Mechatronics Design and Energy-Efficient Navigation of a Heavy-Duty Omni-Directional Mecanum Autonomous Mobile Robot

Li Xie

A THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN ENGINEERING,
THE UNIVERSITY OF AUCKLAND, 2018.

Abstract

The Ilon Mecanum wheel is one of the practical omnidirectional wheel designs in industry but it consumes excessive energy due to its motion inefficiency. Omnidirectional autonomous mobile robots without any nonholonomic constraints can reach the same goal pose from the same start pose via too many feasible trajectories, each consuming different amounts of energy. It is both economically beneficial and academically interesting to optimize the omnidirectional trajectories of holonomic Mecanum autonomous mobile robots for energy minimization. The aim of this PhD research was to first develop a heavy-duty omnidirectional Mecanum mobile robot that can perform fully functional autonomous navigation for research purposes, and then to realize energy-efficient autonomous navigation on the designed robot via both online and offline optimal motion planning research studies.

A heavy-duty omnidirectional Mecanum robot platform was designed and developed from the beginning. The robotic control system architecture that allows the robot to perform autonomous navigation was realized by fusing an industrial automation control system and open-source autonomous navigation technologies. Then an experimentally validated novel energy consumption model of the four-wheel omnidirectional Mecanum mobile robot was proposed based on a comprehensive understanding of the kinematics, dynamics and energy flow of the robot. In order to realize energy-efficient autonomous navigation, both online and offline optimal planners were presented in this research. The proposed online planner extended the Dynamic Window Approach (DWA) by means of a new energy-optimal omnidirectional velocity search technique for the purposes of optimizing motional power consumption and reducing energy consumption for autonomous navigation. The offline planner proposed a method of trajectory generation that interpolates the task-space trajectory of the omnidirectional robot in polynomial spline functions and then searches for the energy-optimal trajectories using genetic algorithms. Both proposed online and offline planning methods were experimentally validated.

Acknowledgements

Foremost, I would like to express my deep gratitude to and acknowledge the support and guidance of my supervisors Prof. Weiliang Xu and Dr. Karl Stol. I am very grateful for their help as patient and wise advisors and mentors in my PhD study.

I also appreciate the University of Auckland for granting me the UoA Doctoral Scholarship to financially support my living costs and consistently provide research funding. The awarded opportunity of studying the PhD course at the University of Auckland brought me to an important period of my life.

Besides my supervisors, I would like to particularly thank Christian Scheifele and Christian Henkel for their peerless contributions to this project. The designed robot – AuckBot – would not be the same without your quality work. My appreciation also goes to other contributors who have made direct efforts to this project: Theng Kiat Chua, Wolfgang Herberger, Chaojie Hu, Christopher Jarrett and Marcel Pelzer.

I give thanks to family and friends who have been part of my life journey. Appreciate everyone that appears in my life. Most importantly, I would like to thank my father, Chaohuai Xie, and mother, Chaohua Li, for everything.

In the end, I wish for world peace.

Table of Contents

List of Figures

List of Tables

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

School of Graduate Studies
AskAuckland Central
Alfred Nathan House
The University of Auckland
Tel: +64 9 373 7599 ext 81321
Email: postgradinfo@auckland.ac.nz

Co-Authorship Form

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work**. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 3: Section 3.3, 3.4, 3.6, 3.8, 3.9 are extracted from a co-authored work - Li Xie, Christian Scheifele, Weiliang Xu, and Karl A. Stol. "Heavy-duty omni-directional Mecanum-wheeled robot for autonomous navigation: System development and simulation realization." In Mechatronics (ICM), 2015 IEEE International Conference on, pp. 256-261. IEEE, 2015.

Nature of contribution by PhD candidate	The PhD candidate (Li Xie) is the lead designer and developer in this published robotic design work, engaged in every part of the robotic design and development, and wrote the majority of the published paper text.
Extent of contribution by PhD candidate (%)	70%

CO-AUTHORS

Name	Nature of Contribution
Christian Scheifele	As a temporary designer and developer in the robot design work, his main contributions include engaging in proposing the idea of the robotic control system architecture, implementing the data communication within the robotic control system and implementing a robotic simulation system
Weiliang Xu	As the main supervisor of the PhD candidate, Weiliang Xu supervised the robot design project and research project, proofread and edited the paper text.
Karl Stol	As the co-supervisor of the PhD candidate, proofread and edited the paper text.

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

Name	Signature	Date
Karl Stol		8/3/18
Christian Scheifele		08.03.2018
Peter Xu		09/03/18

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

School of Graduate Studies
AskAuckland Central
Alfred Nathan House
The University of Auckland
Tel: +64 9 373 7599 ext 81321
Email: postgradinfo@auckland.ac.nz

Co-Authorship Form

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work**. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 4 is extracted from a co-authored work - Li Xie, Wolfgang Herberger, Weiliang Xu, and Karl A. Stol. "Experimental validation of energy consumption model for the four-wheeled omnidirectional Mecanum robots for energy-optimal motion control." In Advanced Motion Control (AMC), 2016 IEEE 14th International Workshop on, pp. 565-572. IEEE, 2016.

Nature of contribution by PhD candidate	The PhD candidate (Li Xie) is the main contributor of this paper, who came up with the methodology, conducted the experiments and analyzed the results, wrote all the text of the paper
Extent of contribution by PhD candidate (%)	85%

CO-AUTHORS

Name	Nature of Contribution
Wolfgang Herberger	Assisted conducting the experiments.
Weiliang Xu	As the main supervisor of the PhD candidate, Weiliang Xu supervised the research, proofread and edited the paper text.
Karl Stol	As the co-supervisor of the PhD candidate, proofread and edited the paper text.

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

Name	Signature	Date
Karl Stol		8/3/18
Wolfgang Herberger		8/3/18
Peter Xu		09/03/18

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

School of Graduate Studies
AskAuckland Central
Alfred Nathan House
The University of Auckland
Tel: +64 9 373 7599 ext 81321
Email: postgradinfo@auckland.ac.nz

Co-Authorship Form

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work**. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 5 is extracted from a co-authored work-Li Xie, Christian Henkel, Karl Stol, and Weiliang Xu. "Power-minimization and energy-reduction autonomous navigation of an omnidirectional Mecanum robot via the dynamic window approach local trajectory planning." International Journal of Advanced Robotic Systems 15, no. 1 (2018): 1729881418754563.	
Nature of contribution by PhD candidate	The PhD candidate (Li Xie) proposed a modified methodology based on the co-author - Christian Henkel's methodology from another published work, conducted new experiments and analyzed the results, wrote all the text of the paper.
Extent of contribution by PhD candidate (%)	70%

CO-AUTHORS

Name	Nature of Contribution
Christian Henkel	Christian Henkel originally came up with the idea of using Dynamic Window Approach for energy-efficient local path planning and published his own paper about it - Christian Henkel, Alexander Bubeck, and Weiliang Xu. "Energy efficient dynamic window approach for local path planning in mobile service robotics." IFAC-PapersOnLine 49, no. 15 (2016): 32-37. In this paper 'Power-minimization and energy-reduction autonomous navigation of an omnidirectional Mecanum robot via the dynamic window approach local trajectory planning', Christian Henkel engaged in discussing the results, and proofread and edited the paper.
Weiliang Xu	As the main supervisor of the PhD candidate, Weiliang Xu supervised the research, discussed the results, proofread and edited the paper text.
Karl Stol	As the co-supervisor of the PhD candidate, discussed the results, proofread and edited the paper text.

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

Name	Signature	Date
Christian Henkel		7.3.17
Karl Stol		8/3/18
Peter Xu		09/03/18

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

School of Graduate Studies
AskAuckland Central
Alfred Nathan House
The University of Auckland
Tel: +64 9 373 7599 ext 81321
Email: postgradinfo@auckland.ac.nz

Co-Authorship Form

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work**. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 6 is extracted from an unsubmitted co-authored work - Li Xie, Karl Stol, Weiliang Xu, "Energy-Optimal Motion Trajectory of an Omnidirectional Mecanum Wheeled Robot via Polynomial Functions"

Nature of contribution by PhD candidate	The PhD candidate (Li Xie) engaged in proposing, improving and discussing the methodology of this research paper, conducted the experiments and analyzed the results, wrote all the text of the paper.
Extent of contribution by PhD candidate (%)	75%

CO-AUTHORS

Name	Nature of Contribution
Weiliang Xu	As the main supervisor of the PhD candidate, Weiliang Xu originally proposed the methodology, enaged in improving the methodology, supervised the research, discussed the results, proofread and edited the paper text.
Karl Stol	As the co-supervisor of the PhD candidate, discussed the results, proofread and edited the paper text.

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

Name	Signature	Date
Karl Stol		8/3/18
Peter Xu		9/03/18

Chapter 1. Introduction

Robotics is changing the world all the time. It is changing the way people live and work, and it is also changing the way industries process and manufacture.

1.1. Background

Autonomous Navigation Mobile Robotics

Automation technology plays an important role in industry, and automation systems are widely used in many tasks. Independent mobility of automation systems brings additional value to the production process so that the number of autonomous mobile robots is increasing across all industries. Their ability to autonomously reach several goal positions and perform functions as required is in demand.

Autonomous navigation mobile robots can constantly sense the surrounding environment, know its position, react to unforeseen situations and navigate towards a goal pose. The involved key technologies in an autonomous navigation system include perception, localization, navigation and motion planning. Research into these technologies has been conducted for decades. To design and develop an autonomous navigation mobile robot from scratch, a fully functional autonomous navigation system is easily implemented because many open source software providers have available collections of tools, libraries and packages. Instead, the challenges focus on robotic hardware development, locomotion control systems, the interface between the navigation system and the locomotion control system, and setup and configuration of the navigation system on the target robot.

Locomotion Mechanism - Omnidirectional Mecanum Wheel

The very first topic of mobile robotics is locomotion. Robotic locomotion highly determines the application range of robots, and vice versa. The wheel-based locomotion mechanism is arguably the most widely utilized robotic locomotion due to its good efficiencies, stable balance, and simple implementation. Omnidirectional mobility that comes with certain types of locomotion mechanisms, is a useful ability for mobile robots. Holonomic drives based on omnidirectional wheels are more manoeuvrable, and thus more convenient to transport in confined, congested and highly dynamic environments than the conventional differential drive.

However, omnidirectional wheels tend to face more difficulties in controllability and/or stability than conventional wheels.

The Mecanum wheel, as one of the practical omnidirectional wheel designs, was invented in 1975 by Ilon in Sweden. The patent of the Mecanum wheel was bought by the US Navy in 1975 for military purposes. The company Airtrax then purchased the rights of the Mecanum wheel patent in 1997 and Airtrax was the first company to commercialize omnidirectional industrial forklifts. Recently the company KUKA developed heavy-duty Mecanum-based autonomous mobile platforms and robots. The Mecanum wheel has the advantages of high load capacity and simple direction control over other omnidirectional wheel designs, while the complex roller design of the Mecanum wheel leads to random slippage, high-speed vibration and motion inefficiency.

Motivation of Research

It would be a great economic benefit to industry to reduce the energy consumption of the Mecanum robots. The Mecanum wheel's omnidirectional mobility is favoured for industrial robots. It has been applied in factory workshops, warehouses and hospitals. Sometimes it is even the only option for special situations. More importantly, its high load capacity contributes to heavy and unusual payload applications such as heavy-duty forklifts, omnidirectional wheel chairs, military lift trucks, et cetera. However, its inefficient kinetic energy use increases the energy consumption of the Mecanum robot, especially for heavy-duty tasks.

Filling a research gap was proposed in this PhD study. Energy-efficient autonomous navigation via optimal trajectory planning had been researched very little for omnidirectional mobile robots. As a holonomic drive, an omnidirectional autonomous Mecanum mobile robot can reach the same goal pose through various paths and via different velocity profiles. The low energy efficiency of the Mecanum wheel urgently requires optimizing the omnidirectional trajectories of the holonomic Mecanum mobile robots for energy reduction. Thus, it is both economically beneficial and academically interesting to reduce the energy consumption of the autonomous Mecanum mobile robot by optimizing its trajectories.

1.2. Project Aim and Objectives

The aim of this PhD research project was to first develop a heavy-duty omnidirectional Mecanum mobile robot that performs fully functional autonomous navigation for research

purposes, and to then realize energy-efficient autonomous navigation on the robot via both online and offline optimal motion planning algorithms. The project aim was realized through the completion of the following four objectives:

- Mechatronics Design of a Heavy-Duty Omnidirectional Mecanum Autonomous Robot

  The first objective was developing a four-wheel Mecanum autonomous mobile robot for research purposes. In this PhD research project, the purpose-built robot was mainly used to validate the proposed algorithms in the experiments. The design requirements and manufacturing standards were set up so that the Mecanum robot can be used as an autonomous warehouse forklift prototype in future projects. The robotic development consists of three main parts: robotic hardware design, locomotion control system design and autonomous navigation system implementation. The robotic platform was manufactured by mechatronics design including mechanical design, electric design and servomechanism design. The locomotion control system was designed to govern the omnidirectional motions of the robot, based on Beckhoff Automation technology. The cutting-edge autonomous navigation technology from the open-source community was implemented and configured on the robot. This robot was ready to conduct experiments not only for validating energy-optimal motion planning algorithms in this PhD project, but also can be used for any other future autonomous navigation research.

- Energy Consumption Model of the Four-Wheeled Omnidirectional Mecanum Robot

  It is a common approach nowadays to start an energy-efficient motion planning problem by building an energy model for the target robot. The second objective was to derive an energy consumption model of the Mecanum robot. The omnidirectional motions of the Mecanum drive are involved with complex mechanisms that are closely related to the energy consumption of the robot. Based on a thorough understanding of the Mecanum wheel's dynamics and the energy consumption analysis of the mobile robot, the model was able to accurately estimate the Mecanum robot's energy consumption of following any given trajectories for the latter energy-efficient motion planning research.

- Energy-Optimal Online Local Trajectory Planning for Autonomous Navigation

  Autonomous navigation is an important function for a modern industrial mobile robot. The online local trajectory planner is a key component of an autonomous navigation

system because the autonomous mobile robot must be able to avoid unexpected obstacles. The third objective of this PhD project was to achieve energy-efficient autonomous navigation via an optimal online local trajectory planner. The proposed local trajectory planning method both reduced overall energy consumption in the global sense for autonomous navigation, and reactively avoided unexpected obstacles.

- Energy-Optimal Offline Global Motion Planning of the Omnidirectional Robots

  In statically known environments, the Mecanum-wheeled industrial mobile service robots execute repetitive motion tasks. The fourth objective was to find an offline global motion trajectory planning method for such applications. Given the path points including a stationary initial pose and a stationary goal pose with or without a sequence of via points, the proposed planner globally found the optimal task-space trajectories that minimizes the energy consumption of the Mecanum mobile robot.

1.3. Structure of the Thesis

The most relevant literature is summarized in Chapter 2. The details of designing the heavy-duty autonomous four-wheel omnidirectional Mecanum mobile robot is reported in Chapter 3. A conference paper based on the robotic control system architecture development was presented at the IEEE International Conference on Mechatronics (ICM) in 2015 [1]. The completion of the second objective – an Energy Consumption Model of a Four-Wheeled Omnidirectional Mecanum Robot – is described in Chapter 4. A conference paper based on the proposed energy consumption model was presented in the IEEE 14th International Workshop on Advanced Motion Control (AMC) in 2016 [2]. Chapter 5 details the proposed technique of extending the Dynamic Window Approach for completing the third objective – Energy-Optimal Online Local Trajectory Planning for Autonomous Navigation – a journal paper has been accepted to be published in the International Journal of Advanced Robotic Systems [3]. Chapter 6 summarizes an approach using a polynomial-based trajectory generation technique to achieve the fourth objective – Energy-Optimal Offline Global Motion Planning of the Omnidirectional Robots. Based on the research of this objective, a journal paper was submitted for peer review. This PhD thesis is concluded, and future work is suggested in Chapter 7.

Chapter 2. Literature Review

In this chapter, the most relevant literature is reviewed and summarized. The key topics include the Mecanum wheel, the Mecanum robot design, the current state of the art of the autonomous navigation system, existing energy consumption modelling technique and existing energy-efficient motion planning methods.

2.1. Mecanum Wheel and Four-Wheeled Mecanum Mobile Robot

This section focuses on the introduction of the Mecanum wheel. It starts with background information about omnidirectional wheels. Then applications, drawbacks, kinematics, dynamics and control methods of the Mecanum wheel are properly described.

2.1.1 Omnidirectional Wheel Applications

Omnidirectional manoeuvrability is an outstanding ability for mobile robots to conveniently transport in congested, confined and highly dynamic environments. Mobile platforms need a good manoeuvrability to conveniently move in tight spaces and easily dodge obstacles. The locomotion design for a mobile robot is mainly selected by its working environment. The conventional wheel is an extremely popular locomotion mechanism because it is efficient, stable and easy to implement.

This makes omnidirectional wheels very useful because they provide a complete manoeuvrability that conventional wheels do not [4]. Without non-holonomic constraints, the motion trajectories of omnidirectional mobile platforms are no longer exclusive to straight trajectories and limited types of curved trajectories. Omnidirectional motions such as crab-like sideway motions, diagonal motions and spin-in-place motions can be exclusively performed by the omnidirectional mobile drive. These omnidirectional motions can effectively simplify the non-holonomic motion trajectories, e.g. the sideways motion is a one-step approach to overcome the excessive parallel parking procedures that may be the only option for differential drive platforms in certain situations. Limited by non-holonomic constraints, nonholonomic mobile platforms only follow straight-line paths and curved-line paths and have to orient their noses toward tangent directions of the path. The omnidirectional mobile drive is able to move instantaneously in any direction from any orientation. Basically, the omnidirectional drive can follow any shape of path. The effectiveness of the omnidirectional motion is superior to robotics.

The operating space is limited in many working environments, and the following omnidirectional mobile platforms/robots are applied. For example, Airtrax omnidirectional forklifts are used in warehouses; KUKA omnidirectional mobile platforms are found in the factory workshops; omnidirectional wheelchairs are used in hospitals and omnidirectional mobile robots were designed to attend the Robot Soccer World Cup RoboCup.

2.1.2 Omnidirectional Wheel Designs

According to the detailed studies of [5], there are two categories of omnidirectional drives: conventional omnidirectional wheel designs and special omnidirectional wheel designs. The conventional designs include active caster wheels and steered wheels. The special designs include the universal omni wheel and the Mecanum wheel. An overview is given in Table I to list the main advantages and disadvantages of each design.

Table I Advantages and Disadvantages of Omnidirectional Wheel Designs

	continuous wheel contact and robustness to floor irregularity	simple to control	simple mechanical design	heavy duty	No need of steering system
caster wheel	√	×	×	√	×
steering wheel	√	×	×	√	×
Omni wheel	×	√	√	×	√
Mecanum wheel	×	√	×	√	√

The special designs tend to utilize additional free-rolling rollers, which are attached to each wheel. These rollers alternately contact the ground, and this causes discontinuous wheel-ground contact. But both as special design, the Omni wheel and the Mecanum wheel are simple to control and operate for omnidirectional motions. In particular, a steering system is not necessary to both drives. The Mecanum wheel has a more complex mechanical design and is more taxing to manufacture. In return, the Mecanum wheel is more suitable to take heavy-duty tasks than the Omni wheel.

2.1.3 Ilon Mecanum Wheel

The Ilon Mecanum wheel is now one of the most practical omnidirectional wheel designs in industry [6]. The Mecanum wheel was invented in 1972-1973 by Swedish engineer and

inventor Bengt Erland Ilon. The wheel is named after the Swedish company Mecanum AB, where he was working when he invented it. Its patent was then bought by the US Navy in 1975 and used to design different scales of the mobile platforms for military purposes [7]. The Mecanum wheel, shown in Figure 1, is a near round-shaped wheel, which has an actively rotating hub (silver colour in Figure 1) with a certain number of passively rolling rollers (black rubber colour in Figure 1) mounted around the wheel circumference at an angle – conventionally, that angle is 45°. The geometry of the roller is a specially designed curvature so that the curved geometry makes the overall shape of the Mecanum wheel remain circular from the side view of the Mecanum wheel. More information about the geometry design of the rollers can be found in [8].

Figure 1 The Mecanum wheel used on the designed robot in this project. This heavy-duty Mecanum wheel has 7 rollers mounted around the wheel circumference. Each roller is attached to the wheel at both sides of the roller. The wheel has a weight of 8 kg, a radius of 0.11 m and a width of 0.15 m.

2.1.4 Omnidirectional Motions of the Mecanum Drive

Just as the conventional car wheel operates, the Mecanum wheel is rotated by active motor torque on the ground and a driving force is generated between the ground and the wheel due to the ground static friction. But the angled roller that is in contact with the ground, generates an angled driving force acting on the Mecanum wheel with respect to the ground. A Mecanum drive consists of most commonly four Mecanum wheels, or sometimes more for heavy-duty applications. Because each Mecanum wheel is independently controlled by an individual motor, the angled driving forces of the Mecanum wheels can be in opposite directions along the rotation axis of the rollers and have different amounts of magnitudes. The combination of every angled driving force from each Mecanum wheel can be in any direction [9]. The direction of the combined force can be controlled only by the velocities of the Mecanum wheel so the Mecanum robot is capable of omni-directional motion without wheel steering. Motion is achieved merely by implementing different velocity combinations of the Mecanum wheels, as

shown in Figure 2. Simply controlling the rotations of each Mecanum wheel without any complex steering system can easily operate, manage and control the omnidirectional motions of the drive.

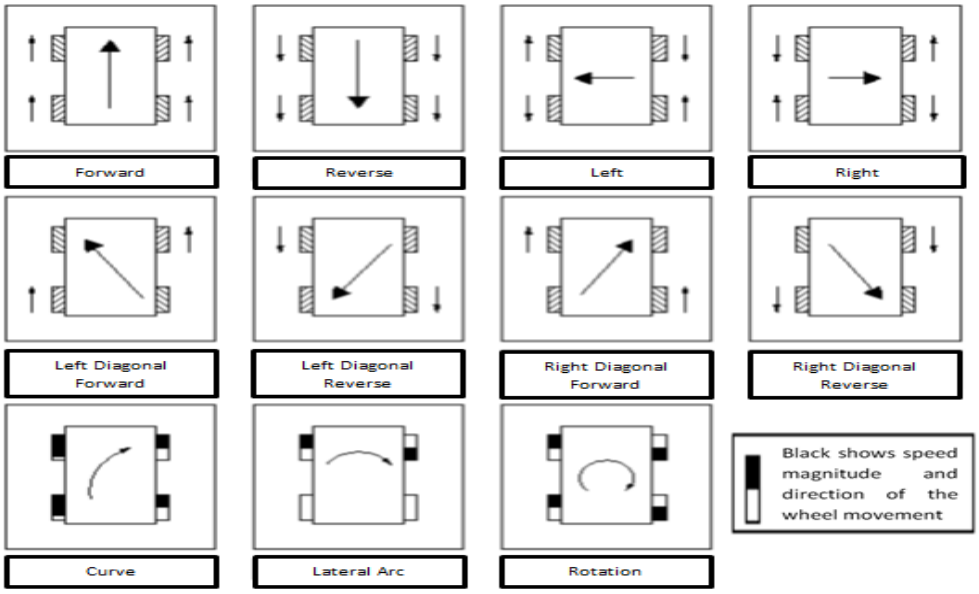


Figure 2 Combinational wheel actuations for general motion [10].

There are two types of the Mecanum wheel design, distinguished by the way the rollers are attached to the wheel hub. The first type attaches the roller to the hub at the centre of the roller, as shown in Figure 3. This type of roller is easily manufactured at low cost, and it is usually used as the kits of small-scale robots. The second type of Mecanum wheel supports the roller from both sides of the roller, as shown in Figure 1. This second type tends to have much better loading capability than the first type.

Compared with other omnidirectional wheels, the Mecanum wheel is a lot more suitable for taking heavy-duty tasks. This is the critical reason why the Mecanum wheel has much wider practical applications, especially in industry, than any other omnidirectional wheel. These days, most of the heavy-duty Mecanum wheels utilized in industry belong to the second type.

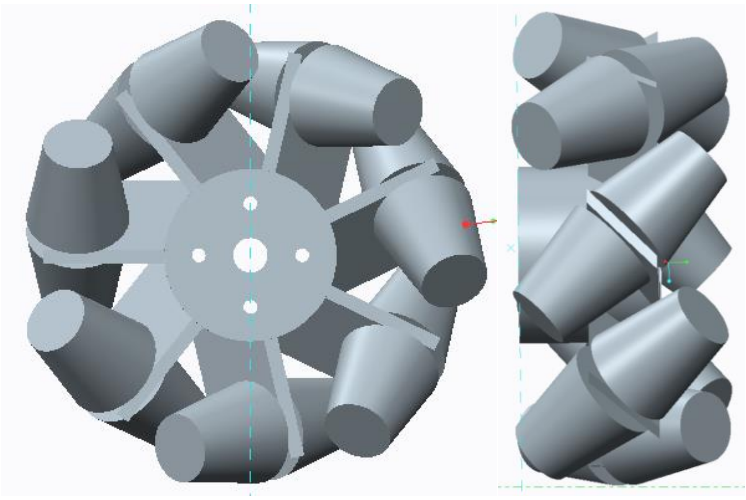
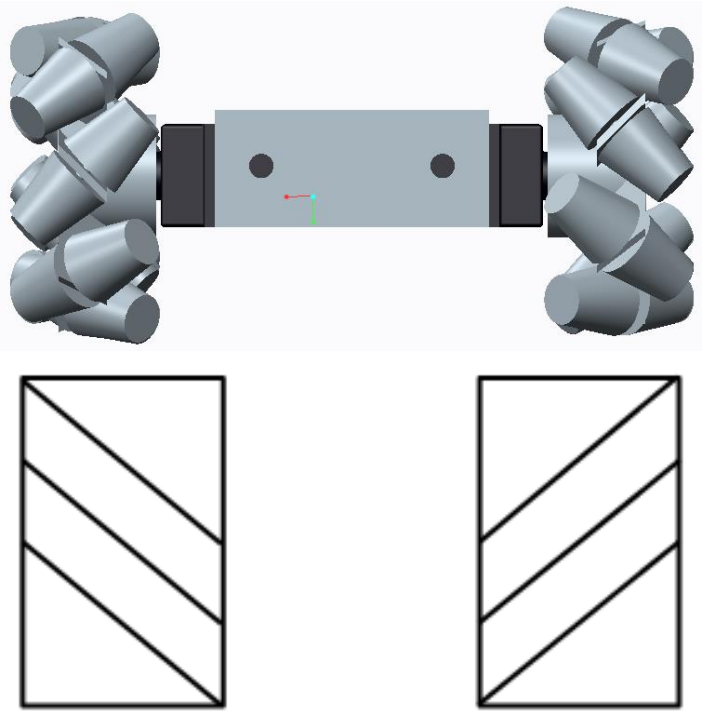


Figure 3 The Mecanum wheel from CAD. The roller attachment is at the centre of the roller.

In order to have a Mecanum drive properly working, there are must be two kinds of the Mecanum wheels on the Mecanum platform. As shown in Figure 2, the orientations of the rollers of the Mecanum wheels can be categorized into two kinds, which are distinguished by the orientation of the roller arrangement. Using the Mecanum wheel in Figure 3 as an example, the roller that is in contact with the ground is oriented like Figure 4 (a), if viewing the Mecanum wheel from above.

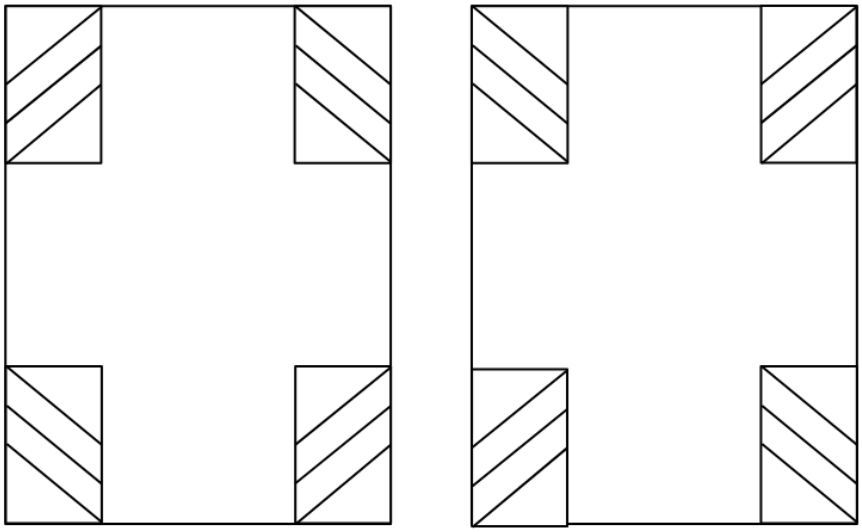


(a) (b)

Figure 4 The roller orientations of two Mecanum wheels, viewed from above.

Thus, there are two ways to organize the Mecanum wheel on the Mecanum robot and both ways are able to have the Mecanum drive working properly, as shown in Figure 5. The wheel arrangement in Figure 5 (a) is named an *'O'* arrangement and the wheel arrangement in Figure 5 (b) is named an *'X'* arrangement, in this thesis.

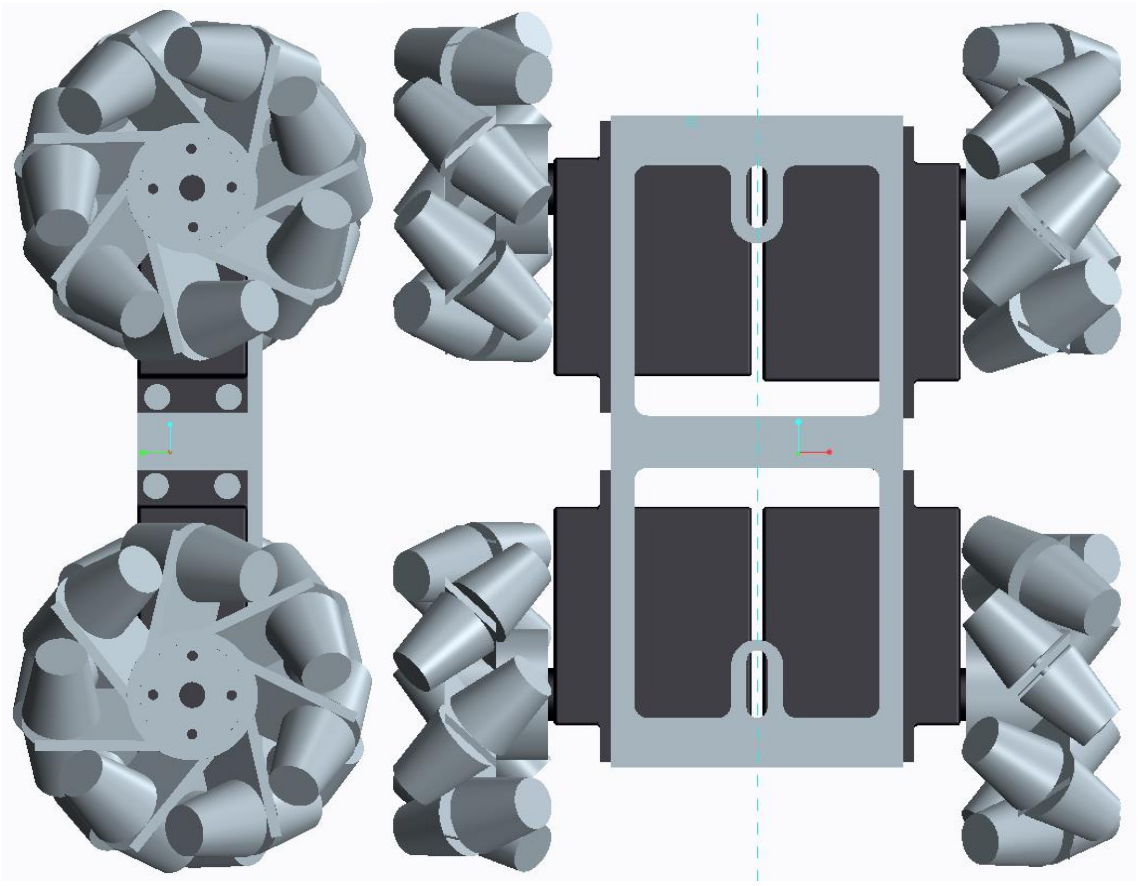
The illustrated thread of the roller is the one that is in contact with the ground, viewed from above. It is worth mentioning that, it is very important to clarify which roller is illustrated in the figures. This is because the roller on the top of the Mecanum wheel has exactly the opposite orientation to the roller contacting with the ground, on the same Mecanum wheel. Otherwise, this may cause much confusion. The Mecanum robot in this PhD project utilizes the *'O'* wheel arrangement of Figure 5 (a).



(a) *'O'* arrangement (b) *'X'* arrangement

Figure 5 *'O'* and *'X'* wheel arrangements on working Mecanum robots.

The *'O'* wheel arrangement of Figure 5 (a) is also shown in the CAD model of Figure 6 below for better illustration purposes. Viewing Figure 6 (b) from above, the roller arrangement seems to be *'X'* in Figure 5 (b). But it is important to remember that the roller on the top of the Mecanum wheel has exactly the opposite orientation to the roller contacting with the ground, on the same Mecanum wheel. Thus, the roller that is in contact with ground in Figure 6 (b) is *'O'* as in Figure 5 (a).



(a) (b)

Figure 6 One type of the Mecanum wheel arrangement viewed from above.

2.1.5 Mecanum Applications

The Mecanum wheel has a lot of applications in various fields, including military, industry, healthcare and academic. The Mecanum wheels have been applied in critical environments for the US Navy, NASA and ESA. Mecanum wheel-based products have been commercialized by companies such as Airtrax, KUKA, Broetje and MIAG for decades. The wheelchairs whose locomotion mechanisms utilize the Mecanum wheels serve for patients. Researchers from many academic institutions build their own Mecanum robots for research purposes. Many of the following Mecanum wheel applications cope with heavy-duty transportation tasks.

Military

As mentioned earlier, the patent of the Mecanum wheel was bought by the US Navy in 1975. The US Navy developed many different sizes of Mecanum-based vehicles, ranging from 30 kg up to 2000 kg. These vehicles are named by the US Navy as Omni-Directional Vehicles

(ODVs) and ODVs are used to transport items on military ships. A picture of a US Navy ODV is shown in Figure 7.

Figure 7 The US Navy Omni-Directional Vehicle ODV[1].

The National Aeronautics and Space Administration (NASA) is also interested in Mecanum wheel applications. The OmniBot Mobile Base [11, 12] is a project at the Kennedy Space Centre to develop remotely controlled Mecanum mobile vehicles operating in hazardous environments, as shown in Figure 8. The Kennedy Space Centre used to aim to test the OmniBot Mobile Base for autonomous mobile vehicles.

Figure 8 The NASA OmniBot Mobile Base[2].

[1] ODV: Omni-Directional Vehicle by the U.S. Navy. http://www.arrickrobotics.com/robomenu/odv.html.
[2] OmniBot: The NASA OmniBot From Kennedy Space Center. https://spinoff.nasa.gov/spinoff2000/ard8.htm.

NASA and the European Space Agency (ESA) are currently conducting ongoing exploration studies on the Mars Cruiser One project. The Mars Cruiser One [13] is a large Mecanum-wheel rover of 4550 kg weight and 4.5m × 9.5m × height 3.8m dimensions, designed by Architecture and Vision. It is designed to accommodate astronauts and to operate on the Moon and Mars in the future, as shown in Figure 9.

Figure 9 The concept design of the Mars Cruiser One[3].

Industry

Figure 10 Airtrax industrial forklifts SIDEWINDER ATX-3000[4].

The US Navy sold the rights of the Mecanum wheel patent to the company Airtrax in 1997, in order to together develop omnidirectional forklift trucks. The US Navy utilizes these forklifts

[3] Mars Cruiser One: Mars Cruiser One by NASA and ESA. http://www.architectureandvision.com/portfolio/011-marscruiserone-2007.
[4] SIDEWINDER ATX-3000: Airtrax industrial forklifts Sidewinder. http://www.vetexinc.com/vehicles/sidewinder.html.

on the decks of aircraft carriers. Airtrax was the first company to commercialize omnidirectional industrial forklifts. A model of Airtrax ATX300 is shown in Figure 10. Unlike the conventional forklift, which is usually equipped with relatively large front wheels to support front-loading goods, the Airtrax Mecanum forklift utilizes equal sizes of front and back wheels.

The company KUKA offers its omniMove drive technology [14], which is also based on the Mecanum wheel. KUKA autonomous mobile platforms are performing heavy-duty transportation tasks shown in Figure 11.

Figure 11 KUKA omniMove heavy-duty mobile platforms[5].

[5] OmniMove: KUKA mobile platforms. https://www.kuka.com/en-ch/products/mobility/mobile-platforms.

KUKA also has a variety of autonomous intelligent robots based on the omniMove technology, such as the KMR iiwa in Figure 12 (a), the KMR QUANTEC in Figure 12 (b), the KUKA moiros in Figure 12 (c) and so on. Based on the holonomic mobility of Mecanum wheels, robots are offered more functional flexibilities from the multiple-degrees-of-freedom robotic manipulators.

(a) (b) (c)

Figure 12 The KUKA Mecanum robots. (a) KMR iiwa. (b) KMR QUANTEC. (c) KUKA moiros[6].

MIAG applied the Mecanum wheeled Omnidrive Carrier System for material handling and innovative transport systems. Figure 13 shows the MIAG Mecanum mobile platform used to move aircraft components for assembly purposes.

Figure 13 The MIAG Mecanum Omnidrive Carrier System[7].

Figure 11 and Figure 13 present typical examples of why heavy-duty omnidirectional mobile locomotion is important and in demand in industry. When assembling huge-scale components,

[6] KUKA mobile robots. https://www.kuka.com/en-de/products/mobility/mobile-robots
[7] MIAG Mecanum carrier system. http://www.miag.de/index.php?m=13&page=30

a convenient way to instantaneously rotate the yaw orientation is handy. The Mecanum wheel is a practical solution.

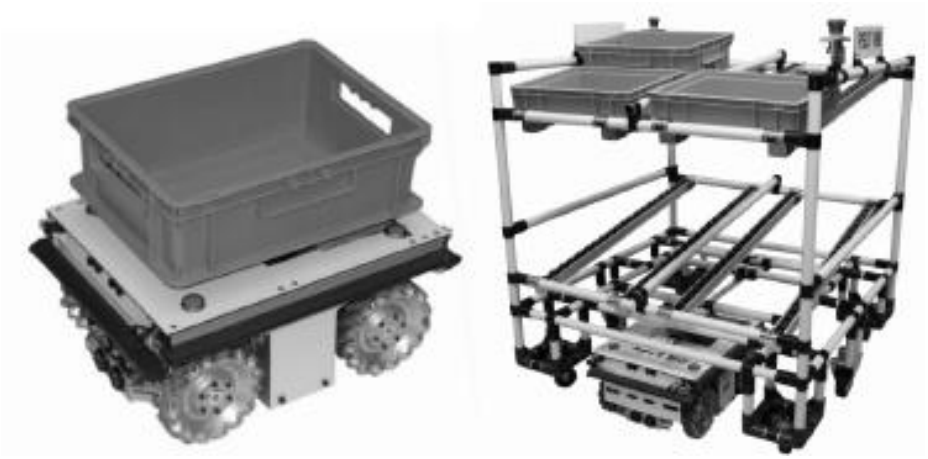


Figure 14 The Mecanum goods container vehicle (left) and the Mecanum trolley vehicle (right) [15].

The Mecanum wheels have also been proposed to build a small goods container vehicle and a trolley vehicle, as shown above in Figure 14.

Healthcare

In the field of healthcare, many Mecanum-based wheelchairs have been proposed, including OMNI, CIIPS and iRW, as shown in Figure 15.

Figure 15 The Mecanum wheelchairs. OMNI (left), CIIPS (middle) and iRW (right) [16-18].

Academic

There are many Mecanum robots designed for academic research purposes. The URANUS in Figure 16 was built in 1985 at the Carnegie Mellon School of Computer Science.

Figure 16 URANUS omnidirectional Mecanum mobile robot [19].

Mecanum Omni-1 and Omni-2 were designed and developed at the Electrical and Electronic Engineering Department of the University of Western Australia.

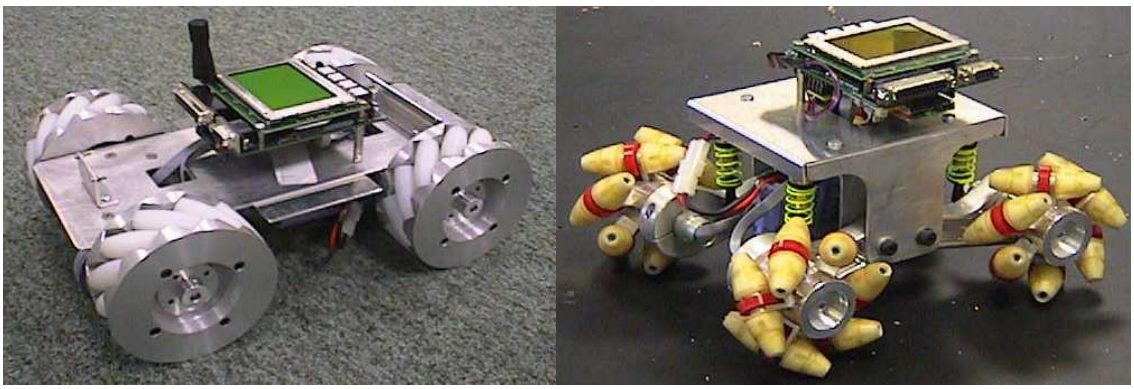


Figure 17 Mecanum robots: Omni-1 (left) and Omni-2 (right)[8].

