotlib.pyplot as plt
import numpy as np
from scipy import signal
from vrep import *
from . import utils

# alias so we can call modes by name or number
mode_alias = { 0 : "stop", "stop" : 0,
               10 : "pseudo_joystick", "pseudo_joystick" : 10,
               20 : "follow_the_leader", "follow_the_leader" : 20,
               21 : "follow_the_leader_const", "follow_the_leader_const" :
                   21,
               30 : "side_joystick", "side_joystick" : 30,
               31 : "side_tracking", "side_tracking" : 31,
               40 : "front_joystick", "front_joystick" : 40,
               42 : "front_vel_joystick", "front_vel_joystick" : 42}


#######################################################################
#                                                                     #
#   FOLLOWER MODES (ALGORITHMS)                                       #
#                                                                      #
########################################################################

class Stopped(object):
    def __init__(self, follower):
        self.follower_vehicle = follower
        pass

    def start(self):
        pass

    def stop(self):
        pass

    def follow(self, l_m, theta_m):
        follower = self.follower_vehicle
        #  We send the vals below to vrep as packed floats, so we have to
            set them to something
        #  (not None)
        follower.V_d = 0.0
        follower.Omega_d = 0.0
        follower.omega_L_d = 0.0
        follower.omega_R_d = 0.0


class FollowerMode(object):
    def __init__(self, follower, winch=None):
        self.follower_vehicle = follower
        # If no winch is given, use the default (first) winch on follower
            vehicle
        if winch is None:
            # TODO: check the vehicle actually has a winch
            self.winch = self.follower_vehicle.winches[0]
        else:
            self.winch = winch
        self.vrep_sim = True


class JoystickFollower(FollowerMode):
    # For code which is common to all joystick follower modes
    def __init__(self, follower):
        FollowerMode.__init__(self, follower)


    class TetherBuffer(object):
        def __init__(self):
            self.l = None
            self.size = None  # current num of points in the buffer
            self.limit = None  # max num of points in the buffer
            self.ds_min = None  # min step s for recording a new point

class PseudoJoystick(JoystickFollower):
    def __init__(self, follower):
        JoystickFollower.__init__(self, follower)

        self.l_d = 1.0  # the desired tether length

        # proportional constants for control
        self.kV = 5.0  # velocity gain
        self.kO = 12.0  # ang velocity gain


    def start(self):
        # Start following using this mode
        # Any startup code goes here
        pass

    def stop(self):
        # Stop following using this mode
        # Any shutdown code goes here
        pass

    def follow(self, l_m, theta_m):
        # Main leader following algorithm
        # Given the input tether data, calc the desired vehicle speed and
            heading

        follower = self.follower_vehicle

        # Find the difference between the length and our desired length
        e_l = l_m - self.l_d

        # Mode 0: Pseudo Joystick Control
        phi = theta_m  # desired heading angle
        if e_l > 0:
            V_d = self.kV * e_l
        else:
            V_d = 0

        Omega_d = self.kO * phi

        # Pass speed and heading to vehicle's kinematics module
        follower.V_d = V_d
        follower.Omega_d = Omega_d
        follower.transform_kinematics(V_d, Omega_d)
        # print("wL: {0}\t wR: {1}".format(follower.omega_L_d, follower.
            omega_R_d))

        # return speed and heading just in case the caller wants to use
            them directly
        return V_d, Omega_d



class SideJoystick(JoystickFollower):
    def __init__(self, follower):
        JoystickFollower.__init__(self, follower)

        # proportional constants for control
        self.kV = 1.25 # velocity
        self.kO = 1.8 # ang velocity omega

        self.l_d = 1.0 # the desired tether length
        self.theta_d = 0.5*math.pi

        self.V_max = 100.0  # max speed
        self.Omega_max = 100.0  # max ang velocity

    def start(self):
        # Start following using this mode
        # Any startup code goes here
        pass

    def stop(self):
        # Stop following using this mode
        # Any shutdown code goes here
        pass

    def follow(self, l_m, theta_m):
        # Main leader following algorithm
        # Given the input tether data, calc the desired vehicle speed and
            heading
```

```python

        follower = self.follower_vehicle

        # Find the difference between the length and our target length
        e_l = l_m - self.l_d
        e_theta = theta_m - self.theta_d

        # Mode: Naive Side Following
        if e_theta < 0:
            V_d = -self.kV * e_theta
        else:
            V_d = 0

        # ang vel component omega (note CCW is positive)
        Omega_d = self.kO * e_l

        # prevent turning when the velocity is low
        if V_d == 0:
            Omega_d = 0

        print("V_d: {0}; Omega_d: {1}".format(V_d, Omega_d))
        if V_d > self.V_max: V_d = self.V_max
        if Omega_d > self.Omega_max: Omega_d = self.Omega_max

        # Pass speed and heading to vehicle's kinematics module
        follower.V_d = V_d
        follower.Omega_d = Omega_d
        follower.transform_kinematics(V_d, Omega_d)

        # return speed and heading just in case the caller wants to use
            them directly
        return V_d, Omega_d


class FrontJoystick(JoystickFollower):
    def __init__(self, follower):
        JoystickFollower.__init__(self, follower)

        # proportional constants for control
        self.kV = 1.0  # velocity
        self.kO = 1.5  # 1.8 # ang velocity omega

        self.l_d = 1.0 # the desired tether length
        self.theta_d = math.pi  # Note this is right on the angle wrap
            boundary!

        self.V_max = 100.0  # max speed
        self.Omega_max = 100.0  # max ang velocity

    def start(self):
        # Start following using this mode
        # Any startup code goes here
        pass

    def stop(self):
        # Stop following using this mode
        # Any shutdown code goes here
        pass

    def follow(self, l_m, theta_m):
        # Main leader following algorithm
        # Given the input tether data, calc the desired vehicle speed and
            heading

        follower = self.follower_vehicle

        # Find the difference between the length and our target length
        e_l = l_m - self.l_d
        e_theta = theta_m - self.theta_d


        # Wrap e_theta
        if e_theta > math.pi:
            e_theta = e_theta - 2 * math.pi
        if e_theta < -math.pi:
            e_theta = e_theta + 2 * math.pi