M.E.G.A.N, which is short for Mapped Environment Guided Autonomous Navigator, was developed at the Mechatronics and Robotics Research Group at Massey University. One set of Mecanum wheels and one set of conventional wheels are both put on the M.E.G.A.N. The robot is able to move more efficiently using the conventional wheels, and performs omnidirectional motions using the Mecanum wheels when necessary.

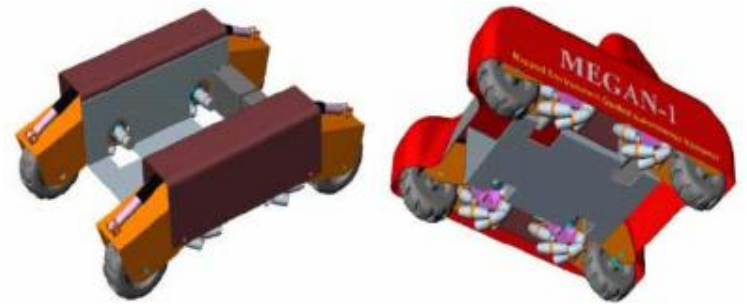


Figure 18 M.E.G.A.N.-1 [9].

[8] Omni-1 and Omni-2: Omni-1 and Omni-2 by the UWA. http://robotics.ee.uwa.edu.au/eyebot/doc/robots/omni.html.

The Mecanum shopping Trolley was designed at the FZI Research Centre for Information Technology at the Karlsruhe Institute of Technology in Germany.

Figure 19 Mecanum shopping Trolley [20].

2.1.6 Drawbacks of the Mecanum Wheel

The Mecanum wheel has wide applications and its industrial applications mainly take advantage of its omnidirectional manoeuvrability and heavy-duty loading ability. However, the Mecanum wheel has three major inherent drawbacks: random wheel slippage, high-speed vibration, and low energy efficiency [1]. To improve its functional performance and to further widen its applications, these issues must be addressed urgently.

When the Mecanum platform is operating, each Mecanum wheel is involved with a very complicated mechanism. This is initialized by the rollers. The freely rolling rollers are the only ground-contacting parts of the Mecanum wheel. Even though the specially curved geometry of the roller makes the Mecanum wheel able to rotate like a normal round wheel on the ground, only a point contact or a tiny area contact exists between the ground and roller due to the roller's placement angle and curved surface. When a working Mecanum wheel is rotating on the ground, most of the time the Mecanum wheel only has one roller contacting the ground. The roller-ground contact moves from one side of the roller to the other side. Sometimes two rollers may simultaneously contact the ground when the contact point discretely transfers from one roller to the next coming roller. Thus, the roller takes both rolling friction and sliding friction, and has a complex friction analysis to distinguish both frictions.

The rollers and their small area of contact cause the Mecanum wheel's first inherent problem – wheel slippage. Wheel slippage seems to be a very common problem for many

omnidirectional wheel designs such as the Omni wheel. Because both the Mecanum wheel and the Omni wheel rely on the freely rolling roller to perform omnidirectional motion, unlike conventional wheels, the freely rolling roller, as the wheel-ground contacting part, exists relative to motion from the ground. Both the small wheel-ground contact area and the freely rolling rollers make the mobile platform very vulnerable to slippage.

The Mecanum wheel's second problem is wheel vibration. When the Mecanum wheel is being rotated, the contact area continuously moves from one side of the roller to the other side and then discretely jumps to the next coming roller, and this same procedure repeats. When the roller-ground contact continuously moves on the same roller, the stability of the Mecanum mobile platform is highly determined by the curvature of the roller. In theory, the perfect curvature makes the Mecanum wheel rotate as smoothly as a perfect round wheel. However, [21] shows that the radius of a tested Mecanum wheel changes within a 5% range as the roller contact point is moving. In addition, the discrete transition of the contact area from the finishing end of one roller to the starting end of another roller results in wheel vibration, too. Because the continuously and discretely moving contact area causes wheel vibration, the wheel vibration is more obvious when the Mecanum wheel rotates at high speed.

Thirdly, the Mecanum wheel trades off manoeuvrability against energy efficiency [9]. The Mecanum wheel is very inefficient of energy consumption to perform omnidirectional motions. The driving force due to the motor torque is divided into effective wheel drive force and ineffective roller rolling force by the angled roller. The ineffective roller rolling force is wasted because roller rolling does not contribute to any movement at all. The angled effective wheel drive force, parts of which may further cancel each other, contributes to the complete manoeuvrability of the Mecanum robot. This leads to the third problem – low energy efficiency.

Fourthly, the Mecanum wheel is poorly robust to floor irregularity. In most cases, the Mecanum wheel is only used on plane ground. The limitation of the Mecanum wheels that only allows operating on plane ground has not been a critical issue for its industrial application yet so far, as many industrial workshops are operated on plane ground. And there are many robotic mobile applications, designed for planar motion only.

2.1.7 Kinematics Model of the Mecanum Robot

Because explicitly modelling the motion of the Mecanum platform is quite challenging, a simplified kinematics model is widely used to describe the motion of the Mecanum platform. The moving tiny contact of the freely rolling rollers that are attached around the Mecanum wheel, leads to a complex mechanism. This complex mechanism must be dealt with if explicitly modelling the motion of the Mecanum platform. However, based on some simplifications – such as perfect roller geometry, a completely free-rolling roller, a fixed contact point at the wheel centre, as well as others – the earliest and the most widely used kinematic model was derived in 1987 by Muir and Neuman from Carnegie-Mellon University [22]. By applying a coordinate transform matrix in relation between the four-wheel coordinate frame, the robot rigid body coordinate frame and the static ground coordinate frame, kinematics modelling is easily built by using a Jacobian matrix. The kinematics model transforms the robotic omnidirectional motions into the rotation tasks of each wheel. Based on their work, the most commonly used kinematic matrix is deduced in the equation shown in Figure 20.

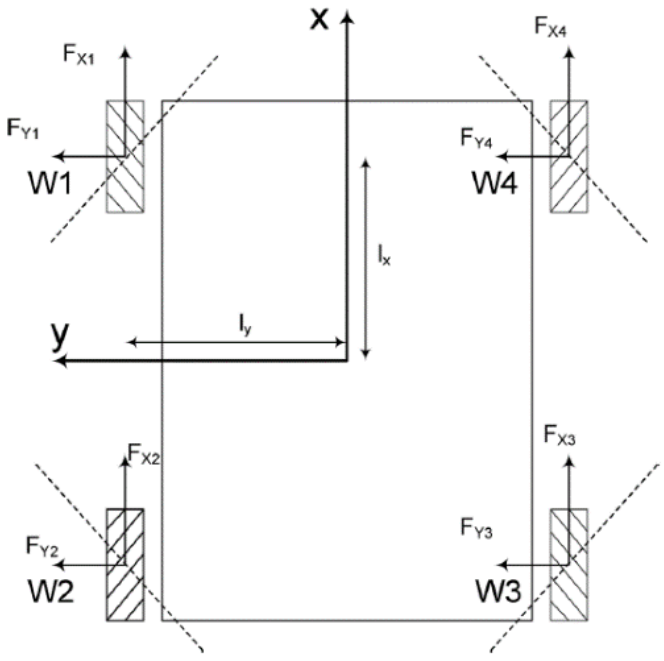


Figure 20 Kinematic matrix of a four-Mecanum-wheel vehicle. Sourced from [21].

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{pmatrix} = \frac{r}{4} \begin{bmatrix} -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ \frac{1}{l_x + l_y} & \frac{-1}{l_x + l_y} & \frac{-1}{l_x + l_y} & \frac{1}{l_x + l_y} \end{bmatrix} \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{pmatrix} \quad (1)$$

All the parameters in the kinematics are easily measured from the robot. $[x \quad y \quad \emptyset]$ is the robotic coordinate system, r is the radius of the Mecanum wheel, and l_x and l_y stand for the distances of the centre of mass of the wheel from the y-axis and x-axis in the robotic coordinate system. The rotations of the wheels are defined as $[\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4]$. There are many successful control methods based on such kinematics modelling applied on the four-Mecanum wheel robot.

Even though the Mecanum wheel is notorious for its slippage, the proposed kinematics model by Muir and Neuman does not consider any sources of the position error that are mainly caused by the wheel slippage and may be caused by other position error factors. Recently, a modified kinematic equation derived by considering the sources of the position error was proposed [23] and the experiment showed a more accurate result. Their proposed position error factors include slippage, bearing/axle friction and point contact friction.

2.1.8 Dynamics of the Mecanum Wheel

There are two approaches to derive the dynamics modelling of the Mecanum robots. The relatively common and simple one is to use Lagrange's equation to derive the sum kinetic energy of the vehicles [24-27]. The relatively more popular Lagrange method aims to generate a state equation as a control law for control purposes, based on the kinematic energy equation of the vehicle and the derived Lagrange's dynamics model as shown below.

$$K = \frac{1}{2} m (\dot{x}^2 + \dot{y}^2) + \frac{1}{2} J_w \dot{\theta}^2 + \frac{1}{2} J_w \sum_{i=1}^{4} \dot{\theta}_i^2 \tag{2}$$

$$\frac{d}{dt}\left(\frac{\partial k}{\partial \dot{\theta}_i}\right) - \frac{\partial k}{\partial \theta_i} = \tau_i \tag{3}$$

K stands for the kinematic energy of the robot. m is the mass of the robot, J_w is the moment of inertia of the robot in the z direction which is positive direction normal to the xy plane according to the right-hand rule, J_w is moment of inertia of the Mecanum wheel along its rotation axis, and τ_i is the torque of the wheel.

Then the state equation can be generated by combining these two equations. Similar to the kinematics model, the dynamics model was built by neglecting wheel slippage. The kinematic energy equation does not consider the energy loss from any motion-related frictions such as

sliding friction and rolling friction. The only friction considered is the viscous friction term from the Lagrange model.

The other less used approach is to use Newtonian mechanics [21, 28-30]. The Newton-Euler methods aim to describe the causes and characteristics of the robot's motion by using Newton's second law and Euler's second law. The relationship between motor torque and vehicle motion is stated in the Newton-Euler dynamics model. This approach requires a thorough understanding of vehicle mechanics. It initially converts motor torque to wheel tractive force according to motor-wheel mechanics, and then combines each wheel tractive force to vehicle driving force according to Mecanum mechanics, so the Newton-Euler method's complexity makes it a less handy approach than Lagrange's approach. Particularly [30], which is a recent and the most comprehensive Newtonian dynamics modelling for a Mecanum-wheel vehicle so far.

2.1.9 Control of the Mecanum Robot

It is common to apply the kinematics model of the Mecanum drive to close-loop control the motion of the Mecanum platform. Because the simplifications of the kinematics model are impractical for most of the manufactured Mecanum wheels, the open-loop control via the kinematics model is not reliable. The Mecanum robot may easily deviate from the planned motion trajectories via an open-loop control due to the random slippage of the Mecanum wheel. In order to cope with the wheel slip, the robotic position needs to be rectified or the robotic velocity requires to be compensated. Based on the kinematics model of the Mecanum robot, a closed-loop control can achieve this. At least two types of sensors are required: the shaft encoders and the position/velocity sensor. The shaft encoders report the real-time wheel rotation information to the robotic control system. The position or velocity sensors such as a vision camera, laser scanner sensor, optical flow sensor, world motion capture sensor and GPS detect the real-time position/velocity of the robot in the world frame. Either the positions or the velocities in both XY translations and Z rotation are needed for a 2D mobile robotic planar motion control. It is more recommended to apply independent closed-loops for each degree of freedom motion X, Y and Z.

Many control methods based on the kinematics model have been proposed to control the Mecanum robots [24-27, 29, 31-36]. PID control [37], fuzzy control [38], symptomatic rectification position corrective control [26] and feedback linearization method via Lyapunov stability theory [27] are the popular control approaches due to the easy implementation based

on the kinematic equations. [23] proposed a modified kinematics model by proposing position error factors, which were solved by controlled laws. The refined kinematic equation of motion better describes the accuracy of the position and reduces the load from the control method.

The dynamics of the Mecanum wheel are involved with complicated analysis of various frictions. Utilizing the dynamics model to control the Mecanum robot is not a mainstream. The back stepping method via the Lyapunov stability theory is proposed to control the Mecanum wheels using the Lagrange's equation-based dynamic model [21, 25]. In addition, [30] experimentally validated the proposed dynamics model by using the dynamics model to model the motion of the Mecanum platform.

2.2. Mechatronics Design of a Heavy-Duty Autonomous Omnidirectional Mecanum Robot

One of the research objectives in this PhD project is to design a fully functional autonomous Mecanum robot for the purposes of academic research and warehouse forklift prototype. To design and develop an autonomous navigation Mecanum-based mobile robot, the KUKA omniRob was studied. The architecture of the KUKA omniRob control system solves the coupling issue between the low-level Mecanum omnidirectional motion control system and the high-level intelligent autonomous navigation system. Beckhoff Automation's PC-based motion control system and ROS's autonomous navigation system are selected to develop the Mecanum robot in this PhD project. The designed motion control system can cope with the wheel slippage.

2.2.1 KUKA OmniRob

The KUKA omniRob was a new industrial Mecanum robot when this PhD project started. From January of 2010 to March of 2014, the project TAPAS – Robotics-enabled Logistics and Assistive Services for the Transformable Factory of the Future – aimed for transformable robot-based solutions in automation and logistics. One target of the project TAPAS was the creation of mobile robots with manipulator arms based on the existing robotic prototypes. The omnidirectional Mecanum wheel was selected by the omniRob designers for its complete manoeuvrability, which made the transportation task more flexible. The robotic manipulator with multiple degrees of freedom was selected for more flexible collection tasking. Both flexible transporting tasking ability and flexible collecting tasking ability are very outstanding in the industry of logistics.

KUKA Laboratories GmbH worked on the mobile robot omniRob shown in Figure 21. The mobile base of omniRob is an omnidirectional holonomic Mecanum-wheeled platform, which has been GmbH's technology product for a long time (other Mecanum robot designs are also seen in Figure 13). The arm is a KUKA LBR 4+ manipulator.

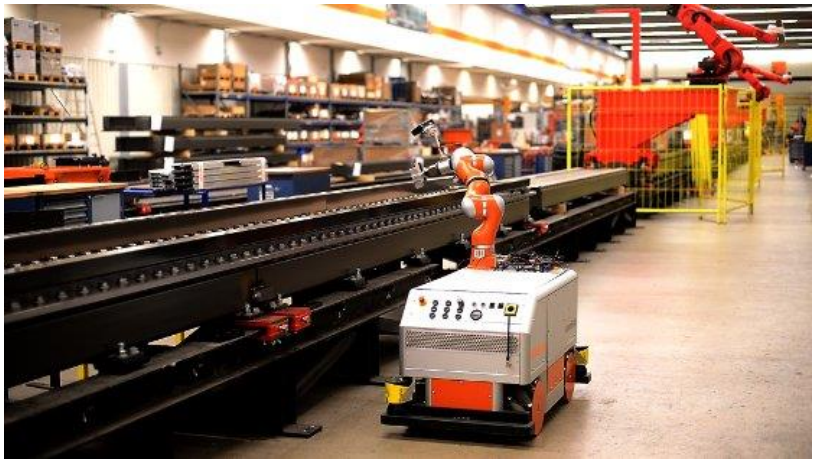


Figure 21 KUKA omniRob working in an automatic factory [39].

2.2.2 Architecture of the OmniRob Control System

The architecture of the control system in the KUKA omniRob robot consists of a controller system and a navigation system [40]. In the controller system, a Controller-PC is responsible to move the robot by controlling the motor rotations. The omniRob applies automation solutions from the company Beckhoff. The controller system also deals with the wheel slippage and some omniRobs utilize laser scanners for true position/velocity detection. The Controller-PC of the omniRob installs the KUKA Sunrise operating system. The navigation system takes care of all the intelligent autonomous navigation tasks. KUKA developed its own navigation system in a Navigation-PC. Both PC systems communicate with each other via an industrial Ethernet-Switch. Inspired by KUKA omniRob's design, the same system architecture is utilized to design the Mecanum robot system in this project.

Beckhoff Automation

The company Beckhoff Automation is a manufacturer that provides automation technology solutions. Its PC-based automation control system and real-time Ethernet are the key product features. Beckhoff Automation is the automation solution provider of the KUKA Mecanum omniRob. Beckhoff Automation technology is also utilized in this project as the motion control system.

2.2.3 Robot Operating System - ROS

The Robot Operating System (ROS) is a robotics software framework, which offers ready-to-use collections of libraries, tools and packages, and provides a convenient, advanced, and robust robot framework for developing robot applications. ROS.org is an open community for the robot developers all over the world. Intelligent robotics are developing and evolving so fast that the features and functionalities of intelligent robotic systems are improving all the time. However, implementation of reliable and robust software in the robotic system is very complex. An insignificant behaviour to humans tends to be very sophisticated implementing as robotic software. For robot designers, robotic system development from scratch and continuous upgrade and maintenance consumes a lot of effort and time. For robot researchers, the current state-of-the art intelligent robotic system should be implemented on purpose-built robots first in order to research further potential improvement possibilities. Even for amateur robot fans, a ready-to-use system that does not require expert knowledge is very useful. Thus, ROS as a robotics software framework provides open source packages to implement functionalities of intelligent robots on the robotic platform and encourages collaboration from the people active in the field of robotics all over the world for the ROS[9] community.

"The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms."

Other available robot frameworks include Player Project, Yet Another Robot Platform (YARP), OROCOS Project, CARMEN, ORCA Robotics, MOOS and Microsoft Robotics Studio.

<u>ROS Navigation Stack</u>

ROS provides an open-source navigation stack for the autonomous navigation research community. Thanks to ongoing modifications and maintenance from the community, the ROS navigation stack represents cutting-edge autonomous navigation technology in the field of autonomous mobile robots. The ROS navigation stack is the main reason why ROS is selected as the robotic system framework for this PhD project.

[9] ROS: The Robot Operating System. http://www.ros.org/about-ros.

2.2.4 Wheel Slippage

Slippage is a common problem for most omnidirectional drives. For the Mecanum wheel there are two reasons to cause the slippage problem, and both reasons are related to the roller-based tread of the Mecanum wheel. Because of the slippage, the following two situations must be faced by Mecanum robot designers. Firstly, random wheel slippage causes a severe position error in the shaft optical encoder dead-reckoning method, which is very commonly applied in wheel mobile robots. A robust self-contained dead-reckoning method is required, so that the robot based on Mecanum wheels can perform autonomous navigation functions in an industrial environment like a warehouse. Secondly, the Mecanum wheel has a different amount of slippage when the Mecanum robot is performing different omnidirectional motions, e.g. the Mecanum robot travels a different lateral distance from longitudinal distance with the same amount of wheel rotations [39].

To deal with wheel slippage, a closed-loop control system is applied in this project to compensate for the wheel slippage with the help of true velocity/position information that is collected by sensors such as IMUs, optical flow sensors, laser scanners and motion capture systems. The visual dead-reckoning based on the cameras or optical flow sensors was utilized [37]. By processing the frames captured from the camera or counting the pixel changes of the image captured from the sensor, the system estimates the real-time velocity of the robot and then compensates for the travelling distance loss from the slippage.

2.2.5 Vibration

The vibration of the Mecanum wheel usually happens at high-speed operations. The high-speed vibration of the Mecanum wheel becomes more severe in the heavy-duty case. The heavy-duty Mecanum wheel vibration may create unacceptable noise and even affect the stability of the motor currents and the power system, which creates safety issues. At the moment, the Mecanum applications are normally at low speeds so this has not been a critical issue yet and the vibration issue is not dealt with in this project.

2.3. State of the Art of Autonomous Navigation

Autonomous navigation is an important field of mobile robotics. Navigation is defined as the act of planning and guiding towards the goal location. To navigate, an autonomous mobile robot constantly locates its position in one given map or the self-built representation of

environment, and continuously updates the feasible route to guide itself approaching goal locations. Autonomous navigation has been deeply studied in the field of mobile robotics for many decades. To perform robust autonomous navigation, there are many sophisticated techniques in localization, path planning, and motion control that meet desirable optimization criteria: mostly time and distance [4]. However, energy-optimal autonomous navigation aiming to reduce the energy consumption for omnidirectional mobile robots has not been systematically investigated yet. As part of the robot design objective, the robot should be able to perform autonomous navigation functions. In this section, the ROS navigation stack is set as an example to present the state of the art of the autonomous navigation technology nowadays.

2.3.1 ROS Navigation Stack System Architecture

The ROS navigation stack operates like a high-level intelligent navigation system that receives a goal pose from the user and then automatically generates velocity commands. The user can select the goal pose on the ROS costmap[10] built by the ROS navigation stack's simultaneous localization and mapping (SLAM). The generated velocity commands are constantly sent to a low-level motion control system, here named a mobile base. While the mobile base is moving towards the goal pose by following the velocity commands, sensory information about the surrounding environments keeps sending to the ROS navigation stack. The local sensory information is important to constantly update the robotic costmap so that the ROS navigation can react to unexpected situations.

As the motion control task is executed at the mobile base, the navigation stack system is only responsible for motion planning. A prior condition for motion planning is localization – the navigation stack system should know where the robot is all the time. A probabilistic localization system, AMCL[11], that is based on the adaptive Monte Carlo localization approach is implemented in the navigation stack. After the SLAM costmap is built, the particle filter algorithm is applied to track the robot in this known map. This 2D occupancy grid costmap stores the sensory data of the world. The cost value of each map cell tends to represent the distance from an obstacle cell, while some particular cost values stand for specific situations. ROS provides a laser-based SLAM package for the mapping functionality named Gmapping[12].

[10] Costmap: ROS Navigation Stack Costmap. http://wiki.ros.org/costmap_2d.
[11] AMCL: ROS Navigation Stack Adaptive Monte Carlo Localization. http://wiki.ros.org/amcl.
[12] Gmapping: ROS Navigation Stack Gmapping. http://wiki.ros.org/gmapping.

The motion planning of the ROS navigation is in a cascaded setup according to ROS nav_core[13]. Motion planning consists of both path planning and trajectory planning. The navigation stack system[14] employs both deliberative and reactive schemes, and utilizes both a global planner and a local planner in its motion planning [41]. In terms of the deliberative scheme, the global planner finds a feasible collision-free path from the current position to the goal position. Because a mobile robot is usually regarded as a 2D point in path planning, orientation of the robot is not taken into account. Based on the global path, the local planner generates a velocity trajectory for the robot to reach the goal pose. In there, the robotic velocity trajectory per sampling time, which is also known as control command, is updated every control cycle. The local trajectory planner works online in such a way mainly because in autonomous navigation a mobile robot should be able to react to its surrounding environment that is subject to change. Thus, the local trajectory planner is under the reactive scheme so that the local planner can deal with unexpected situations that cannot be foreseen by the global path planner.

2.3.2 Online and Offline Planning

Motion planning of mobile service robots is classified into offline and online planning [42]. An offline planner, which globally pre-plans a path and/or a trajectory from the current pose to the goal pose, is usually applied for repetitive tasks in a static known environment. The online planner updates the local trajectories every control cycle while reaching the goal pose, in order to react to its surrounding environment that is subject to change. Compared to the online planner, the offline planner cannot cope with unforeseen situations such as unknown obstacles, but its optimization is more reliable in the global sense.

2.3.3 Deliberative, Reactive and Hybrid Navigation System

In terms of control architecture, the autonomous navigation system can be categorized as deliberative navigation, reactive navigation, and their hybrid. The deliberative navigation system usually utilizes offline planning and the reactive navigation system usually utilizes online planning.

[13] Nav_core: ROS Navigation Stack nav_core. http://wiki.ros.org/nav_core.
[14] Navigation stack: ROS Navigation Stack. http://wiki.ros.org/navigation.

Deliberative Navigation System and Offline Planning

Deliberative (centralized) navigation, also commonly used as map-based navigation, requires a global map of the working environment, either provided by users or self-built by sensors. The deliberative navigation system finds the path to the goal in the map and generates the trajectory tasks via offline planning. Then, the motion controller executes the trajectory tasks in the real world. The standard procedures of the deliberative navigation are shown in Figure 22, but not necessarily in that exact order. Deliberative navigation assumes that all the information about the environment is already known, and the environment is static or fully controlled. The motion planning in deliberative navigation is offline and usually global.

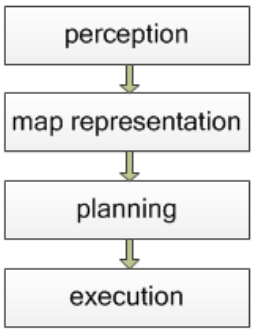


Figure 22 Deliberative navigation.

Deliberative navigation globally plans actions of mobile robots in advance, highly relying on the prior knowledge of the environment. The planner of the deliberative navigation is usually offline so that the robots are unable to agilely deal with imperfections of the environment knowledge. Deliberative navigation planning is involved with massive amounts of computation due to the nature of global planning and, as the deliberative navigation planning is usually offline, computation time is not a critical criterion. In the practical applications, deliberative navigation requires either static environments or fully controlled environments. It is commonly used in repetitive logistic tasks. Absolute global optimization can be guaranteed to be realized in deliberative navigation.

Reactive Navigation and Online Planning

Reactive navigation copes with the various unmodeled or unexpected uncertainties that cannot be solved by deliberative navigation. Dynamic environment or unexpected events lead to an uncertain map and/or environment representation, or sometimes the robot that is located in an unknown environment does not even have a map representation. Reactive navigation only reacts to the local perceptive environment. Behaviour-based reactive methodology is a

common approach. The actuators' execution commands are generated by various behaviours in immediate response to the sensory information. Sometimes the execution commands are coordinated from various behaviours. The standard procedures of reactive navigation are shown in Figure 23.

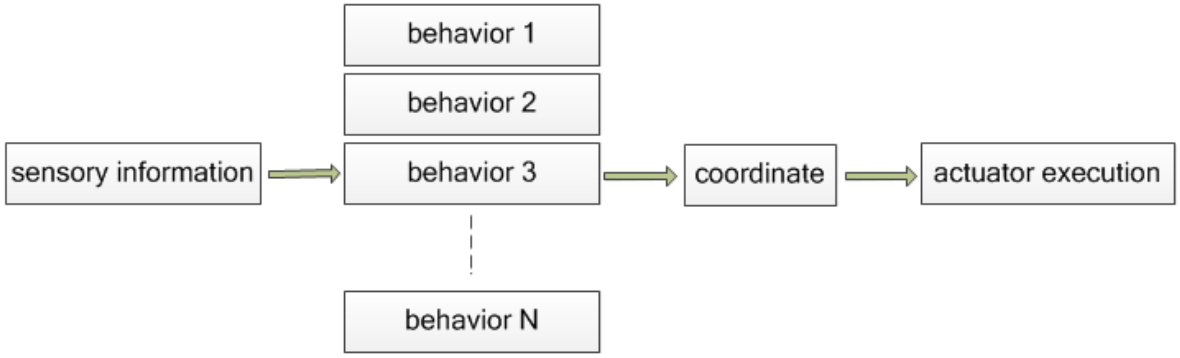


Figure 23 Behaviour-based reactive navigation procedures.

When facing the dynamic environment, partially known environment or partially controlled environment, reactive navigation planning must be applied. Reactive navigation first stimulates the robotic mobile actions in response to the most recent local perceptive information and then selects a feasible motion plan to accommodate the surrounding local environment. Reactive navigation planning is different in many ways. As the environment is subject to change in a very unpredictable way, reactive navigation planning is online. This requires for every control cycle, a limited amount of computation and a time-critical tasking. The reactive navigation lacks global sense so that it may be trapped by the local environment such as the limit cycle path problem [43]. In addition, reactive navigation is locally planned with respect to the instantaneous environments. Absolute global optimization is very difficult to achieve in reactive navigation.

Hybrid Navigation System

A hybrid navigation system hires and benefits from both the deliberative scheme and reactive scheme. The hybrid navigation system is a common navigation solution that employs a global path planner and a local trajectory planner respectively [4, 41]. The ROS navigation stack is a typical example of the hybrid system. The high-level path planner globally finds a complete collision-free route from an initial position to a goal position in the representation of the environment, a map for example. This can avoid the limit cycle path problem and can plan an optimal path (for example shortest path). Then the low-level trajectory planner locally makes motional decisions to continuously track one segment of the route by another. The local

trajectory planner aims to follow the global path as a global guide and, in the meantime it is always ready for path deviation to deal with any unexpected circumstances. After passing the unexpected obstacles for example, the local trajectory planner guides the robot to safely return to the old global path. Or instead of returning, a more advanced intelligent hybrid system plans a new global path while the robot deviates from the old path.

2.4. Dynamic Window Approach

The Dynamic Window Approach (DWA) [44] was originally a local obstacle-avoidance method with a similar approach to vector field and vector histogram approaches. The basic scheme performs forward simulation to find the admissible controls and velocities that allow the robot to stop before hitting obstacles whiling respecting the robot's kinematic and dynamic constraints. Then an optimization is performed over these admissible controls to find the one with the highest utility. Different suggestion of the utility function are proposed in [44-48]. Brock et al. [45] extended the original DWA in [44] by looking at holonomic robots while Fox et al. considered synchro-drive ones, and more importantly, adds information about connectivity to the goal globally by introducing a NF navigation function. In [46], they reformulate the DWA approach as a model predictive control (MPC) by assuming a holonomic robot model and choose acceleration as the control signal in contrast to [44], where velocity is used to control the robot. Another two recent DWA improvements in [47] and [48] guarantee a global convergence to the goal by introducing a new objective function and considers the shape of the robot using MPC respectively. In addition, the DWA has already been implemented in the ROS navigation stack system as a local trajectory planner.

Based on these studies, the DWA is proven to be a well-known and stable approach for both holonomic and non-holonomic drives with excellent experiment results. More importantly, the DWA easily accommodates extensions, being flexible with its utility function. The original DWA utility function have been successfully reformulated by MPC, so an energy cost function based on an energy prediction model should be possibly able to find the energy-optimal trajectory. Additionally, the original DWA approach was designed to perform obstacle avoidance. The DWA-based energy-optimal trajectory planning can be proposed as a sound technique in energy-minimizing reactive navigation.

2.5. Energy Consumption Model

According to the recent existing methods of energy-efficient motion planning[49-51], a reliable energy consumption model for the target robot is a common approach to start the energy-efficient motion planning problem. The energy consumption of the mobile robot is a complex problem involving a lot of influencing factors. About 10 years ago, due to there being a lot of influencing factors, [52] chose to use a polynomial fitting model and [53] simply used gravity and simple friction as energy cost. But in recent years, the energy consumption models were built by listing the major factors that influence robotic consumption. Their proposed factors were experimentally validated to address the complex robotic energy consumption model for their application robots. However, the energy consumption model for the Mecanum mobile robot has not been studied yet.

2.6. Energy Efficient Motion Planning

Conserving energy is a global strategy to solve environmental issues now. Energy-reduction techniques are important to solve many problems in various fields, particularly the global warming issue and optimal logistics solutions. Increasing energy efficiency is the most direct way to secure and sustain energy. In the mobile robotics field, an energy-efficiency system releases the design pressures from the battery capacity and system consumption. The mobile robot usually sources its motion power from a limited-capacity battery and the operating duration is determined by how much power the robotic system consumes and how much energy the battery stores. Many energy conserving approaches focused on increasing the energy efficiency of the robot components such as battery, controller and actuators. Efficient energy utilization of on-board electronics devices applying recent technologies such as power electronics [54-56] is a direct mitigation of the problem. Power management and multitasking scheduling are also promising techniques for energy-efficient designs of autonomous mobile robots [57]. Recently energy-efficiency motion planning strategies have shown effective energy-saving results. Many researchers have shown strong interest in how to reduce energy consumption of mobile robots via optimal motion planning.

Motion planning including both path planning and trajectory planning, has been well researched, and flourish with many mature and sophisticated algorithms, but mostly aiming for the shortest length or the least time. Research on energy-efficient motion planning methods has not been studied enough yet, in particular for omnidirectional mobile robots. To reach a goal

position from a start position, there are an infinite number of feasible paths. It takes different amounts of energy for one mobile robot to follow each path. Even along the same path, one mobile robot that executes different velocity profile consumes a different amount of energy. The mobile robot may abuse energy due to inappropriate accelerations or decelerations. Mei et al. introduced a method of reducing energy consumption of mobile robots to search an open area via energy-efficient motion planning [52]. The results showed that the energy efficiencies of different paths and different velocity profiles to search an open area were obviously dissimilar. Their studies focused on energy-efficient searching, exploration and deployment of mobile robots, however, instead of autonomous navigation.

2.6.1 Energy Efficient Motion Planning Proposal for Omnidirectional Mecanum Robot

Recall that low energy efficiency is one of the inherent disadvantages of the Mecanum wheel. Actually, low energy efficiency is also a common drawback for many omnidirectional wheel designs such as the Omni wheel. The Mecanum wheel trades off manoeuvrability against energy efficiency. Unlike other nonholonomic drives, the omnidirectional Mecanum mobile robot has complete manoeuvrability, which provides a lot more feasible path plans to reach the same goal pose. The differential drive as a common nonholonomic drive can only move along straight lines and differential curves. The holonomic Mecanum drive can move along any types of continuous lines. In addition, different trajectory plans to move along one path consume different amounts of energy. This can be caused by different velocity profiles, which consume different amounts of energy, and different omnidirectional motion forms, each of which has different energy efficiency. Thus, one feasible method to solve the low energy efficiency issue of the Mecanum wheel is to develop energy-optimal autonomous navigation motion planning methods for the Mecanum mobile robots.

2.6.2 Existing Methods

There are many mature studies investigating optimal path planning methods but typically to find the shortest as the geometry-optimal. Regarding the energy consumption of the mobile robot, if a shorter route contains many sharp turns, the robot may consume more energy due to frequent decelerations, changes of directions, and accelerations. A longer route may require less energy if the robot does not have to accelerate often [52]. [49] is a recent journal paper studying minimizing energy consumption of wheeled mobile robots via optimal path and trajectory planning.

Some researchers study energy-minimizing path planning without considering the energy consumption of mobile robots. Sun and Reif concentrated on the problem of energy-minimizing path planning on terrains considering only friction and gravity [53]. The energy cost in their study was only based on gravity and friction. [58] proposed an energy-minimizing path planning for robots to finish all tasks in various locations and stay alive. Liu and Sun added energy-related criteria, which is based on an energy model of a wheeled mobile robot, into the A* algorithm for energy-minimum path planning [49]. This energy-efficient path planning method does not consider the robotic motion trajectories. Different motion trajectories of a robot consume different amounts of energy to follow the same path. It is not convincing to find the energy minimum path without optimizing the robot motion trajectories. In addition, the path planning method assumes that the wheeled mobile robot is a point without orientations. But this assumption does not apply properly to omnidirectional mobile robots. The orientation of an omnidirectional mobile robot is not necessarily in the tangential direction of the path as the orientation of the differential mobile robot is. Thus, it is necessary to consider robot motion and current orientation in energy-optimal path planning for omnidirectional mobile robots. To follow the energy-minimum path, Liu and Sun proposed a parameterized smooth trajectory planning with minimum energy consumption specific to nonholonomic wheeled mobile robots. But their proposed trajectory planning is specific to nonholonomic wheeled mobile robots. Prior to parameter optimization, the robotic orientations of the waypoints are set for the purpose of generating trajectories to cope with nonholonomic constraints. The holonomic omnidirectional drive is able to move instantaneously in any direction from any orientation. Different omnidirectional motions have different energy efficiencies so that the trajectory of the robotic orientations should be also optimized for the Mecanum robot.

[49] provides a recognised methodology of planning an energy-minimum path and trajectory for a nonholonomic wheeled mobile robot has the following useful information. An energy consumption model is firstly established for the wheeled mobile robot. The established energy consumption model is taken as a solid foundation for an energy-efficient motion planning study. [51] also takes the same action by starting the study from building an energy consumption model for the target robot. Then, a cost function of the robotic energy consumption is derived from the energy consumption model. The A* algorithm-based path planner is used to find the collision-free energy-optimal path by using the energy cost function. Based on the existing sophisticated path planning method, an energy consumption criterion is newly considered and an energy-optimal A* algorithm path planning technique is proposed.

Due to the sophisticated global path planning algorithm development, there are many path planning methods to optimize the path, for example, A* search algorithm [49, 59], A* heuristic search algorithm [60], Voronoi diagram [61] and probabilistic roadmaps [62]. A proper energy-optimal path planning method for FOMRs should be easy to find. To follow the found path in [49], a non-holonomic wheeled mobile robot applies a secondary-smoothing algorithm. A cubic Bezier curve-based trajectory planner is used to smooth the trajectory while optimizing the energy consumption. However, path planning and trajectory planning are separately optimized in [49]. Energy-optimal path planning without considering robot motion is not convincing enough.

After planning the energy-optimal path, the next challenge is globally optimizing the trajectory along the result path. Along the same given path, energy consumption differences depending on different velocity profiles can be noticeable. Along each type of path, one needs to find velocity profiles for the Mecanum robot so as to minimize the energy consumed. The robot may consume more energy due to frequent and inappropriate decelerations and accelerations. The Mecanum robot when operating different omnidirectional motions as shown in Figure 2 has different energy efficiencies. To solve such energy-optimal motion planning problems, [51] provides a recognised methodology to find minimum-energy trajectory planning on a straight-line path for three-wheeled omnidirectional mobile robots based on Omni wheels. [51] as a recent journal paper with a similar research interest presents some useful ideas. An accurate dynamics model that considers the Coriolis force and actuator motor dynamics is built first. Based on the dynamics model, an experimentally validated cost function of the robotic energy consumption that is drawn from the battery to motors, is derived. Pontryagin's minimum principle is used to find the energy-minimum rotational velocity trajectory and a novel minimum algorithm is used to find the energy-minimum translation velocity trajectory. The found velocity trajectories have been executed on the robot via an implemented control system in experiments. Kim et al. studied minimum-energy trajectory planning of both nonholonomic and holonomic drives on a straight-line path [50, 51]. Their results showed that following the same straight-line path via different velocity profiles cost different amounts of energy.

However, their studies were restricted to straight-line paths. Even though a planned path normally consists of two types of path segments, straight-line path and curved-line path, curved-line path is not studied in [51]. A curved-line path is more dynamic and more complex than a straight-line path. There are many ways used to describe the curved-line paths, including the clothoid [63], circular arc, and other parametric curved lines with irregular curvatures such

as Dubins curves and Bezier curves [64]. Their study also requires stationary states in the beginning and at the end, and hence become much less practical. It is not practical, nor energy efficient to make mobile robots come to a full stop at the end of each path segment. Most importantly, those minimum-energy motion planning methods cannot deal with reactive avoidance of unforeseen obstacles for autonomous navigation. Thus, a new planning method that can cope with any types of paths and reactive obstacle avoidance should be proposed for energy-efficient autonomous navigation of the holonomic Mecanum mobile robots.

2.6.3 Power Minimization via Motion Planning

Robotic power consumption is the robotic energy consumption per time. It is interesting and important to optimize the robotic power consumption via motion planning. Because mobile robots usually source energy from batteries that store limited energy, most of the existing energy-efficient studies aim to minimize energy consumption instead of power consumption. However, power consumption is also a critical criterion to design and control mobile robots with self-sustaining power supplies such as solar power [65-68]. Maintaining robotic power consumption closely below the solar power supply can sustain the robot in operation for a long time. Energy-conservation techniques are proposed to improve energy efficiency for mobile robots by reducing the power consumption of the robotic electronic components in [57]. The issue of power minimization via motion planning has not been fully studied, in particular for holonomic mobile robots with redundant manoeuvrability.

2.6.4 Optimization Algorithms

Most existing energy-optimal trajectory planning was based on complex mathematical optimization algorithms e.g. finding energy-optimal solutions relying on constrained polynomial parameterized trajectories [69], using optimal control theory to find energy-optimal velocity profile to follow straight-line paths [51], applying calculus of variation to find smoothness-optimal trajectory as energy conservation [70] and a modification of the Newton algorithm for nonholonomic energy-optimization motion [71]. The main advantage of applying mathematical optimization algorithms such as optimal control theory or calculus of variation is the guarantee of optimal or near-optimal solutions. However, the algorithms usually require the planning problem formulation to have certain levels of constraints such as fixed initial and final stationary states, specific shapes of path, defined path-description functions, unique path characters like smoothness, or specified nonholonomic constraints. It costs a lot of time and

effort to fit these algorithms into autonomous navigation. Kim et al. have conducted many studies on both online and offline energy-minimization trajectory planning of both nonholonomic and holonomic drives on straight-line paths [50, 72-74] and minimum-time trajectory planning following specified bounded-curvature paths [63]. They showed 5% to 10% energy saving compared to the trapezoidal velocity profile. However, their studies are restricted to specified shapes of paths. The problem was formulated as that the initial and final states of the mobile robots are both in a full stop state. The aforementioned optimization algorithms are restricted from further application in the trajectory planning of autonomous navigation.