        V_d = -1.0 * self.kV * e_l

        # # Limit vel to forwards only
        # if V_d < 0.0:
            # V_d = 0.0

        phi = e_theta  # desired heading angle
        Omega_d = self.kO * phi

        if e_theta > math.pi/2 or e_theta < -math.pi/2:
            Omega_d = 0.0

        print("e_l: {0}; e_theta: {1}".format(e_l, e_theta))
        print("V_d: {0}; Omega_d: {1}".format(V_d, Omega_d))
        if V_d > self.V_max: V_d = self.V_max
        if Omega_d > self.Omega_max: Omega_d = self.Omega_max

        # Pass speed and heading to vehicle's kinematics module
        follower.V_d = V_d
        follower.Omega_d = Omega_d
        follower.transform_kinematics(V_d, Omega_d)

        # return speed and heading just in case the caller wants to use
            them directly
        return V_d, Omega_d


class FrontVelJoystick(JoystickFollower):
    def __init__(self, follower):
        JoystickFollower.__init__(self, follower)

        # proportional constants for control
        self.kV = 1.0  # velocity
        self.kO = 0.8  # 1.8 # ang velocity omega

        self.l_d = 1.0 # the desired tether length
        self.theta_d = math.pi  # Note this is right on the angle wrap
            boundary!

        self.V_max = 100.0  # max speed
        self.Omega_max = 100.0  # max ang velocity

        # tether length buffer
        self.l_m_buffer = self.TetherBuffer()
        self.l_m_buffer.size = 0
        self.l_m_buffer.limit = 5
        self.l_m_buffer.l = self.l_m_buffer.limit * [0.0]
        self.l_m_buffer.ds_min = 0.0

        # tether angle buffer
        self.theta_m_buffer = self.TetherBuffer()
        self.theta_m_buffer.size = 0
        self.theta_m_buffer.limit = 5
        self.theta_m_buffer.l = self.theta_m_buffer.limit * [0.0]
        self.theta_m_buffer.ds_min = 0.0

    def start(self):
        # Start following using this mode
        # Any startup code goes here
        pass

    def stop(self):
        # Stop following using this mode
        # Any shutdown code goes here
        pass

    def follow(self, l_m, theta_m):
        # Main leader following algorithm
        # Given the input tether data, calc the desired vehicle speed and
            heading

        follower = self.follower_vehicle
        dt = follower.dt

        # Tether length buffer
        #  shift each array value up one
        for i in xrange(self.l_m_buffer.limit-2, -1, -1):
            self.l_m_buffer.l[i+1] = self.l_m_buffer.l[i]
        #  add the new values to the start of the buffer
        self.l_m_buffer.l[0] = l_m
        self.l_m_buffer.size = self.l_m_buffer.size + 1
        if self.l_m_buffer.size > self.l_m_buffer.limit - 1:
            self.l_m_buffer.size = self.l_m_buffer.limit - 1

        # Tether angle buffer
        #  shift each array value up one
        for i in xrange(self.theta_m_buffer.limit-2, -1, -1):
            self.theta_m_buffer.l[i+1] = self.theta_m_buffer.l[i]
        #  add the new values to the start of the buffer
        self.theta_m_buffer.l[0] = theta_m
        self.theta_m_buffer.size = self.theta_m_buffer.size + 1
        if self.theta_m_buffer.size > self.theta_m_buffer.limit - 1:
            self.theta_m_buffer.size = self.theta_m_buffer.limit - 1

        # Use end points of buffer for pos 1 and 2
        l_1 = self.l_m_buffer.l[-1]
        l_2 = self.l_m_buffer.l[0]
        theta_1 = self.theta_m_buffer.l[-1]
        theta_2 = self.theta_m_buffer.l[0]
        dx = l_2 * np.cos(theta_2) - l_1 * np.cos(theta_1)
        dy = l_2 * np.sin(theta_2) - l_1 * np.sin(theta_1)

        time_span = dt * (self.l_m_buffer.size + 1)
        V_tip_x = dx / time_span
        V_tip_y = dy / time_span

        # # Find the difference between the length and our target length
        # e_l = l_m - self.l_d
        # e_theta = theta_m - self.theta_d

        # # Wrap e_theta
        # if e_theta > math.pi:
            # e_theta = e_theta - 2 * math.pi
        # if e_theta < -math.pi:
            # e_theta = e_theta + 2 * math.pi

        # Set target vel to same as tip vel
        V_d = np.hypot(V_tip_x, V_tip_y)
        phi = np.arctan2(V_tip_y, V_tip_x)


        Omega_d = self.k0 * phi

        if V_d > self.V_max: V_d = self.V_max
        if Omega_d > self.Omega_max: Omega_d = self.Omega_max

        # Pass speed and heading to vehicle's kinematics module
        follower.V_d = V_d
        follower.Omega_d = Omega_d
        follower.transform_kinematics(V_d, Omega_d)

        # return speed and heading just in case the caller wants to use
            them directly
        return V_d, Omega_d


class TrackingFollower(FollowerMode):
    def __init__(self, follower):
        FollowerMode.__init__(self, follower)

    class PointBuffer(object):
        def __init__(self):
            self.x = None
            self.y = None
            self.s = None
            self.size = None  # current num of points in the buffer
            self.limit = None  # max num of points in the buffer
            self.ds_min = None  # min step s for recording a new point

    def point_tracker(self, Px, Py, point_buffer):
        # Track the position of a point and record the history in an array/
            buffer
        # Usually the tracked point is the tether tip, but it may also be a
             side offset point, or something else

        # calculate distance travelled
        if point_buffer.size < 1:
            s_t = 0
            ds = 1e12
        else:
            dx = Px - point_buffer.x[0]
            dy = Py - point_buffer.y[0]
            ds = math.sqrt(dx*dx + dy*dy)
            s_t = point_buffer.s[0] + ds

        # check if the leader has travelled more than ds_min
        if ds >= point_buffer.ds_min:
            #  shift each array value up one
            for i in xrange(point_buffer.limit-2, -1, -1):
                point_buffer.x[i+1] = point_buffer.x[i]
                point_buffer.y[i+1] = point_buffer.y[i]
                point_buffer.s[i+1] = point_buffer.s[i]

            #  add the new values to the start of the buffer
            point_buffer.x[0] = Px
            point_buffer.y[0] = Py
            point_buffer.s[0] = s_t

            point_buffer.size = point_buffer.size + 1
            if point_buffer.size > point_buffer.limit - 1:
                point_buffer.size = point_buffer.limit - 1

        return s_t, point_buffer


    def find_closest_point(self, Px, Py, point_buffer):
        # find the min distance (rho) between point Px,y and point buffer
```

```python
        # xs and ys are tables/arrays of x and y points. n is the number of points
        n = point_buffer.size
        if n < 2:
            return 0, 0.0

        rho_min_sq = 1e12
        path_index = 0  #  index of min point
        for i in xrange(0, n):
            dx = Px - point_buffer.x[i]
            dy = Py - point_buffer.y[i]
            rho_sq = dx*dx + dy*dy
            if rho_sq < rho_min_sq:
                rho_min_sq = rho_sq
                path_index = i

        return path_index, math.sqrt(rho_min_sq)


    def find_lookahead_point(self, point_buffer, start_point, delta):
        #  find a target point on the leader path, which is at least delta ahead of the robot
        #  delta is measured along the s-distance of the path, NOT in a straight line

        n = point_buffer.size

        #  set a sensible default target (0 is the index of the leader position)
        target = 0
        for i in xrange(start_point, -1, -1):
            # print(i, paths[i], paths[startPoint])
            if point_buffer.s[i] - point_buffer.s[start_point] >= delta:
                target = i
                break

        Ttarget_x = point_buffer.x[target]
        Ttarget_y = point_buffer.y[target]
        return target, Ttarget_x, Ttarget_y


    def find_const_dist_target(self, point_buffer, l_d):
        # find a target point on the leader path, which is at least l_d behind the leader
        # l_d is measured in a straight line from the leader, NOT along the s-distance of the path
        # set a sensible default target (0 is the index of the leader position)

        # set a sensible default target (0 is the index of the leader position)
        target = 0
        # loop over leader path, starting at leader position
        for i in xrange(0, point_buffer.size, 1):
            # print('bufSize: ', n, '; i: ', i, '; ')
            dx = point_buffer.x[i] - point_buffer.x[0]
            dy = point_buffer.y[i] - point_buffer.y[0]
            dist = math.sqrt(dx*dx + dy*dy)
            if dist >= l_d:
                target = i
                break

        Ttarget_x = point_buffer.x[target]
        Ttarget_y = point_buffer.y[target]
        return target, Ttarget_x, Ttarget_y

```

```python


class FollowTheLeader(TrackingFollower):
    def __init__(self, follower):
        TrackingFollower.__init__(self, follower)

        self.l_d = 1.0  # the desired tether length (OxyDog default=0.6m)
        self.delta = 0.5 # lookahead distance

        # proportional constants for control
        self.kV = 1.0  # velocity
        self.kO = 3.0  # ang velocity omega
        self.phi = None  #  relative angle to the target point

        # leader/tip position buffer
        self.tip_buffer = self.PointBuffer()
        self.tip_buffer.size = 0
        self.tip_buffer.limit = 300
        self.tip_buffer.x = self.tip_buffer.limit * [0.0]
        self.tip_buffer.y = self.tip_buffer.limit * [0.0]
        self.tip_buffer.s = self.tip_buffer.limit * [0.0]
        self.tip_buffer.ds_min = 0.01


    def start(self):
        # Start following using this mode
        # Any startup code goes here
        pass

    def stop(self):
        # Stop following using this mode
        # Any shutdown code goes here
        pass

    def follow(self, l_m, theta_m):
        # Main leader following algorithm
        # Given the input tether data, calc the desired vehicle speed and
            heading

        follower = self.follower_vehicle
        winch = self.follower_vehicle.winches[0]

        # calculate the position of the leader/tip
        Tx = follower.x + winch.offset_x * math.cos(follower.heading) +
            winch.offset_y * math.sin(follower.heading) + l_m * math.cos(
            follower.heading + theta_m)
        Ty = follower.y + winch.offset_x * math.sin(follower.heading) +
            winch.offset_y * math.cos(follower.heading) + l_m * math.sin(
            follower.heading + theta_m)

        # track leader position (Leader path stored in global tables
            bufferTx, bufferTy, bufferTs)
        s_t, self.tip_buffer = self.point_tracker(Tx, Ty, self.tip_buffer)

        # find the point on the leader path which is closest to the robot
        s_tmin, rho = self.find_closest_point(follower.x, follower.y, self.
            tip_buffer)
        # Note: s_tmin is an array index, not really an s (distance) value

        Tmin_x = self.tip_buffer.x[s_tmin]
        Tmin_y = self.tip_buffer.y[s_tmin]

        # look ahead on the leader path to find our target direction
        # Note: Ttarget is T(stmin + delta) in past papers
        target, Ttarget_x, Ttarget_y = self.find_lookahead_point(self.
            tip_buffer, s_tmin, self.delta)

        self.target_x = Ttarget_x  # store these values for updating vrep
            dummies
        self.target_y = Ttarget_y

        # Find our desired heading
        # absolute angle Phi
        Phi = math.atan2(Ttarget_y - follower.y, Ttarget_x - follower.x)
        # angle relative to robot
        self.phi = Phi - follower.heading
        self.phi = utils.wrap_angle(self.phi)

        # Find the difference between the length and our desired length
        e_l = l_m - self.l_d

        # desired speed, ang velocity
        V_d = self.kV * e_l
        Omega_d = self.kO * self.phi

        # Pass speed and heading to vehicle's kinematics module
        follower.V_d = V_d
        follower.Omega_d = Omega_d
        follower.transform_kinematics(V_d, Omega_d)

        print("omega_L_d: ", follower.omega_L_d, "omega_R_d: ", follower.
            omega_R_d)

        # return speed and heading just in case the caller wants to use
            them directly
        return V_d, Omega_d


class FollowTheLeaderConst(TrackingFollower):
    def __init__(self, follower):
        TrackingFollower.__init__(self, follower)

        self.l_d = 1.2  # the desired tether length (OxyDog default=0.6m)