2.7. Summary

Based on the summarized literature review of the relevant topics, there are many practical Mecanum applications in various fields. In particular, the Mecanum wheel's high load capacity prioritizes the Mecanum drive over other omnidirectional wheel designs for heavy-duty tasks. However, the Mecanum wheel's further application was hindered by its three inherent disadvantages: random wheel slippage, high-speed vibration, and low energy efficiency. There are many research interests about the Mecanum wheel by virtue of these three main inherent disadvantages and those research interests in heavy-duty Mecanum robots are expected to be explored, and particularly the low energy efficiency of the Mecanum wheel still lacks research in the academic field and solving this problem would benefit its further practical application.

Omnidirectional mobility is a privileged ability for mobile robots, particularly in confined, congested and highly dynamic environments. The Mecanum wheel, due to its unique heavy-duty transporting ability in the omnidirectional wheel family is the most common heavy-duty omnidirectional manoeuvrability solution in the industry [6, 12, 16, 18, 75, 76]. Like many other omnidirectional drives, the Mecanum drive trades off energy efficiency for manoeuvrability. Its low-efficiency working principle for generating omnidirectional motions consumes a large amount of energy, particularly for heavy-duty tasks in industry [9]. The amount of energy consumed by industrial heavy-duty Mecanum robots such as the Airtrax Mecanum forklift and the KUKA autonomous Mecanum mobile platform is enormous. Each omnidirectional motion such as forward, sideways, diagonal or curved motion results in different amounts of energy efficiencies. A four-wheel omnidirectional Mecanum drive on a plane ground is a holonomic and redundant system, so reaching one given goal pose, it has many motion plans, each consuming different amounts of energy. Every motion plan comprises a path and a trajectory. Different paths of different lengths and different curvatures have an

influence on the robotic energy consumption. Even to follow the same path, multiple options of trajectories involved with various combinations of omnidirectional motions are eligible, as shown in Figure 24, so, it is both academically interesting and economically beneficial to solve the systematic problem of energy-optimal trajectory planning for autonomous navigation of Mecanum mobile robots.

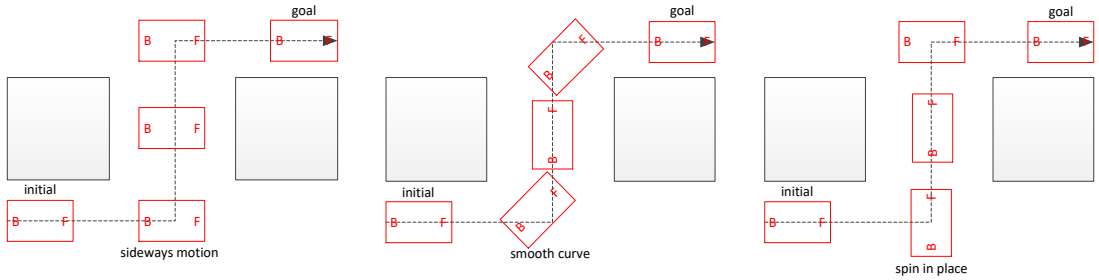


Figure 24 Multiple omnidirectional trajectories to follow the same path.

This PhD research focused on studying an energy-optimal autonomous navigation approach for four-wheel omnidirectional Mecanum mobile robots. The following research gaps were declared in this PhD research project. First, a heavy-duty four-wheeled Mecanum omnidirectional mobile robot was developed as a research platform for future autonomous navigation experiments. All the proposed energy-efficient autonomous navigation methods in this PhD project were implemented in the Mecanum robot for experiment validation. Second, because the energy consumption model for the Mecanum mobile robot had not been studied yet, a novel energy consumption model aiming to evaluate the energy consumption of the Mecanum mobile robot while conducting various motion trajectories, was established and experimentally validated. Energy-efficient autonomous navigation had been explored very little for the Mecanum robot. The motion trajectories of the Mecanum mobile robot were optimized in both deliberative navigation and reactive navigation, in order to realize the energy-optimal autonomous navigation. This requires an offline energy-optimal global motion planning method to deal with deliberative navigation and an online energy-optimal local motion planning for reactive navigation. Lastly, both the global offline motion planning problem and the local online motion planning problem were solved in this project.

Energy-Efficient Autonomous Navigation

In order to realize energy-optimal autonomous navigation via motion planning for the omnidirectional mobile robot, both deliberative planning and reactive planning were considered in this study. The motion planning of the ROS autonomous navigation consists of

deliberative path planning and reactive trajectory planning. It is common to have global path planning, which arranges a path from the start to the goal in the map, and local trajectory planning, which generates a trajectory profile along the arranged path and reactively avoids unforeseen obstacles. Deliberative navigation assumes that the working environment is completely known and static. Under such an assumption, motion planning can be both global and offline. Offline motion planning has a lot of potential to optimize robotic energy consumption. In deliberative planning, a method that simultaneously optimizes both the path and motion of the omnidirectional robot for the least energy consumption is required. The method plans an energy-optimal path from the initial pose to the goal pose, and in the meantime, generates an energy-optimal trajectory along the path. The reactive navigation must be able to cope with unexpected situations so that the motion planning is usually local and online. This raises the challenge of how to achieve optimization of energy consumption in the global sense via local motion planning. In reactive planning, an energy-optimal motion planning method that can reactively avoid unexpected obstacles is needed.

Chapter 3. Mechatronics Design of a Heavy-Duty Omnidirectional Mecanum Robot

3.1. Introduction

In this PhD research project, a heavy-duty omnidirectional four-wheeled Mecanum autonomous mobile robot was designed and developed from scratch, as a warehouse forklift operating in industrial environments for the purposes of conducting the energy-optimal autonomous navigation research. This robot, named AuckBot, is suitable for and capable of conducting experiments for validating the proposed energy-optimal motion planning algorithms, which are to be introduced in the next several chapters. It should be highlighted that the designed robot is not only limited to this energy-optimal autonomous navigation research, but also for any other future research. In addition, the designed robot will be eventually developed to be an omnidirectional Mecanum warehouse forklift prototype in future projects. Now, the robot is able to perform fully functional autonomous navigation.

3.2. Overview

The robotic design and development in this PhD project include the following four parts. Firstly, the electromechanics of the robot were designed to transmit electric power to the rotating Mecanum wheels. This part of the design also comprises the chassis of the robot that carries all the components. The transmission system that transmits the servo motor power to the Mecanum wheels and the electric system that supplies power to the robot were designed to manufacture the robotic hardware. The second part is that a locomotion control system was developed to omnidirectionally drive the Mecanum robot. Industrial servo solutions are utilized to control the rotational velocity of the Mecanum wheel axis and thus to execute the omnidirectional motion of the robot. Thirdly, an intelligent autonomous navigation system was implemented on the Mecanum robot. The ROS navigation stack is utilized and configured on the robot as a high-level autonomous navigation system. Lastly, a completed robotics control system was realized by fusing both the locomotion control system and autonomous navigation system. The low-level locomotion control system that is based on Beckhoff technology, receives and executes the motion commands from the high-level ROS autonomous navigation system that copes with perception, localization, cognition, trajectory planning and trajectory following.

3.2.1 Robotics Control System Architecture

A robotics control system architecture was implemented to perform autonomous navigation functions on the robot. The implemented robotic control system architecture is inspired by and based on the KUKA omniRob. The robotic control system is able to both perform high-level intelligent navigation tasks and control low-level motion execution. A low-level Controller-PC governs the motion execution of the robot platform, while a high-level Navigation-PC deals with navigation tasks such as perception, localization and cognition. The Controller-PC has the qualities of real-time motion tasking and locomotion control. The selection of an industrial controller is preferred to meet the requirements of robustly operating the Mecanum mobile forklift-prototype robot in an industrial environment. The Navigation-PC is a Linux laptop running the Robot Operating System for open-source autonomous navigation technology from a leading robotics community. The ROS navigation stack represents cutting-edge autonomous navigation technology. Lastly, a combined system of both the Controller-PC and the Navigation-PC was developed for the target robot. The system architecture synergistically combines a Beckhoff industrial PC as the Controller-PC serving low-level motion execution and a ROS laptop as the Navigation-PC coping with high-level intelligent navigation tasks. The most difficult technical issue in this part of project was a robust communication between the Controller-PC and the Navigation-PC.

3.2.2 Progress of the Robotic Enhancements

Figure 25 Versions 1, 2 and 3 of the robot. Version 1 was designed and manufactured from scratch, with the Beckhoff automation system implemented for simple omnidirectional motions. The operation commands can be inputted into the robot via a touch screen. Four 12-volt batteries are used as the energy source of the robot. The Mecanum wheels are assembled outside the robotic chassis. Version 2 was vastly improved by adding and combing the ROS autonomous navigation stack to the robot and with the Beckhoff system. The combined system allows the robot to perform both high-level navigation tasks and low-level motion execution for conducting experiments. A range of sensors have been successfully implemented on the robot for trial purposes. The wheel placement was redesigned to be inside the robotic chassis for smoother and more stable motion. Version 3 is the final form of the designed robot. It focuses on the maintenance, organization, grooming and most importantly design finalization.

The robot has been enhanced three times throughout this project. More design details of versions 1, 2 and 3 of the robot can be found in [77], [1] and [46], respectively. The progress of the robotic modifications is shown in Figure 25 and details are presented in the following sections. The first version of the robot was designed and manufactured from scratch. Only the Beckhoff automation system was implemented in the robot. The robot could perform simple omnidirectional motions of the Mecanum drive as manual operation commands. The second version of the robot was vastly improved by adding and combining the ROS autonomous navigation stack to the robot and with the Beckhoff system. An exteroceptive range of sensors were interfaced to the robotic system. The electric circuit design was also refined, and the wheel placement was redesigned for smoother and more stable motion. Version 2 of the robot was eligible and feasible for conducting experiments. The combined system allowed the robot to perform both high-level navigation tasks and low-level motion execution. Various sensors were successfully implemented on the robot for trial purposes. Much better motion benefited from the new wheel placement design.

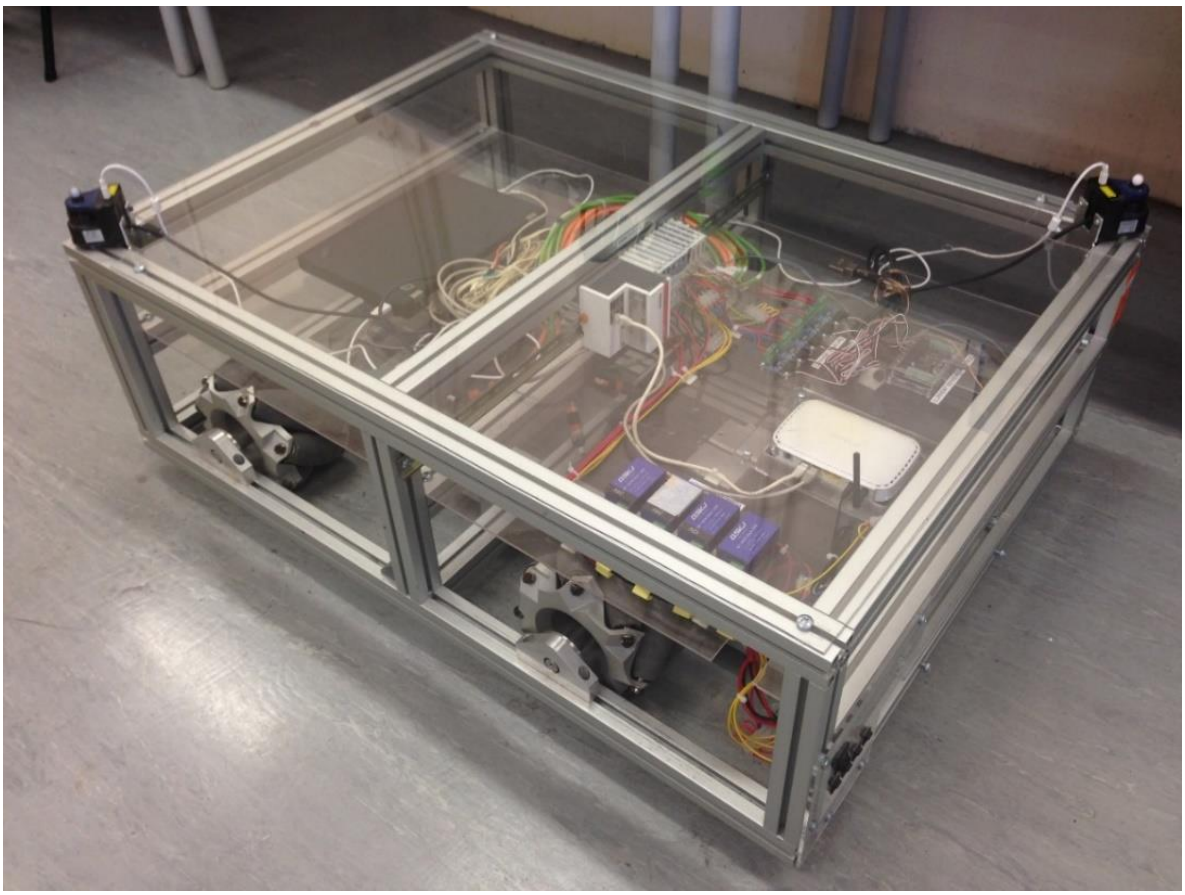


Figure 26 Version 3 of the robot. This version is the final form of the robot. The robot has a size of 1.2m × 0.9m × 0.4m. There are three modular layers inside the robot. The top layer of the chassis only has two laser scanners. The middle layer contains the major electronics components and the bottom layer takes the responsibilities of the mechanical transmission system and stores the robotic battery. More details can be found later.

Last but not the least, the final design of the robot was finalized as version 3. Because the second version had already met the research requirements and experimental standards, the third version of the robot focused on the maintenance, organization, grooming and most importantly design finalization. The robot conducts heavy-duty tasks in lab environments, so it is necessary to take regular maintenance, especially for the mechanical parts. The organization for electric components, wires and battery save more space and is planned for future add-ons. Besides, everything is isolated inside the robotic casing with transparent materials so that the on-board devices are not disturbed by the outside environment and the working state of the robot can be observed all the time. Only laser scanners were selected out of the exteroceptive sensory collections as the sole environmental sensor in Figure 26. An illustrated CAD model of the Mecanum mobile robot was built in PTC Creo. The CAD model is shown in Figure 27.

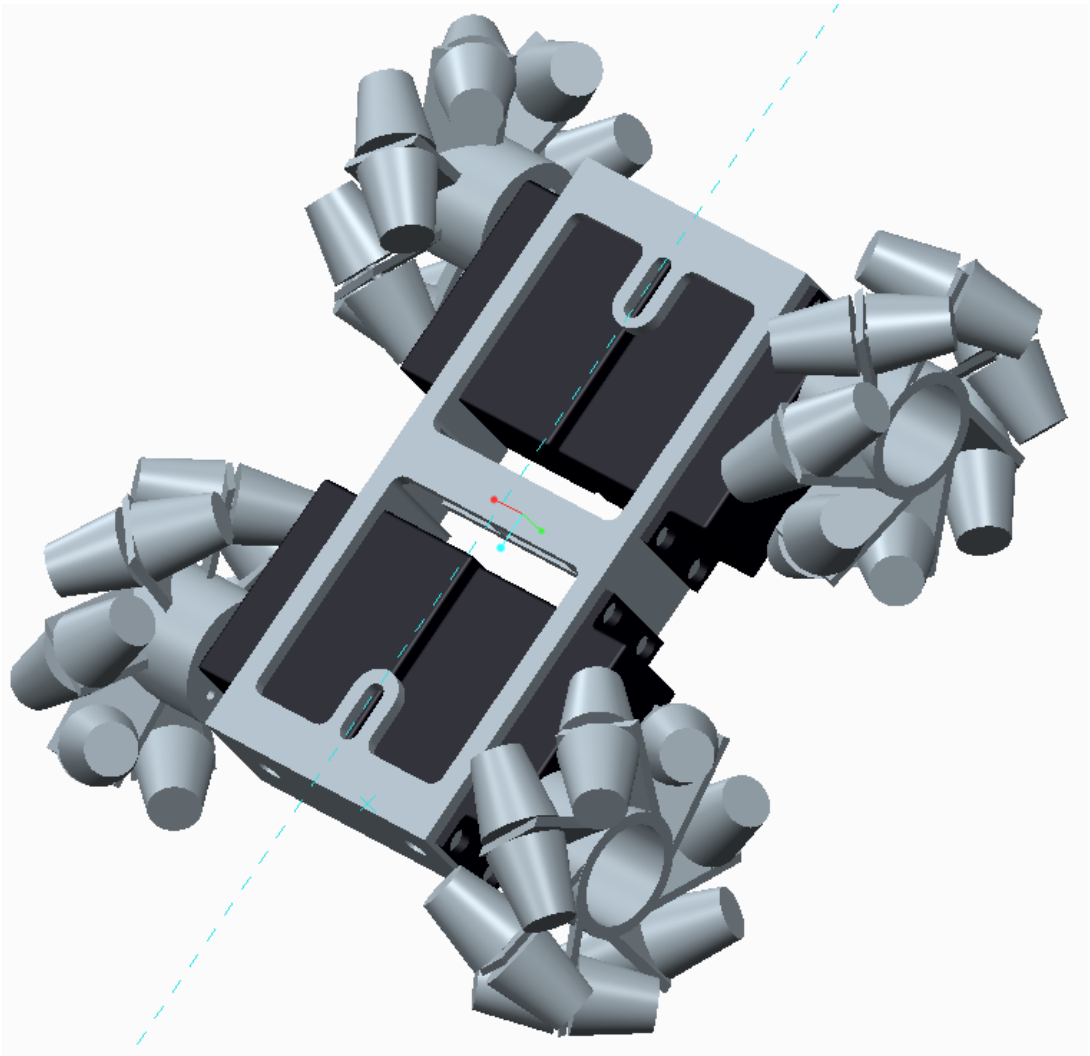


Figure 27 CAD model of the Mecanum mobile robot, built by PTC Creo. It is a simplified model of the designed robot, without detailing the robotic electronics components. Only a cuboid chassis containing four servomotors and assembling four Mecanum wheels is represented in the CAD model.

In the rest of this chapter, the full descriptions of the setup and the special features of the designed Mecanum robot are introduced. The focus lays on the robotic hardware development, locomotion control system, setup and configuration of the navigation system on the robot, and the interface between the navigation system and the locomotion control system.

3.2.3 Design Requirements

The robotic design aim is to design a heavy-duty omnidirectional four-wheeled Mecanum autonomous navigation mobile robot. The following technical requirements were met in this project.

- Hardware: A Mecanum mobile platform that is able to carry 400 kg loading weight.
    - Robotic chassis
        - It must provide enough space to carry all the robotic components and must have spare space for future expansion.
        - It must be able to load up to 400 kg weight.
    - Mechanical transmission
        - It must be able to stably deliver the motor rotating power to rotate the Mecanum wheels and to be rigidly assembled on the robotic chassis.
        - It consists of a motor base, bearing housing, wheel hub and drive shaft.
    - Servomechanism
        - It is a completed system that consists of motors, gearbox, drives and central controller
    - Robotic electric system
        - It provides electric energy to power up the entire mobile robot.
        - This consists of the battery and electric circuit design
- Software: A robotic control system
    - Controller-PC based locomotion control system
        - The locomotion control system controls the rotations of the servomotors to execute the given omnidirectional motion commands.
    - Navigation-PC based autonomous navigation system

- The autonomous navigation performs the full autonomous navigation function and outputs the omnidirectional motion commands.

- A combined robotic control system
  - A combined robotic control system comprises the Controller-PC based locomotion control system and Navigation-PC based autonomous navigation system.

3.3. Electromechanics

The robotic electromechanics supply motive power for the Mecanum mobile robot. The battery-based source provides electric power to the servomotors that rotate the driving Mecanum wheels via the designed transmission system.

3.3.1 Mechanical Design – Mecanum wheel, Chassis and Transmission System

Mecanum Wheel

The Mecanum wheel, shown in Figure 28, is described as an active wheel with a designed number of passive rollers attached around its circumference at a certain angle – conventionally, that angle is 45°. The geometry of the roller is specially curved so that the side shape of the Mecanum wheel remains circular. Four heavy-duty Mecanum wheels were provided at the beginning of the project. Each has a weight of 8 kg, a radius of 0.11 m and a width of 0.15 m. There are seven rollers, angled at 45° and attached surround the wheel circumference. The maximum weight that each wheel can support is 100 kg.

Figure 28 The Mecanum wheel of the designed robot. The heavy-duty Mecanum wheel supports 100 kg loading weight and it has a weight of 8 kg, a diameter of 0.22 m and a width of 0.15 m. There are 7 rollers attached to the wheel at 45° degrees, on both sides of the rollers for heavy-duty purposes.

Robot Chassis

The chassis of the robot is designed to support up to 400 kg weight. The dimension of the chassis is 1.2 m x 0.9 m x 0.4 m, as shown in Figure 26. The chassis is assembled by T-slotted aluminium extrusion solutions.

Transmission System

The function of the transmission system is to deliver the motor rotating power to rotate the Mecanum wheel. The designed transmission system was assembled on the robotic chassis. The transmission system consists of a motor base, bearing housing, wheel hub and motor shaft, as shown in Figure 29 and Figure 30. The Beckhoff servomotor is fixed on the robotic chassis by a right-angle motor base. Each Mecanum wheel is mounted on a motor shaft by a wheel hub. Two tapered roller bearings with houses support the motor shaft on both sides of the Mecanum wheel. Motor base, wheel hub and motor shafts are made of stainless steel. Bearing houses are made of aluminium.

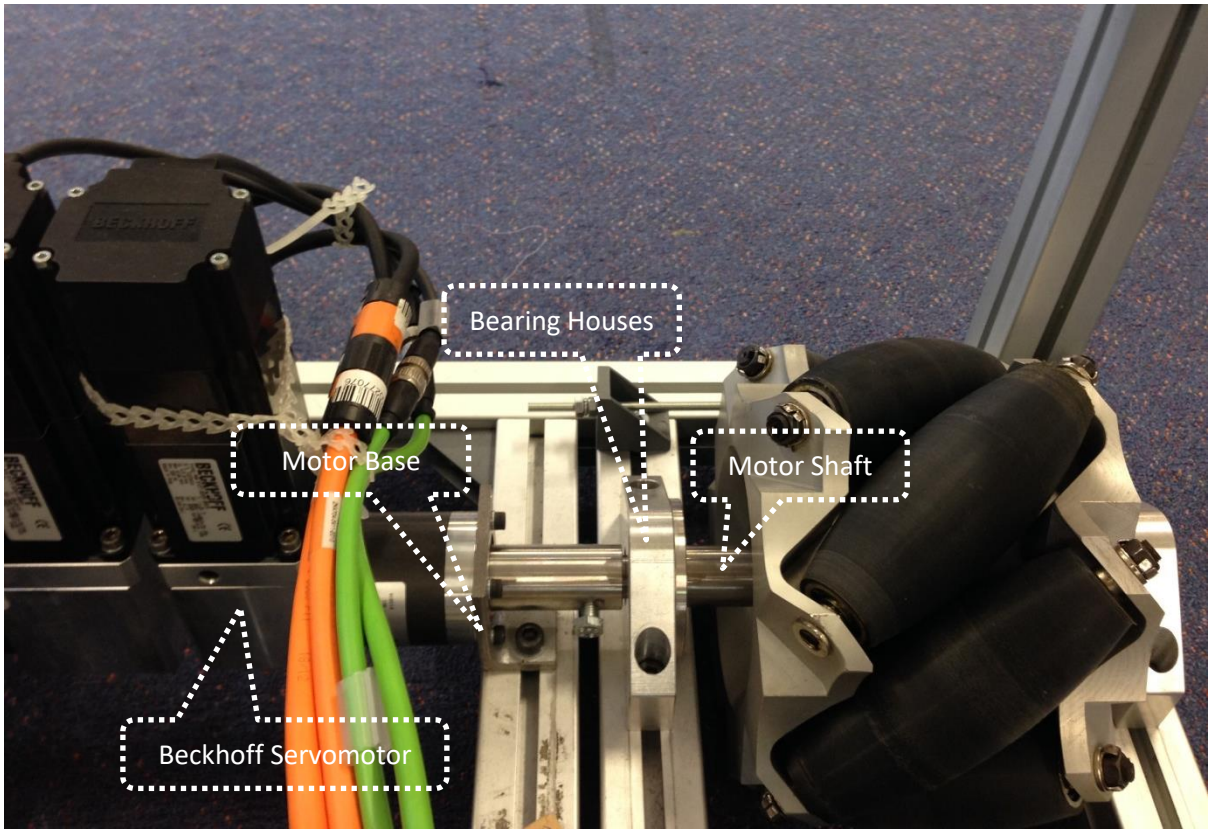


Figure 29 Transmission system of the robot. Every Beckhoff Automation servomotor is attached to the robotic chassis by the motor base. A motor shaft links the servomotor and the Mecanum wheel. Two bearing houses support the motor shaft on both sides of the Mecanum wheel and are assembled on the robotic chassis.

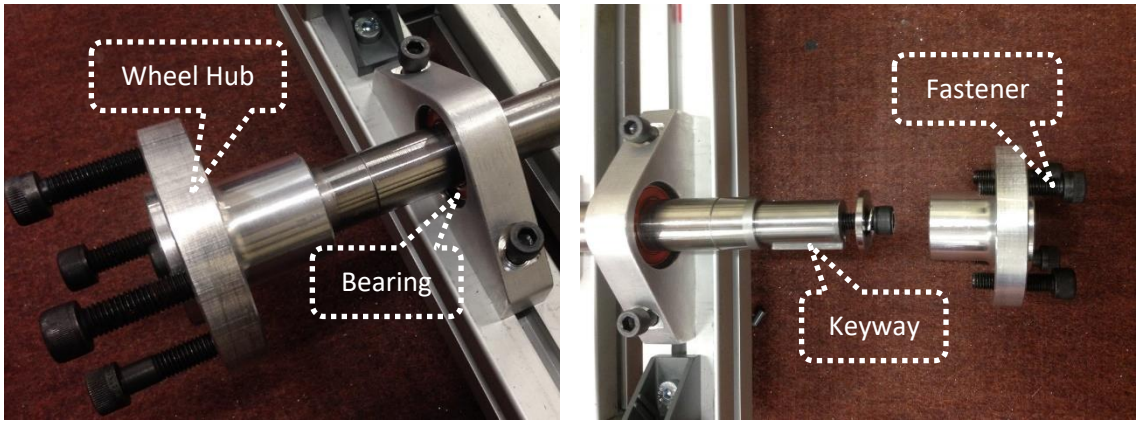


Figure 30 The wheel hub of the robot. The wheel hub is used to attach the Mecanum wheel to the motor shaft, with four fasteners. At the end of the motor shaft, there is a designed keyway to assemble the wheel hub.

3.3.2 Servomechanism

Four Beckhoff AM3121 servomotors including gear units independently provide rotating power to rotate each of four Mecanum wheels. The supply voltages of the embedded PC controller and the servomotors are 24 V and 48 V, respectively. The company Beckhoff provides off-the-shelf servomechanism solutions, including the motor drive and motor.

Servo Drive

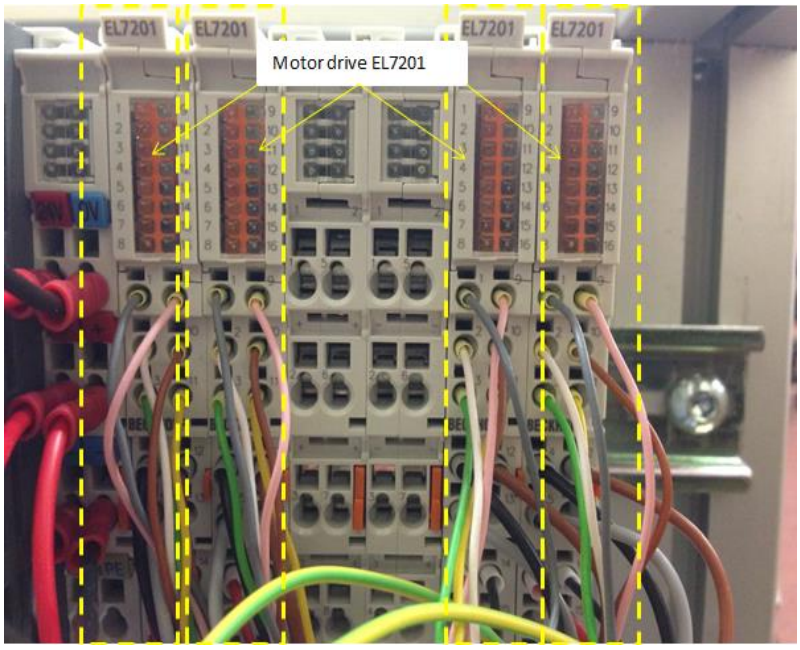


Figure 31 Servomotor terminal EL7201.

The servo terminal EtherCAT EL7201 shown in Figure 31 works as the motor drive, which connects the embedded PC controller to each servomotor. The main responsibilities of the

servo terminals include fast control technology, industrial PLC control, monitoring parameters (voltage, current, motor load) and compact PC-based controller interface.

Servo Motor

In Figure 29, a synchronous servo motor is utilized to power-rotate each Mecanum wheel and thus, to drive the Mecanum robot. The rated torque is 15 Nm, the maximum power is 140 W, the required voltage supply is 48 V dc, and the rated current is 4.6A.

3.3.3 Electric design

Energy Source

To supply electric power to the mobile robot, an on-board battery is a common and feasible option of energy source. The power consumption of the robot ranges from 80 W to 100 W. The servomotor requires at least 48 V to operate, so a capacity of 20 AH is a proper amount of electric charge for the battery to drive the robot for long hours. For fast charging and long-life cycle, the LifePO4 battery was chosen and purchased as the energy source for the robot.

Circuit Diagram

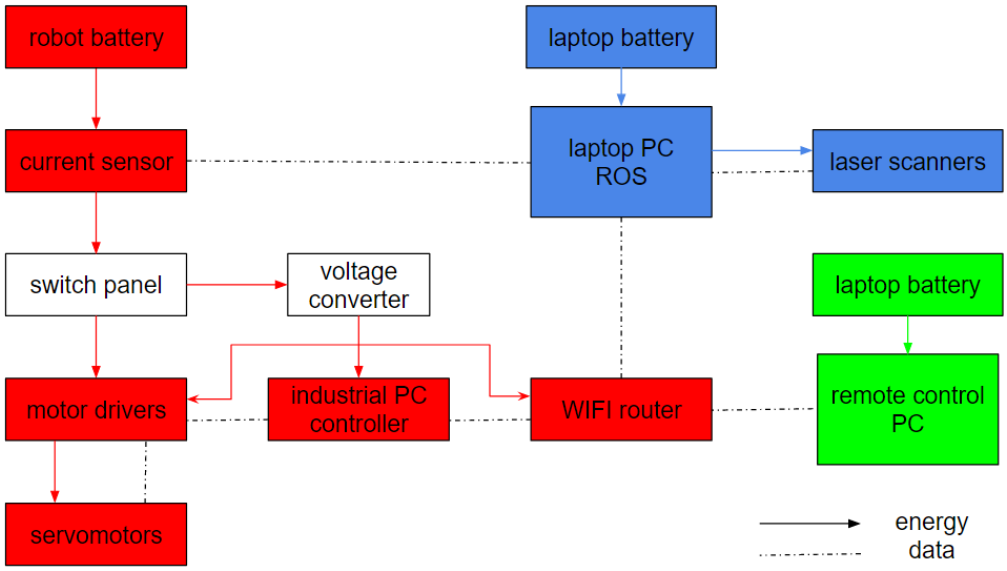


Figure 32 The electric circuit diagram of the Mecanum robot. The arrow vectors show the energy flow and the dash-dotted lines represent the data communication. Three independent electric circuits exist in the Mecanum robot. Most importantly, the red colour electric circuit represents the main energy consumption of the robot. The battery provides the electric energy and the current sensor measures the amount of robotic electric energy consumption. Most of the energy is consumed by the Beckhoff automation system. The blue colour electric circuit represents the energy consumption of the ROS PC and the green colour electric circuit represents the energy consumption of the remote-control PC.

Figure 32 presents the final electric circuit design of the Mecanum robot. The robotic on-board battery mainly provides the electric energy for the Beckhoff automation system including the industrial PC, motor terminal drives and servomotors, and the on-board electronic devices including the WIFI router. Another two laptop PCs are used for running the ROS navigation system and robot remote control. Both laptop PCs have their own laptop batteries. The laser scanners are connected directly to the ROS laptop PC for energy supply and data exchange. The colours red, blue and green represent three electric supply systems. Both switch panel and voltage converter have extremely high energy efficiencies so that they are assumed to have no energy consumption and are marked in white. The current sensor is used to measure the energy consumption of the robot, for the purpose of research. The servomotors require 48V voltage; the industrial PC and the motor drivers require 24V voltage, and the wireless network router requires 12V voltage power supply, so three DC-to-DC voltage converters are utilized on the robot. The laser scanners can be powered via the USB of the ROS laptop PC. The wireless network router is responsible for connecting all three sub-systems together, which are the industrial PC, the ROS PC and the robot remote control PC. An illustration of the robot with labelling of the main components is shown below in Figure 33.

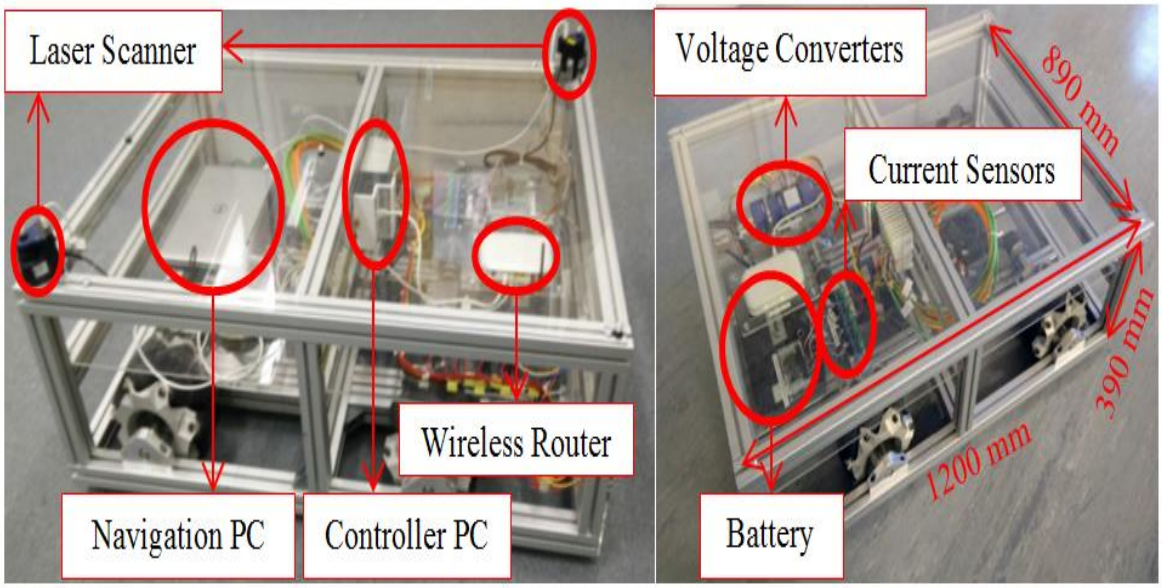


Figure 33 Main components of the robots. On the top layer of the chassis, two Hokuyo UBG-04LX-F01 scanning laser rangefinders are attached on two diagonal corners for localization and obstacle avoidance purposes. On the middle layer, a Linux-OS notebook Navigation PC running the ROS Navigation Stack receives sensory data from the laser rangefinders via USB and is connected to the Beckhoff Automation industrial PC via Ethernet. The industrial Controller PC is placed in the centre of the robot and its main task is to control the Beckhoff Automation servomotors. The blue boxes are the 48-to-24 voltage converters. A white wireless router connects the ROS PC, the industrial PC and most importantly, the user command PC for remote control purposes. To the top right of the router, there are the current sensor and its drive kit. On the bottom layer, there are a single 48-volt battery as the robotic energy source, four servomotors, four Mecanum wheels and the mechanical transmission system.

3.4. Locomotion Control System

In order to realize omnidirectional motions on the Mecanum robot, the locomotion control system assigns the corresponding rotation tasks to each motor according to the kinematics of the Mecanum wheel. The electromechanics of the robot can drive each Mecanum wheel independently by an exclusive servomotor. The Beckhoff-based controller is in charge of the locomotion control system. An overview of the locomotion control system is shown below in Figure 34. To avoid confusion, it is worth mentioning that even though the schematics of the locomotion control system below shows end-supported Mecanum wheels, the physical AuckBot and this research study focus on centrally supported wheel arrangement.

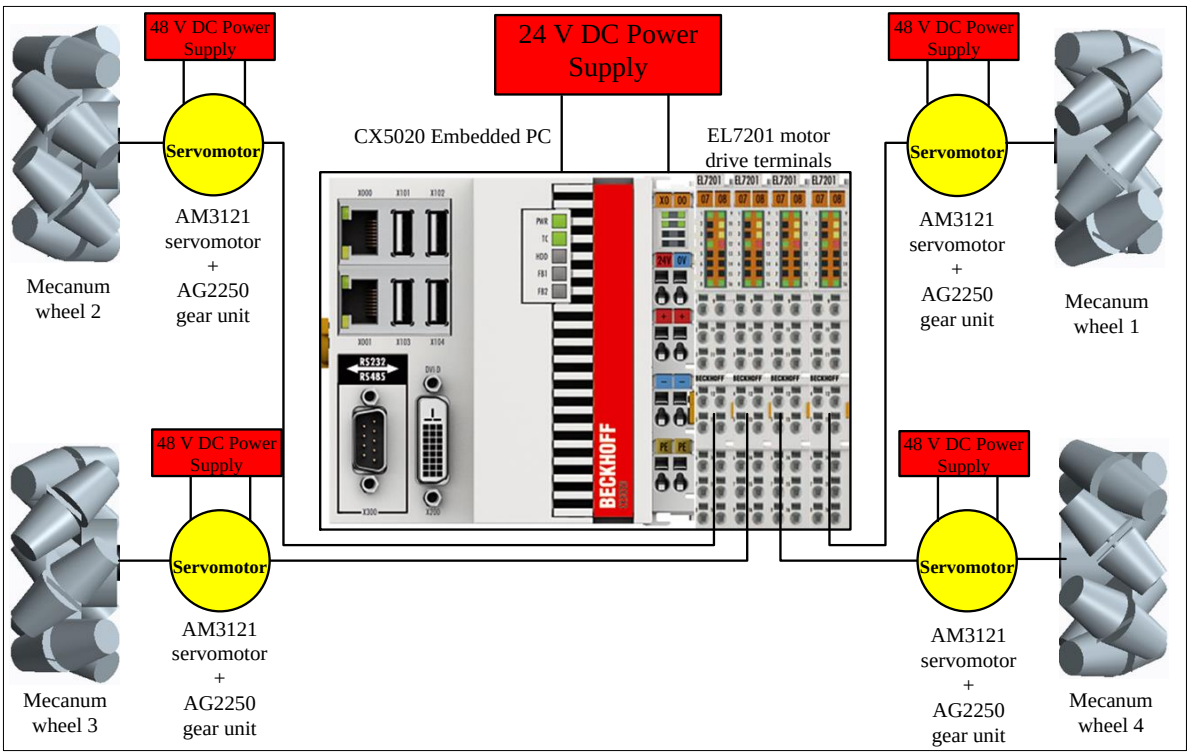


Figure 34 The schematics of the locomotion control system. The Beckhoff Automation Industrial CX5020 Embedded PC is the controller of the locomotion control system. The embedded PC is powered by a 24-volt DC power supply and it connects to four EL7201 motor drive terminals. Each drive terminal directly controls a servomotor to rotate the Mecanum wheel. The servomotors are powered by a 48-volt DC power supply.

3.4.1 Beckhoff Automation

The locomotion control system is a design based on Beckhoff Automation technologies. Beckhoff Automation provides automation technology solutions in the modern industrial standards and Beckhoff technology serves for accurate and stable motor controllability. Figure 35 shows the technology solutions provided from Beckhoff Automation and used on the robot. The Beckhoff CX5020 embedded PC-based Controller-PC runs a TwinCAT PLC program that

applies NC-PTP, a numerical controller point-to-point function via the EL7201 motor drive terminals to operate the motor axis. The TwinCAT NC motion control PLC library provides a function block that was standardised by PLCopen to access NC-PTP.

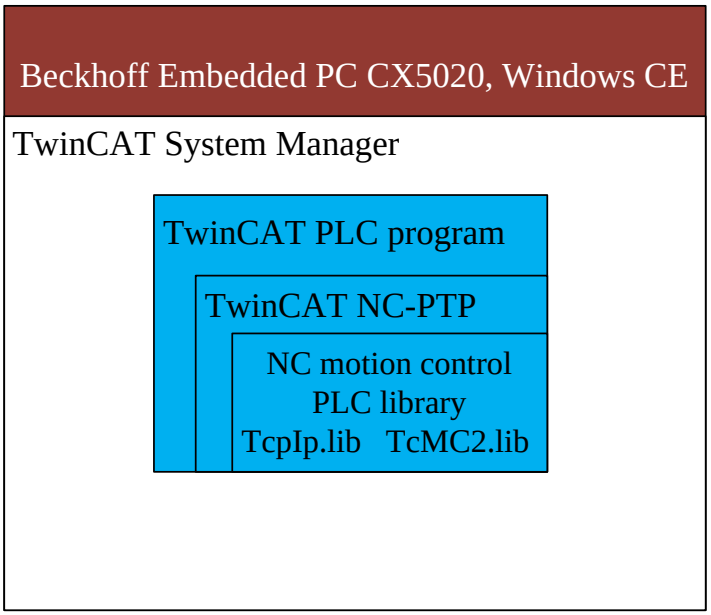


Figure 35 Beckhoff Automation solutions on the robot.

3.4.2 Beckhoff Servomotor Control

The functionality of the Beckhoff servomotor works as Figure 36 illustrates. The TcMC2 library offers PLCopen standardised MC_MoveAbsolute and MC_MoveVelocity function blocks. Depending on the received either positioning of the wheel axis from MC_MoveAbsolute or velocity control of the wheel axis from MC_MoveVelocity, the PLCopen library generates the desired encoder position of the wheel axis. Via a position control with a P controller and a velocity control with PID, either positioning or velocity control at customised rates will be achieved as shown in Figure 36 below.

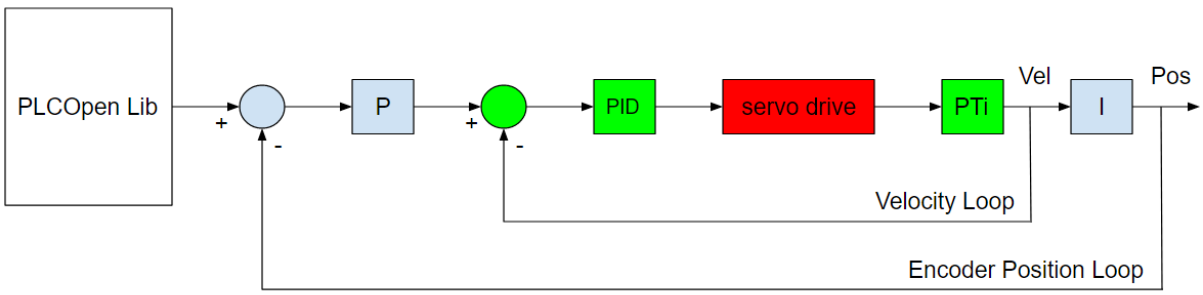


Figure 36 Beckhoff servomotor motion control system. It includes a position control with a P controller and a velocity control with a PID controller.

3.4.3 Locomotion Control of the Mecanum Robot

The locomotion control system of the robot is shown in Figure 37. According to the kinematics of the Mecanum robot, the Beckhoff embedded PC assigns the rotation tasks to each servomotor. The reverse kinematics model translates the robotic omnidirectional motion into each wheel's motion commands.

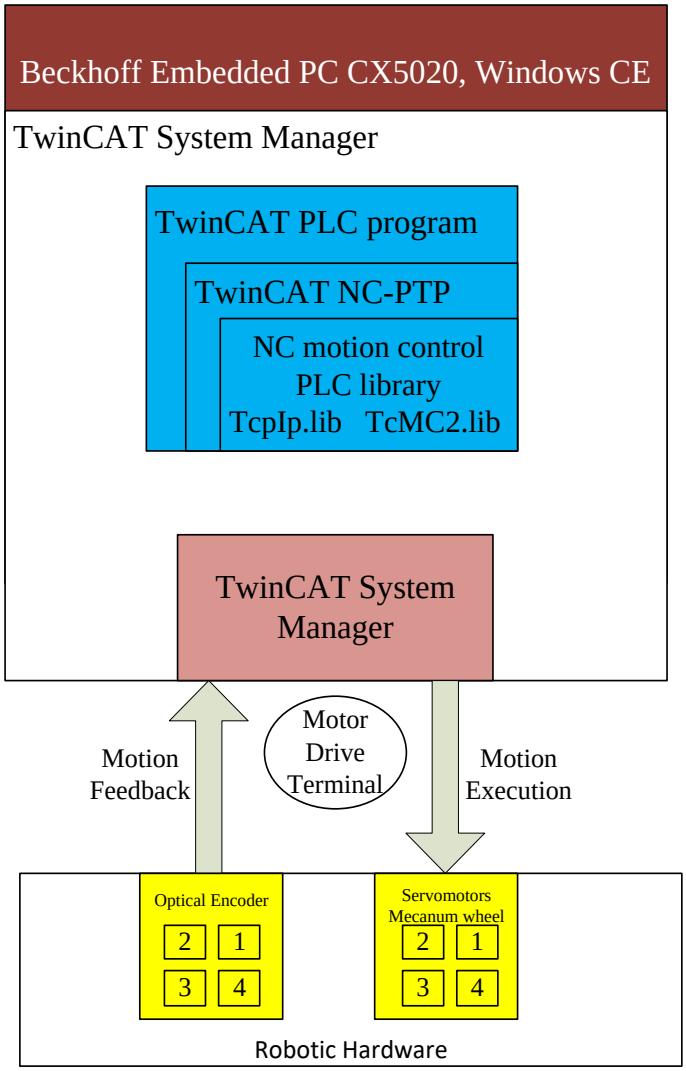


Figure 37 Locomotion control of the robot.

The robotic electromechanics transmit electric power of the servomotor to the Mecanum wheels. Thus, the robot can perform omnidirectional motions and the Beckhoff embedded PC-based locomotion control system is able to drive the Mecanum mobile robot as commanded.

3.5. Autonomous Navigation System

To implement fully the autonomous navigation function on the robot, an autonomous navigation system was accomplished to perform perception, localization, obstacle avoidance,

trajectory planning and trajectory following. The ROS robotics software framework was applied to implement the autonomous navigation system on the designed prototype robot, in this PhD research project as shown in Figure 38.

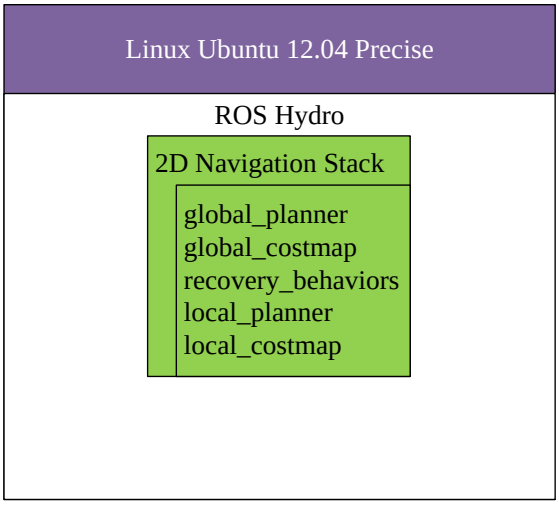


Figure 38 ROS with 2D Navigation Stack.

3.5.1 Robot Operating System

Recall that the Robot Operating System (ROS) is a robotics software framework that provides open source packages to implement functionalities of intelligent robots on the robotic platform. The functionalities of the developing intelligent robots are evolving and improving fast. Implementation of reliable and robust software in the robotic system to behave intelligently is a complex task. Development from scratch and continuous update consume a lot of effort and time. An insignificant behaviour to humans tends to be very sophisticated implementing as robotic behaviour, so ROS provides an open community to the robotics developers over the world and encourages collaboration from each other.

3.5.2 ROS Graph Concepts

Running in the Ubuntu Linux system, the ROS has a structured communication among multiple ROS *nodes*. A ROS *node* is an executable process. The *roscore* containing a collection of programs is the master of the ROS-based system that manages the *node* communications. The methods of communication with one *node* another include *topic*, *service* and *parameter server*. A robotics system usually has lots of *nodes*, merging into a tree-like graph communication structure.

Topics exchange data in *messages* over *nodes*, working as one-way buses for unidirectional streaming communication. A *message* is a ROS data structure of many types. *Nodes* do not directly communicate with each other. Instead, *nodes publish* to a *topic* to generate relevant data and *nodes subscribe* to a *topic* to receive relevant data. Instead of the one-way communication by *topic*, *service* offers request/reply interaction communication over *nodes* for a distributed system. A *node* sends a request *message* to and awaits a reply *message* from a node that offers a service. The *Parameter server* is designed to easily and globally access static data such configuration parameters. The *Parameter server* can save and provide data as a note book for *nodes* at runtime. It can be reached via network APIs by *nodes*.

3.5.3 2D Navigation Stack

The ROS 2D navigation stack provides multiple packages to implement autonomous navigation functions on an arbitrary mobile robot. An overview of the architecture of the ROS 2D navigation stack is presented below in Figure 39. Overall, the navigation stack receives a goal pose and then continuously generates motion commands to a mobile robot; meanwhile, the navigation stack constantly takes in information from odometry and sensors, for the purposes of localization and environment perception.

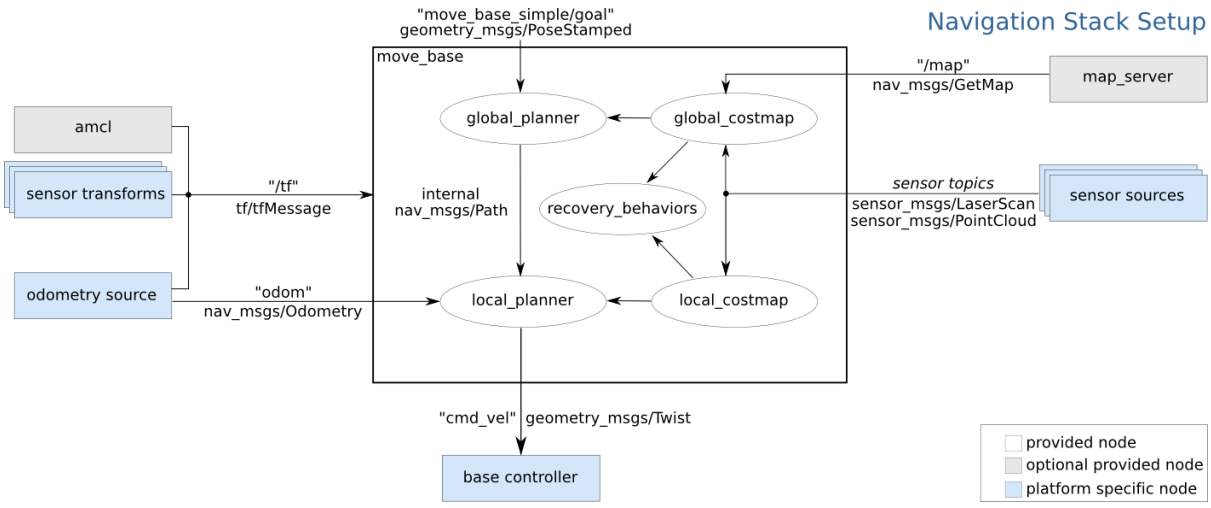


Figure 39 2D Navigation Stack [78].

The *move_base* package contains a collection of essentially provided nodes that implement general-purpose autonomous navigation functionality on any qualified mobile robot. In a navigation task, *move_base* navigates the *base_controller* of the mobile robot to the given goal pose by associating the *global_planner* with the *local_planner*. The *global_planner* finds a feasible global path to reach the goal pose in the global map from the *global_costmap*. Based

on the localization information, the *local_planner* computes the velocity commands to follow the given global path in the local map. The local map is constantly updated in the *local_costmap* using online sensor sources. The *global_costmap* with the predefined global map gives reference to the *local_costmap* to build the local map while the online local map in the *local_costmap* updates the global map in the *global_costmap*. The *recover_behaviors* will take place if the move_base fails during a navigation task.

Outside the *move_base* block, the *amcl* node serves as a particle filter localization system for the mobile robot to localize in a given map based on the adaptive Monte Carlo localization approach. The mobile robot may have a global coordinate frame, a robot coordinate frame, and a laser scanner coordinate frame, etc., so the package *tf* offers the transformation calculation over these coordinate frames in a tree-like structure. An *odometry source* is required to obtain and publish velocity information of the robot. The *map_server* node saves the global map data generated by a SLAM mapping *service* and offers the global cost map *service* to the *global_costmap*.

3.6. Control System Architecture

A robotics control system architecture, consisting of a low-level Controller-PC and a high-level Navigation-PC, was implemented on the designed robot in this research project. The Beckhoff embedded PC-based locomotion control system that executes the motion commands acts as the Controller-PC in the robotics control system architecture and governs the locomotion of the robot platform system. With the intelligent robotics framework on top of the locomotion control system, ROS acts as the Navigation-PC providing current state of the art autonomous navigation functionality. In order to interface both systems, both the Beckhoff system and the ROS had to be deeply studied to come up with a feasible way via TCP/IP to enable data communication.

3.6.1 Controller-PC

In the robotic control system, the Controller-PC manages the motion command executions of the robotic mobile base. The Beckhoff embedded PC was assigned as the Controller-PC to govern the locomotion control system. In the Controller-PC, TwinCAT transforms the robotic velocity into the motor velocities so that the locomotion control system can execute the velocity commands. The locomotion control system is able to drive the Mecanum mobile robot as commanded by either manual operation or automation system. The Beckhoff technology

allows the robotic electromechanics to control the wheel axis and the locomotion control system executes the omnidirectional motion commands. The optical encoders return the shaft rotations as motion feedback to the Controller-PC. The Beckhoff embedded PC running TwinCAT is assigned as the Controller-PC as shown in Figure 40.

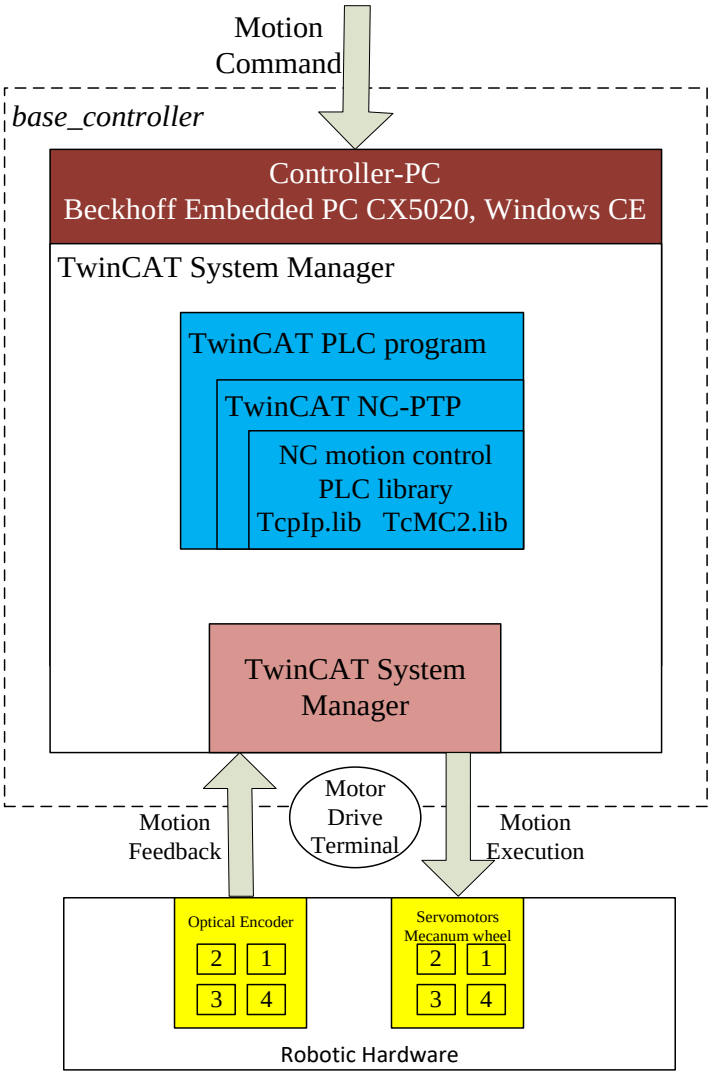


Figure 40 Controller-PC.

However, it is beyond the computational capability of the provided Beckhoff system to perform advanced autonomous navigation tasks. It is also both time-consuming and costly to create a new navigation system in the Beckhoff system.

3.6.2 Navigation-PC

The Navigation-PC performs complex autonomous navigation tasks such as perception, localization, cognition, navigation and motion planning. ROS on which leading research in the robotics field is developing and whose active community offers free open source code, was

utilized as the intelligent robot framework of autonomous navigation. In this project, a laptop PC installing the Ubuntu Linux system and running ROS was assigned as the Navigation-PC. The ROS navigation stack program runs in the Navigation-PC. As shown in Figure 41, a higher-level Navigation-PC on top of the Controller-PC deals with autonomous navigation tasks and outputs the resultant motion commands to the Beckhoff-based locomotion control system to execute.

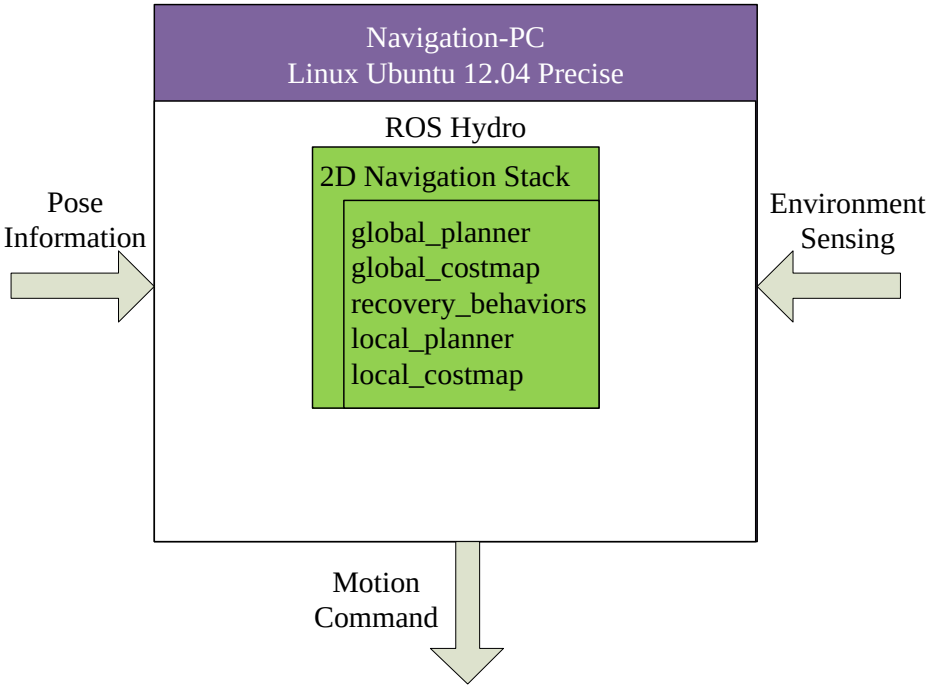


Figure 41 Navigation-PC.

3.6.3 Interface

In order to combine the Controller-PC and the Navigation-PC as a united system, a feasible communication method to exchange data is required. An overview of the interface is shown below in Figure 42.

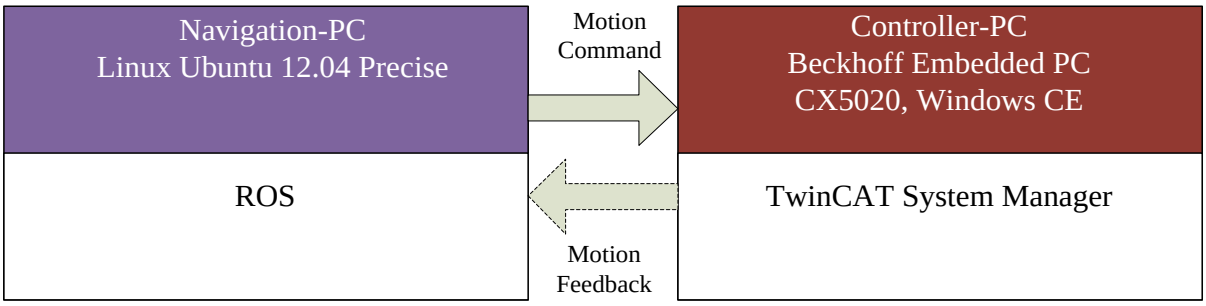


Figure 42 Interface between Controller-PC and Navigation-PC.

The Navigation-PC sends the desired robotic velocity as the motion commands to the Controller-PC. In the Controller-PC, the embedded PC takes care of the low-level motion execution task. TwinCAT transforms the robotic velocity into the motor velocities so that the locomotion control system can execute the velocity commands. The shaft encoders of the servo motors feed the motor rotation information back to TwinCAT. Then the Controller-PC returns the motion feedback in terms of the robotic pose estimation to the Navigation-PC.

The TwinCAT remains executing the latest velocity command until the Controller-PC receives an updated velocity command from the Navigation-PC. ROS is not real-time but the TwinCAT controls the servo motors with a constant control cycle of 1 ms, so the data exchange between the Navigation-PC and the Controller-PC does not meet the real-time criterion as shown in Figure 43.

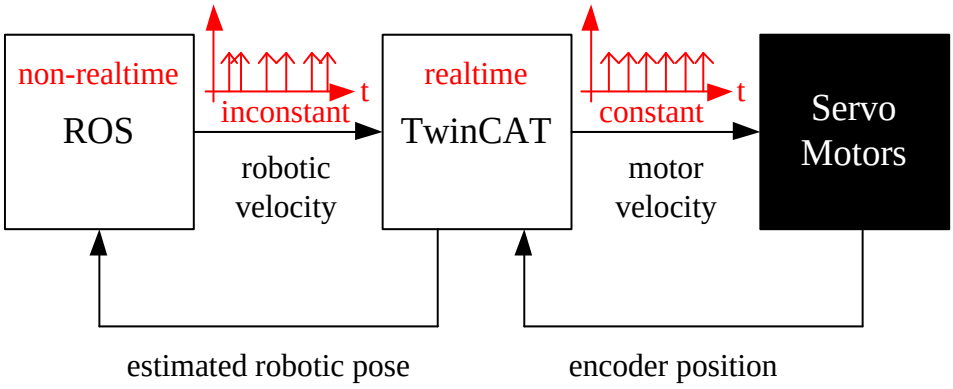


Figure 43 Non-realtime ROS and realtime TwinCAT.

3.6.4 Combined Control System

The combined robotic control system utilized industrial automation solutions to control the robotic locomotion and open-source robot framework for autonomous navigation functions on the current state of the art. Both the high-level Navigation-PC and the low-level Controller-PC are suitable for extensibility of future research. In order to interface both systems, TCP/IP is proposed to enable data communication.

3.7. Closed-Loop Motion Control

A closed-loop control system is required to compensate for the wheel slippage for an accurate and reliable motion trajectory tracking function. The Mecanum wheel suffers random slippage, which causes errors in the optical encoder dead-reckoning method that is commonly applied in

the position control of wheel mobile robots. This function is necessary for the Mecanum robot to properly operate and essential for conducting experiments.

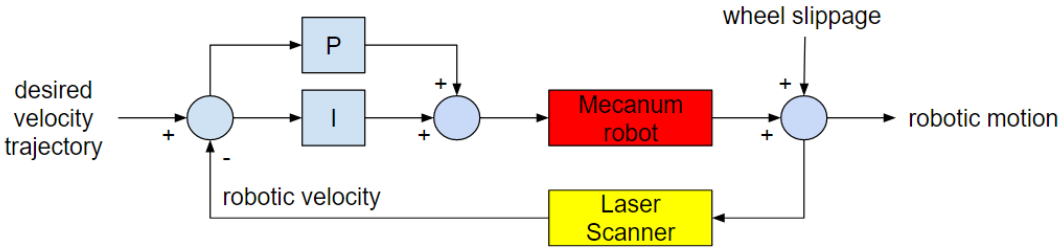


Figure 44 The PI closed-loop velocity trajectory control system.

Several means of real-time detecting of the robotic omnidirectional velocities have been practically applied and properly worked for the robot. The Vicon motion capture system is the most reliable way to feedback the robotic velocity trajectory to the robot. But the motion capture system requires the system infrastructure and it is only available in a busy laboratory. A Kinect camera has been used to estimate the robotic velocities via image processing techniques. The performance of Kinect-based visual dead reckoning is directly proportional to the amount of time and effort that was put into the image process tuning. Other methods involving drifting errors have also been used to function within short-distance situations for the robot, including mechanical spherical sensor [79] in Figure 45, an optical flow sensor in Figure 46, and IMUs in Figure 47, simple SLAM and Reactive Navigation using Kinect and IMU sensors [80], and a 2D Navigation Stack using Autonomous Architecture System using the Spherical encoder and optical flow sensors [1].

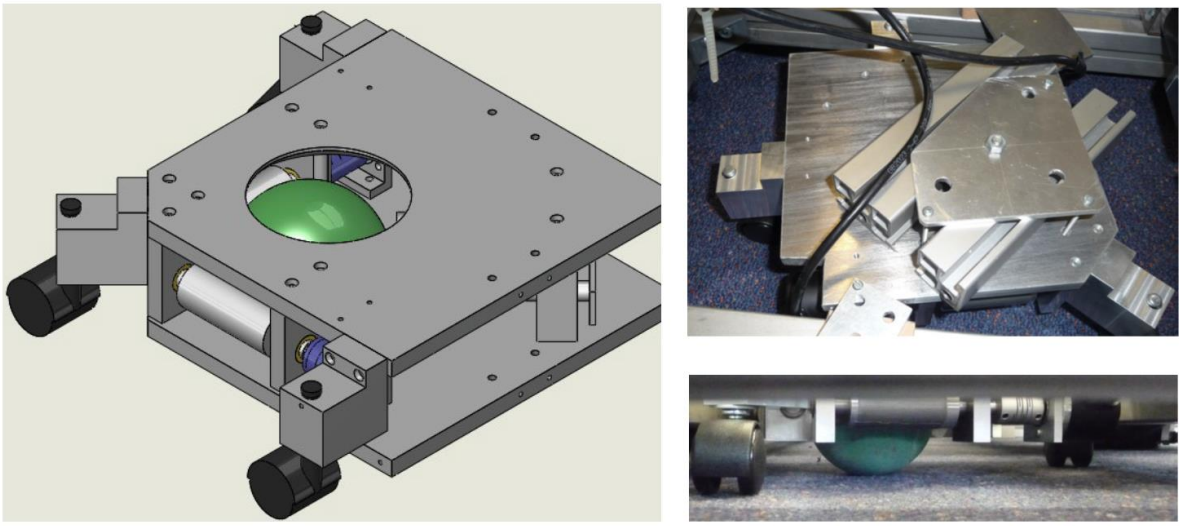


Figure 45 Spherical encoder design. This encoder imitates the mechanical mouse by containing a rubber ball rolling inside the sensor. Two sensors detect the motion of the rubber ball. [79]

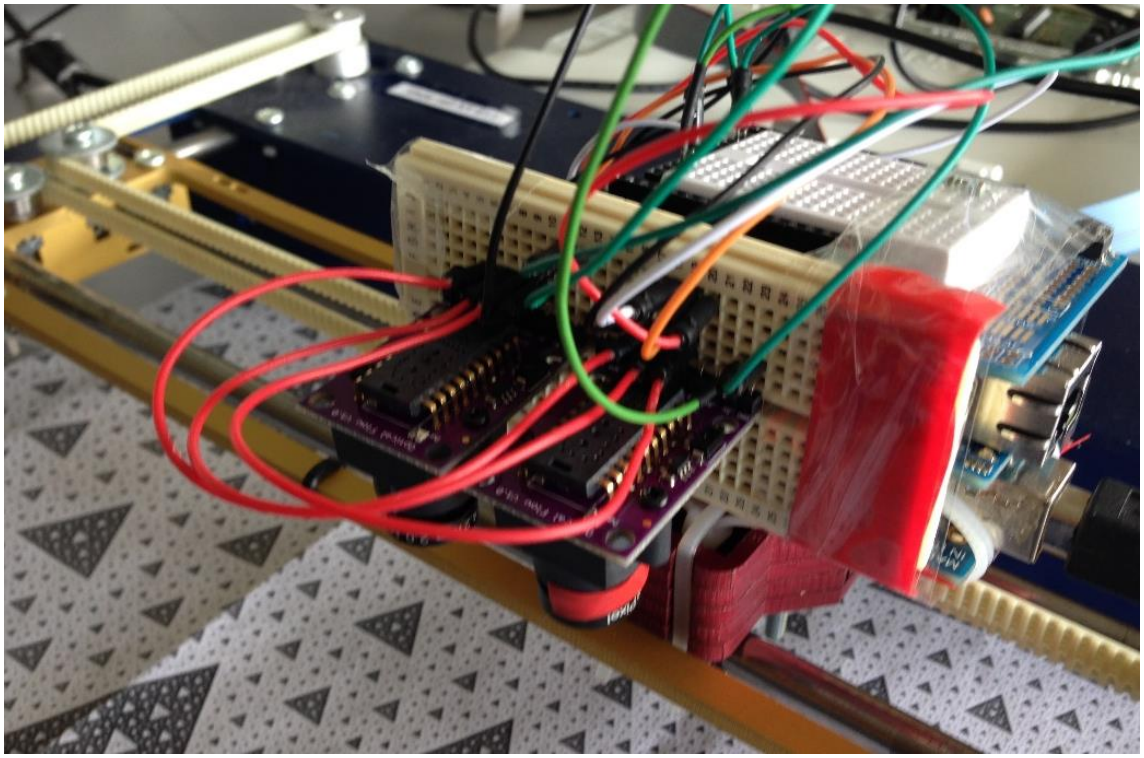


Figure 46 Optical flow sensor, tested using XY motor control table. Two off-the-shelf optical flow sensors are connected to Arduino to process data.

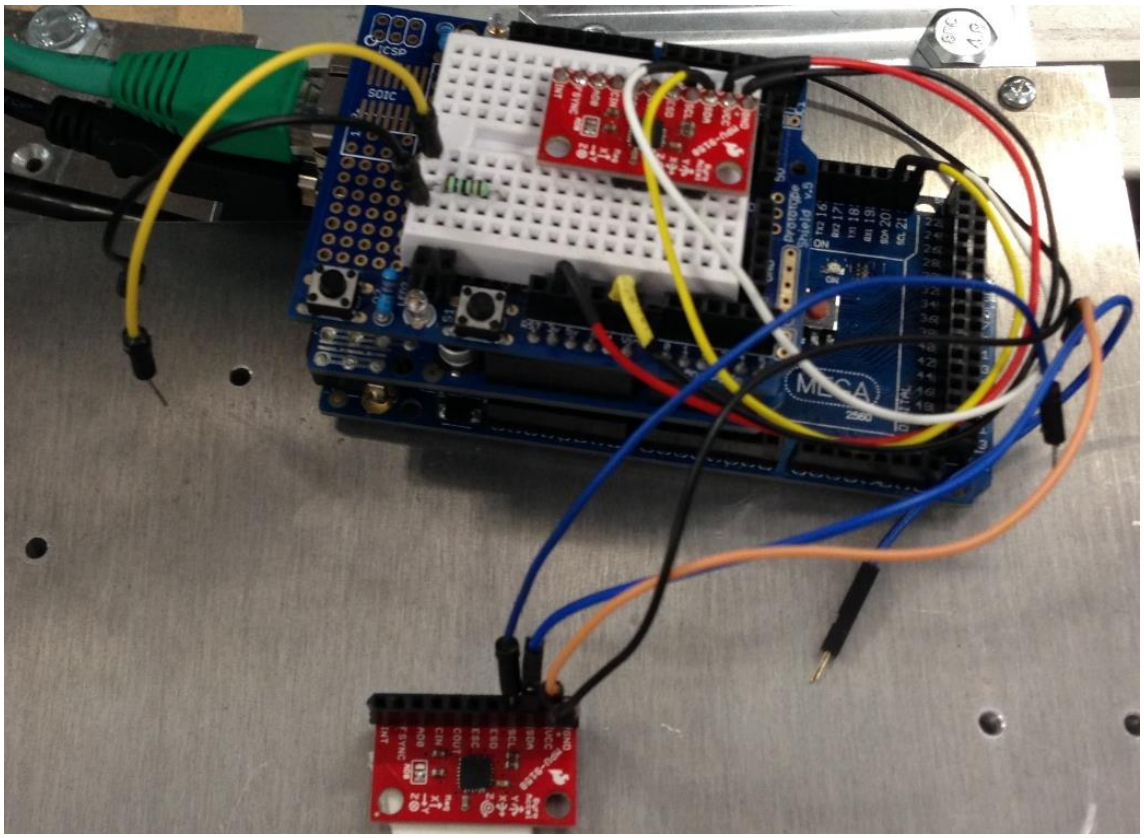


Figure 47 IMU interface with the robotic system via Arduino.

The final solution is to utilise the laser scanners, as shown in Figure 33. As a self-contained sensor, the laser scanners regularly detect the distance changes between the robot and the surrounding objects. The resultant velocity of the robot is computed via a ROS open-source package *laser_scan_matcher*, which can be used as a stand-alone odometry estimator.

The velocity trajectory of the robot is controlled using the PI controller. The omnidirectional velocities of the robot, consisting of v_x,v_y and v_{theta} are controlled independently by three different sets of PI tuning parameters.

3.8. Implementation

3.8.1 Controller-PC

The Beckhoff embedded PC runs the TwinCAT PLC program to perform its allocated locomotion control tasks as the Controller-PC in the robotic system. The primary task is to control the servomotors so that the Mecanum robot can move in any direction. Then the Controller-PC records the motion feedback of the Mecanum robot by reading the servomotor encoders.

Motion Execution by Positioning of Wheel Axis

The embedded PC runs the PLC program. The motor drive terminals connecting to the embedded PC constantly monitor and control four servomotors, each independently powering one Mecanum wheel via the transmission system. The control system of the servomotor positioning and the manual tuning approach can be found in [77].

In order to control the servomotors, both the TwinCAT PLC and the TwinCAT numerical controller (NC) axis control were used. The TwinCAT PLC provides a programming environment for the use of IEC61131-3 PLC programming language. The TwinCAT NC is associated with setting the axis control of the servomotors. With the library functions offered from the TwinCAT NC point-to-point (PTP), a simple servomotor velocity control PLC program was written in structured text. The TwinCAT PLC programming environment is shown below in Figure 48.

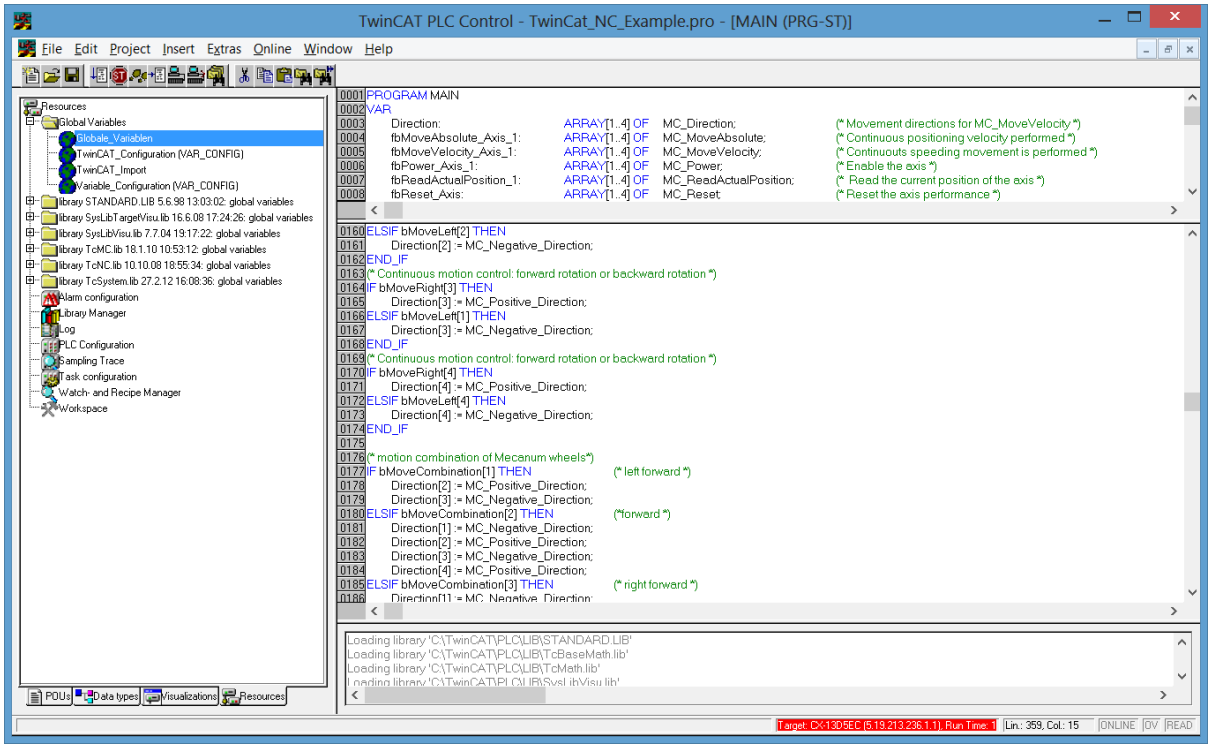


Figure 48 TwinCAT PLC programming environment.

TwinCAT NC configuration station within the TwinCAT System Manager is shown below in Figure 49. All the motor drive terminals are controlled via the NC Axes.

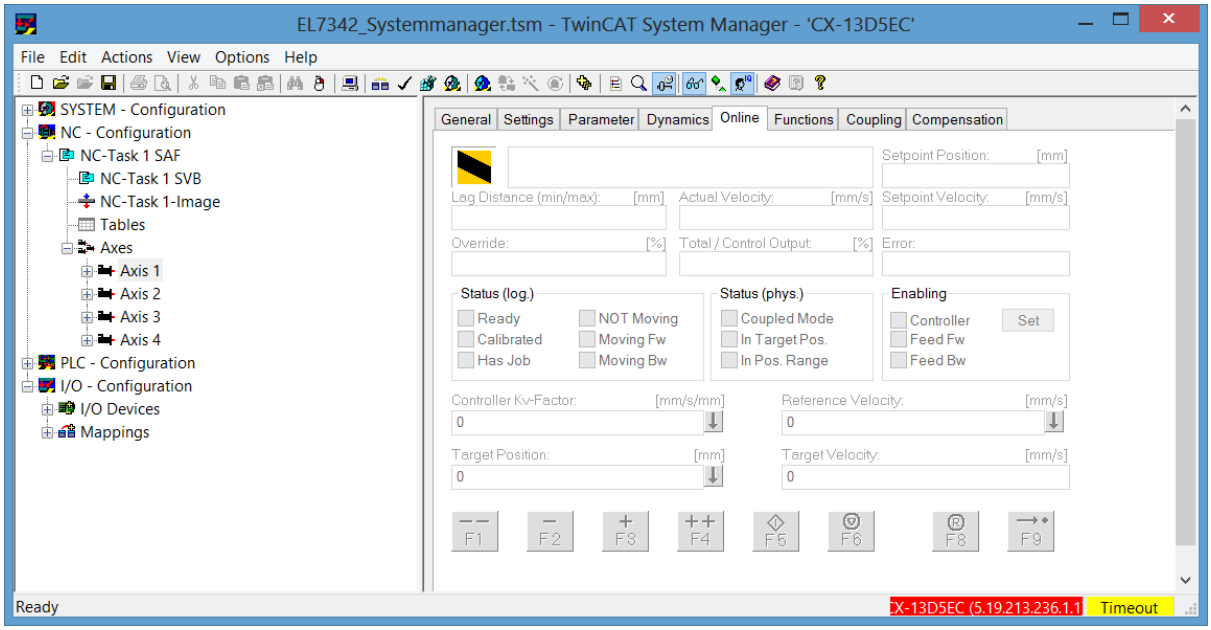


Figure 49 TwinCAT NC configuration station.

The TwinCAT NC-PTP function blocks from the Motion Control library shown in Figure 50 are used in this project and enable configuration between the TwinCAT PLC and the TwinCAT NC.

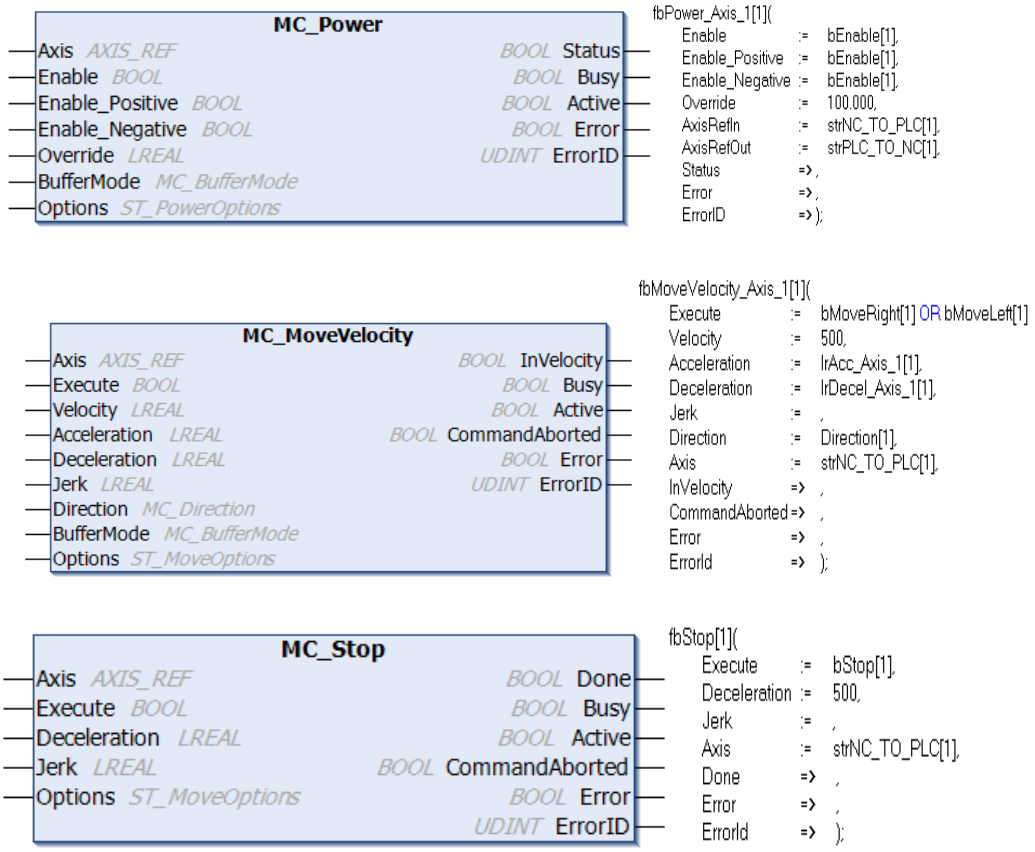


Figure 50 TwinCAT NC-PTP function blocks.