        # proportional constants for control
        self.K_p = -2.0
        self.K_a = 0.5
        self.K_b = -2.0

        self.phi = None  #  relative angle to the target point

        # leader/tip position buffer
        self.tip_buffer = self.PointBuffer()
        self.tip_buffer.size = 0
        self.tip_buffer.limit = 300
        self.tip_buffer.x = self.tip_buffer.limit * [0.0]
        self.tip_buffer.y = self.tip_buffer.limit * [0.0]
        self.tip_buffer.s = self.tip_buffer.limit * [0.0]
        self.tip_buffer.ds_min = 0.01

    def start(self):
        # Start following using this mode
        # Any startup code goes here
        pass

    def stop(self):
        # Stop following using this mode
        # Any shutdown code goes here
        pass

    def follow(self, l_m, theta_m):
        # Main leader following algorithm
```

```python
        # Given the input tether data, calc the desired vehicle speed and heading

        follower = self.follower_vehicle
        winch = self.follower_vehicle.winches[0]

        # calculate the position of the leader/tip
        Tx = follower.x + winch.offset_x * math.cos(follower.heading) + winch.offset_y * math.sin(follower.heading) + l_m * math.cos(follower.heading + theta_m)
        Ty = follower.y + winch.offset_x * math.sin(follower.heading) + winch.offset_y * math.cos(follower.heading) + l_m * math.sin(follower.heading + theta_m)

        # track leader position (Leader path stored in global tables bufferTx, bufferTy, bufferTs)
        s_t, self.tip_buffer = self.point_tracker(Tx, Ty, self.tip_buffer)

        # find the point on the leader path which is closest to the robot
        s_tmin, rho = self.find_closest_point(follower.x, follower.y, self.tip_buffer)
        # Note: s_tmin is an array index, not really an s (distance) value
        Tmin_x = self.tip_buffer.x[s_tmin]
        Tmin_y = self.tip_buffer.y[s_tmin]

        # look ahead on the leader path to find our target direction
        # Note: Ttarget is T(stmin + delta) past papers
        target, Ttarget_x, Ttarget_y = self.find_const_dist_target(self.tip_buffer, self.l_d)

        self.target_x = Ttarget_x  # store these values for updating vrep dummies
        self.target_y = Ttarget_y

        # Find our desired heading
        # absolute angle Phi
        Phi = math.atan2(Ttarget_y - follower.y, Ttarget_x - follower.x)
        # angle relative to robot
        self.phi = Phi - follower.heading
        self.phi = utils.wrap_angle(self.phi)
        # print('Phi: ', math.degrees(Phi), 'phi: ', math.degrees(self.phi))

        # find magnitude of angle phi
        phi_mag = math.sqrt(self.phi * self.phi)

        # Control laws (see Endo/Tani IROS 2011 paper)
        V_d = -self.K_p * (l_m - self.l_d) * (1 - self.K_a * phi_mag)
        Omega_d = -self.K_b * self.phi

        if V_d < 0:
            V_d = 0

        # Pass speed and heading to vehicle's kinematics module
        follower.V_d = V_d
        follower.Omega_d = Omega_d
        follower.transform_kinematics(V_d, Omega_d)

        # return speed and heading just in case the caller wants to use them directly
        return V_d, Omega_d


class SideTracking(TrackingFollower):
    def __init__(self, follower):
        TrackingFollower.__init__(self, follower)

        self.l_d = 0.1  # the desired tether length (OxyDog default=0.6m)
        self.delta = 0.6 # lookahead distance

        # proportional constants for control
        self.kV = 1.0  # velocity
        self.kO = 3.0  # ang velocity omega

        self.phi = None  #  relative angle to the target point

        self.offset_angle = math.pi/2
        self.offset_length = 1.0

        # leader/tip position buffer
        self.tip_buffer = self.PointBuffer()
        self.tip_buffer.size = 0
        self.tip_buffer.limit = 10
        self.tip_buffer.x = self.tip_buffer.limit * [0.0]
        self.tip_buffer.y = self.tip_buffer.limit * [0.0]
        self.tip_buffer.s = self.tip_buffer.limit * [0.0]
        self.tip_buffer.ds_min = 0.01


        # side offset point position buffer
        self.side_buffer = self.PointBuffer()
        self.side_buffer.size = 0
        self.side_buffer.limit = 300
        self.side_buffer.x = self.side_buffer.limit * [0.0]
        self.side_buffer.y = self.side_buffer.limit * [0.0]
        self.side_buffer.s = self.side_buffer.limit * [0.0]
        self.side_buffer.ds_min = 0.01

        self.Kx = None
        self.Ky = None
        self.target_x = None  # store these values for updating vrep
            dummies
        self.target_y = None


    def start(self):
        # Start following using this mode
        # Any startup code goes here
        pass

    def stop(self):
        # Stop following using this mode
        # Any shutdown code goes here
        pass

    def follow(self, l_m, theta_m):
        # Main leader following algorithm
        # Given the input tether data, calc the desired vehicle speed and
            heading

        follower = self.follower_vehicle
        winch = self.follower_vehicle.winches[0]

        print("l_m: ", l_m, "theta_m: ", math.degrees(theta_m))

        # calculate the position of the leader/tip
        Tx = follower.x + winch.offset_x * math.cos(follower.heading) +
            winch.offset_y * math.sin(follower.heading) + l_m * math.cos(
            follower.heading + theta_m)
        Ty = follower.y + winch.offset_x * math.sin(follower.heading) +
            winch.offset_y * math.cos(follower.heading) + l_m * math.sin(
            follower.heading + theta_m)
        print("l_m: ", l_m, "theta_m: ", math.degrees(theta_m))
        print('heading: ', math.degrees(follower.heading), 'follower.x: ',
            follower.x, 'follower.y: ' , follower.y)
        print('Tx: ', Tx, 'Ty: ', Ty)


        # track leader position (Leader path stored in global tables
            bufferTx, bufferTy, bufferTs)
        s_t, self.tip_buffer = self.point_tracker(Tx, Ty, self.tip_buffer)
        print('Tx, y:', Tx, Ty)