Motion Feedback and Behaviour Model

While the motion commands are being executed, the servomotors and the Mecanum wheels should be rotating to drive the Mecanum robot. After the shaft encoders of the servomotors start to detect the rotational information, the robotic system will know that the motion commands are under execution. The rotational information is converted into robotic motion as the motion feedback by the behaviour model. The motion feedback is returned to the Navigation-PC.

Both velocity and pose of the robot are interesting feedback to the robotic system. It is worth mentioning that motion feedback is not suitable for control purposes such as localization. Because the Mecanum wheel suffers the random and obvious slippage issue, encoder-based dead reckoning does not provide accurate pose estimation.

Assuming a slippage-free situation, the encoder readings are converted into the wheel rotation and further the robotic motion using the kinematics model. The robotic velocity $(\dot{x}, \dot{y}, \dot{\emptyset})$ is in relation to the robotic coordinate system and the robotic pose (X, Y, Z) is in relation to the global coordinate system as shown in Figure 51.

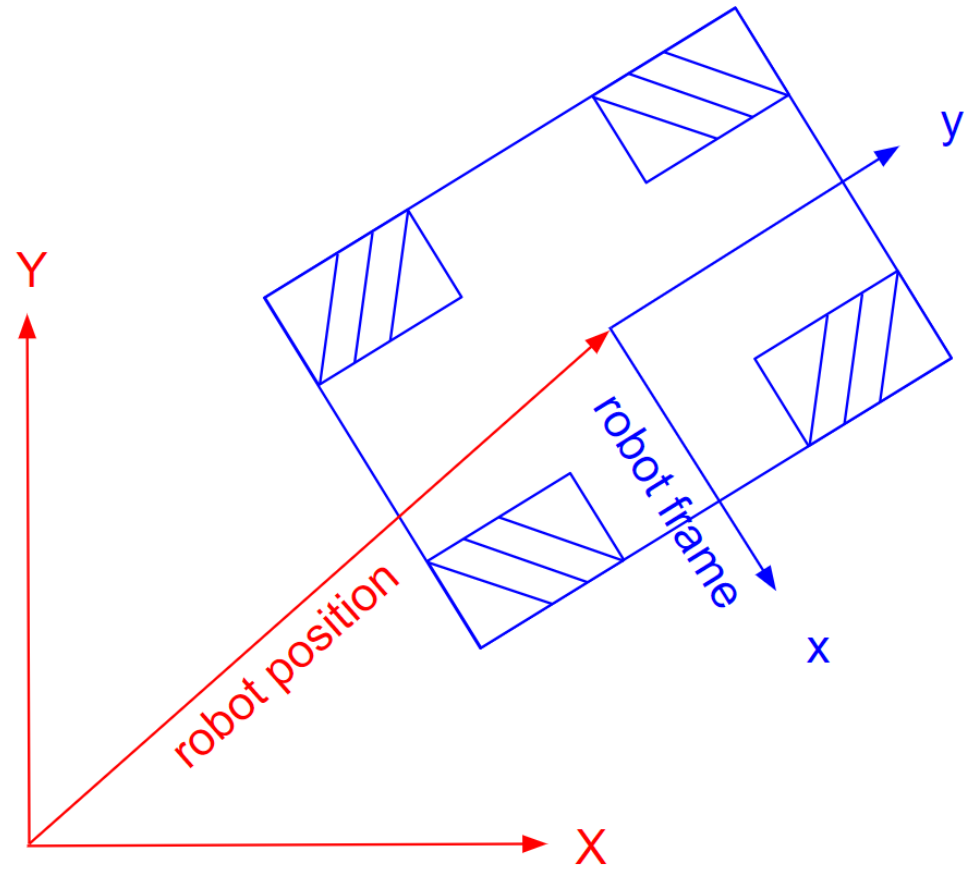


Figure 51 Global coordinate system.

The shaft encoders of the servomotors output the angular positions of the motor shaft to the motor drive terminals. After considering the gear ratio, the motor shaft rotations are recorded as the absolute wheel axis positions $AbsPosAxis_i(t)$ in the Controller-PC. The wheel rotational velocities $\dot{\theta}_i$ are then calculated as follows.

$$\dot{\theta}_i = \frac{AbsPosAxis_i(t) - AbsPosAxis_i(t - \Delta t)}{\Delta t} \tag{4}$$

Based on the kinematics model of the four-wheeled Mecanum mobile robot, the wheel rotational velocities are converted into the robotic omnidirectional velocities, as shown in Figure 20.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ \frac{1}{l_x + l_y} & \frac{-1}{l_x + l_y} & \frac{-1}{l_x + l_y} & \frac{1}{l_x + l_y} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix} \tag{5}$$

In order to receive the current robotic position, the robotic velocities need to be continuously integrated by time.

$$\begin{bmatrix} x_{world}(t+1) \\ y_{world}(t+1) \\ \emptyset_{world}(t+1) \end{bmatrix} = \begin{bmatrix} x_{world}(t) \\ y_{world}(t) \\ \emptyset_{world}(t) \end{bmatrix} + \begin{bmatrix} \cos(\emptyset_{world}(t)) & -\sin(\emptyset_{world}(t)) & 0 \\ \sin(\emptyset_{world}(t)) & \cos(\emptyset_{world}(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x_{world}(t) \\ \Delta y_{world}(t) \\ \Delta \emptyset_{world}(t) \end{bmatrix} \quad (6)$$

The origin of the global coordinate is the start position

$$\begin{bmatrix} x_{world}(t=0) \\ y_{world}(t=0) \\ \emptyset_{world}(t=0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (7)$$

3.8.2 Navigation-PC

A notebook computer runs the control software of the Navigation-PC to function as a robotic navigation system. Receiving the internal status information of the robot from the Controller-PC, the Navigation-PC performs the tasks such as SLAM, global path planning, local trajectory planning, reactive obstacle avoidance and so on. The control software of the Navigation-PC is the ROS 2D navigation stack, which provides the current state-of-the-art and open-source algorithms of full autonomous navigation functionality. But it still takes a lot of effort to implement the navigation stack on any arbitrary robot. This section describes the setup and configuration of the ROS navigation stack on the Mecanum robot.

Navigation Stack Preconditions

The preconditions for the ROS navigation stack to execute on an arbitrary robot can be categorized into hardware and software requirements. The hardware requirements include a Linux Ubuntu laptop running ROS, planar laser scanners and a mobile base that can execute velocity commands in form of v_x,v_y and v_{theta}. A Dell Latitude E6440 notebook computer is used as the Navigation-PC. Two Hokuyo UBG-04LX-F01 scanning laser rangefinders connect to the Navigation-PC, for the purpose of functioning SLAM. The Controller-PC can execute velocity commands as required and the locomotion control system assigns the corresponding rotation tasks to each servomotor that drives the Mecanum wheel. The software requirements consist of a */tf* transformation tree for the Mecanum robot and all the sensor data published in the ROS message types. In this project, the distribution of the ROS that is utilized is Hydro Medusa.

Transformation Tree

In order to describe a working mobile robot, many different coordinate frames must be involved. Starting with the largest, a world coordinate frame describes the robotic pose. The robot has a robotic base frame to describe the robotic motion with respect to itself. The sensory information that the laser scanner collects is in the laser scanner frame. It can be really confusing and messy to deal with frame transformations among multiple coordinate systems, sometimes hundreds of coordinate systems for a complex robot. Thus, a very important function that ROS provided is called */tf* transformation tree. With all the coordinate frames related to each other, a tree structure of transformations is formed. The ROS system utilises *subscribe* and *publish* to convert from one frame to another. Any point, vector or pose matrix in one frame can be converted to the ones in any other frame by *subscribe*. And the frame updates can be sent to the system by *publish*.

ROS uses its unified standard of message descriptions to communicate among the ROS nodes. This is known as message type, which is easy for ROS to deliver a message inside the system. Different sub-systems may be programmed by different languages. A unified standard of message descriptions accelerates the speed of generate source code for the message type in different target languages. The *nav_msgs* contains the common messages that are communicated within the ROS navigation stack. The *nav_msgs/Odometry* defines the position with the orientation and velocity of the robot. The *geometry_msgs/Twist* is a six-dimensional velocity. The robot *subscribes* the */cmd_vel* topic for the *Twist* messages. The Mecanum robot is omnidirectional so its velocity message is in the x, y and theta directions.

Platform Specific Nodes

Recall the navigation stack setup illustrated in Figure 39. The ROS node components that are marked in light blue are the platform specific nodes. The odometry source of the robot is the robotic velocity information, which is published via the *odom* topic. Two odometry sources were implemented. Firstly, the shaft optical encoders of the servomotors collect the rotation information of the Mecanum wheels. the kinematics model and the Controller-PC utilizes dead reckoning to return the odometry information. Because the Mecanum wheel suffers random slippages, this odometry information is not reliable. This dead reckoning odometry is utilized to monitor the motion execution progress of the robot. The second odometry source is based on the laser scanners. The laser scanners regularly detect the distance changes between the scanner and the surrounding objects. The resultant velocity of the robot is computed via

laser_scan_matcher, which can be used as a stand-alone odometry estimator. ROS provides a special message type *sensor_msgs*/LaserScan[15]. The related information or sensory data that a general given scan from the laser scanner contains can be saved in this message type.

The purpose of the Mecanum robot in this PhD project is to conduct experiments for energy-efficient motion planning research. In order to monitor the electricity usage, a current sensor is connected to the robotic electric circuit. Phidgets 1122 AC/DC current sensor was used to measure the overall current of the robotic battery.

Navigation Configuration

When all the hardware and software preconditions are satisfied, it is time to setup and configure[16] the navigation stack on the designed Mecanum robot. Completing all the above node software, setting up the necessary hardware and running ROS allows the Mecanum robot to be able to publish coordinate frame information using *tf*, receive laser scanner sensory information, publish robotic odometry information via *Odometry message* and send velocity commands out. The navigation configuration procedures include the robotic configuration launch file, the local and global costmap configuration, the base local planner configuration, the navigation launch file and the AMCL configuration.

The robotic configuration launch file brings up all the hardware and initializes all the *tf* publishes. First of all, both the laser scanners and the current sensor are brought up. Second, the odometry *tf* of the robot is launched. Third, in this robot, a TwinCAT driver is also designed to bring up the Controller-PC system. Fourth, the *tf* configuration of the robot is brought up.

Both global costmap and local costmap are set up and configured. First, a common configuration for both costmaps is set up. This mainly includes obstacle tolerance information, robotic footprint, obstacle observation source and a lot more. The obstacle tolerance information determines the minimum distance before the robot starts to put the obstacle in the map. The robotic footprint describes the size and shape of the robot. This determines the additional distance the robot centre should keep away from the obstacle. The obstacle observation source is set as the laser scanners. The global costmap contains all the information about the robot world for long-term plans. The local costmap only stores information from the

[15] http://library.isr.ist.utl.pt/docs/roswiki/navigation(2f)Tutorials(2f)RobotSetup(2f)Sensors.html#Publishing_PointClouds_over_ROS.
[16] ROS Navigation Stack Robot Setup. http://wiki.ros.org/navigation/Tutorials/RobotSetup.

robotic local perception. More details about the global costmap configuration and the local costmap configuration can be found on the ROS Navigation Stack Costmap website[17].

The base local planner configuration sets up the base_local_planner[18]. The designed robot utilizes the DWA-based local planner. Last but not the least, the navigation launch file contains all the configuration launch files so that the navigation stack can be launched by a single file. In addition, AMCL configuration details can be found in the ROS AMCL website[19].

Sensor Data

Various sensors have also been interfaced to the robot system. When conducting different navigation experiments, different sensors might be needed. Proprioceptive sensors such as optical encoder, current sensor and IMU indicate the internal state of the robot. Exteroceptive sensors such as spherical encoder [79], optical flow sensor, Hokuyo laser scanner and Microsoft Kinect acquire information from the environment. Other than laser scanner sensors used as the navigation sensors, various sensors can partially or fully do the same job. The optical encoder, IMU, optical flow sensor and spherical encoder are used for odometry research. Microsoft Kinect and Hokuyo laser scanners are used for localization research. The current sensor is used for energy-optimal navigation research. A robust interface between the system and the sensors is the most difficult technical issue.

3.8.3 Combined Control System

A combined control system was implemented by combining the Controller-PC and the Navigation-PC. By following the implementation procedures in Section 3.8.1 and Section 3.8.2, the Controller-PC and the Navigation-PC can independently perform their own functions. The Controller-PC relies on the Beckhoff TwinCAT system to control the servo motors in the standard of the industrial automation. The Mecanum wheels are driven accordingly and the Mecanum platform can execute omnidirectional motion. The Navigation-PC applies the configured ROS navigation stack for complex autonomous navigation tasks. The open-source navigation stack is highly reliable thanks to the global developers and maintainers in the ROS community. The key step to realize the combined control system architecture is the communication connectivity. The Beckhoff TwinCAT system and ROS do not support any direct interface with each other. Windows CE, which runs on the Controller-

[17] Costmap: ROS Navigation Stack Costmap. http://wiki.ros.org/costmap_2d.
[18] Base_local_planner: ROS Navigation Stack Base Local Planner. http://wiki.ros.org/base_local_planner.
[19] AMCL: ROS Navigation Stack Adaptive Monte Carlo Localization. http://wiki.ros.org/amcl.

PC, does not support any ROS node at all. In the meantime, Beckhoff does not provide any finished ADS Router for the Ubuntu Linux OS. The Navigation-PC has many available nodes in the purpose of interfacing with different external devices and it is open to many connectivity options. It is the Controller-PC running TwinCAT that restricts the methods of connectivity.

TwinCAT connectivity

The Controller-PC has a limited number of connectivity methods. The connectivity of the Beckhoff embedded PC is supported by TwinCAT Automation Device Specification ADS. The ADS protocol is the communication technology that exchanges data within the TwinCAT system. According to the TwinCAT system architecture, each module of the software within the TwinCAT system is recognised as a stand-alone device. For example, TwinCAT PLC and TwinCAT NC are recognised as two individual devices in the TwinCAT. Thus, the ADS protocol has the capability to transport messages to the TwinCAT system from the ROS system or the Ubuntu Linux PC device. Simply regarding the Navigation-PC as an additional device within the TwinCAT system, the ADS protocol can govern the data exchange via the TCP/IP connections among all the devices in the network. Over an ADS router, motion command and feedback messages are transported to the Controller-PC and stored as TwinCAT PLC variables.

TCP/IP

Beckhoff offers the TwinCAT TCP/IP Server to receive and send streams via the Transmission Control Protocol (TCP) and/or User Datagram Protocol (UDP) ports. The related function blocks from the TcpIp.Lib were used to program one TCP/IP server in the TwinCAT PLC, which can directly interact with the servo control. In the ROS, one node communicates to another via topic connections, which utilize TCP protocol or UDP protocol. So, simply TCP/IP socket connections was applied in the Navigation-PC by programming a TCP/IP client into a ROS node. This is done by using the Berkeley sockets library. The TCP/IP servers in the TwinCAT wait for incoming connections from the TCP/IP clients in the ROS node. The IP address of the Controller-PC and the port numbers of the created servers are written in the clients to register the servers. The data structure of the TCP stream message is shown in Figure 52. The TCP streams are communicated between the TCP/IP servers in the TwinCAT system and the TCP/IP clients in the ROS system, as shown in Figure 53. The contents of the TCP streams include the motion commands from the Navigation-PC to the Controller-PC and motion feedback from the Controller-PC to the Navigation-PC.

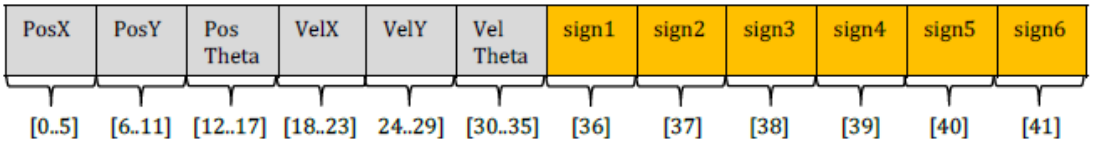


Figure 52 Data structure of TCP stream message.

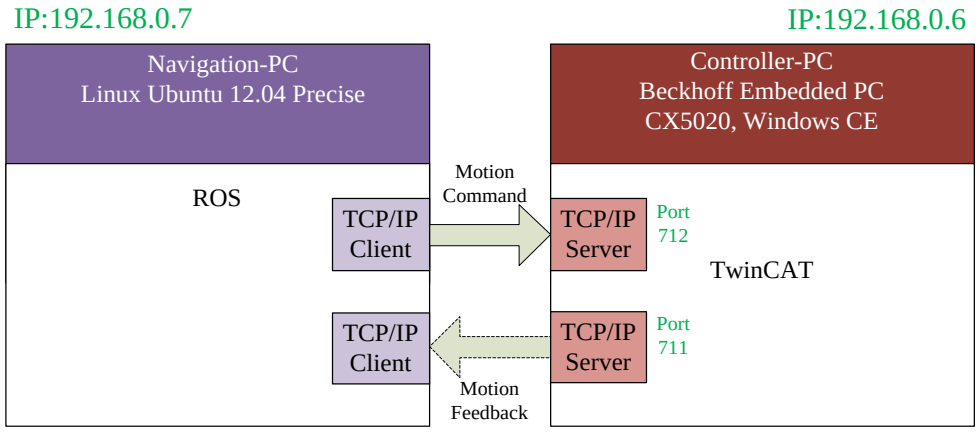


Figure 53 TCP/IP connection.

ROS TCP/IP clients in the Navigation-PC first need to connect to the TwinCAT TCP/IP server in the Controller-PC. An IP socket creation program was written in the initialization of the TCP/IP client based on the Berkeley sockets API for C++ (Programming IP Sockets on Linux). The motion feedback information can then be received by a TCP/IP client from the TwinCAT TCP/IP server. The motion feedback information is published to the ROS navigation stack in the form of *nav_msgs/Odometry*. In addition, after subscribing the *cmd_vel* topic, the other TCP/IP client formats the *twist* motion commands into the TCP steam message and forwards the stream to the TwinCAT server.

Architecture of the Combined Control System

The combined control system[20] has been implemented by combining the Controller-PC and the Navigation-PC, and the system is able to control the Mecanum mobile platform. The overall architecture of the combined control system is shown below in Figure 54. After connecting the Navigation-PC to the Controller-PC via TCP/IP, the combined control system controls the Mecanum mobile platform. According to the architecture, the Controller-PC of the system

[20] The original idea of this combined control system was proposed, and communication method was implemented by Christian Scheifele. A published jointly authored paper contains this contribution of his: Xie, L., et al. *Heavy-duty omni-directional Mecanum-wheeled robot for autonomous navigation: System development and simulation realization*. in *Mechatronics (ICM), 2015 IEEE International Conference on*. 2015. IEEE..

governs the servomotor to control the motion of the robot. The Navigation-PC receives environmental sensory information via laser scanners to plan autonomous navigation tasks.

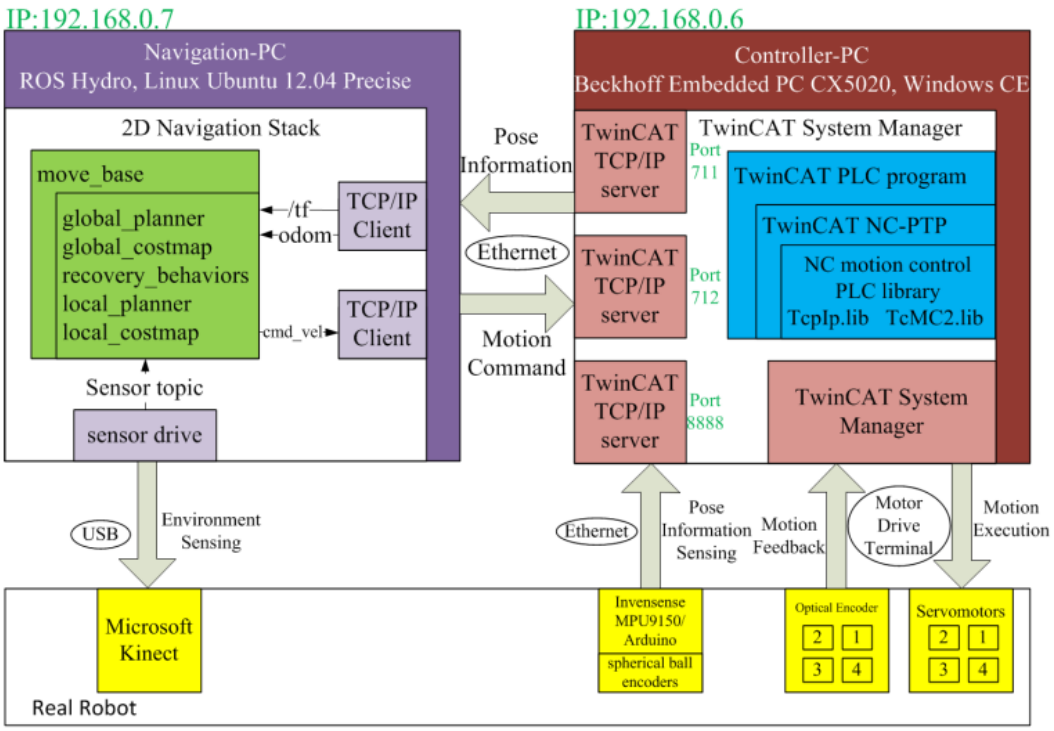


Figure 54 The architecture of the combined control system. The Navigation-PC and the Controller-PC are connected via Ethernet by TCP/TP. The Navigation-PC deals with the autonomous navigation tasks. The ROS navigation stack receives environmental information from the laser rangefinders and sends the robotic motion commands to the Controller-PC. The Controller-PC controls the motion of the robot. The TwinCAT system executes the servomotor rotations based on the motion commands.

3.8.4 Environmental Sensor – Laser Scanner

The laser scanner was the only environmental sensor installed in the final version of the designed Mecanum robot. Each laser scanner has a coverage of 240 degrees, as shown in Figure 55. Two laser scanners are enough for the robot to observe the 360 degrees surrounding environment and to perform fully functional autonomous navigation. The Hokuyo UBG-04LX-F01 scanning laser rangefinders were utilized.

The laser scanners are connected directly to the ROS laptop PC for energy supply and data exchange. The laser scanners can be powered via the USB of the ROS laptop PC. The laser scanner-related sensory data is saved in a special ROS message type *sensor_msgs*/LaserScan. The laser scanner has three essential functionalities to the autonomous Mecanum robot. The laser scanner scans the surrounding environmental profile to generate an environmental map for the robot. A laser-based SLAM is applied for the mapping functionality. The laser scanner also detects any unexpected obstacles as the obstacle sensor. The unexpected obstacle

avoidance function of the autonomous navigation system requires the obstacle sensor. Lastly but not the least, the laser scanner as a self-contained sensor observes the robotic relative position with position-known objects. The Mecanum robot requires robotic velocity/position information to compensate for its wheel slippage issue.

Throughout the project, the following sensors were considered, integrated and tested: optical encoder, IMU, optical flow sensor, spherical encoder and Microsoft Kinect. All showed their potential to provide partial or full environmental sensing data for the ROS navigation stack. However, this project does not focus on selection of the navigation sensor. The laser scanner simply performs better as the sole sensor on the robot than the other sensors in the aspect of easy integration, high accuracy, multiple functions, shorter process time and good reliability.

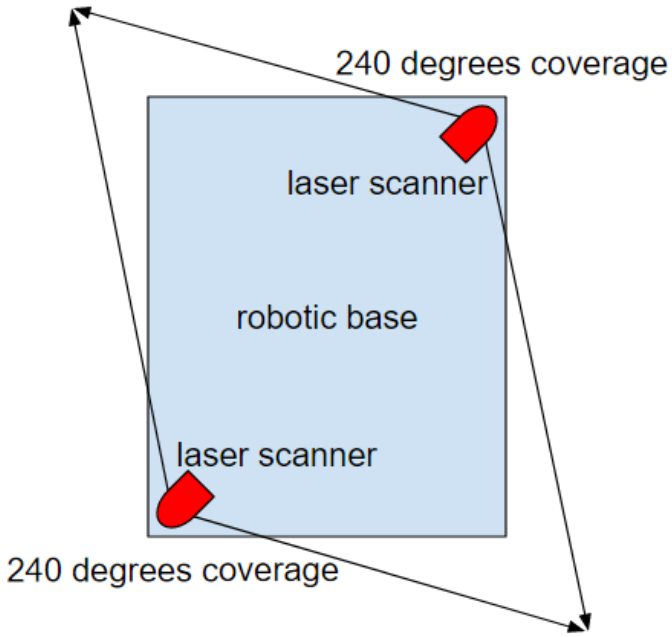


Figure 55 The coverage of the laser scanners on the robot. Two Hokuyo UBG-04LX-F01 scanning laser rangefinders are attached on two diagonal corners for localization and obstacle avoidance purposes. Each laser rangefinder has a 240-degree coverage. There are two blind spots around the robot's other two corners.

3.9. Virtual Simulation

A computer virtual simulation based on ISG-virtuos[21] for virtual AuckBot was designed and validated. A computer virtual simulation is important in engineering design. First, safety should

[21] ISG: ISG-Virtuos. http: //www.isg-stuttgart.de/en/isg-virtuos/virtuos.html.

be guaranteed in the simulation before conducting experiments, especially on the heavy-duty robot. A reduction of experiment time can be expected in simulation. In addition, an arbitrary testing environment is easily set up in the simulation. The behaviour of the real robot AuckBot is simulated to be a virtual robot in the simulation system, which offers virtual experiments for the virtual robot to be tested. The chosen simulation system is supported by ISG-virtuos because ISG-virtuos is able to fully integrate into the TwinCAT system. ISG-virtuos can be installed and optionally integrated into the TwinCAT. ISG-virtuos software consists of Virtuos M for the simulation model, Virtuos S for simulation solving and Virtuos V for simulation visualization.

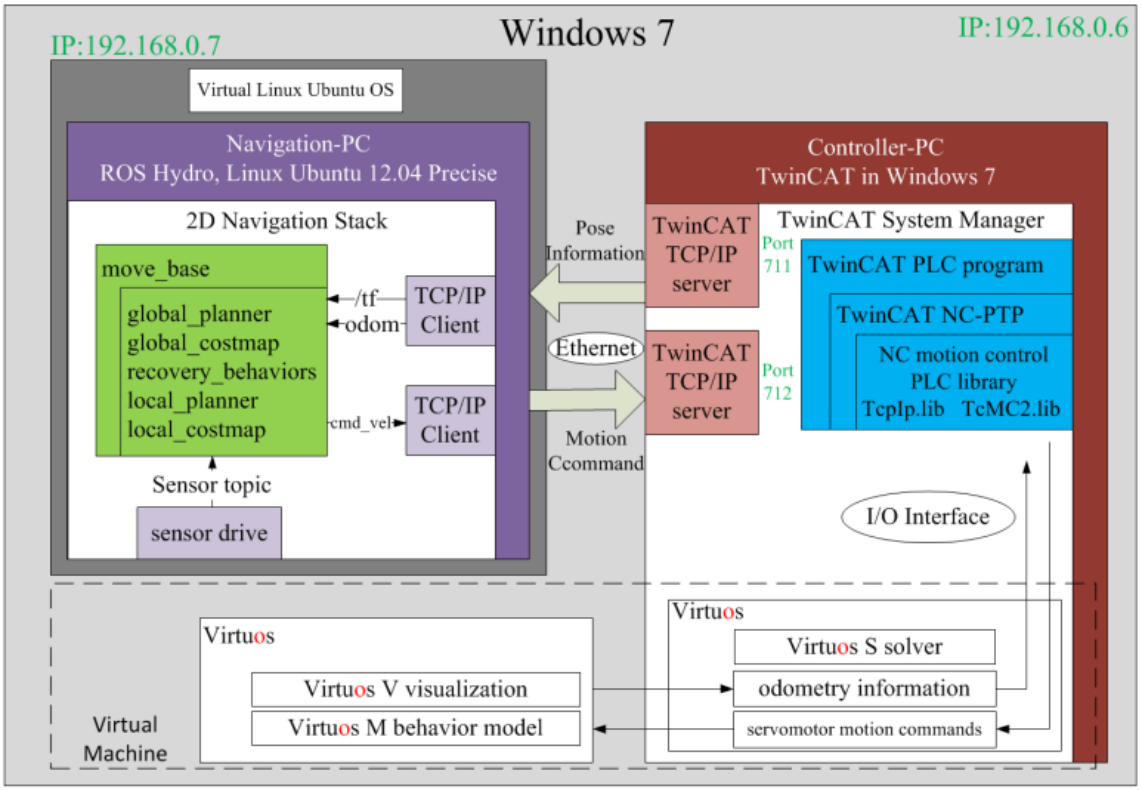


Figure 56 Simulation system architecture in a one-PC solution. A Windows PC runs a Linux in a virtual OS and the virtual Linux OS runs the ROS navigation stack. The Windows PC also runs the TwinCAT software. Both the virtual Navigation-PC and the Controller-PC are connected by the TCP/IP. Instead of connecting to the real Mecanum machine, the combined control system connects to a virtual machine. The virtual machine is simulated in the ISG Virtuos.

The AuckBot simulation system has three function aspects: behaviour model, visualization and monitoring. A behaviour model is created based on the forward kinematic equation in the Virtuos M. To set up the simulation, the virtual robot, rather than the real robot, is connected to the TwinCAT PLC via the TwinCAT system manager. A one-PC solution of simulation system architecture is shown in Figure 56. The motion command, which was sent to the servomotors, is now sent to the ISG-virtuos. Given the motion command from the Controller-

PC, ISG-virtuos can calculate the robot's dynamics and odometry information in the Virtuos S. The robot's dynamics are sent to Virtuos V to generate the visual motion of the virtual robot. Odometry information is returned to the Controller-PC. During the procedure, all the values in the simulation are available to be displayed in Virtuos M. A simplified drawing of the Mecanum robot is used in the ISG Virtuos as shown in Figure 57. The virtual simulation can be real-time while the robot is operating in the real world. The CAD model of the Mecanum mobile robot shown in Figure 27 can be planted into ISG Virtuos and used as a better visual representation.

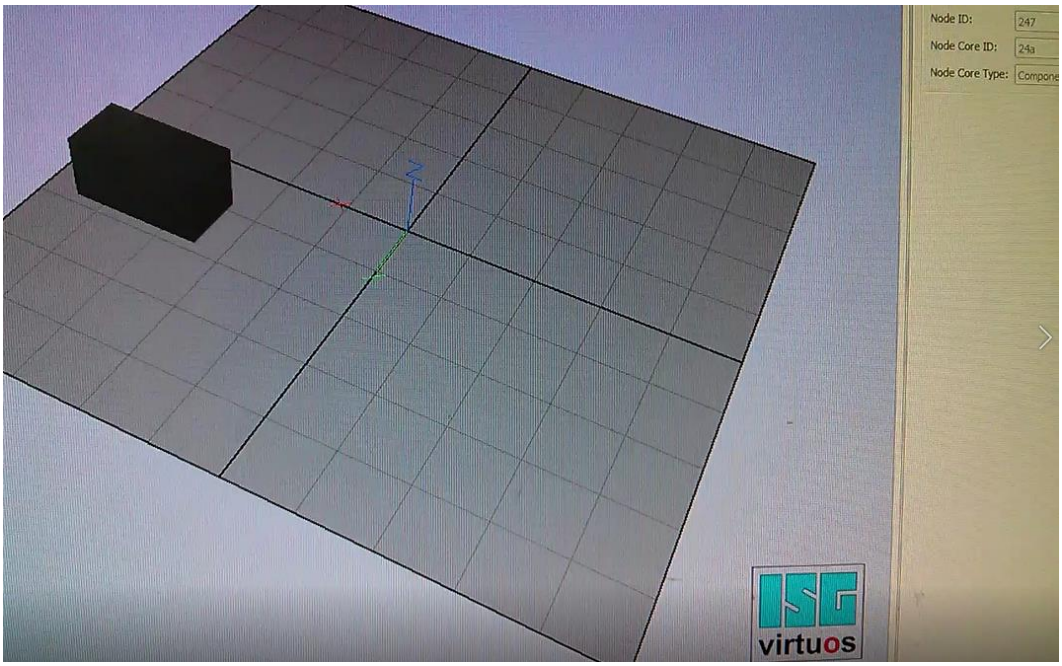


Figure 57 A soft real-time virtual simulation in ISG Virtuos.

3.10. Operation Instructions and Maintenance

The instructions for bringing up the robotic navigation system and operating the robot are summarized in Appendix A. The instructions for building the ROS costmap are written in Appendix B. The robot takes heavy-duty tasks, so it is necessary to take regular maintenance – especially for the mechanical parts. The robot platform has also been modified in a safe and organized standard. The modification procedure includes platform layout organization and grooming. The organization of electric components, wires, and battery saves more space and there are plans for future add-ons. Everything has been isolated inside the robot with transparent materials so that the electric circuit is not disturbed by the outside environment and the working state of the robot can be observed all the time.

3.11. Summary

In order to conduct research on the Mecanum wheel and develop an omnidirectional Mecanum warehouse forklift prototype, a Mecanum robot was manufactured in this PhD project at the University of Auckland. The heavy-duty omnidirectional four-wheeled Mecanum autonomous mobile robot was designed and developed from scratch, for the purposes of conducting the energy-optimal autonomous navigation research and in the future as a warehouse forklift prototype operating in industrial environments.

The entire mechatronics design of AuckBot includes the following parts. The electromechanics of the robot were designed to transmit electric power to the rotating Mecanum wheels. The robotic chassis was designed to carry all the robotic components, the transmission system transmits the servo motor power to the Mecanum wheels, and the electric system supplies power to the robot. The Controller-PC and the Navigation-PC were developed. The Controller-PC governs the locomotion control system and the locomotion control system is able to drive the Mecanum mobile robot as commanded by either the manual operation or the automation system. The Navigation-PC runs the ROS navigation stack software program and performs complex autonomous navigation tasks such as perception, localization, cognition, navigation and motion planning. The combined robotic control system was implemented by completing the communication problem between the Controller-PC and the Navigation-PC. Thus, the locomotion control system and the autonomous navigation system of the Mecanum robot were successfully fused.

Now, the robot has satisfied the requirements for conducting research experiments in this PhD project. A working Mecanum robot is important to study the energy consumption of the Mecanum wheel and conduct experiments to validate the proposed energy consumption model. With the robotic velocity controlled by a closed loop, the robot is able to overcome the wheel slippage issue and can move in desired omnidirectional velocities. Thus, the robot can conduct experiments for validating the energy-efficient motion trajectory planning algorithm. The robot is able to perform fully functional autonomous navigation so that the robot can conduct experiments for validating energy-efficient navigation algorithms. In the future, the designed Mecanum robot is to be further developed to be an omnidirectional Mecanum warehouse forklift prototype. To complete the forklift prototype development in the future, a robotic manipulator arm can be installed on the platform and interfaced to the robotic system.

Chapter 4. Energy Consumption Model of the Four-Wheeled Omnidirectional Mecanum Robot

4.1. Introduction

This chapter proposes a novel energy consumption model for the four-wheeled omnidirectional Mecanum mobile robot. According to the summarized literature review in Chapter 2, modelling the energy consumption of the target mobile robot is a common approach to solving the energy-efficient motion planning problem, and the energy consumption model for the Mecanum robot has not been built yet. Inspired by the recent existing energy consumption modelling methods of mobile robots, the proposed model considered the mainstream energy, consumed by each electric component of the robot. This model was mathematically implemented in MATLAB and experimentally validated on the purpose-built robot.

A reliable energy consumption model is essential to conduct this energy-optimal autonomous navigation research project. The omnidirectional motions of the Mecanum drive are involved with complex mechanisms, which are also closely related to the energy consumption of the robot. Putting the problem of energy-optimal motion planning for the Mecanum robot on top makes it more complicated. Thus, building an accurate energy consumption model that estimates the Mecanum robot's electric energy consumption of following any given trajectories is the key to challenging the energy-optimal motion planning problem. Depending on the different approaches to the problem, there are different ways to apply the model. For example, a derived energy consumption cost function can be derived from the model and co-opted to the A* search algorithm for finding the energy-minimum path instead of the shortest path that the original A* does [49]. A solid energy consumption analysis is based on a thorough understanding of the Mecanum wheel's dynamics and the electric consumptions of the mobile robotic devices. The considered energy consumptions include the idle energy required for stand-by and the motional energy consumed for kinetic energy transformation, overcoming robotic traction resistances, electric dissipation in the motor armatures and mechanical dissipation in the actuators. The main challenge of this study was from the complicated dynamics of the omnidirectional Mecanum wheels.

In this chapter, the dynamics model of the Mecanum drive is first reviewed, and the proposed energy consumption model is then established based on the dynamics model and the robotic energy analysis. Practical implementation and experimental validation are summarized lastly.

4.2. Dynamics Model of Four-Wheeled Omnidirectional Mecanum Robot

A thorough understanding of the Mecanum wheel's mechanism is the foundation of this research study. This study was about reducing the energy consumption of the Mecanum mobile robot by optimizing the motion planning and establishing an energy consumption model for the target robot is a common approach to the energy-efficient motion planning problem. Both motion planning and energy consumption modelling are closely related to the dynamics analysis of the target robot. Thus, kinematics and dynamics of the four-wheeled Mecanum robot are first reviewed in this section. In particular, a variety of frictions exist on the Mecanum wheel to either assist or resist the omnidirectional motions of the robot. Friction analysis of the Mecanum wheel is presented in detail. Based on the existing dynamics analysis, a dynamics model of the four-wheeled Mecanum drive is derived in this study. The Mecanum mobile platforms utilize three, four or even more Mecanum wheels. A four-wheeled Mecanum platform was considered in this study because this is the most common Mecanum drive.

4.2.1 Kinematics Analysis

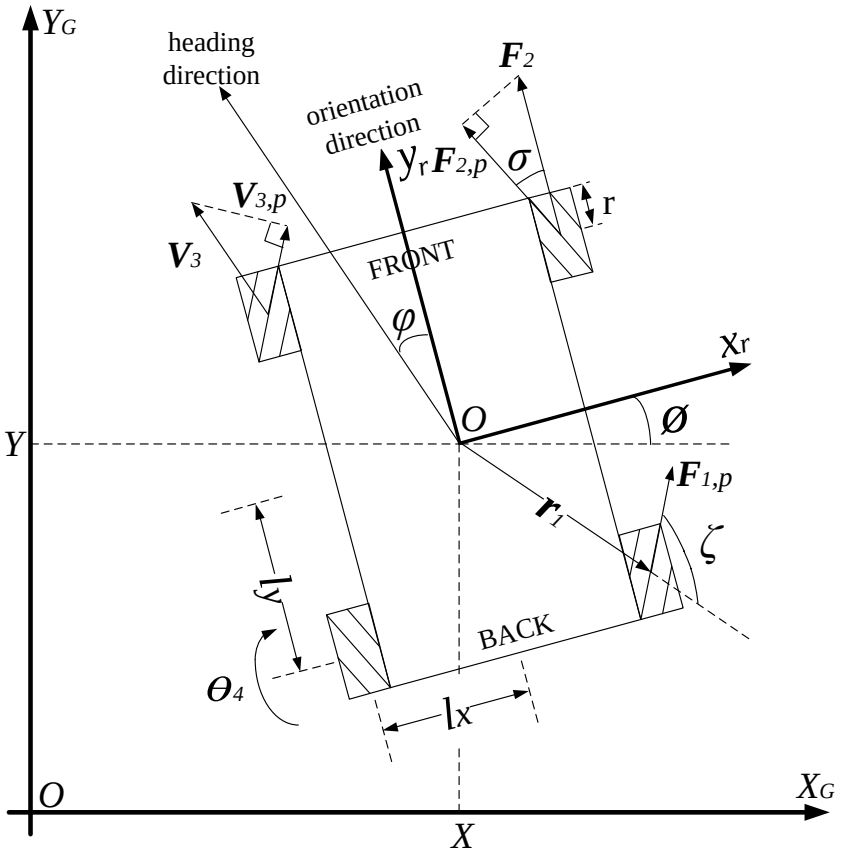


Figure 58 Schematics of the robot, viewed from above.

The frame of the designed four-wheeled Mecanum robot is a rectangular structure as the schematic diagram shows in Figure 58. The schematic diagram is viewed from above, but the roller arrangements present the rollers that are in contact with the floor. As shown in the schematic diagram, there are four labelled Mecanum wheels assembled on the robot and the threads on the wheel illustrate how the rollers of each Mecanum wheel are arranged. The roller of the first and the third Mecanum wheel contacting the ground is placed from top right to bottom left. In order to avoid confusion, the front and back sides of the robot must be indicated first. The front of the robot is defined as the side of chassis where the second and third wheels are mounted. It is a good practice to label the front and back sides on the robot.

In this study, the robotic centre of mass is assumed to be at the centre of the geometry for simplification. A robotic coordinate system $Ox_r y_r$ is rigidly attached at the mass centre of the chassis and is related to the global coordinate system $OX_G Y_G$. The centre of the mass coordinate or the robotic coordinate system $Ox_r y_r$ is rotated by an angle $\emptyset$ to get the translational motion in the global coordinate system $OX_G Y_G$. Thus, angle $\emptyset$ is the angular displacement or orientation of the robot. Unlike the differential car-like drive, the robot does not necessarily have to move in its front direction or back direction. Thus, the pose of the robot is represented by its translational displacements X and Y and orientation $\emptyset$ as $\boldsymbol{q}$ in the global coordinate frame and $\boldsymbol{q} = [X \quad Y \quad \emptyset]^T$ in $OX_G Y_G$. The pose of the robot in the robotic coordinate system is $\boldsymbol{q}_r = [x \quad y \quad \emptyset]^T$.The velocity relationship between $\dot{\boldsymbol{q}}$ and $\dot{\boldsymbol{q}}_r$ is shown in the equations below and $\boldsymbol{R}$ is the rotation matrix.

$$\dot{\boldsymbol{q}} = \boldsymbol{R}(\emptyset)\dot{\boldsymbol{q}}_r \tag{8}$$

$$\boldsymbol{R}(\emptyset) = \begin{bmatrix} \cos\emptyset & -\sin\emptyset & 0 \\ \sin\emptyset & \cos\emptyset & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

By assuming no slip in the tangential directional for all the Mecanum wheels, the kinematics of the robot, whose wheel rollers are arranged as in Figure 58, is shown by the equation below.

$$\dot{\boldsymbol{q}}_r = \frac{r}{4}\boldsymbol{K}\dot{\boldsymbol{\theta}} \tag{10}$$

$$\boldsymbol{K} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ \dfrac{1}{l_x + l_y} & \dfrac{1}{l_x + l_y} & \dfrac{-1}{l_x + l_y} & \dfrac{-1}{l_x + l_y} \end{bmatrix} \tag{11}$$

$\boldsymbol{K}$ is the forward kinematics matrix, the wheel radius is represented as r and the rotations of the wheels are defined as $\boldsymbol{\theta} = [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4]^T$. As shown in Figure 58, θ_4 represents the rotation of the robot's fourth Mecanum wheel and the positive direction is also illustrated in the schematics diagram. l_x and l_y stand for the distances of the centre of mass of the wheel from the y-axis and x-axis in Ox_ry_r, respectively.

There are many symbols showing at each wheel, which will be utilized later in this chapter. Here, these symbols are briefly introduced. In the schematics diagram, $\boldsymbol{r}_1$ is a distance vector from the origin of the robotic frame to the centre of the first wheel and $\boldsymbol{F}_{1,p}$ is the effective driving force of the first wheel. ζ is the angle between $\boldsymbol{r}_1$ and $\boldsymbol{F}_{1,p}$. At the second wheel, $\boldsymbol{F}_2$ is the driving force and σ is the roller arrangement angle of the Mecanum wheel. $\boldsymbol{V}_3$ is the velocity of the third wheel in the global coordinate system and $\boldsymbol{V}_{3,p}$ is the reprojection of $\boldsymbol{V}_3$ along the roller arrangement. $\boldsymbol{V}_3$ should be in the same direction as the robotic heading direction, as shown in the schematics diagram. Because the Mecanum robot is holonomic, its motion does not have to be aligned with its orientation. The angle between the heading direction of the robotic motion and the robotic orientation is φ, which is measured anticlockwise from the y-axis in the global coordinate system.

4.2.2 Dynamics Analysis

To understand how the omnidirectional motion of the Mecanum robot is brought into being, this section details how all the motor thrusts are converted to an omnidirectional tractive force on the mobile platform and then describes how the omnidirectional tractive force contributes to the robotic motion using Newtonian mechanics after compensating for resistive frictions. Each actively rotating wheel of the robot is driven by an independent motor and obtains a driving force from the ground surface. Due to the passive free-rolling rollers that are placed surround the circumference of the wheel at an angle $\sigma = 45°$, the driving force is angled. After combining all the angled driving forces from every Mecanum wheel, a sum tractive force acts on the Mecanum mobile platform. This tractive force can be in any desired direction or the so-called omnidirectional, depending on the magnitude and direction of each wheel's rotation. Thus, the Mecanum mobile platform is able to perform omnidirectional motion without steering its wheels. Simple omnidirectional motion primitives can be realized just by different combinations of wheel rotation directions at the same speed. These omnidirectional motion primitives include forward/backward, strafing left/right and spin clockwise/anti-clockwise.

They can be easily controlled by rotating each wheel according to Figure 2. Complex omnidirectional motions such as curved motions require the kinematics model to determine the rotational speeds of each wheel.

Without even considering the frictions, a large portion of the driving force is not directly contributing to the robotic motions. Part of the motor thrust is completely wasted by the roller in order to generate the angled driving force. One component of the driving force contributes to rotating the passive free-rolling rollers, which does not move the Mecanum platform at all. The other component of the driving force is then used to generate the robot's sum tractive force. The robot's sum omnidirectional tractive force is the result of further cancelling all the angled driving forces with each other to some extent. The extent is dependent on the types of omnidirectional motion that the robot is performing, so the omnidirectional motions generally have low energy efficiencies and different omnidirectional motions are involved with different amounts of energy loss.

Following the approach from the study [30], a two-phase force analysis is conducted to review the dynamics of the Mecanum robot. The first phase of the force analysis is conducted in an ideal case without resistive frictions. Then the second phase introduces and identifies various resistive frictions existing on each Mecanum wheel. Some modifications and simplifications have been made.

Force analysis of the Mecanum robot in a non-resistance case

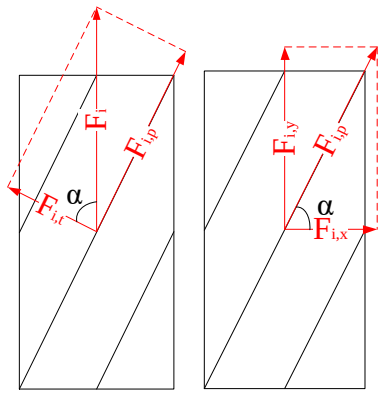


Figure 59 Components of the driving force, decomposed by the wheel roller.

When the servomotor of the robot rotates its shaft, the Mecanum wheel is driven by the motor torque via the designed transmission system. Due to the friction between the roller and the ground, a driving force F_i is acting on the wheel i of the robot. Here, the first or the third wheel

is set as the example. In this non-resistance case, the friction that causes driving force F_i is assumed to be more than enough to enable motion and all other the resistive frictions are assumed to be small enough to be neglected.

Because the rollers of the Mecanum wheel are attached and surround the wheel circumference at an angle of 45°, the driving force F_i is separated into two components $F_{i,p}$ and $F_{i,t}$. The component $F_{i,p}$ is in parallel to the rotation direction of the roller and the other component $F_{i,t}$ is in the transverse direction. The rollers can freely roll, so the transverse force component $F_{i,t}$ does not contribute to the traction of the robot at all. This part of the driving force is ineffective to the motion of the robot. The parallel force component $F_{i,p}$ is named as the effective force that enables the motion of the robot. As shown in the equation below, $\sin\alpha$ represents the efficiency of the Mecanum wheel. As $\alpha = 45°$, any rotating Mecanum wheel has an efficiency of 70.7%.

$$F_{i,p} = F_i \cdot \sin\alpha \tag{12}$$

In order to study the sum driving force of the robot, all the parallel/effective forces of each Mecanum wheel are combined together. In the robotic coordinate system Ox_ry_r, $F_{i,p}$ is decomposed into $F_{i,x}$ and $F_{i,y}$ along the x-axis and y-axis, as shown in Figure 60.

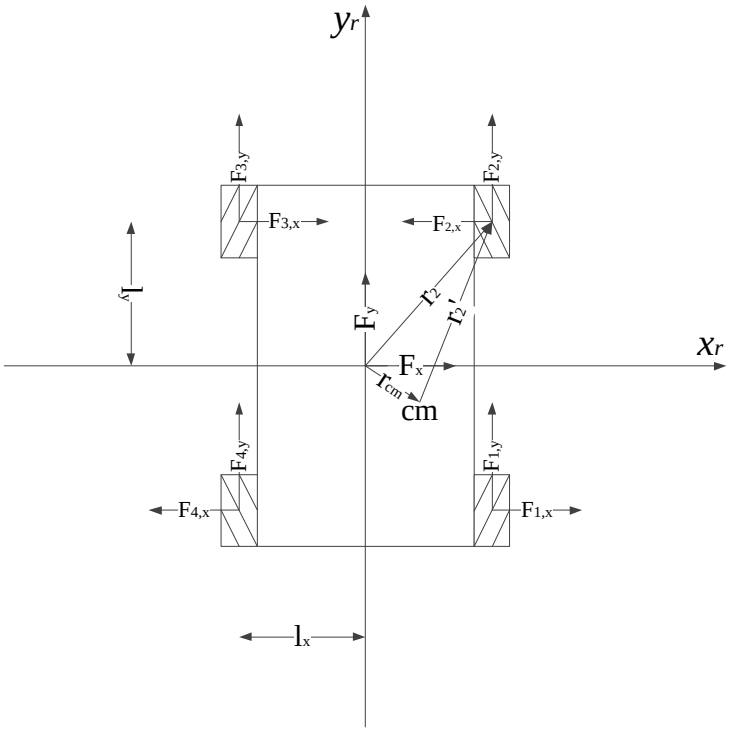


Figure 60 Effective force analysis in the robotic coordinate.

The origin of the robotic coordinate system Ox_ry_r is at the centre of the robotic geometry. $\boldsymbol{r}_{cm}$ is the position vector of the robotic centre of mass. The robotic centre of mass is assumed to be at its centre of the geometry so that the robotic coordinate system is overlapped with the centre of mass coordinate. Thus, $\boldsymbol{r}_{cm}$ is set as zero here. The position vector $\boldsymbol{r}_i$ of wheel i in the robot coordinate system is equal to the position vector $\boldsymbol{r}_i'$ of wheel i in the centre of mass coordinate system.

$$\boldsymbol{r}_{cm} = 0 \tag{13}$$

$$\boldsymbol{r}_i = \boldsymbol{r}_i' \tag{14}$$

The directions of $F_{i,x}$ and $F_{i,y}$ of each Mecanum wheel are determined by the roller axle arrangement of the wheel and the rotation direction of the wheel. Taking the wheel roller axle arrangements in Figure 58 and Figure 60 as an example, the roller axles of the first and third Mecanum wheel of the robot that are in contact with the ground are placed on the wheel from top right to bottom left.

$$F_x = \sum_{i=1}^{4} F_{i,x} = \sin\alpha \cdot \cos\alpha \cdot \sum_{i=1}^{4} (-1)^{i+1} \cdot sgn(\theta_i) \cdot F_i \tag{15}$$

$$F_y = \sum_{i=1}^{4} F_{i,y} = \sin^2\alpha \cdot \sum_{i=1}^{4} sgn(\theta_i) \cdot F_i \tag{16}$$

$$F_R = F_x \cdot \hat{u}_x + F_y \cdot \hat{u}_y \tag{17}$$

The direction of the driving force F_i is determined by the rotational direction $sgn(\theta_i)$ of the wheel i. The expression $(-1)^{i+1}$ takes the roller orientation arrangement of the wheel i into consideration. After combining all the effective components of the driving forces of the Mecanum wheels along both the x-axis and y-axis, the total force in the robotic coordinate system F_R consisting of F_x and F_y is acting on the robot. Intuitively, $\hat{u}_x$ and $\hat{u}_y$ are the unit vectors towards the positive directions of the x-axis and y-axis, respectively.

In order to determine the robot's translation motion in the global coordinate system, the force components in the robotic coordinate system Ox_ry_r need to be transferred into the ones in the global coordinate system OX_GY_G by the orientation angle $\emptyset$.

$$\begin{bmatrix} F_X \\ F_Y \\ T_Z \end{bmatrix} = \begin{bmatrix} \cos\emptyset & -\sin\emptyset & 0 \\ \sin\emptyset & \cos\emptyset & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ T_z \end{bmatrix} \quad (18)$$

The total force F_G in the global coordinate system is given by the X_G-axis component force F_X and the Y_G-axis component force F_Y. $\hat{u}_X$, and $\hat{u}_Y$ are the unit vectors towards the positive directions of X-axis and Y-axis, respectively.

$$F_G = F_X \cdot \hat{u}_X + F_Y \cdot \hat{u}_Y \quad (19)$$

The translational motion of the robot in the global coordinate system is calculated by the rotational torque T_Z on the robot. T_Z is caused by the wheel position vector $\boldsymbol{r}_i$ in the robotic coordinate system and the driving force of the wheel.

$$T_Z = \sum_{i=1}^{4} \boldsymbol{r}_i \times \boldsymbol{F}_i \quad (20)$$

In this case, the resistive frictions of the Mecanum wheels have not been considered. But the motion of the robot can be estimated by using F_X, F_Y and T_Z in Newtonian mechanics. The estimated motion of the robot helps understand the movements of the Mecanum wheels, based on which the resistive frictions can be determined.

Resistive friction analysis of the Mecanum robot

According to a recent publication [30], the resistive friction on each Mecanum wheel is independently analysed. Two categories of resistive friction exist and appear on the Mecanum wheels: static friction and dynamic friction. Static friction appears between the surfaces of two objects, which are at rest relative to each other. This friction must be overcome in order to start movement between the two objects. If the objects are moving against each other, dynamic friction appears. This must be overcome to sustain the movement. Usually, the static friction coefficient is higher than the dynamic friction coefficient. It is assumed that both static and dynamic friction coefficients are the same [30], meaning that no such distinction will be made.

Resistive friction between the Mecanum wheel and the ground is difficult to analyse, due to the omnidirectional motion of the Mecanum mobile platform, and the angled roller design of the Mecanum wheel. Unlike other wheels, the Mecanum wheel is more complex to include the resistive effects in the general motion of a Mecanum-wheeled vehicle. Standard wheels create either forward or backward overall motion so that the resistive effects can be concluded in

terms of the whole vehicle. However, the Mecanum wheels rotating at different angular velocities create an omnidirectional motion. Each Mecanum wheel's resistive effects must be analysed independently. The general state of motion of each wheel is described as "rolling with sliding".

When the Mecanum robot performs omnidirectional motions, the Mecanum wheel is likely to move towards any direction. Depending on the movements and rotation of the Mecanum wheel, either type of two resistive frictions is acting on the Mecanum wheel. The sliding friction f_s occurs if the moving surfaces rub against each other and the rolling friction f_r is the friction that resists the rolling of a wheel on a surface.

$$f_{i,s} = \mu_s N_i \cdot sgn(V_{i,p})\hat{u}_{i,p} \tag{21}$$

$$f_{i,r} = \mu_r N_i \cdot sgn(V_{i,p})\hat{u}_{i,p} \tag{22}$$

μ_s and μ_r are the sliding friction coefficient and the rolling friction coefficient, respectively. N_i is the normal force on the wheel i. Because the robotic centre of the mass is at the centre of the geometry, N_i is exactly one fourth of the total weight of the robot. V_i is the velocity of the wheel i in the global coordinate system. $V_{i,p}$ is the projection of V_i along the $\hat{u}_{i,p}$ direction. The direction of either friction is determined by $V_{i,p}$ relative to the ground in the projection along the $\hat{u}_{i,p}$ roller parallel direction as shown in Figure 60.

For the first and the third wheels:

$$\hat{u}_{i,p} = \cos(\alpha + \emptyset)\,\hat{u}_X + \sin(\alpha + \emptyset)\,\hat{u}_Y \tag{23}$$

For the second and the fourth wheels:

$$\hat{u}_{i,p} = \cos(180 - \alpha + \emptyset)\,\hat{u}_X + \sin(180 - \alpha + \emptyset)\,\hat{u}_Y \tag{24}$$

Furthermore, the viscous friction $f_{i,v}$, influenced by the viscous friction coefficient μ_v appears if there is relative motion between the robot and the ground.

$$f_{i,v} = \mu_v N_i \cdot V_{i,p} \tag{25}$$

Modelling robotic motion

The following presents the standard procedures of modelling the motion of the robot. Firstly, the general wheel velocity V_i of each wheel is estimated.

$$V_i = V_{cm} + \begin{bmatrix} \cos\emptyset & -\sin\emptyset \\ \sin\emptyset & \cos\emptyset \end{bmatrix} [\boldsymbol{\theta}_i \times \boldsymbol{r}_i] =$$

$$V_{cm} - \dot{\emptyset}[(y_i \cos\emptyset + x_i \sin\emptyset)\hat{u}_X + (y_i \sin\emptyset - x_i \cos\emptyset)\hat{u}_Y] \quad (26)$$

V_{cm} is the velocity of the robotic centre of the mass in the global coordinate system. In this study, $V_{cm} = \dot{\boldsymbol{q}}$. V_{cm} can be estimated by using F_X, F_Y and T_Z in the Newtonian mechanics. x_i and y_i are the positions of the wheel i along x-axis and y-axis in Ox_ry_r. From here, estimate the $V_{i,p}$ by using the following two equations for wheel 1, 3 and for wheel 2, 4, respectively.

$$V_{i,p} = V_{i,X} \cos(\alpha + \emptyset) + V_{i,Y} \sin(\alpha + \emptyset) \quad (27)$$

$$V_{i,p} = V_{i,X} \cos(180 - \alpha + \emptyset) + V_{i,Y} \sin(180 - \alpha + \emptyset) \quad (28)$$

At last, $F_{i,p}$ is estimated. Follow the flowchart in Figure 61 to determine the resistive frictions and update $F_{i,p}$. Quantity ε is a very small positive number as a tolerance limit.

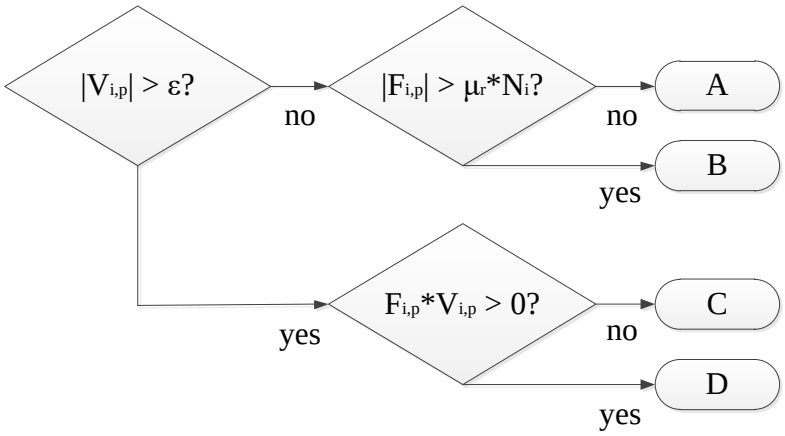


Figure 61 The flowchart of determing wheel resistive frictions.

According to the flowchart above, there are four situations to update $F_{i,p}$. The following four equations present how to update $F_{i,p}$ for all four situations.

$$\overrightarrow{F_{i,p}} = \overrightarrow{F_{i,p}} - sgn(F_{i,p})\mu_r N_i \widehat{u_{i,p}} \quad \text{for Situation A} \quad (29)$$

$$\overrightarrow{F_{i,p}} = \vec{o} \quad \text{for Situation B} \quad (30)$$

$$\overrightarrow{F_{i,p}} = \overrightarrow{F_{i,p}} - sgn(V_{i,p})\mu_s N_i \widehat{u_{i,p}} - \mu_v N_i \overrightarrow{V_{i,p}} \quad \text{for Situation C} \quad (31)$$

$$\overrightarrow{F_{i,p}} = \overrightarrow{F_{i,p}} - sgn(V_{i,p})\mu_r N_i \widehat{u_{i,p}} - \mu_v N_i \overrightarrow{V_{i,p}} \quad \text{for Situation D} \quad (32)$$

Finally, the updated $F_{i,p}$ is used to recalculate the corresponding F_X, F_Y, T_Z, F_G and $\dot{\boldsymbol{q}}$.

4.2.3 Dynamics Model

To summarize the dynamics of the Mecanum robot, a dynamics model of the Mecanum robot is derived in this study based on the analysis of [30]. According to Newtonian mechanics, the following equation is presented.

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \begin{bmatrix} F_X \\ F_Y \\ T_Z \end{bmatrix} \quad (33)$$

$$\boldsymbol{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix} \quad (34)$$

m is the mass of the robot and I is the moment of inertia of the robot in z the direction.

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{R}(\emptyset)\begin{bmatrix} F_x \\ F_y \\ T_z \end{bmatrix} \quad (35)$$

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{R}(\emptyset)\boldsymbol{C}(\alpha)\boldsymbol{S} \quad (36)$$

$\boldsymbol{C}(\alpha)$ is a matrix to convert each Mecanum wheel's traction into the FOMR's traction.

$$\boldsymbol{C}(\alpha) = \begin{bmatrix} \cos\alpha \cdot sgn(\dot{\theta}_1) & -\cos\alpha \cdot sgn(\dot{\theta}_2) & \cos\alpha \cdot sgn(\dot{\theta}_3) & -\cos\alpha \cdot sgn(\dot{\theta}_4) \\ \sin\alpha \cdot sgn(\dot{\theta}_1) & \sin\alpha \cdot sgn(\dot{\theta}_2) & \sin\alpha \cdot sgn(\dot{\theta}_3) & \sin\alpha \cdot sgn(\dot{\theta}_4) \\ \frac{1}{\sin\alpha}\boldsymbol{r}_1 \times \hat{u}_{F_1} & \frac{1}{\sin\alpha}\boldsymbol{r}_2 \times \hat{u}_{F_2} & \frac{1}{\sin\alpha}\boldsymbol{r}_3 \times \hat{u}_{F_3} & \frac{1}{\sin\alpha}\boldsymbol{r}_4 \times \hat{u}_{F_4} \end{bmatrix} \quad (37)$$

$$\boldsymbol{S} = \begin{bmatrix} |\boldsymbol{F}_{1,p} + \boldsymbol{f}_{1,s/r} + \boldsymbol{f}_{1,v}| \\ |\boldsymbol{F}_{2,p} + \boldsymbol{f}_{2,s/r} + \boldsymbol{f}_{2,v}| \\ |\boldsymbol{F}_{3,p} + \boldsymbol{f}_{3,s/r} + \boldsymbol{f}_{3,v}| \\ |\boldsymbol{F}_{4,p} + \boldsymbol{f}_{4,s/r} + \boldsymbol{f}_{4,v}| \end{bmatrix} \quad (38)$$

Intuitively, $\hat{u}_{F_i}$ is the unit vector along the driving force F_i. $\boldsymbol{f}_{i,s/r}$ means that either $\boldsymbol{f}_{i,s}$ or $\boldsymbol{f}_{i,r}$ takes effect on the wheel at one time.

4.3. Energy Consumption Model

A novel energy consumption model for the Mecanum mobile robot was proposed in this study for the purpose of getting to know the energy consumption of the Mecanum robot and improving the energy efficiency of the Mecanum wheel. Energy consumption of a mobile robot is complex because it is influenced by many factors. According to the literature review, it is a common approach to study energy-efficient motion planning via first building an energy consumption model for the target robot [49-51]. Based on the experimentally validated influencing factors of the existing recent literature review, the energy consumption of the four-wheeled omnidirectional Mecanum robot was modelled in this chapter by considering the idle energy required for stand-by and the motional energy consumed for kinetic energy transformation, overcoming robotic traction resistances, electric dissipation in the motor armatures and mechanical dissipation in the actuators. Then an energy cost function can be derived to estimate the energy consumption of the robot conducting the given trajectories, based on the energy consumption model, the kinematic model and the dynamics model.

4.3.1 Methodology and Robotic Energy Consumption Analysis

The method of establishing this proposed energy consumption model is mostly inspired by [49]. The energy consumption of a mobile robot consists of several main components. Their proposed methodology is to simply list the major components. The difference between this study and their study is that their proposed energy consumption model is built specifically for differential wheeled robots. The novelty of this study is an energy consumption model of the holonomic omnidirectional Mecanum robot and the challenge is to consider the complex dynamics of the Mecanum wheel. Compared to the differential drive, the omnidirectional Mecanum drive has much more complex dynamics.

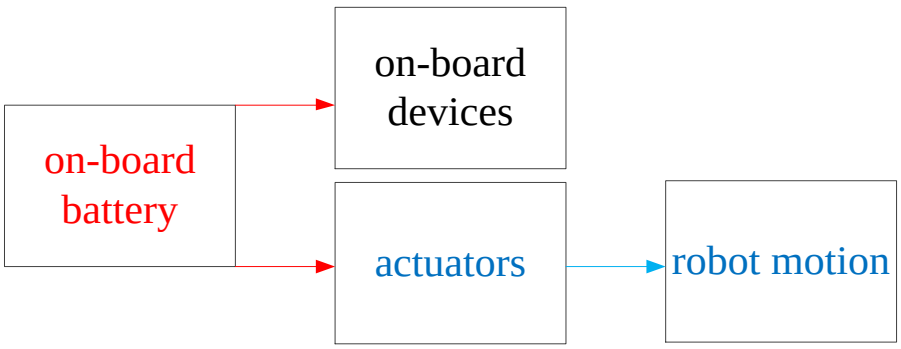


Figure 62 Battery energy supply of a mobile robot.

Normally the energy of a mobile robot is supplied by its on-board battery. The energy consumption of all the on-board electrical devices is sourced from the battery. The battery is also usually the only energy source of the mobile robot. Because the main purpose of building this model is to help study energy-optimal motion trajectory planning, the actuators' energy consumption is differentiated from any other devices' energy consumption as shown in Figure 62. The energy consumption of other robotic on-board devices whose power consumption is independent of the robotic motion, is proportionally related to the duration of the motion trajectory. Here, the energy consumption is the time integral of the power consumption. The electric motors convert the electrical energy into mechanical energy and rotate the Mecanum wheels to generate robotic motion. The energy consumption of the actuators is non-linearly related to the robotic motion trajectory. High-speed motion may result in very high power consumption but in return the entire motion only lasts for a short duration. Low-speed motion consumes low power consumption but has a long task duration. Both power consumption and task duration must be optimized for minimum energy consumption. The red electric supply system in Figure 32 is considered as the robotic energy consumption in this study.

4.3.2 Energy Consumption Modelling

Based on the analysis above, there are two categories of energy consumed by the robot. The first category of energy is consumed at a constant rate, in other words, these energy consumptions have a constant power consumption over time. The idle energy E_{idle} is used to describe all the energy consumptions belonging to the first category. The second category of energy is consumed to initialize and sustain the robotic motions, containing the motional energy consumptions E_{motion} whose powers are dependent on the occurring motions of the mobile robot. This category includes kinetic energy, frictional dissipation, armature dissipation and mechanical dissipation. Thus, the total energy consumption of the robot E_{total} consists of the idle energy E_{idle} and motional energy E_{motion}.

$$E_{total} = E_{idle} + E_{motion} \tag{39}$$

$$E_{motion} = E_{kinetic} + E_f + E_e + E_m \tag{40}$$

Idle Consumption

The idle energy E_{idle} is consumed by the on-board electrical devices. In the design Mecanum robot, E_{idle} includes an embedded PC, sensors, electric drives and other electrical devices. The

power consumption of these devices is experimentally found to be constant over time. P_{idle} is used as the power consumption of all the on-board electrical devices. The only relation of the idle energy consumption to robotic motion trajectory is the motion duration t.

$$E_{idle} = P_{idle}t = \int_{t_i}^{t_f} P_{idle}\, dt \tag{41}$$

Kinetic Consumption

In order to initialize the robotic motions, part of the energy consumed in the actuators must be transferred into the robot kinetic energy. It is important to know that the Beckhoff servomotors used on the robot are not able to recycle the kinetic energy. Otherwise, an additional recycling term should be added in the equations below.

$$E_{kinetic} = \frac{1}{2}m\left(\dot{X}^2 + \dot{Y}^2\right) + \frac{1}{2}J_Z\dot{\emptyset}^2 + \frac{1}{2}J_w\left(\dot{\theta}_1^2 + \dot{\theta}_2^2 + \dot{\theta}_3^2 + \dot{\theta}_4^2\right) \tag{42}$$

$$E_{kinetic} = \int_{t_i}^{t_f} m\left(\dot{X}\ddot{X} + \dot{Y}\ddot{Y}\right) + J_Z\dot{\emptyset}\ddot{\emptyset} + J_w\left(\dot{\theta}_1\ddot{\theta}_1 + \dot{\theta}_2\ddot{\theta}_2 + \dot{\theta}_3\ddot{\theta}_3 + \dot{\theta}_4\ddot{\theta}_4\right) dt \tag{43}$$

$$E_{kinetic} = \int_{t_i}^{t_f} (\dot{\boldsymbol{q}}^T \boldsymbol{M}\ddot{\boldsymbol{q}} + J_w\dot{\boldsymbol{\theta}}^T\ddot{\boldsymbol{\theta}})dt \tag{44}$$

J_Z is the moment of inertia of the robot along the laboratory Z axis and J_w is the moment of inertia of the wheel along its rotation axis.

Frictional Dissipation

In order to sustain the robot's motion, part of the energy must overcome the traction resistance. According to the friction analysis in the dynamics model, the Mecanum wheel is suffering sliding friction f_s, rolling friction f_r or both at the same time depending on its omnidirectional motion. In addition, viscous friction f_v exists between the robot and the ground.

$$E_f = \int_{t_i}^{t_f} f_{resistance} \cdot v\, dt = \int_{t_i}^{t_f} \sum_{i=1}^{4} \left(\left|\boldsymbol{f}_{i,s/r} + \boldsymbol{f}_{i,v}\right| \cdot \left|\boldsymbol{V}_{i,p}\right|\right)\, dt \tag{45}$$

Armature Electric Dissipation

While the battery is providing energy to the actuators, part of the energy is electrically dissipated inside the motors. In this study, only armature resistance is considered. Other electric circuit losses are considered as minor loss and are neglected. The dynamics of the motor are also neglected.

$$E_e = R_a \int_{t_i}^{t_f} \boldsymbol{I}_m{}^T \boldsymbol{I}_m \, dt \tag{46}$$

The Mecanum mobile robot contains four electric motors, each independently controlling one Mecanum wheel. $\boldsymbol{I}_m = [I_{m1} \quad I_{m2} \quad I_{m3} \quad I_{m4}]^T$ is the current flow in the electric motors. R_a is the armature resistance of the electric motor.

Mechanical dissipation

While the motors are rotating the Mecanum wheel, part of the energy is mechanically dissipated in the transmission system of the robotic electromechanics. Assuming that the bearing has a very low friction coefficient, only the static friction torque of the motor shaft is considered as the major contribution of the mechanical dissipation.

$$E_m = M_R \int_{t_i}^{t_f} \dot{\boldsymbol{\theta}} \, dt \tag{47}$$

$\boldsymbol{\theta} = [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4]^T$ is the angular velocity of each Mecanum wheel and M_R is the static friction torque existing in the motor.

Energy Consumption Model

The total energy consumption of the robot E_{total} is summarized by the aforementioned major components of the robotic energy consumptions.

$$E_{total} = E_{idle} + E_{kinetic} + E_f + E_e + E_m \tag{48}$$

4.4. Implementation

It is aimed to apply the established energy consumption model for the energy-efficient motion planning study. The energy-optimal motion planning study requires a cost function to evaluate

the energy consumption of the robot following given trajectories. The cost function is essential to the research of energy-optimal autonomous navigation. The proposed energy consumption model was implemented in this study according to such requirements. The proposed energy consumption model in the equation above represents a typical energy consumption of the mobile robot. According to the energy consumption model, there are five mainstream components of energy that consume the electric energy from the on-board battery, which is usually the only energy source of the mobile robot. In order to implement the energy model for energy-optimal motion planning, knowledge of the Mecanum kinematics and dynamics is required to link the robotic motion trajectory to the energy consumption model. For example, the kinematics of the Mecanum robot can convert robotic motion into wheel rotations $\dot{\boldsymbol{\theta}}$ for calculating $E_{kinetic}$ and E_m, and the dynamics of the Mecanum wheel can find the resistive frictions for calculating E_f. This section introduces how to mathematically implement the energy consumption model to evaluate the robotic energy consumption of any given velocity profiles via the dynamics and kinematics models.

According to the model, E_{total} consists of E_{idle}, $E_{kinetic}$, E_f, E_e and E_m. A trajectory function $\dot{\boldsymbol{q}}(t)$ is defined as a given arbitrary velocity profile of the robot and its corresponding motion trajectory duration is defined as t. The motion trajectory duration t can be used to directly calculate E_{idle} directly because P_{idle} is considered to be constant. $\dot{\boldsymbol{q}}$ needs to be first differentiated to obtain $\ddot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$ is then used to calculate $E_{kinetic}$. Based on $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$, $\dot{\boldsymbol{\theta}}$ and $\ddot{\boldsymbol{\theta}}$ can be found according to the kinematics model of the robot. Thus, both $E_{kinetic}$ and E_m can be computed. The resistive frictions acting on each Mecanum wheel can be found based on the dynamics analysis of the robot. E_f can be easily calculated by using the found resistive frictions.

Motor Current Calculation

However, it is less straightforward to calculate E_e. It is necessary to calculate motor current flows $\boldsymbol{I}_m$ first. The servomotors of the robot are four brushless DC three-phase servomotors. The servomotor consists of a control circuit and a three-phase circuit, as shown below in Figure 63.

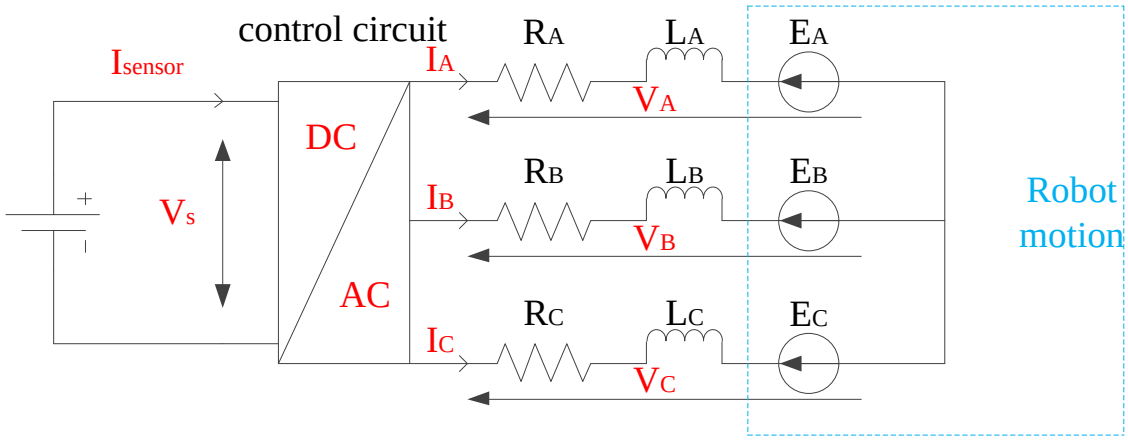


Figure 63 Electrical circuits of brushless DC three-phase servomotors.

A three-phase electric circuit is too complex to implement. However, when the brushless DC motor is in steady-state, it is simply operating as an ordinary brush DC motor; hence the brush DC motor model can be utilized to simulate the performance of the brushless DC motor in such a state. So, an equivalent circuit model can be built in Figure 64.

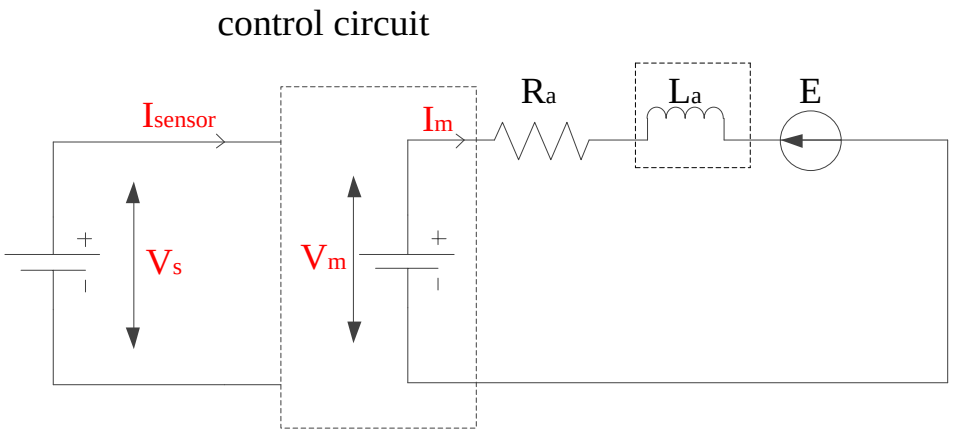


Figure 64 An equivalent brush DC motor circuit model. It is used to replace the three-phase circuit in steady-state.

The model of the DC motor containing inductance, series resistance and back EMF proposed in [81] is used. According to brushed DC motor study in [81], the motor shaft torque vector $\boldsymbol{T}_m$ is related to $\boldsymbol{I}_m$. L_a can be ignored because the electrical response is generally much faster than the mechanical response.

$$\boldsymbol{V}_m = K_e\dot{\theta_m} + R_a I_m + L_a\dot{\theta_m} I_m^{(1.5\sim2)} \tag{49}$$

$$\boldsymbol{V}_m \approx K_e\dot{\boldsymbol{\theta}_m} + R_a\boldsymbol{I}_m \tag{50}$$

$$\boldsymbol{T}_m = k_t\boldsymbol{I}_m \tag{51}$$

The Mecanum robot in this PhD project uses brushless AC servo motors, but the torque is proportional to current in both cases. k_t is the torque constant. The motor shaft torque $\boldsymbol{T}_m$ is directly related to the Mecanum wheel's driving force F_i and thus, to its effective component $F_{i,p}$. Here, n is the gear ratio of the motor gear box and r is the radius of the Mecanum wheel.

$$F_{i,p} = T_m \frac{n}{r} sin\,\alpha \tag{52}$$

$$\boldsymbol{I}_m k_t \frac{n}{r} \sin\alpha = \left[\left|\boldsymbol{F}_{1,p}\right| \quad \left|\boldsymbol{F}_{2,p}\right| \quad \left|\boldsymbol{F}_{3,p}\right| \quad \left|\boldsymbol{F}_{4,p}\right|\right]^T \tag{53}$$

Based on the dynamics analysis of the robot, a matrix equation to calculate $\boldsymbol{I}_m$ is derived.

$$\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{C}^{\boldsymbol{T}})^{-1}\boldsymbol{R}(\emptyset)^{\boldsymbol{T}}\boldsymbol{M}\ddot{\boldsymbol{q}} = \begin{bmatrix} \left|K\boldsymbol{I}_{m1} + \boldsymbol{f}_{1,s/r} + \boldsymbol{f}_{1,v}\right| \\ \left|K\boldsymbol{I}_{m2} + \boldsymbol{f}_{2,s/r} + \boldsymbol{f}_{2,v}\right| \\ \left|K\boldsymbol{I}_{m3} + \boldsymbol{f}_{3,s/r} + \boldsymbol{f}_{3,v}\right| \\ \left|K\boldsymbol{I}_{m4} + \boldsymbol{f}_{4,s/r} + \boldsymbol{f}_{4,v}\right| \end{bmatrix} \tag{54}$$

$$K = sin\,\alpha\, k_t \frac{n}{r} \tag{55}$$

Now E_e can be calculated by the found $\boldsymbol{I}_m$.

Implementation Flowchart

According to the implementation flowchart shown in Figure 65, the sum energy consumption can be calculated based on the robotic energy consumption model, kinematics and dynamics. Initially a given trajectory time function $\dot{\boldsymbol{q}}(t)$ in the global coordinate frame is converted into the robotic trajectory time function $\dot{\boldsymbol{q}}_{\boldsymbol{r}}(t)$ in the robotic coordinate frame, by using the rotation matrix $\boldsymbol{R}$. Once $\dot{\boldsymbol{q}}_{\boldsymbol{r}}(t)$ is found, the general wheel velocity of each wheel $\boldsymbol{V}_i(t)$ and its projection $\boldsymbol{V}_{i,p}(t)$ can be calculated. The dynamics model of the Mecanum robot is utilized to find $\boldsymbol{F}_{i,p}(\mathrm{t})$ and the kinematics model of the Mecanum robot is utilized to find $\dot{\boldsymbol{\theta}}(t)$. Then the resistive frictions $\boldsymbol{f}_{i,r}(\mathrm{t})$, $\boldsymbol{f}_{i,s}(\mathrm{t})$ and $\boldsymbol{f}_{i,v}(\mathrm{t})$ are computed for each wheel, according to Figure 61. Last but not the least, $\boldsymbol{I}_m(t)$ is calculated based on the simplified DC motor circuit model. Now all the essential parameters to calculate each component of the robotic energy consumption are ready. Based on the proposed energy consumption model, the robotic energy consumption can be easily computed, and the energy consumption model can be implemented accordingly.