        Kx, Ky = self.side_offset(self.tip_buffer, self.offset_angle, self.
            offset_length)

        self.Kx = Kx  # store these values for updating vrep dummies
        self.Ky = Ky


        print("Kx {0}; Ky {1};".format(Kx, Ky))

        if Kx != None and Ky != None:

            s_k, self.side_buffer = self.point_tracker(Kx, Ky, self.
                side_buffer)

            # find the point on the side tracked path which is closest to
                the robot
            s_kmin, rho = self.find_closest_point(follower.x, follower.y,
                self.side_buffer)
            # Note: s_kmin is an array index, not really an s (distance)
                value

            Kmin_x = self.side_buffer.x[s_kmin]
            Kmin_y = self.side_buffer.y[s_kmin]

            # look ahead on the leader path to find our target direction
            # Note: Ttarget is T(stconst) in Endo/Tani paper
            target, target_x, target_y = self.find_lookahead_point(self.
                side_buffer, s_kmin, self.delta)

            self.target_x = target_x  # store these values for updating
                vrep dummies
            self.target_y = target_y

            # Find our desired heading
            # absolute angle Phi
            Phi = math.atan2(target_y - follower.y, target_x - follower.x)
            # angle relative to robot
            self.phi = Phi - follower.heading
            self.phi = utils.wrap_angle(self.phi)

            dx = Kx - follower.x
            dy = Ky - follower.y
            e_l = math.sqrt(dx*dx + dy*dy) - self.l_d
        else:
            self.phi = 0.0
            e_l = 0.5

        # compute left and right desired velocities
        follower.omega_L_d = (self.kV * e_l - self.k0 * 0.5 * follower.b *
            self.phi) / follower.r
        follower.omega_R_d = (self.kV * e_l + self.k0 * 0.5 * follower.b *
            self.phi) / follower.r
        print("omega_L_d: ", follower.omega_L_d, "omega_R_d: ", follower.
            omega_R_d)

```

```

    def side_offset(self, point_buffer, offset_angle, offset_length, n=4):

        Tx = point_buffer.x[0]
        Ty = point_buffer.y[0]

        if point_buffer.size < n:
            print('waiting for enough points')
            Kx, Ky = None, None
        else:
            # get the first few values from buffer T
            ax = point_buffer.x[0:n]
            ay = point_buffer.y[0:n]
            bx = np.diff(np.array(ax))
            by = np.diff(np.array(ay))

            # find average point
            Px = np.mean(ax)
            Py = np.mean(ay)

            # find average vector
            Mx = np.mean(bx)
            My = np.mean(by)
            #print('Mx: ', Mx, 'My: ', My)

            # rotate vector
            Mxrot, Myrot = utils.vec_rotate(Mx, My, offset_angle)
            # scale vector to the side offset length
            Mxrot2, Myrot2 = utils.vec_scale_to_length(Mxrot, Myrot,
                offset_length)
            # find side offset point K
            Kx = Tx + Mxrot2
            Ky = Ty + Myrot2

        return Kx, Ky



######################################################################
#                                                                    #
#    VEHICLES                                                        #
#                                                                    #
######################################################################

class Vehicle(object):
    """An arbitrary vehicle. Could be a 2 wheeled robot, a buggy, a snake.
    """
    def __init__(self):
        self.name = None
        self.V = None   # Vehicle speed (along local x axis)
        self.Omega = None   # Vehicle rot velocity (about local z axis)
        self.V_d = None   # _d: desired speed
        self.Omega_d = None   #

        # Following are in inertial ref frame, with origin at vehicle start
            point
        self.x = None   # x position
        self.y = None   # y position
        self.heading = 0.0   # Vehicle heading angle (Gamma in IROS paper)
        self.s = None   # Distance travelled (positive scalar, along path)

        self.x_0 = 0.0   # _0: values from previous time step
        self.y_0 = 0.0   #
        self.heading_0 = 0.0   #
        self.s_0 = 0.0   #

        self.winches = []  # list of winches mounted on this vehicle
        self.tip_interfaces = []  # list of tether tips attached to this vehicle

        self.mode = None  # Follower mode the is vehicle using

        self.dt = None  # time step


    def add_winch(self, winch):
        self.winches.append(winch)

    def add_tip_interface(self, tip_interface):
        self.tip_interfaces.append(tip_interface)

    def set_mode(self, mode):
        """Input mode should be a valid int or string from mode_alias"""
        if isinstance(mode, basestring):
            try:
                mode = mode_alias[mode]
            except:
                print("Error: invalid follower mode passed to set_mode() in {0}".format(self.name))
                raise
        if not mode in mode_alias:
            print("Error: invalid follower mode passed to set_mode() in {0}".format(self.name))
            raise Exception()

        if self.mode is not None:
            # Run the terminate routine for the previous mode
            self.mode.stop()
            pass

        if mode == 0:
            self.mode = Stopped(self)
        elif mode == 10:
            self.mode = PseudoJoystick(self)
        elif mode == 20:
            self.mode = FollowTheLeader(self)
        elif mode == 21:
            self.mode = FollowTheLeaderConst(self)
        elif mode == 30:
            self.mode = SideJoystick(self)
        elif mode == 31:
            self.mode = SideTracking(self)
        elif mode == 40:
            self.mode = FrontJoystick(self)
        elif mode == 42:
            self.mode = FrontVelJoystick(self)

        self.mode.start()


class TwoWheeledRobot(Vehicle):
    def __init__(self):
        Vehicle.__init__(self)
        # Geometry
        self.b = None  # axle track (distance between wheel centers)
        self.r = None  # wheel radius
        # Kinematics
        self.omega_L = None  # Left wheel ang velocity
        self.omega_R = None  # Right wheel ang velocity
        self.omega_L_d = None  # _d: desired ang velocity
        self.omega_R_d = None  #
```