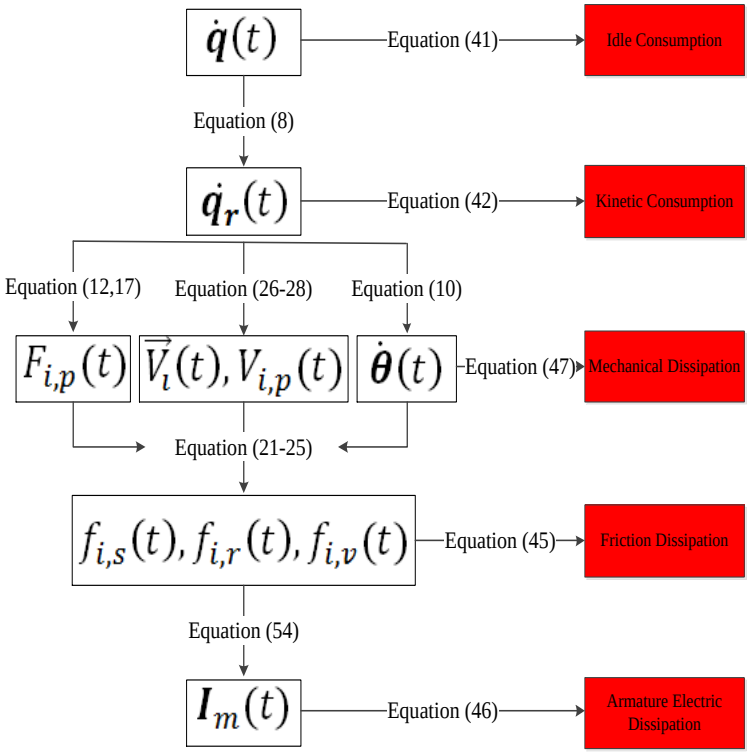


Figure 65 Energy consumption model implementation flowchart.

Energy Cost Function Pseudocode

Given an arbitrary velocity trajectory profile $\dot{\boldsymbol{q}}(t)$, calculate the energy cost $energy\_cost(\dot{X}, \dot{Y}, \dot{\emptyset})$ in the following way:

Find the time duration T

Calculate E_{idle}based on (41)

Calculate $\dot{\boldsymbol{q}}_r(t)$ using (8)

Separate $\dot{\boldsymbol{q}}_r(t)$ into N parts, each part having different constant accelerations

//In each part, calculate the following variables sequentially:

Calculate the mean velocity vel_mean

Calculate the mean acceleration acc_mean

Calculate $E_{kinetic}$using (42)

Calculate $V_{i,p}$based on (26-28)

Calculate $F_{i,p}$ using (12, 17)

Calculate $\dot{\boldsymbol{\theta}}$ based on (10)

Calculate $E_m$based on (47)

Calculate $f_{i,s}, f_{i,r}, f_{i,v}$ based on (21-25)

Calculate $E_f$based on (45)

Calculate $\boldsymbol{I}_m$based on (54)

Calculate $E_e$based on (46)

Calculate E_{total}using (48)

Return $energy\_cost = E_{total}$

4.5. Verification

In order to verify the proposed energy consumption model and to demonstrate the accuracy of the proposed energy consumption model, a simulation was first developed, and the experimental validation was then conducted. A simulation has been implemented in MATLAB based on the proposed energy consumption model. Experiments have been conducted to validate the proposed energy consumption model. The main goal of the following simulations and experiments on the robot is that the comparison of the real-time energy consumptions in the experiments and the estimated energy consumptions in the simulations provides the evidence to prove the accuracy of the proposed energy consumption model. The purpose-built Mecanum robot in Chapter 3 was used for energy consumption model verification purposes.

4.5.1 Simulation Setup

The energy consumption model was programmed in MATLAB by following the guide in Section 4.4. The proposed energy consumption model, mathematically implemented in MATLAB, simulated the energy consumptions of the robot that follows any given velocity trajectory profiles.

Table II Model Parameters of the Robot.

Symbol	QUANTITY	Value
r	wheel radius	0.11 m
l_x	distance from centre of robot to centre of Mecanum wheel in x directions	0.45 m
l_y	distance from centre of robot to centre of Mecanum wheel in y directions	0.60 m
m	mass of robot	94.00 kg
J_z	moment of inertia of robot in z direction	19.40 $kg \cdot m^2$
σ	roller angle	45.00 °
μ_r	rolling friction coefficient	0.18
μ_s	sliding friction coefficient	0.90
μ_v	viscous friction coefficient	0.20
J_w	moment of inertia of the Mecanum wheel along its rotation axis	0.04 $kg \cdot m^2$
R_a	armature resistance of the motor	0.80 Ω
M_R	static friction torque in the actuator	0.02 $N \cdot m$
P_{idle}	idling robotic power consumption	72.00 W
n	gear ratio	40.00
k_t	torque constant	0.14 $N \cdot m/A$

The key model parameters of the designed robot and their values are given in Table II. Parameters, such as the roller angle, the wheel position, and the mass of the robot, were directly measured. Some other parameters such as the gear ratio, the torque constant, and the armature resistance and so on were provided by the Company Beckhoff product specifications. The remaining parameters, such as friction coefficients, and the moment of inertia of robot and wheel, were obtained based on experiment measurements and estimated from engineering drawings.

The designed robot was equipped with four AC servomotors AM3121 with 4×140 W rated power. For data acquisition, a current sensor (Phidgets 1122 Current Sensor AC/DC) was used to measure the overall current of the robotic battery. When the robot was idling, there was an idle current of 1.51A-1.52A detected by the current sensor. The total current from the battery, measured under a stable 48V voltage was utilized to calculate the consumed power as a product of 48V voltage and the measured current.

4.5.2 Experimental Validation

The energy consumption of the designed Mecanum robot following different motion trajectories was recorded in the experiments. The designed Mecanum robot was controlled by the robotic motion control system to follow different motion trajectories. As stated in Section 3.7, the laser scanner-based robotic closed-loop motion control system drove the robot to follow the predefined velocity trajectory profiles. In the meantime, the battery output current

was recorded by the current sensor in real time. In this section, both primitive and complex omnidirectional motions have been tested on both carpet and concrete floor conditions. More advanced experimental validation that has been done to conduct polynomial-based velocity trajectory can be found in Chapter 6 Section 5.2.

Experimental Motion Trajectories

The Mecanum robot is a holonomic mobile platform that can perform omnidirectional motions on the ground. The holonomic mobile platform is able to independently control each degree of freedom of the 2D motion. Thus, three primitive omnidirectional motions include pure translation in the x_r direction (forward/backward) and the y_r direction (sideway left/right), and rotation about the origin of Ox_ry_r ($z_r$orientation/yaw in place clockwise/anti-clockwise) and four complex omnidirectional motions include combinational motion of translation in the x_r direction and translation in the y_r direction, combinational motion of translation in the x_r direction and rotation in the z_r direction, combinational motion of translation in the y_r direction and rotation in the z direction and combinational motion of translation in the x_r direction, and translation in the y_r direction and rotation in the z_r direction. Both primitive and complex omnidirectional motions were tested in the experiments. All the motions were tested on concrete at two speeds: ±0.05 m/s and ±0.1 m/s for translation in the x_r direction, ±0.025 m/s and ±0.05 m/s for translation in the y_r direction, and ±0.0125π rad/s and ±0.025π rad/s for rotation. Each motion test lasted for two seconds, including a 0.15-second acceleration from stationary, 1.75-second constant velocities and a 0.1-second deceleration to stationary. Different ground conditions including carpet and concrete were considered. In total, there are 102 types of omnidirectional motions. Three repetitions of the experiments have been conducted.

Primitive Omnidirectional Motion on Carpet

Omnidirectional motion on the ground plane consists of three motion primitives. The velocity trajectory profiles illustrated below in Figure 66, containing the predefined omnidirectional motion primitives, were sent to the robotic system to drive the Mecanum robot and simulated in the MATLAB. After each run of primitive motion, the robot stayed stationary for two seconds before the next run. Three parts include a 0.15-second acceleration from stationary, 1.75-second constant motion in target velocities and a 0.1-second deceleration to full stop. This

is because the setting of the Beckhoff servomotors constantly requires 0.15 second to accelerate and 0.1 second to decelerate within the range of available velocities.

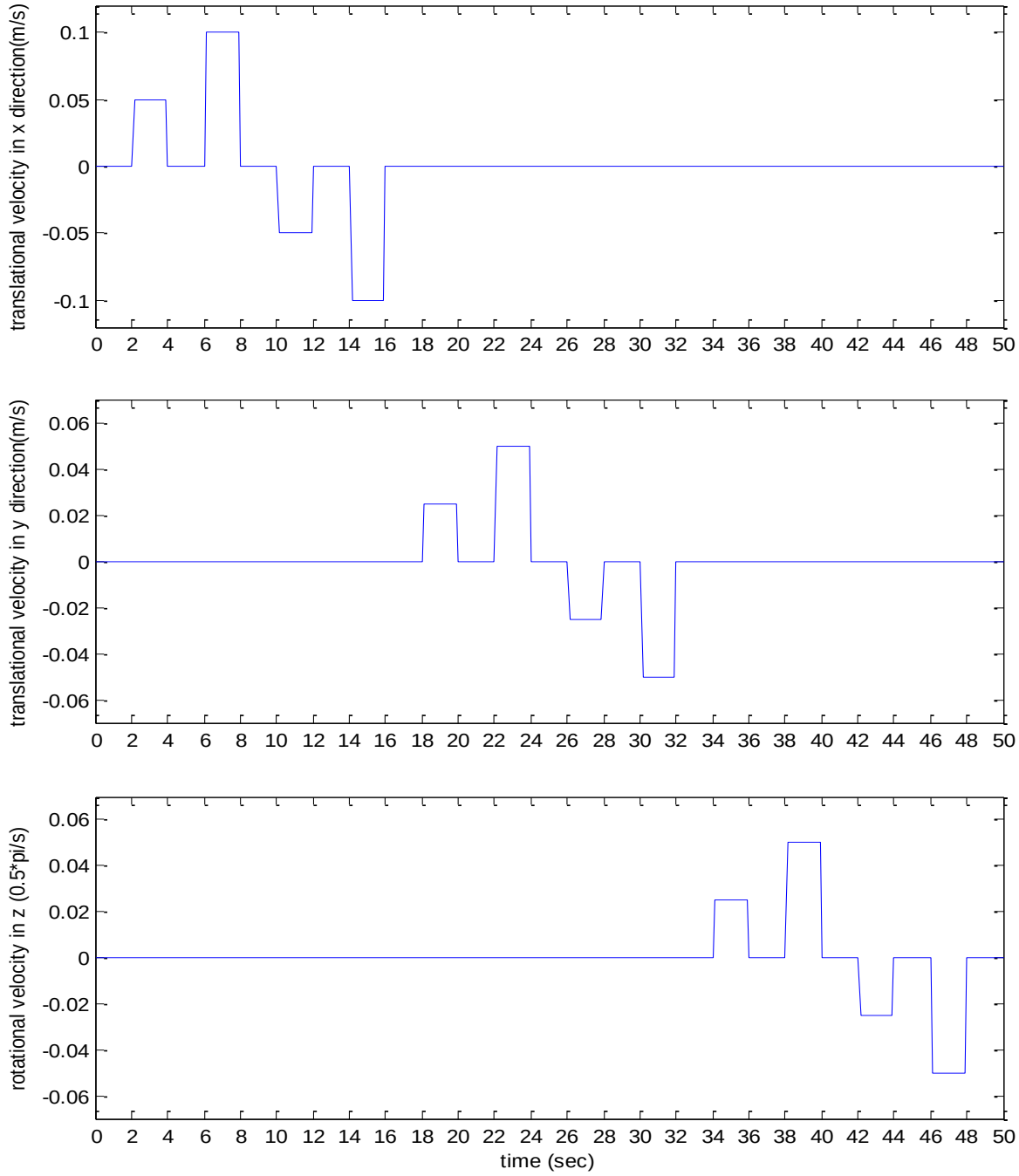


Figure 66 The velocity trajectory profile of the primitive omnidirectional motions.

The MATLAB simulation program was written according to the implementation flowchart in Figure 65. The proposed energy consumption model, implemented in MATLAB, simulated the predicted the energy consumptions according to the described motions above. The simulation expressed the energy consumption in joules. The simulated current was calculated by reversing the simulated energy. A Python test script controlled the designed robot to perform all the two-

second omnidirectional motions. The real-time battery output current that the robot consumed in the experiments was recorded by the current sensors. Both simulated and experimental currents that the robot consumed to perform the set of primitive motions are plotted in Figure 67.

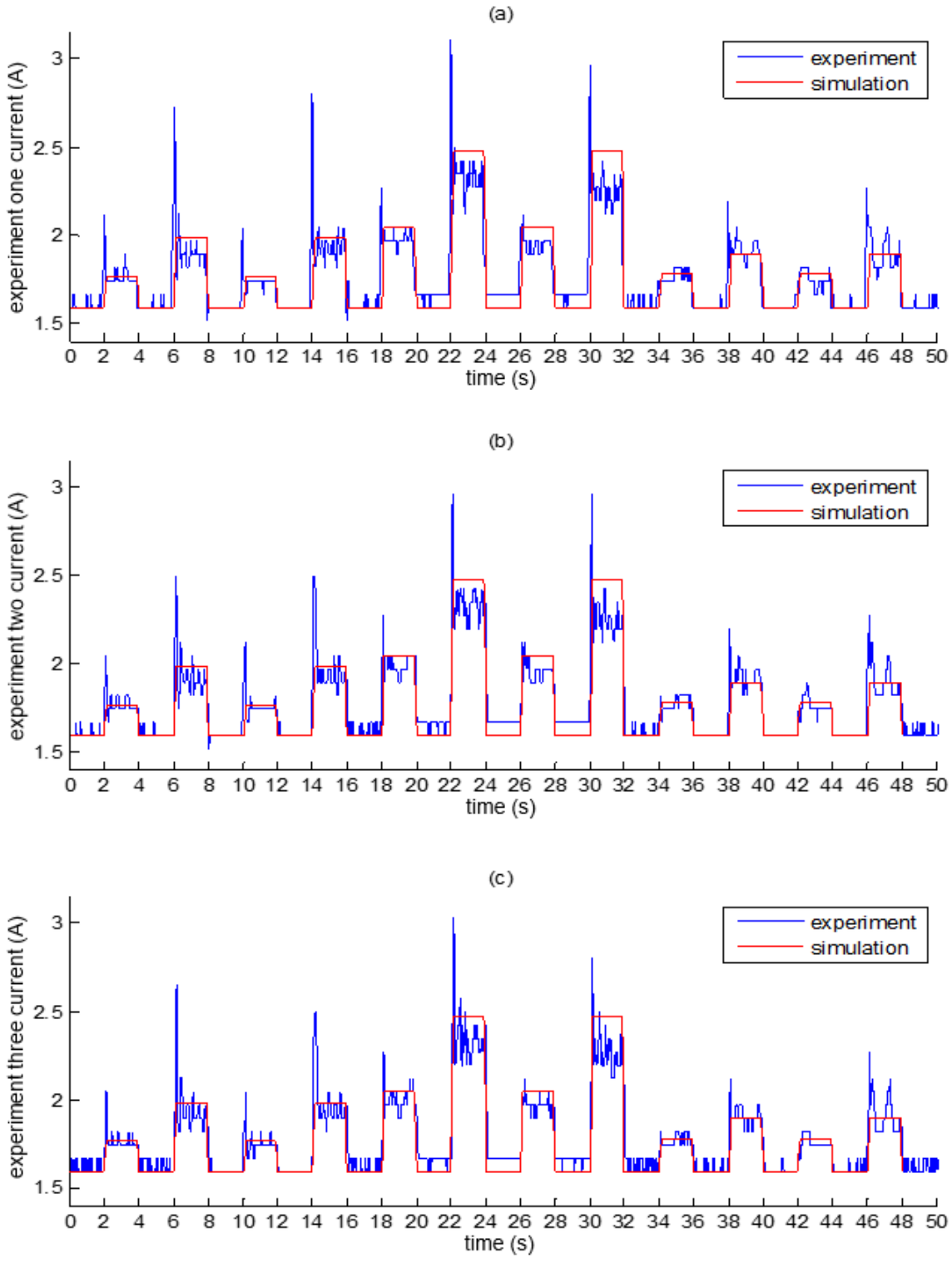


Figure 67 The simulated and experimental current consumptions of the robot.

From the experiment, it was observed that while the robot was idling, the robotic idle current ranged from 1.4A to 1.6A depending on the battery's state of charge. Thus, the same groups of experiments were conducted under the same level of battery's state of charge. When the robot starts to accelerate, a temporary high initial current is detected lasting for around 0.15 seconds, which is the acceleration time. Constant-velocity motion generates a reasonably fluctuating current. It takes about 0.1 seconds to decelerate at the range of the required velocities. The current plot also proves that the robotic servomotors are not able to reserve energy from deceleration. The difference between the simulated and experimental current plots is mainly caused without considering the high starting currents in the servomotors.

The energy consumption results are given above in Table III. Because the proposed model expresses the energy consumption in joules, a product of time integration of the measured current plot in Figure 67 and the battery voltage (48V) is calculated for the consumed total energy. The energy cost is calculated based on steps, each of which includes a two-second stand still and a two-second motion primitive. The motional columns only show the energy cost related to motions by filtering the idle energy consumption.

Table III Simulated and Experimental Energy Consumption of Primitive Motions on Carpet

Motions	Total Energy Consumption (J)				Motional Energy Consumption (J)			
	Exp. 1	Exp. 2	Exp.3	Simulation	Exp. 1	Exp. 2	Exp.3	Simulation
forward, slow	323.3	323.8	324.9	321.0	32.0	32.5	33.6	29.7
forward, fast	338.5	339.5	342.7	340.9	47.2	48.2	51.4	49.6
backward, slow	320.5	320.6	321.1	321.0	29.2	29.3	29.8	29.7
backward, fast	341.2	341.3	341.2	340.9	49.9	50.0	49.9	49.6
sideway right, slow	344.7	346.3	347.2	346.7	53.4	55.0	55.9	55.4
sideway right, fast	385	384.2	384.8	385.8	93.7	92.9	93.5	94.5
sideway left, slow	346.7	348.3	349.3	346.7	55.4	57.0	58.0	55.4
sideway left, fast	376.7	377.0	377.0	385.8	85.4	85.7	85.7	94.5
rotation right, slow	323.9	324.1	323.3	323.3	32.6	32.8	32.0	32.0
rotation right, fast	337.6	337	337.3	335.5	46.3	45.7	46.0	44.2
rotation left, slow	320.5	321.7	322.1	323.3	29.2	30.4	30.8	32.0
rotation left, fast	337	336.9	337.6	335.5	45.7	45.6	46.3	44.2

It is observed that the robot consumed a consistent amount of energy to perform the same omnidirectional motion under the same surface conditions. The total energy consumption includes both the idle energy consumption and the motional energy consumption. The

simulation and experimental results show that the energy consumption model has over 98% accuracy. After getting rid of the idle energy consumption, four fifths and almost all of the filtered simulation results remain at a high accuracy of over 95% and 90% respectively. According to the proposed energy consumption model, the same type of motion primitives at the same speed but a different direction should consume exactly the same amount of energy. In Table III, forward and backward simulation at the same speed consumes the same amount of energy, much like strafing left and right at the same speed, and rotating left and right at the same speed. However, the experimental results illustrate that there are energy consumption differences between the same types of motion primitives (e.g. forward and backward motions), which lead to inaccurate simulation results. The main reason for such experimental results is that AuckBot has different model parameters, friction coefficients in particular, between forward and backward, between strafing left and right, and between rotating left and right.

Other Results

The simulation and experiment results of the primitive omnidirectional motions on concrete, the complex omnidirectional motion on carpet and the complex omnidirectional motion on concrete are summarized in this section.

Table IV Simulated and Experimental Energy Consumption of Primitive Motions on Concrete

Motions	Total Energy Consumption (J)				Motional Energy Consumption (J)			
	Exp. 1	Exp. 2	Exp.3	Simulation	Exp. 1	Exp. 2	Exp.3	Simulation
forward, slow	318.8	318.8	318.6	318.4	27.5	27.5	27.3	27.1
forward, fast	335.4	335.4	334.5	336.2	44.1	44.1	43.2	45.0
backward, slow	315.4	315.9	315.2	318.4	24.1	24.6	23.9	27.1
backward, fast	333.5	333.0	323.6	336.2	42.2	41.7	32.3	45.0
sideway right, slow	323.0	323.4	323.2	324.6	31.7	32.1	31.9	33.3
sideway right, fast	345.9	346.5	346.2	344.0	54.6	55.2	54.9	52.8
sideway left, slow	324.3	323.8	324.0	324.6	33.0	32.5	32.7	33.3
sideway left, fast	346.7	346.5	346.0	344.0	55.4	55.2	54.7	52.8
rotation right, slow	313.7	313.5	313.3	315.8	22.4	22.2	22.0	24.5
rotation right, fast	323.6	324.0	323.5	322.3	32.3	32.7	32.2	31.0
rotation left, slow	313.4	313.1	313.0	315.8	22.1	21.8	21.7	24.5
rotation left, fast	319.2	324.5	324.4	322.3	27.9	33.2	33.1	31.0

Table V Simulated and Experimental Energy Consumption of Complex Motions on Carpet.

Motions	Total Energy Consumption (J)				Motional Energy Consumption (J)			
	Exp. 1	Exp. 2	Exp.3	Simulation	Exp. 1	Exp. 2	Exp.3	Simulation
left forward slow	335.8	338.1	339.4	327.3	44.5	46.8	48.1	50.0
left forward fast	364.6	368.9	371.8	361.7	73.3	77.6	80.5	84.4
right backward slow	338.1	339.8	334.3	327.3	46.8	48.5	43.0	50.0
right backward fast	375.8	369.4	368.0	361.7	84.5	78.1	76.7	84.4
right forward slow	337.9	343.8	338.8	327.3	46.6	52.5	47.5	50.0
right forward fast	379.4	382.1	383.9	361.7	88.1	90.8	92.6	84.4
left backward slow	333.2	335.5	334.2	327.3	41.9	44.2	42.9	50.0
left backward fast	376.4	371.2	384.6	361.7	85.1	79.9	93.3	84.4
left rotation forward slow	318.6	326.3	322.0	309.2	27.3	35.0	30.7	31.9
left rotation forward fast	342.3	345.4	349.7	331.6	51.0	54.1	58.4	54.3
right rotation backward slow	318.2	322.0	325.2	309.2	26.9	30.7	33.9	31.9
right rotation backward fast	342.9	344.0	347.2	331.6	51.6	52.7	55.9	54.3
right rotation forward slow	319.8	325.5	321.5	309.2	28.5	34.2	30.2	31.9
right rotation forward fast	344.5	346.7	345.2	331.6	53.2	55.4	53.9	54.3
left rotation backward slow	317.3	320.0	323.0	309.2	26.0	28.7	31.7	31.9
left rotation backward fast	344.2	341.8	344.6	331.6	52.9	50.5	53.3	54.3
left rotation left slow	331.9	337.2	331.5	323.0	40.6	45.9	40.2	45.7
left rotation left fast	364.7	368.8	369.5	356.3	73.4	77.5	78.2	79.0
right rotation right slow	332.6	333.4	331.8	323.0	41.3	42.1	40.5	45.7
right rotation right fast	365.9	366.6	365.1	356.3	74.6	75.3	73.8	79.0
right rotation left slow	332.4	340.3	336.3	323.0	41.1	49.0	45.0	45.7
right rotation left fast	371.4	376.7	371.3	356.3	80.1	85.4	80.0	79.0
left rotation right slow	334.8	340.4	340.1	323.0	43.5	49.1	48.8	45.7
left rotation right fast	377.3	376.3	373.9	356.3	86.0	85.0	82.6	79.0
left rotation left forward slow	337.4	339.2	336.4	327.8	46.1	47.9	45.1	50.5
left rotation left forward fast	373.7	374.6	383.2	373.2	82.4	83.3	91.9	95.9
right rotation right backward slow	337.9	338.4	343.2	327.8	46.6	47.1	51.9	50.5
right rotation right backward fast	374.8	373.0	380.3	373.2	83.5	81.7	89.0	95.9
right rotation left forward slow	335.5	337.4	340.9	327.8	44.2	46.1	49.6	50.5
right rotation left forward fast	380.3	382.4	384.7	373.2	89.0	91.1	93.4	95.9
left rotation right backward slow	339.3	340.8	348.0	327.8	48.0	49.5	56.7	50.5
left rotation right backward fast	388.0	381.8	385.8	373.2	96.7	90.5	94.5	95.9
left rotation right forward slow	342.8	345.3	343.3	327.8	51.5	54.0	52.0	50.5
left rotation right forward fast	389.7	393.0	391.2	373.2	98.4	101.7	99.9	95.9
right rotation left backward slow	339.0	340.5	345.1	327.8	47.7	49.2	53.8	50.5
right rotation left backward fast	385.7	384.7	387.8	373.2	94.4	93.4	96.5	95.9
right rotation right forward slow	342.4	340.0	338.0	327.8	51.1	48.7	46.7	50.5
right rotation right forward fast	385.0	393.3	384.8	373.2	93.7	102.0	93.5	95.9

left rotation left backward slow	341.0	340.0	342.0	327.8	49.7	48.7	50.7	50.5
left rotation left backward fast	379.7	380.1	384.0	373.2	88.4	88.8	92.7	95.9

Table VI Simulated and Experimental Energy Consumption of Complex Motions on Concrete.

Motions	Total Energy Consumption (J)				Motional Energy Consumption (J)			
	Exp. 1	Exp. 2	Exp.3	Simulation	Exp. 1	Exp. 2	Exp.3	Simulation
left forward slow	324.7	326.8	327.7	327.6	33.4	35.5	36.4	36.3
left forward fast	349.3	360.2	360.8	355.1	58.0	68.9	69.5	63.9
right backward slow	321.0	328.2	328.7	327.6	29.7	36.9	37.4	36.3
right backward fast	345.8	352.5	351.9	355.1	54.5	61.2	60.6	63.9
right forward slow	328.7	324.2	323.3	327.6	37.4	32.9	32.0	36.3
right forward fast	359.5	355.8	356.1	355.1	68.2	64.5	64.8	63.9
left backward slow	328.8	325.7	325.5	327.6	37.5	34.4	34.2	36.3
left backward fast	358.8	355.1	354.5	355.1	67.5	63.8	63.2	63.9
left rotation forward slow	320.5	317.1	316.6	317.0	29.2	25.8	25.3	25.7
left rotation forward fast	341.4	341.5	342.0	339.8	50.1	50.2	50.7	48.5
right rotation backward slow	319.3	318.0	317.6	317.0	28.0	26.7	26.3	25.7
right rotation backward fast	338.2	337.7	338.0	339.8	46.9	46.4	46.7	48.5
right rotation forward slow	320.7	318.0	317.0	317.0	29.4	26.7	25.7	25.7
right rotation forward fast	341.1	340.1	339.3	339.8	49.8	48.8	48.0	48.5
left rotation backward slow	319.0	316.6	316.2	317.0	27.7	25.3	24.9	25.7
left rotation backward fast	338.7	337.0	336.6	339.8	47.4	45.7	45.3	48.5
left rotation left slow	323.6	321.7	321.2	322.2	32.3	30.4	29.9	30.9
left rotation left fast	348.5	348.3	347.4	348.5	57.2	57.0	56.1	57.2
right rotation right slow	323.1	319.0	318.6	322.2	31.8	27.7	27.3	30.9
right rotation right fast	344.7	344.8	345.0	348.5	53.4	53.5	53.7	57.2
right rotation left slow	326.2	322.7	321.5	322.2	34.9	31.4	30.2	30.9
right rotation left fast	352.1	351.1	350.8	348.5	60.8	59.8	59.5	57.2
left rotation right slow	325.6	321.3	321.3	322.2	34.3	30.0	30.0	30.9
left rotation right fast	349.1	349.5	349.4	348.5	57.8	58.2	58.1	57.2
left rotation left forward slow	329.3	327.2	327.0	332.4	38.0	35.9	35.7	41.1
left rotation left forward fast	363.3	361.6	359.7	363.6	72.0	70.3	68.4	72.3
right rotation right backward slow	336.7	335.7	328.8	332.4	45.4	44.4	37.5	41.1
right rotation right backward fast	359.9	355.3	354.2	363.6	68.6	64.0	62.9	72.3
right rotation left forward slow	329.9	329.1	328.8	332.4	38.6	37.8	37.5	41.1
right rotation left forward fast	368.4	365.3	363.5	363.6	77.1	74.0	72.2	72.3
left rotation right backward slow	333.9	329.0	328.1	332.4	42.6	37.7	36.8	41.1
left rotation right backward fast	362.3	359.5	357.3	363.6	71.0	68.2	66.0	72.3
left rotation right forward slow	336.4	339.7	334.3	332.4	45.1	48.4	43.0	41.1
left rotation right forward fast	368.9	372.1	370.4	363.6	77.6	80.8	79.1	72.3
right rotation left backward slow	337.0	335.0	333.5	332.4	45.7	43.7	42.2	41.1
right rotation left backward fast	366.7	367.0	364.9	363.6	75.4	75.7	73.6	72.3

right rotation right forward slow	329.5	328.0	327.8	332.4	38.2	36.7	36.5	41.1
right rotation right forward fast	365.0	364.7	363.5	363.6	73.7	73.4	72.2	72.3
left rotation left backward slow	335.9	333.2	333.0	332.4	44.6	41.9	41.7	41.1
left rotation left backward fast	363.9	365.4	363.3	363.6	72.6	74.1	72.0	72.3

After comparing the simulation results to experimental results, the energy consumption model proved to have over 98% accuracy for primitive omnidirectional motions and over 95% accuracy for complex omnidirectional motions.

4.6. Summary

This chapter introduces a novel energy consumption model for the four-wheel omnidirectional Mecanum mobile robot. The comprehensive energy consumption model of the mobile robot must be understood to develop the energy-optimal autonomous navigation algorithm. The proposed energy consumption model was based on a comprehensive understanding of the kinematics, dynamics and energy flow of the Mecanum robot. To develop an energy-efficient motion planning algorithm including path planning and trajectory planning, kinematics of the mobile robot is essential. Dynamics of the mobile robot describing how the robot motion is initialized and sustained by the driven forces is without doubt related to the energy consumption of the mobile robot. Thus, this model is specific to the four-wheel Mecanum robot by considering the unique dynamics of the Mecanum wheel in omnidirectional motions. Different dynamics should be accommodated in the model in order to cope with other mobile robots. The actuator which converts the electric energy into the motion is the important link between the energy source battery and robot motion. In particular, an omnidirectional mobile drive is involved with complex dynamics and unique kinematics.

The energy consumption of the mobile robot is a complex problem involving many influencing factors. According to the existing methods, building a reliable energy consumption model for the target robot as a solid base is the common approach to solving this problem. The proposed energy consumption model of the four-wheeled Mecanum robot considers the validated factors from the recent energy model works in [49] and [51]. Based on the recently published energy model works from [49] and [51], our proposed energy consumption model considers their experimentally validated factors to address the complex robotic energy consumption model for our robot. The matters consist of the idle energy required for stand-by and the motional energy consumed by kinetic energy transformation, overcoming robotic traction resistances, electric dissipation in the motor armatures and mechanical dissipation in the actuators. But, unlike the

existing energy models, our proposed model is specific to the Mecanum robot. The main challenge was to involve the complicated dynamics of the omnidirectional Mecanum platform.

The proposed model has been experimentally validated in this study. The energy consumption model has been mathematically implemented in MATLAB and experimentally validated on the purpose-built Mecanum mobile robot. The experimental energy consumption results showed the consistency and accuracy of the model to predict energy consumptions of the Mecanum robot. After comparing the simulation results to experimental results, the energy consumption model proved to have over 98% accuracy for primitive omnidirectional motions and over 95% accuracy for complex omnidirectional motions. The energy consumption model would be applied in continuous research of energy-efficiency motion planning for the Mecanum robot. The minor inaccurate results arose from the fact that our experimental robot platform, AuckBot, had a small different amount of model parameters – friction coefficients in particular – between forward and backward, between strafing left and right, and between rotating left and right. Refinements such as different friction coefficients for the same type of omnidirectional motion in different directions will be considered in the future to improve the energy consumption model.

Chapter 5. Energy-Optimal Online Local Trajectory Planning for Autonomous Navigation

5.1. Introduction

Autonomous navigation is one of the important functions of industrial mobile robots. To realize energy-efficient reactive navigation for the autonomous Mecanum robot, the key requirement was that the mobile robot could avoid unexpected obstacles. The modern autonomous navigation system applies a local trajectory planner to achieve this. A local trajectory planner only considers the nearby environment via local perceptions and plans for short-term trajectories. When the autonomous mobile robot accomplishes a navigation task by reaching the goal pose, the resultant trajectory of the robot is a result of many local trajectories. The energy consumption of the robot in one navigation task is determined by the resultant trajectory, which is planned one by one periodically, by the local planner for online obstacle avoidance purpose. How to ensure global energy efficiency via the effort of a local planner was a challenge of this study. Most importantly, there was no existing energy-efficient motion planning study to avoid unexpected obstacles.

For the purpose of reducing the energy consumption of industrial autonomous Mecanum robots, an online local trajectory planning method was proposed in this chapter, aiming for power-minimization and energy-reduction autonomous navigation. The method extended the popular ROS DWA local trajectory planner by integrating a novel energy consumption model of the Mecanum mobile robot as a new energy-related criterion. The contribution of this chapter was that the DWA was extended by means of adding the energy consumption cost function as a new criterion for the purpose of minimizing the power consumption of the autonomous Mecanum robot. Based on the proposed extended Dynamic Window Approach (DWA) that could minimize power consumption, energy-reduction autonomous navigation was proposed via the combinational cost objectives of low power consumption and high speed.

5.2. Methodology

A local trajectory planning method that both reduces overall energy consumption in the global sense for autonomous navigation, and reactively avoids unexpected obstacles, is essential to realize energy-efficient autonomous navigation. The DWA is a well-known reactive collision avoidance navigation concept and has been implemented as a popular online local trajectory

planner in the ROS navigation stack. In addition, DWA has proven performance over a range of extended studies [45, 46, 48], so the DWA was chosen to be extended for the purposes of energy-efficient reactive navigation. The DWA was extended by means of considering a new energy-related criterion. Based on the energy consumption model of the four-wheeled omnidirectional Mecanum mobile robot, the omnidirectional velocities of the robot involving three degrees of freedom were then optimized in the extended DWA-based local trajectory planner.

5.2.1 Existing Methods Review

Energy-Efficient Motion Planning

Energy-efficient motion planning methods have been applied to reduce the energy consumption of mobile robots. Mei et al. introduced a method of reducing energy consumption of mobile robots for open area exploration via energy-efficient motion planning [52]. Up to 51% of energy savings were demonstrated for a search of an open area by comparing the energy efficiencies of different motion trajectories. The energy consumption of the robot was modelled by using a sixth-degree polynomial to fit the relationship of motor speed and power consumption. Sun and Reif concentrated on the robotic energy-minimum path planning problem on terrains [53]. The energy cost in their study was based on gravity and friction. Recently, Liu and Sun proposed an experimentally validated energy model for a wheeled mobile robot by considering kinematic energy transformation, overcoming resistive friction and electric supply of onboard devices. Then, a model-based energy criterion was added into the A* algorithm to find the energy-minimum path [49]. To follow the found path, they proposed parameterized smooth trajectory planning using a cubic Bezier curve for minimum energy consumption. In [69], polynomial parameterization was applied to analytically find the near minimal-energy, real-time and collision-free path while considering for other criteria for a carlike mobile drive. The modified Newton algorithm was used in [71] to optimize the energy consumption in nonholonomic motion planning. But these methods only apply to nonholonomic mobile robots and do not apply to holonomic omnidirectional mobile robots. In addition, these methods do not consider the online local energy-efficient trajectory planning problem. To be precise, these methods cannot deal with avoiding unexpected obstacles.

Kim et al. studied online minimum-energy trajectory planning of both nonholonomic and holonomic drives on a straight-line path [50, 51]. Their energy cost function is the sum energy drawn from the onboard batteries and includes energy dissipation by the motor armature, the

energy overcoming frictions and the kinetic energy of the robot. A closed-form solution of the minimum-energy rotational velocity trajectory was found by using Pontryagin's minimum principle, and the minimum-energy translational velocity trajectory was found using a new researching algorithm. Their results showed that following the same straight-line path via different velocity profiles consumed different amounts of energy, but their studies were restricted to straight-line paths and stationary states in the beginning and at the end, and hence became less practical for autonomous navigation.

Power-Optimization Motion Planning

Because mobile robots usually source energy from batteries that store limited energy, most of the aforementioned energy-efficient studies aim to minimize the energy consumption instead of the power consumption. The robotic energy consumption to complete a task is the integral of its power consumption over the task duration. The power consumption, which is the energy spent per unit of time, is also a critical criterion to design and control mobile robots with self-sustaining power supplies such as solar power [65-68]. Energy-conservation techniques are proposed to improve energy efficiency for mobile robots by reducing power consumption of the robotic electronic components in [57]. The issue of the power minimization via motion planning for holonomic mobile robots with redundant maneuverability has not been fully studied.

5.2.2 Comparisons with Existing Methods

The low-level trajectory planner locally makes motional decisions to track the given path in the meantime ready for deviation in case of unforeseen circumstances. Most existing energy-optimal trajectory planning was based on complex mathematical optimization algorithms, e.g. finding energy-optimal solutions relying on constrained polynomial parameterized trajectories [69], using optimal control theory to find an energy-optimal velocity profile to follow the straight-line path [51], applying calculus of variation to find smoothness-optimal trajectory as energy conservation [70] and a modification of the Newton algorithm for nonholonomic energy-optimization motion [71]. The main advantage of applying mathematical optimization algorithms such as optimal control theory or calculus of variation is the guarantee of optimal or near-optimal solutions. However, the algorithms usually require the planning problem formulation to have certain levels of constraints such as fixed initial and final stationary states, specific shapes of path, defined path-description function, and unique path characters like

smoothness, or specified nonholonomic constraints. It costs a lot of time and effort to fit these algorithms into autonomous navigation.

[49] proposed energy-minimization path planning and energy-minimization trajectory planning. Since the global path planning in autonomous navigation is usually reduced to a 2D representation by assuming the robot to be a point, the proposed path planning methods with minimum energy consumption in [49, 53] are applicable to both nonholonomic and holonomic mobile robots. Thus, this chapter only focuses on the local trajectory planning of autonomous navigation. In [49], the optimal trajectory planning using parameter optimization is proposed. As the proposed motion planning is specific to a two-wheeled mobile robot and due to the dynamics constraints of the two-wheeled mobile robot, the proposed optimal trajectory planning only applies for straight-line and Bezier-curve paths for smoothness and controllability. Instead, in this proposed trajectory planner, a four-wheel Mecanum omnidirectional mobile robot is considered due to its holonomicity and omnidirectional maneuverability. The omnidirectional Mecanum mobile robot can move in any arbitrary direction with any arbitrary orientation. To plan the trajectory of such kind of omnidirectional mobile robot, the proposed trajectory planning is able to serve for omnidirectional mobile robots.

Kim et al. have conducted many studies on both online and offline energy-minimization trajectory planning of both nonholonomic and holonomic drives on straight-line paths [50, 72-74] and minimum-time trajectory planning following specified bounded-curvature paths [63]. They showed 5% to 10% energy saving compared to the trapezoidal velocity profile. In [51], minimum-energy trajectory planning using optimal control theory and Pontryagin's minimum principle on a straight-line path for three-wheeled omnidirectional mobile robots is proposed and is to be extended on curve-line paths in the future. However, their studies are restricted to specified shapes of paths. Additionally, their research problem was formulated as that the initial and final states of the mobile robots are both in the full stop state. In their approach, the initial and final velocities are constrained to be zero, which means that the mobile robots need a complete stop once there is a path type change, e.g. from a straight-line path into curve-line path. This is quite impractical for energy minimization.

The aforementioned optimization algorithms are restricted from further application in the trajectory planning of autonomous navigation. The proposed local trajectory planning technique based on DWA is a generic approach to the online trajectory planning of autonomous

navigation. Firstly, the proposed generic technique is inspired by DWA, so it can apply to various kinds of holonomic or nonholonomic mobile robots. Secondly, the trajectory planning technique, which has been successfully applied as a local trajectory planner in an autonomous navigation system, copes with any shapes of given paths. Thirdly, the local trajectory planner via the proposed technique continuously works on one local path segment by another according to the current state of motion, so it is compatible with online path planners while being able to reactively avoid unforeseen obstacles.

5.2.3 Dynamic Window Approach Methodology Review

To perform robust autonomous navigation, there are many sophisticated techniques to online plan local trajectory. The trajectory planner works online to update the local control commands every control cycle because autonomous mobile robots should be able to react to their surrounding environment that is subject to change. The DWA was originally a reactive collision avoidance technique designed for nonholonomic synchro-drive robots [82], with a similar approach of vector field and vector histogram approaches. DWA has been implemented as a popular online local trajectory planner in the ROS Navigation Stack.