```python
        self.omega_max = np.inf  # Maximum wheel speed
        self.omega_min = -np.inf  # Minimum wheel speed
        self.wheel_pos_L = None  # Left wheel ang pos (this is a sim equivalent of encoder count)
        self.wheel_pos_R = None  # Right wheel...
        self.wheel_pos_L_0 = 0.0  # _0: values from previous time step
        self.wheel_pos_R_0 = 0.0

    def transform_kinematics(self, V_d, Omega_d):
        # Given the desired vehicle speed and ang velocity, calc the wheel speeds
        # compute left and right wheel desired velocities
        self.omega_L_d = (V_d - 0.5 * self.b * Omega_d) / self.r
        self.omega_R_d = (V_d + 0.5 * self.b * Omega_d) / self.r


    def sim_update_wheel_pos(self, clientID, left_motor_handle, right_motor_handle):
        """Get ang position of wheels from vrep, save it to vehicle class instance"""
        _, self.wheel_pos_L = simxGetJointPosition(clientID, left_motor_handle, simx_opmode_oneshot_wait)
        _, self.wheel_pos_R = simxGetJointPosition(clientID, right_motor_handle, simx_opmode_oneshot_wait)


    def sim_update_wheel_speeds(self, clientID, left_motor_handle, right_motor_handle):
        """Get ang speeds of wheels from vrep, save it to vehicle class instance"""
        _, self.omega_L = simxGetObjectFloatParameter(clientID, left_motor_handle, 2012, simx_opmode_oneshot_wait)
        _, self.omega_R = simxGetObjectFloatParameter(clientID, right_motor_handle, 2012, simx_opmode_oneshot_wait)


    def dead_reckon(self):
        # update robot's position etc using dead reckoning (odometry)

        v_L = self.r * self.omega_L  # left wheel vel
        v_R = self.r * self.omega_R  # right wheel vel
        self.Omega = (v_R - v_L) / self.b  # robot ang vel
        self.heading = self.Omega * self.dt + self.heading_0  # robot orientation
        self.V = (v_L + v_R) / 2  # robot speed

        xdot = self.V * math.cos(self.heading)  # robot x vel
        ydot = self.V * math.sin(self.heading)  # robot y vel

        # Check for straight line motion (avoid div/zero error)
        if (self.Omega == 0):
            dx = xdot * self.dt
            dy = ydot * self.dt
        else:
            dx = (self.V / self.Omega) * (math.sin(self.heading) - math.sin(self.heading_0))
            dy = -(self.V / self.Omega) * (math.cos(self.heading) - math.cos(self.heading_0))

        self.x = self.x_0 + dx
        self.y = self.y_0 + dy
        self.s = self.s_0 + math.sqrt(dx*dx + dy*dy)


    def dead_reckon_simple(self):
        # update robot's position etc using simplified dead reckoning (
            odometry)
        s_L = self.r * (utils.wrap_angle(self.wheel_pos_L - self.wheel_pos_L_0))  # left wheel travel
        s_R = self.r * (utils.wrap_angle(self.wheel_pos_R - self.wheel_pos_R_0))  # right wheel travel
        s = (s_L + s_R) / 2
        self.heading = (s_R - s_L) / self.b + self.heading_0
        self.x = s * math.cos(self.heading) + self.x_0
        self.y = s * math.sin(self.heading) + self.y_0
        self.s = self.s_0 + s


    def sim_dead_reckon(self, clientID, vehicle_handle):
        """Simulate perfect dead reckoning getting the exact vehicle position and angle from vrep"""
        error_code, position = simxGetObjectPosition(clientID, vehicle_handle, -1, simx_opmode_oneshot_wait)  # local
        utils.check_error_code(clientID, error_code)
        error_code, orientation = simxGetObjectOrientation(clientID, vehicle_handle, -1, simx_opmode_oneshot_wait)  # local
        utils.check_error_code(clientID, error_code)

        self.x = position[0]
        self.y = position[1]
        self.heading= orientation[2]

        dx = self.x - self.x_0
        dy = self.y - self.y_0
        self.s = self.s_0 + math.sqrt(dx*dx + dy*dy)

    def save_last_timestep_values(self):
        """Copy current position and orientation values to the last_timestep variables"""
        self.x_0 = self.x
        self.y_0 = self.y
        self.heading_0 = self.heading
        self.s_0 = self.s
        self.wheel_pos_L_0 = self.wheel_pos_L
        self.wheel_pos_R_0 = self.wheel_pos_R


######################################################################
#                                                                    #
#   TETHER AND WINCH INTERFACES                                      #
#                                                                    #
######################################################################
#                                                                    #
#                                               Tip                  #
#              Winch                         Interface               #
#              .----.          Tether          .--.                  #
#              |    |------------------------->|  |                  #
#       .-----------.                         .-----------.          #
#       | Follower  |                         | Leader    |          #
#       |  _     _  |   ---->                 |  _     _  |  ---->   #
#       '-(_)---(_)-'                         '-(_)---(_)-'          #
#                                                                    #
######################################################################

class Winch(object):
    """Winch: interface between follower vehicle and a tether """
    def __init__(self):
        self.l_m = 0.0  # Measured tether length
        self.theta_m = 0.0  # Measured tether angle
        self.offset_x = 0.0  # winch offset from vehicle center
        self.offset_y = 0.0
        self.offset_ang = 0.0 # orientation offset (i.e. if winch is
            mounted at an angle)
        self.tethers = []
    def add_tether(self, tether):
        self.tethers.append(tether)

    def sim_update_winch_data(self, clientID, tether_base_handle,
       tether_tip_handle):
        "Return tether length and angle from winch"
        error_code, base_pos = simxGetObjectPosition(clientID,
            tether_base_handle, -1, simx_opmode_oneshot_wait)
        utils.check_error_code(clientID, error_code)

        error_code, tip_pos = simxGetObjectPosition(clientID,
            tether_tip_handle, -1, simx_opmode_oneshot_wait)
        utils.check_error_code(clientID, error_code)

        self.l_m = utils.vec_dist_3d(base_pos[0], base_pos[1], base_pos[2],
             tip_pos[0], tip_pos[1], tip_pos[2])
        theta_abs = math.atan2(tip_pos[1] - base_pos[1], tip_pos[0] -
            base_pos[0])
        error_code, base_orientation = simxGetObjectOrientation(clientID,
            tether_base_handle, -1, simx_opmode_oneshot_wait)
        utils.check_error_code(clientID, error_code)

        gamma = base_orientation[2]

        self.theta_m = theta_abs - gamma

        self.theta_m = utils.wrap_angle(self.theta_m)


class TipInterface(object):
    """Tip_interface: interface between tether and a leader vehicle"""
    def __init__(self):
        self.offset_x = 0.0  # winch offset from vehicle center
        self.offset_y = 0.0
        self.offset_ang = 0.0 # orientation offset (i.e. if winch is
            mounted at an angle)
        self.tethers = []
    def add_tether(self, tether):
        self.tethers.append(tether)


class SimTether(object):
    """Tether for connecting a leader and follower
    This is a sim object; our follower model only knows the measured values
         from the winch, not the real values"""
    def __init__(self):
        self.l = None  # tether length
        self.Theta = None  # tether angle (global ref frame)
```

Bibliography

[1] World Health Organization. (2012). Top 10 causes of death: fact sheet no. 310, [Online]. Available: `http://www.who.int/mediacentre/factsheets/fs310/en/`.

[2] K. Kida, "Home Oxygen Therapy in Japan : Clinical application and considerations for practical implementation," *Japan Medical Association Journal*, vol. 138, no. 12, pp. 99–104, 2011.

[3] R. A. Pauwels, A. S. Buist, P. M. Calverley, C. R. Jenkins, and S. S. Hurd, "Global strategy for the diagnosis, management, and prevention of chronic obstructive pulmonary disease," *American journal of respiratory and critical care medicine*, vol. 163, no. 5, pp. 1256–1276, 2001.

[4] BUPA. (2013). Copd symptoms, causes and treatments, [Online]. Available: `http://www.bupa.co.uk/health-information/directory/c/copd`.

[5] G. W. S. Hoo. (2014). Noninvasive ventilation, [Online]. Available: `http://emedicine.medscape.com/article/304235-overview` (visited on 09/20/2014).

[6] N. P. Care, *Home oxygen service–assessment and review. good practice guide*, 2011.

[7] T. Tada, F. Hashimoto, Y. Matsushita, Y. Terashima, T. Tanioka, and I. Nagamine, "Study of life satisfaction and quality of life of patients receiving home oxygen therapy," *Journal of Medical Investigation*, vol. 50, no. 1/2, pp. 55–63, 2003.

[8] J. P. Janssens, T. Rochat, J. G. Frey, N. Dousse, C. Pichard, and J. M. Tschopp, "Health-related quality of life in patients under long-term oxygen therapy: a home-based descriptive study.," *Respiratory medicine*, vol. 91, no. 10, pp. 592–602, Nov.

1997, ISSN: 0954-6111. [Online]. Available: `http://www.ncbi.nlm.nih.gov/pubmed/9488892`.

[9] D. A. Norman, *The design of everyday things.* Basic books, 2002.

[10] J. Shao, G. Xie, J. Yu, and L. Wang, “Leader-following formation control of multiple mobile robots,” in *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, IEEE, 2005, pp. 808–813.

[11] J. Shao, G. Xie, and L. Wang, “Leader-following formation control of multiple mobile vehicles,” *IET Control Theory & Applications*, vol. 1, no. 2, pp. 545–552, 2007.

[12] S. Carpin and L. E. Parker, “Cooperative leader following in a distributed multi-robot system,” in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, IEEE, vol. 3, 2002, pp. 2994–3001.

[13] C. Samson and K. Ait-Abderrahim, “Mobile robot control. part 1: feedback control of nonholonomic wheeled cart in cartesian space,” 1990.

[14] C. Samson and K. Ait-Abderrahim, “Feedback control of a nonholonomic wheeled cart in cartesian space,” in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, IEEE, 1991, pp. 1136–1141.

[15] C. Samson, “Control of chained systems application to path following and time-varying point-stabilization of mobile robots,” *Automatic Control, IEEE Transactions on*, vol. 40, no. 1, pp. 64–77, 1995.

[16] A. Ohya and T. Munekata, “Intelligent escort robot moving together with human: interaction in accompanying behavior,” in *Proceedings of 2002 FIRA Robot Congress*, 2002.

[17] A. Ohya, Y. Nagumo, and Y. Gibo, “Intelligent escort robot moving together with human-methods for human position recognition,” in *Joint 1st International Confer-*

ence on Soft Computing and Intelligent Systems and 3rd International Symposium on Advanced Intelligent Systems (SCIS & ISIS 2002), 24B5-2, 2002.

[18] Y. Nagumo and A. Ohya, "Human following behavior of an autonomous mobile robot using light-emitting device," in *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, IEEE, 2001, pp. 225–230.

[19] Yaskawa. (2013). Yaskawa roboporter, [Online]. Available: `http://www.yaskawa.co.jp/en/company/csr2010/09.htm`.

[20] Panasonic. (2008). Panasonic robotic porter, [Online]. Available: `http://robot.watch.impress.co.jp/cda/news/2008/02/28/923.html`.

[21] T. Yoshimi, M. Nishiyama, T. Sonoura, H. Nakamoto, S. Tokura, H. Sato, F. Ozaki, N. Matsuhira, and H. Mizoguchi, "Development of a person following robot with vision based target detection," in *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 5286–5291. DOI: `10.1109/IROS.2006.282029`.

[22] H. Takemura, N. Zentaro, and H. Mizoguchi, "Development of vision based person following module for mobile robots in/out door environment," in *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, Dec. 2009, pp. 1675–1680. DOI: `10.1109/ROBIO.2009.5420414`.

[23] Segway Japan. (2013). Person following robot and simulator, [Online]. Available: `http://www.segway-japan.co.jp/robot-soft/casestudy/tracking.html`.

[24] Stewart Golf. (2014). X9 follow, [Online]. Available: `http://www.stewartgolf.com/x9follow/`.

[25] E. Mumm, S. Farritor, P. Pirjanian, C. Leger, and P. Schenker, "Planetary cliff descent using cooperative robots," *Autonomous Robots*, vol. 16, no. 3, pp. 259–272, 2004.

[26] M. Krishna, J. Bares, and E. Mutschler, “Tethering system design for dante ii,” in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, IEEE, vol. 2, 1997, pp. 1100–1105.

[27] T. Takayama and S. Hirose, “Development of “Souryu I and II” – connected crawler vehicle for inspection of narrow and winding space,” *Journal of Robotics and Mechatronics*, 2003.

[28] M. Arai, Y. Tanaka, S. Hirose, H. Kuwahara, and S. Tsukui, “Development of “Souryu-IV” and “Souryu-V:” serially connected crawler vehicles for in-rubble searching operations,” *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 31–65, 2008. DOI: `http://dx.doi.org/10.1002/rob.v25:1/2`.

[29] M. Nohmi, D. N. Nenchev, and M. Uchiyama, “Trajectory planning and feedforward control of a tethered robot system,” in *Intelligent Robots and Systems’ 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, IEEE, vol. 3, 1996, pp. 1530–1535.

[30] S. Na, H.-S. Ahn, Y.-C. Lee, and W. Yu, “A tethering device for mobile robot guidance,” *International Journal of Advanced Robotic Systems*, vol. 6, no. 2, 2009.

[31] H.-S. Ahn, S.-I. Na, Y.-C. Lee, and W. Yu, “A controller design of a tethered-robot guiding system,” in *Proceedings of Int. Conf. On Ubiquitous Robots and Ambient Intelligence*, Citeseer, 2006, pp. 43–46.

[32] E. Fukushima and S. Hirose, “Hyper-tether, a proposal,” in *Proceedings of the 15th Robotics Society of Japan Conference*, 1997, pp. 453–454.

[33] K.-h. Kim, J.-u. Chu, and Y.-j. Lee, “Steering-by-Tether and Modular Architecture for Human-Following Robot,” *2006 SICE-ICASE International Joint Conference*, pp. 340–343, 2006. DOI: `10.1109/SICE.2006.315704`. [Online]. Available: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4108852`.

[34] R. Kimura, K. Tsubata, M. Kaneko, C. Ishibiki, S. Itai, and Y. Miwa, "Development of a burden carrier using the string as an interface [japanese]," *JSME annual meeting*, vol. 2005, no. 4, pp. 359–360, Sep. 2005. [Online]. Available: `http://ci.nii.ac.jp/naid/110006191051/en/`.

[35] J. Ogawa, R. Kimura, C. Ishibiki, S. Itai, and Y. Miwa, "The burden carrier utilizing string redundancy for following human [japanese]," *Proceedings of the 2006 JSME Conference on Robotics and Mechatronics*, vol. 2006, 2006. [Online]. Available: `http://ci.nii.ac.jp/naid/110008694213/en/`.

[36] A. Terashima, C. Ishibiki, S. Itai, and Y. Miwa, "The burden carrier utilizing redundancy of a string [japanese]," *Proceedings of SI2006, Conference on Systems Integration*, pp. 1102–1103, 2006.

[37] E. F. Fukushima, N. Kitamura, and S. Hirose, "A new flexible component for field robotic systems," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, IEEE, vol. 3, 2000, pp. 2583–2588.

[38] T. Kamegawa, E. Fukushima, and S. Hirose, "Research on Hyper-Tether. No.4: Cooperative steering control for multiple robots connected in series," in *JSME Robomec*, (in Japanese), 1999.

[39] P. Debenest, E. Fukushima, and S. Hirose, "Proposal for automation of humanitarian demining with buggy robots," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, Oct. 2003, 329–334 vol.1. DOI: `10.1109/IROS.2003.1250649`.

[40] G. Endo, A. Tani, E. Fukushima, S. Hirose, M. Iribe, and T. Takubo, "Study on a practical robotic follower to support daily life - development of a mobile robot with hyper-tether for home oxygen therapy patients," in *System Integration, 2009. SII 2009. IEEE/SICE International Symposium on*, Jan. 2009, pp. 71–76. DOI: `10.1109/SI.2009.5384552`.

[41] Bullet Physics. (2014). Bullet physics library overview and specification, [Online]. Available: `http://bulletphysics.org/wordpress/`.

[42] E. Rohmer, S. Singh, and M. Freese, "V-rep: a versatile and scalable robot simulation framework," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, New Jersey, USA, Nov. 2013, pp. 1321–1326.

[43] G. Endo, A. Tani, E. F. Fukushima, S. Hirose, M. Iribe, and T. Takubo, "Development of a Leader-following Mobile Robot to Support Home Oxygen Therapy Patients," *Journal of the Robotics Society of Japan*, vol. 30, no. 8, pp. 779–787, 2012, ISSN: 0289-1824. DOI: `10.7210/jrsj.30.779`.

[44] Motion Analysis. (2013). Motion capture products, [Online]. Available: `http://www.motionanalysis.com/`.

[45] G. Endo, Y. Iemura, E. F. Fukushima, S. Hirose, M. Iribe, R. Ikeda, K. Onishi, N. Maeda, T. Takubo, and M. Ohira, "Study on a practical robotic follower to support home oxygen therapy patients—questionnaire-based concept evaluation by the patients," in *Rehabilitation Robotics (ICORR), 2013 IEEE International Conference on*, IEEE, 2013, pp. 1–5.

[46] R. Green, "Faster math functions," *Sony Computer Entertainment America Developers Workshop*, 2003. [Online]. Available: `http://www.research.scea.com/gdc2003/fast-math-functions_p1.pdf`.

[47] S. Little, M. Elkholy, G. Chalmers, A. Farouk, K. Patel, and N. Thomson, "Predictors of nocturnal oxygen desaturation in patients with copd," *Respiratory medicine*, vol. 93, no. 3, pp. 202–207, 1999.

[48] Tokyo Institute of Technology. (2015). Strategic innovation program (sip), [Online]. Available: `http://www.sms.titech.ac.jp/sip/`.

[49] PIO Ota. (2015). Ota-city industrial promotion organization, [Online]. Available: `http://pio-ota.jp/`.

[50] Yamasyou Co. Ltd. (2015). Yamasyou company information, [Online]. Available: `http://yamasyou.co.jp/`.

[51] Japan Airport Terminals Co. Ltd. (2015). Haneda aiport operations, [Online]. Available: `http://www.tokyo-airport-bldg.co.jp/company/en/`.

[52] Tokyo Institute of Technology. (2015). Strategic innovation program: cart robot project, [Online]. Available: `http://www.sms.titech.ac.jp/sip/projects.html#projects01`.

[53] Yamaha Motors. (2014). Joy unit x wheel pack, [Online]. Available: `http://www.yamaha-motor.co.jp/wheelchair/lineup/joyunitx/`.