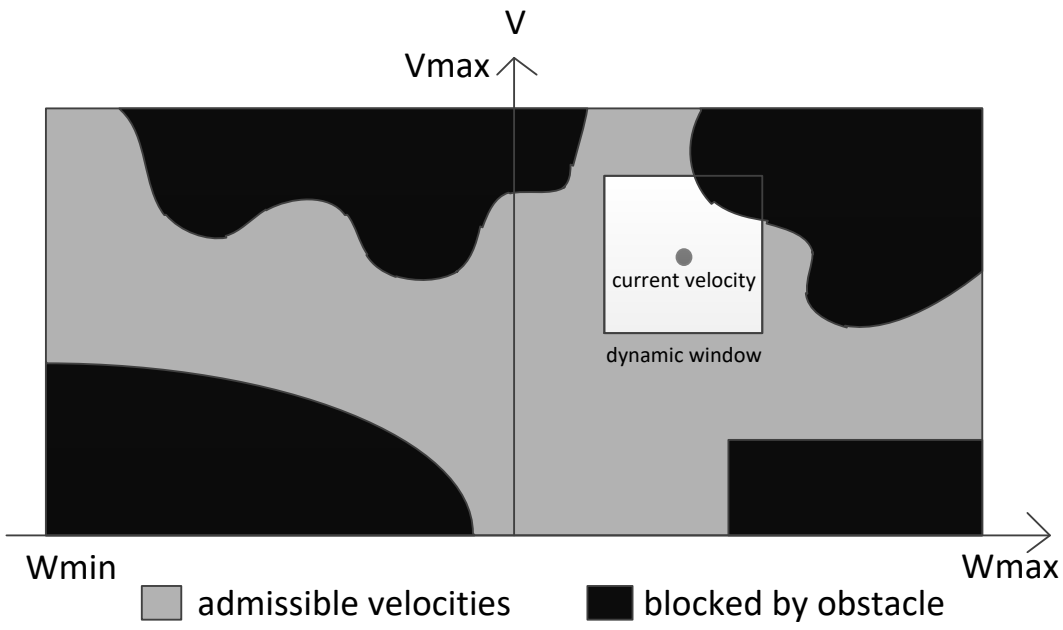


Figure 68 The dynamic window of synchro-drive robots. The possible velocity trajectories of the robots are limited by the maximum translational and rotational velocities. Due to the locations of the surrounding obstacles and the maximum accelerations, some velocity trajectories are not feasible. At such velocities, the robot cannot safely stop before hitting the obstacles. These infeasible trajectories are shown in black colour. The rest of the trajectories are considered as admissible velocities, shown in grey colour. The reachable admissible velocities are further restricted within a white dynamic window, centred at the current velocity. The area of the dynamic window is determined by maximum accelerations and the control cycle time.

DWA continuously updates the velocity command for mobile robots to execute for the coming short period of time interval while respecting the kinematic and dynamic constraints of the robots. According to the kinematic constraints of a synchro-drive robot, a search space of velocity commands is the set of circular trajectories described by pairs (v, w) of translational and rotational velocities. The basic scheme of DWA performs a forward simulation to find all admissible velocities that allow the robot to safely stop before hitting an obstacle. The admissible velocities are selected based on current position, current velocity, acceleration limit and locations of surrounding obstacles. Then reachable admissible velocities are further restricted within a dynamic window, as shown in Figure 68. The dynamic constraints of the robot such as acceleration capability and the time interval determine the reachable admissible velocities based on current velocity.

An optimization process is then performed over these admissible velocities within the dynamic window to pick the one with highest utility or least cost, e.g. maximal robot alignment with the goal direction, furthest trajectory away from the obstacles and highest speed magnitude. Different suggestions of the utility functions are proposed in [45-48, 82]. Brock et al. extended the original DWA by looking at holonomic robots and more importantly, added information about connectivity to the goal globally by introducing a NF navigation function [45]. In [46], they reformulated DWA as a model predictive control (MPC) by assuming a holonomic robot model and chose accelerations as control commands in the search space in contrast to the original DWA, where velocity is used to control the robot. Another two recent DWA improvements [47, 48] guarantee a global convergence to the goal by introducing a new objective function and considers the shape of the robot using MPC respectively. Based on these aforementioned studies, DWA has been proven to be a stable technique for both holonomic and non-holonomic drives with excellent experimental results. More importantly, DWA has been shown to easily accommodate with extensions, being flexible with its utility function.

The open source Robotics Operating System (ROS) software library provides convenient setup and configuration of the 2D navigation stack on a robot, including the DWA local planner package. The default cost function of the DWA local planner package in ROS has three criteria, which determine how much the robot should stay close to the given path, attempt to reach its local goal and attempt to avoid obstacles.

5.3. Extended Dynamic Window Approach

DWA continuously updates the velocity command for mobile robots to execute for the coming short period of time interval while respecting the kinematic and dynamic constraints of robots. The DWA has two steps: search space and optimization process. In this study, the DWA employed a cost function involving the energy-related criterion to search for the optimal omnidirectional velocity trajectories in a cuboid space.

5.3.1 Cuboid Search Space

The Mecanum robot is holonomic and its velocity trajectory is described using a triplet of omnidirectional velocities $(\dot{X},\ \dot{Y},\ \dot{\emptyset})$. Hence, a cuboid search space that contains all the allowable translational velocities $\dot{X}$ and $\dot{Y}$ and rotational velocities $\dot{\emptyset}$ of the robot is drawn in Figure 69 for illustration purposes.

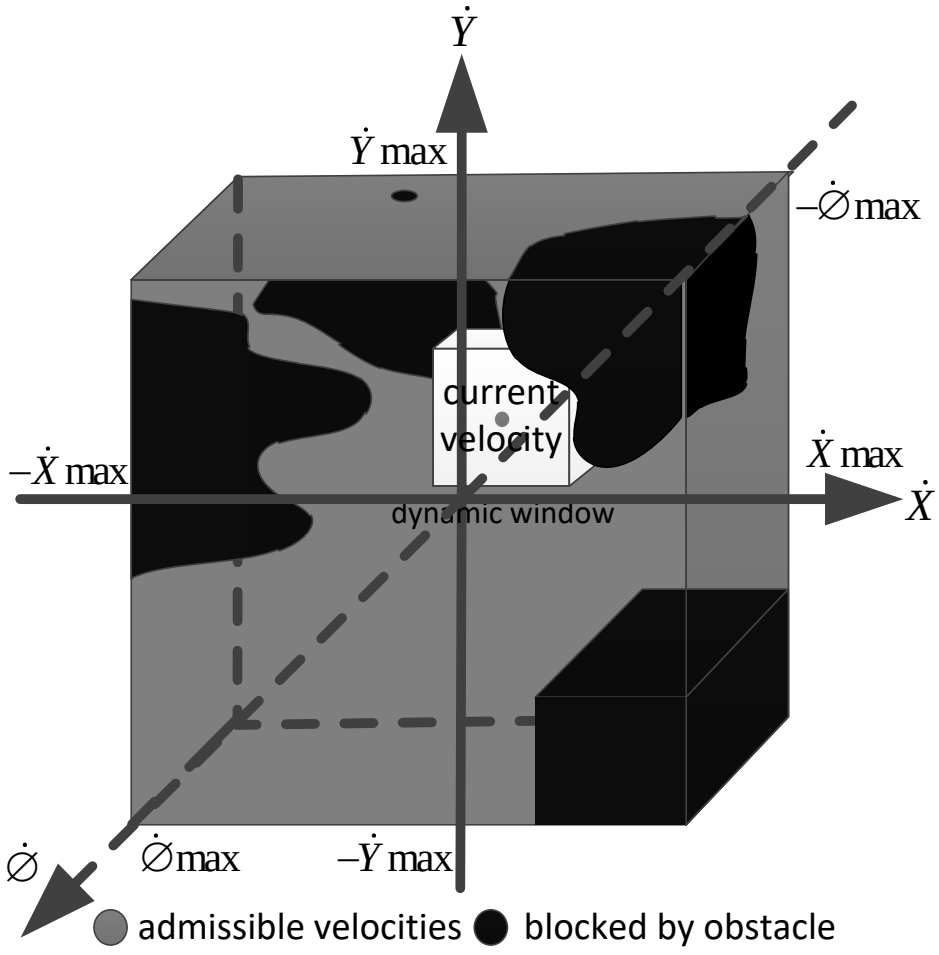


Figure 69 Cuboid search space of the DWA. The Mecanum robot is able to independently control its translational velocities $\dot{X}$ and $\dot{Y}$ and rotational velocities $\dot{\emptyset}$ so a cuboid dynamic window is presented.

The dimension of the space is defined by the maximum velocities $\dot{X}_{max}$, $\dot{Y}_{max}$, and $\dot{\emptyset}_{max}$ of the Mecanum robot in both positive and negative directions. Part of the cuboid search space is blocked by obstacles. The omnidirectional velocities in these blocked search spaces are unsafe because the robot cannot stop before hitting nearby obstacles at such velocities. The rest of the safe omnidirectional velocities are the so-called admissible velocities. According to the DWA [82], the cuboid search space distinguishes the admissible velocities that still allow the robot to stop before hitting an obstacle by considering the dynamic limits of the robot. The admissible velocities that are then further restricted within a local dynamic window are defined as reachable admissible velocities. The dimension of the dynamic window or the range of the reachable admissible velocities is determined by the forward-simulation time of the dynamic window and the dynamic constraints of the robot. The centre of the dynamic window usually lies at the current velocity in the search space. A cuboid dynamic window is located according to the current velocities within the search space. Considering the limits of omnidirectional accelerations, all the admissible velocities in the cuboid dynamic window are reachable within a short time interval T_{DW}. After each time interval T_{DW}, the search space and the dynamic window will be updated. It is important to mention that the DWA evaluates the optimal velocities within each local time window that rolls forward. Every local trajectory shares the same amount of time.

5.3.2 Optimization Process

All the reachable admissible omnidirectional velocities within the cuboid dynamic window are potential candidates of velocity command for the next control cycle. An optimal triplet of omnidirectional velocities is picked relying on an optimization process. Different suggestions of objective function criteria can be found [45-48, 82]. Trajectory planning of autonomous navigation aims to follow the given global paths and to avoid unforeseen obstacles, so path track and obstacle clearance are the desirable behaviours of a DWA-based trajectory planner. Other preferred partialities such as global goal approach, alignment with local goal heading, speed magnitude and so on, depend on design purposes.

$$COST = \alpha \cdot path\_distance + \beta \cdot goal\_distance$$

$$+\gamma \cdot obstacle\_occupation + \delta \cdot energy\_consumption \quad (56)$$

In this study, the cost function above that based on the ROS dwa_local_planner [41] is put forward for an optimal triplet of omnidirectional velocities. The local trajectory planer is aimed to track the given path and approach the local goal while avoiding any unforeseen obstacles.

Here, the first three terms are associated respectively with how much the robot stays close to the given path, how fast the robot approaches the local goal, and how far the robot stays away from an obstacle. In this PhD project, the fourth criterion of robotic energy consumption is considered and added into the DWA optimization process. Two different ways to implement the energy-related term $energy\_consumption$ have been both studied, depending on whether the global information is available to the local trajectory planning or not. The coefficients α, β, γ and δ are four normalized weights.

5.3.3 Energy Consumption of the Local Trajectory within the Dynamic Window

To be utilized in the optimization process, the energy criterion should be mathematically expressed as a cost function. Because the global information is not always available to the local trajectory planner, a local trajectory energy cost function is first introduced. The local trajectory energy cost function only considers the robotic energy consumption within the dynamic window. Based on the energy consumption model proposed in Chapter 4, a local trajectory energy cost function C_{lt} is defined to calculate the minimum energy consumed to attain and sustain the omnidirectional velocity triplet locally within the dynamic window.

$$C_{lt} = E_{total}\left(\dot{X}_{DW}, \dot{Y}_{DW}, \dot{\phi}_{DW}, T_{DW}\right) = E_{idle}(T_{DW}) + E_{motion}\left(\dot{X}_{DW}, \dot{Y}_{DW}, \dot{\phi}_{DW}, T_{DW}\right) \quad (57)$$

The optimization process of the conventional DWA only takes effect within the local dynamic window, whose size is determined by the constant time interval of control cycle T_{DW}. The idle energy consumption $E_{idle}(T_{DW})$ is the same for all the candidate velocity trajectories – the reachable admissible omnidirectional velocity. The cost term $E_{idle}(T_{DW})$ does not take effect in the optimization process at all.

$$C_{lt} = E_{motion}\left(\dot{X}_{DW}, \dot{Y}_{DW}, \dot{\phi}_{DW}, T_{DW}\right) \quad (58)$$

Only $E_{motion}\left(\dot{X}_{DW}, \dot{Y}_{DW}, \dot{\phi}_{DW}, T_{DW}\right)$ evaluates the energy-optimal omnidirectional velocity over a constant period of time T_{DW}. Thus, C_{lt} optimizes P_{motion} of the trajectory within the dynamic window. In addition, C_{lt} also optimizes P_{motion} of the trajectory in the global sense for an autonomous navigation task. This will be mathematically proven later.

However, the local trajectory energy cost function C_{lt} is lack of foresight. C_{lt} only finds the energy-minimum velocity trajectory within the local dynamic window timeframe. Thus, C_{lt} utilized in the optimization process prefers slow motion. This is because low-speed motion tends to consume less energy locally over the same period of time T_{DW}. But slow motion that increases the task duration of the overall resultant trajectory consumes more energy globally in the rest of path. Energy consumption is the integral of power consumption over time. Additional action is necessary to consider the time influence on energy consumption. With global information, an extra global trajectory energy cost function is suggested to be added into the energy criterion. Without the global information, high weight ratios of β and δ in (12) are suggested as a compensation method.

5.3.4 Energy Consumption of the Global Trajectory outside the Dynamic Window

When global information is available to the local planner, considering the robotic energy consumption by the rest of the planned global path in $energy\_consumption$ compensates for the lack of foresight in the local planning. Because computation time is critical to the on-line planner, an energy heuristic cost function C_{Heu} is defined as the heuristic energy consumption by the rest of the path.

$$C_{Heu} = E_{idle}(T_r) + E_{motion}\left(\dot{X}_{end}, \dot{Y}_{end}, \dot{\emptyset}_{end}, T_r\right) \tag{59}$$

$$T_r = \frac{L_r}{\sqrt{\dot{X}_{end}^2 + \dot{Y}_{end}^2}} \tag{60}$$

By assuming no further acceleration, T_r is the time for the robot to finish the rest of the path at a translational speed at the end of the local velocity trajectory. $\dot{X}_{end}$, $\dot{Y}_{end}$ and $\dot{\emptyset}_{end}$ are the omnidirectional velocities at the end of the local trajectory, while L_r is the length of the rest of the path.

A combinational trajectory energy cost function C_E is defined by combining C_{lt} and C_{Heu}. Both C_{lt} and C_{Heu} are weighted with respect to the local trajectory length L_t and the remaining global path length L_r.

$$C_E = C_{lt}\frac{L_t}{L_r} + C_{Heu}\frac{(L_r - L_t)}{L_r} \tag{61}$$

5.3.5 Compensation Method without the Global Information

The velocity trajectories are evaluated within the local time window that rolls forward. The energy consumption is optimized in the constant-time window and hence contributes to minimizing the overall average power consumption of the resulting trajectories in the global sense for autonomous navigation. Thus, δ determines the bias of optimizing the power consumption against other cost objectives.

Mathematical Proof of Overall Average Power Consumption Minimization

This section mathematically proves that a sequence of energy-minimization velocity trajectories that are evaluated by the DWA results in the overall average power minimization in the global sense for autonomous navigation.

The DWA local trajectory planner plans the entire robotic velocity trajectory within a local constant-time dynamic window that rolls forward. It is assumed that, given a global path to follow, the rolling-forward dynamic window separates the global path into n pieces of path segments after the robot finishes the autonomous navigation task. The velocity trajectory to follow every path segment is planned within one dynamic window. Thus, the time duration t of each velocity trajectory is the same but the length of each path segment is not necessarily the same, as shown below in Figure 70.

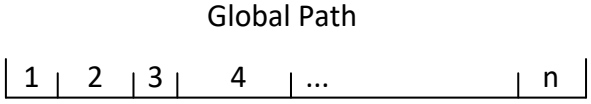


Figure 70 Path segments separated by the dynamic windows.

$$t_1 = t_2 = t_3 = t_4 = \cdots = t_n = t \tag{62}$$

The overall time T in the global sense is:

$$T = t_1 + t_2 + t_3 + t_4 + \cdots + t_n = n \cdot t \tag{63}$$

The energy consumptions of the velocity trajectories $e_1, e_2, e_3, e_4, \ldots$, and e_n are positive numbers and are also minimized within each corresponding dynamic window. The overall energy consumption E in the global sense is:

$$E = e_1 + e_2 + e_3 + e_4 + \cdots + e_n \tag{64}$$

Thus, the overall average power consumption P in the global sense is:

$$P = \frac{E}{T} = \frac{e_1 + e_2 + e_3 + e_4 + \cdots + e_n}{n \cdot t} \tag{65}$$

$$P = \frac{average(e_1, e_2, e_3, \ldots, e_N)}{t} \tag{66}$$

t is a constant value. If $average(e_1, e_2, e_3, e_4, \ldots, e_N)$ is minimized, the overall power consumption will be minimized. P is directly proportional to the average energy consumption per the number of dynamic windows $average(e_1, e_2, e_3, \ldots, e_N)$. Because each energy consumption of $e_1, e_2, e_3, \ldots$, and e_n is minimized within each dynamic window, $average(e_1, e_2, e_3, \ldots, e_N)$ is guaranteed to be the minimized value.

One possible way to achieve a lower value than $average(e_1, e_2, e_3, e_4, \ldots, e_N)$ is to both increase the values in one or many of $e_1, e_2, e_3, e_4, \ldots, e_n$ and expect for smaller additional $e_{n+1}, e_{n+2}, \ldots$ so that $average(e_1^*, e_2^*, \ldots, e_n^*, e_{n+1}^*, \ldots) < average(e_1, e_2, \ldots, e_N)$. First, the larger energy consumption means to result in energy-inefficient omnidirectional motions or faster speeds in the local trajectory planning. Because the DWA-based local trajectory planner has other criteria such as $goal\_distance$ to pick fast-speed trajectories, a fast-speed trajectory will be picked rather than an energy-inefficient trajectory if both consume the same amount of energy consumption. In DWA-based local trajectory planning that relies on a given global path to approach the goal pose, faster speeds cannot result in more additional e_{n+1}, e_{n+2}, etc.. Second, if the minimized energy consumption for $e_1, e_2, e_3, e_4, \ldots, e_n$ has been maintained to a very low level, it is less likely for the additional minimized $e_{n+1}, e_{n+2}, \ldots$ to be a lower level. Thus, the overall average power consumption is also minimized in the global sense. The DWA cost objective $energy\_consumption$ determines the general energy consumption level in every dynamic window. So, $energy\_consumption$ determines $average(e_1, e_2, e_3, \ldots, e_N)$ and P.

Reducing Energy Consumption without the Global Information

It is difficult to realize overall energy optimization via only the effort of local trajectory planning. The accumulation of the energy optimization in each local window does not necessarily achieve overall energy optimization in the global sense for autonomous navigation.

The local trajectory energy cost function C_{lt} optimizes the motional power of the resultant trajectory in an autonomous navigation task. However, C_{lt} also results in high task duration of the resultant trajectory. The locally energy-optimal velocity trajectories tend to be slow motions

that consume less energy locally within the dynamic window but much more energy globally by the rest of the path than fast motions. As both low task duration and low motional power are two essential conditions to reduce overall energy consumption of the resultant trajectory in an autonomous navigation task, a compensation method which decreases task duration of the resultant trajectory is needed. Another criterion, $goal\_distance$, controls speed by minimizing the distance to the local goal from the endpoint of the trajectory. So, the combinational cost objectives of high speed and optimal motional power is put forward to reduce energy consumption. Both high weight ratios of β and δ in the DWA cost function are suggested as a compensation method.

5.4. Implementation

The purpose-built Mecanum robot, AuckBot had the ROS framework installed in the robotic control system. A ROS local planner package, dwa_local_planner, an implementation of the Dynamic Window Approach is available in the 2D Navigation Stack. The dwa_local_planner has been easily configured for FOMR due to its holonomy. The default cost function of the ROS DWA local planner package has three criteria, which determine how much to stay close to the given path, how fast to attempt reaching its local goal and how far to attempt avoiding obstacles. A new criterion based on the validated energy consumption model was added into the ROS DWA local trajectory planner. The pseudocode of robotic energy consumption cost function can be found in Chapter 4 Section 4 and the implementation flowchart in Figure 65. Through scaling different weights of each cost function terms (coefficients $\alpha\ \beta\ \gamma\ \delta$), the extent of energy consumption influenced by the energy cost term and the correlation between the energy term and other three original cost terms are important conclusions in this trajectory planning research.

5.5. Experimental Verification

The proposed DWA-based technique aimed to optimize the power consumption and reduce the energy consumption of any autonomous navigation task by acting as a local trajectory planner. To investigate the functionalities of the proposed technique in the application of autonomous navigation, this section conducted a series of experiments. In the following experiments, each experiment was a complete navigation task, which applied the proposed local trajectory planner to track global paths. The resultant global trajectory of autonomous navigation was determined

by the local trajectory planner, so the characters of the resultant trajectory were global, including task duration, sum energy, idle energy, kinetic energy and kinetic power.

5.5.1 Experimental Design

To experimentally validate the proposed local trajectory planning method of the extended DWA for autonomous navigation, both deliberative and reactive situations were considered. In the deliberative situation, map-based paths were fed into the local trajectory planner, including both straight-line paths and curved-line path. Three common types of straight-line paths included longitudinal path, lateral path and diagonal path because the Mecanum mobile robots could perform omnidirectional motions. Additionally, the reactive situation of placing unexpected obstacles in the testing paths was also investigated via experiments. This required the DWA-based local trajectory planner to agilely avoid unexpected obstacles. When the local perception of the robot detected an unexpected obstacle along the path, the local trajectory planner must react by deviating from the obstacle. Thus, five test paths consisted of the forward path, the sideways path, the diagonal path, the curved path and the obstacle path as shown in Figure 71.

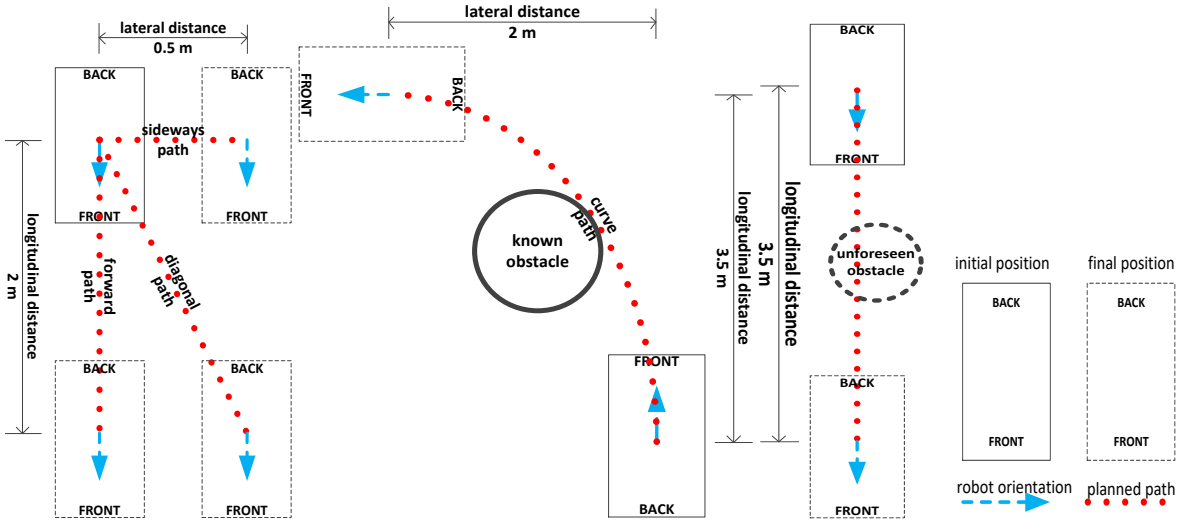


Figure 71 Test paths. (a) Straight-line paths in Scenario One. The straight-line paths include forward, diagonal and sideway paths. (b) Curved-line path in Scenario Two. (c) Reactive obstacle avoidance path in Scenario Three.

Making sure that the global path planner of the ROS 2D Navigation Stack generated those test paths to the local DWA planner, three experimental scenarios were designed in Figure 71. In Scenario One, open ground without any known or unforeseen obstacles, if the navigation task places a goal position with the same initial orientation two meters forward away, the global path planner will generate a straight-line two-meter forward path to the local trajectory planner.

In Scenario One, open ground without any known or unforeseen obstacles, if the navigation task places a goal position with the same initial orientation half a meter sideways away, the global path planner will generate a straight-line half-meter sideways path for the local trajectory planner. In Scenario One, open ground without any known or unforeseen obstacles, if the navigation task places a goal position with the same initial orientation two meters forward and half a meter sideways away, the global path planner will generate a straight-line diagonal path to the local trajectory planner. In Scenario Two, ground with a known obstacle without unforeseen obstacles, if the navigation task places a goal position with a 90° orientation change 3.5 meters forward and two meters sideways away, the global path planner will generate a curved-line path to the local trajectory planner. In Scenario Three, open ground with an unforeseen obstacle, if the navigation task places a goal position with the same initial orientation 3.5 meters forward away, the global path planner will generate a straight-line 3.5-meter forward path to the local trajectory planner. Once the unforeseen obstacle is detected within the local perception, a sudden alternative path will be generated, and the trajectory planner will reactively deal with the sudden appearance of the obstacle.

Experiment Setup

Figure 72 Experimental environment setup. A remote control PC was connected to the robotic system via Wi-Fi. The boundary of the robotic playground was made of Kraft paper.

The experiments were conducted in a laboratory environment with a concrete ground condition shown in Figure 72. Three aforementioned experimental scenarios were set up. Five common types of paths, with which local trajectory planners usually cope, including straight-line forward path, straight-line sideway path, straight-line diagonal path, curved-line path and reactive obstacle path were also set up. Applying the Navigation Stack, Auckbot was able to build a costmap of the experimental environment using the SLAM technique. Furthermore, two kinds of experiments were conducted to investigate the newly fused DWA cost function. The first one was to prove the function of power optimization and the second one was to investigate the performance of the energy reduction. Behaviours of the trajectory planning could be changed dramatically by changing the weights of each cost term in the cost function. The feasible default weights of the ROS DWA cost function were kept as reference. In the following sections, the weights of each cost term were presented as weight ratios in percentages. Current sensors measured the energy consumption of AuckBot every 0.03 second. Laser range scanners recorded the translational displacement and rotational position of AuckBot every second. Each result came from 3-5 repetitions. The following figures illustrate the test paths that were generated in the ROS navigation system based on the experimental environment.

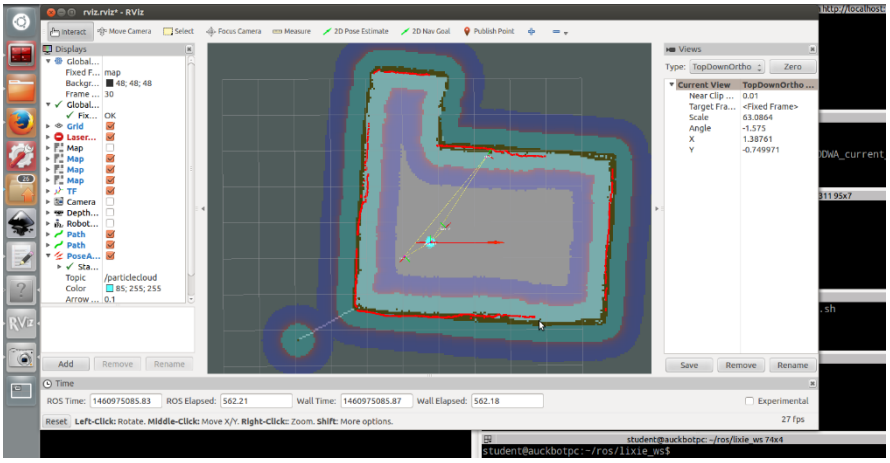


Figure 73 Straight forward path shown in the ROS map.

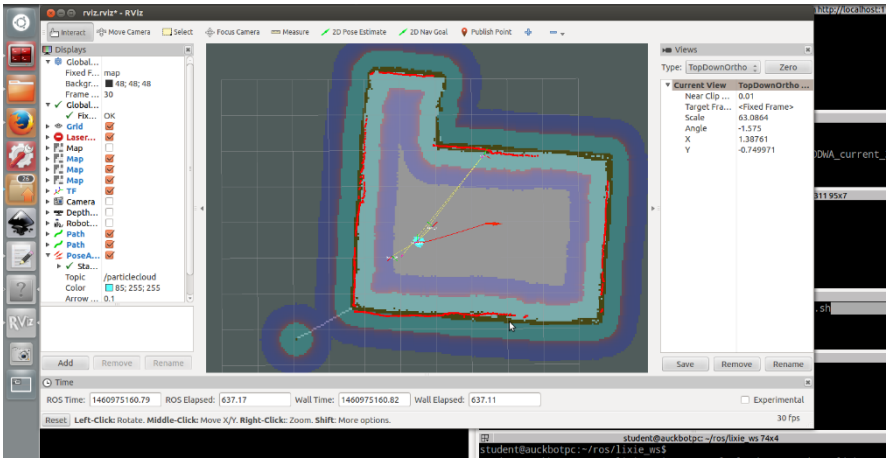


Figure 74 Straight diagonal path shown in the ROS map.

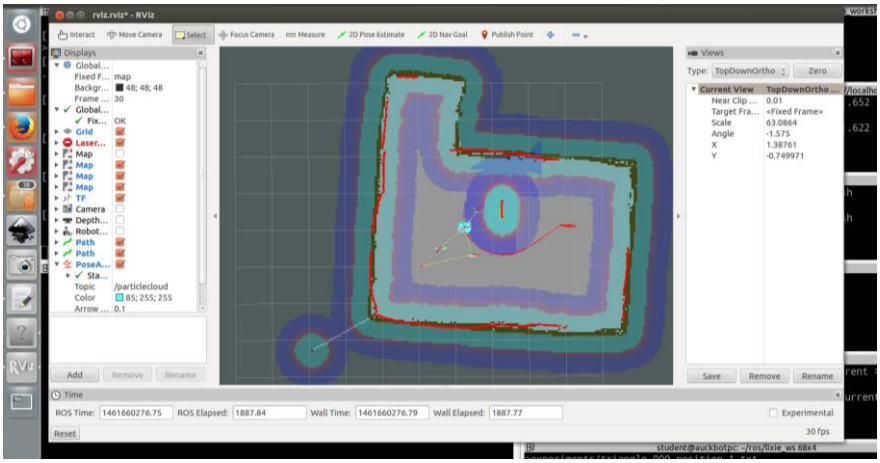


Figure 75 Curved path shown in the ROS map.

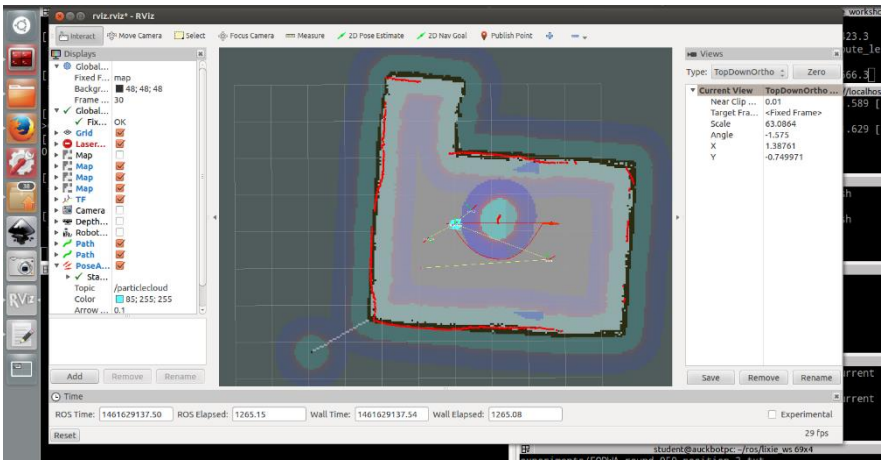


Figure 76 Reactive obstacle avoidance path shown in the ROS map.

As shown in Figure 76, a straight test path was initially generated in the ROS map. As soon as the robot detected the unexpected obstacle in the way, an alternative curved path was reactively re-generated. This was in accordance with Scenario Three of Figure 71 (c).

5.5.2 Experimental Validation of the Power-Optimal Local Trajectory Planning

Given the planned path, the local trajectory planner determines the resulting trajectory for autonomous navigation of the robot by online updating the local velocity control commands every control cycle. In the ROS dwa_local_planner [http://wiki.ros.org/dwa_local_planner], the sim_time is 1.7 seconds, controller_frequency is 20 Hz and the sim_granularity is 0.025 meters. The sim_time is the entire time of the DWA simulating trajectories within one dynamic window as T_{DW}, the controller_frequency is the frequency at which the dynamic window is called and the sim_granularity determines the step size between the points in the trajectories.

Since each term of the ROS DWA cost function is mathematically expressed in different units according to Eq. (20), the tuning parameters are on different scales. For the purpose of clear expression, the ratios α, β, γ and δ are used in the experimental studies to represent the weighting of path closeness, speed, obstacle avoidance and motional power optimization, respectively, by calculating the corresponding cost percentage.

$$\delta = \frac{e\_bias \cdot energy\_consumption}{COST} \tag{67}$$

Based on the default values given in the ROS dwa_local_planner, the robot was initially tuned by using $p\_bias = 32$, $g\_bias = 50$, $o\_bias = 0.01$ and $e\_bias = 0$. To demonstrate the effectiveness of the proposed energy-optimal omnidirectional velocity search technique in optimizing the power consumption for autonomous navigation, δ was varied from 0.00%, 20.00%, 50.00%, 66.66% to 80.00% by only increasing e_bias.

The total time for the robot to complete the planned paths is defined as the task duration. The total energy was calculated by the product of the time integral of the measured battery current and the battery voltage (48 volts). The idle power was calculated to be 72.7W due to the 1.51-1.52A observed idle current. The idle energy was calculated based on the idle power and the task duration. Then, the motional energy was calculated by subtracting the idle energy from the total energy. The motional power was calculated by dividing the motional energy by the task duration. Task duration, energy and power of the resulting trajectories are summarized in Table VII. It is shown that the increase in δ reduces both the total power and the motional power of the robot in all of the scenarios, which proves the effectiveness and significance of introducing the energy consumption into the cost function for optimizing the robotic power consumption. Reducing the idle power of the robotic electronics components was not within

the scope of this study. The motional energy consumptions do not always decrease with decreasing motional power as shown in Figure 77.

Table VII Experimental Results of Optimizing Power Consumption.

δ(%)	0.00%	20.00%	50.00%	66.66%	80.00%
FORWARD					
task duration(s)	27.9	26.8	29.9	32.4	49.2
total energy(J)	2209.9	2110.2	2323.2	2510.7	3745.6
idle energy(J)	2027.8	1947.8	2171.5	2358.9	3579.2
motional energy(J)	182.1	162.4	151.7	151.8	166.4
total power(W)	79.2	78.7	77.7	77.5	76.1
idle power(W)	72.7	72.7	72.7	72.7	72.7
motional power(W)	6.5	6.1	5.1	4.7	3.4
SIDEWAY					
task duration(s)	24.5	23.2	19.2	24.5	30.2
total energy(J)	2004.8	1860.7	1490.1	1879.8	2310.2
idle energy(J)	1783.1	1690.3	1394.7	1782.9	2194.1
motional energy(J)	221.7	170.4	95.4	96.9	116.1
total power(W)	81.8	80.2	77.6	76.7	76.5
idle power(W)	72.7	72.7	72.7	72.7	72.7
motional power(W)	9.0	7.3	5.0	4.0	3.8
DIAGONAL					
task duration(s)	28.2	28.6	30.2	31.6	40.9
total energy(J)	2271.2	2250.5	2380.3	2481.8	3179.2
idle energy(J)	2051.8	2082.5	2199.7	2300.2	2977.1
motional energy(J)	219.4	168.0	180.6	181.6	202.1
total power(W)	80.5	78.7	78.8	78.5	77.7
idle power(W)	72.7	72.7	72.7	72.7	72.7
motional power(W)	7.8	5.9	6.0	5.7	4.9
CURVE					
task duration(s)	45.8	46.1	49.7	57.4	70.7
total energy(J)	3813.4	3803.8	4084.8	4701.6	5676.0
idle energy(J)	3333.4	3349.7	3612.5	4171.7	5141.3
motional energy(J)	480.0	454.1	472.3	529.9	534.7
total power(W)	83.3	82.5	82.2	81.9	80.3
idle power(W)	72.7	72.7	72.7	72.7	72.7
motional power(W)	10.5	9.9	9.5	9.2	7.6
OBSTACLE					
task duration(s)	41.8	44.4	45.6	59.1	90.8
total energy(J)	3570.7	3740.2	3728.7	4631.0	6974.7
idle energy(J)	3042.5	3226.9	3318.1	4299.2	6602.5
motional energy(J)	528.2	513.3	410.6	331.8	372.2
total power(W)	85.4	84.2	81.8	78.4	76.8
idle power(W)	72.7	72.7	72.7	72.7	72.7
Motional power(W)	12.6	11.6	9.0	5.6	4.1

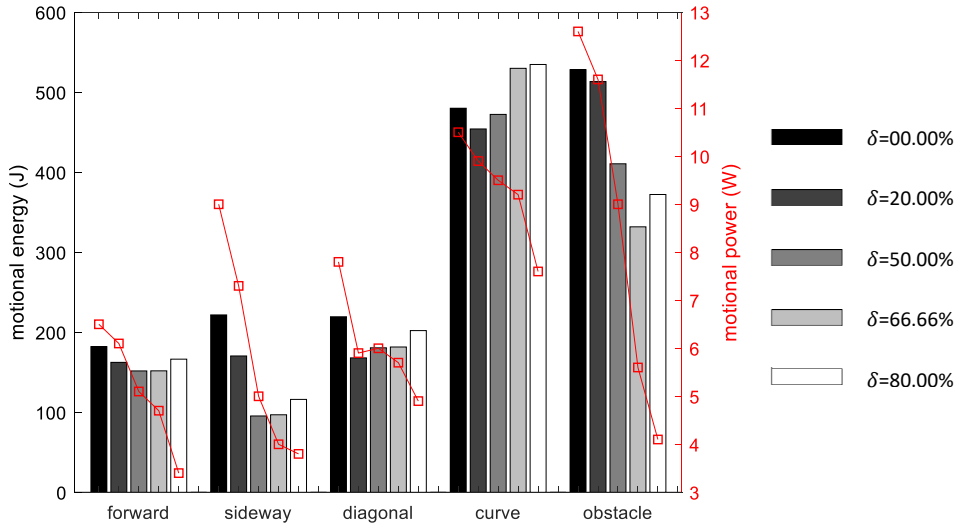


Figure 77 Motional energy and motional power in the experiments. The increase in δ reduces the motional power of the robot in almost all of the scenarios. This proves the effectiveness and significance of introducing the energy consumption into the cost function for optimizing the robotic power consumption. The motional energy consumptions did not always decrease with the decreasing motional power.

In the experiments, two laser range scanners were used to record the translational and rotational displacements of the robot and detect the unforeseen obstacles near or around the robot. Given the sideways straight-line path in Figure 71 (a), two resulting trajectories of the robot are depicted in Figure 78 (a) and Figure 78 (b) when $\delta = 20.00\%$ and $\delta = 80.00\%$, respectively, in the cost function. Both resulting trajectories show that the robot reached the goal pose within a reasonable goal tolerance. When $\delta = 80.00\%$, the robot performed a sideways motion with the heading φ angle remaining 90º.

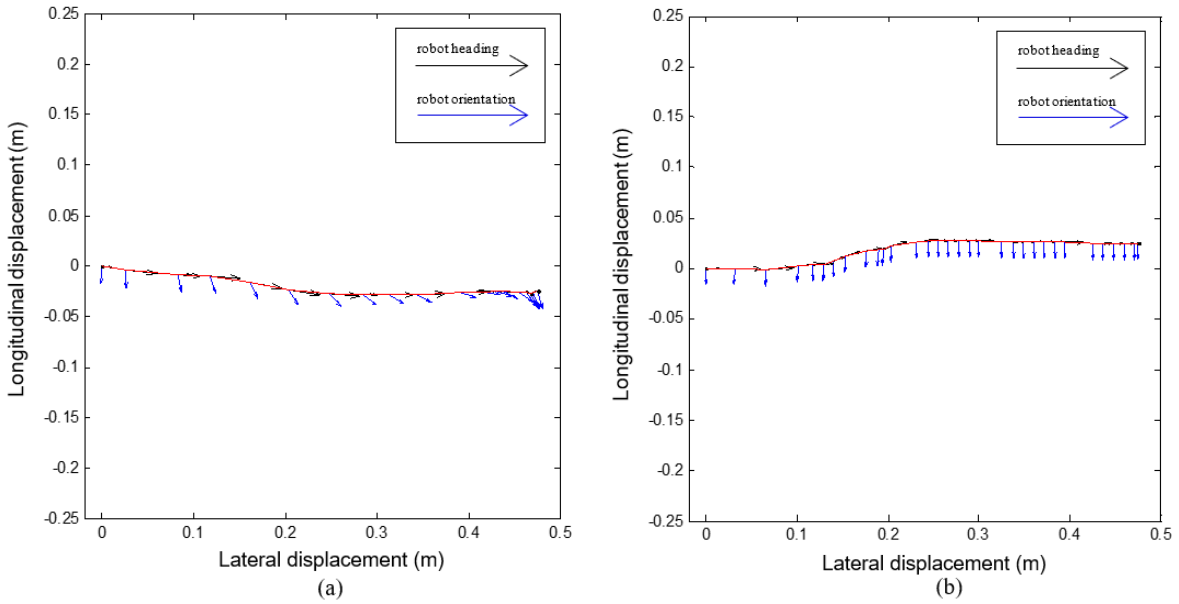


Figure 78 Trajectory of following a sideways straight-line path. (a) Energy weight ratio of 20.00%. (b) Energy weight ratio of 80.00%.

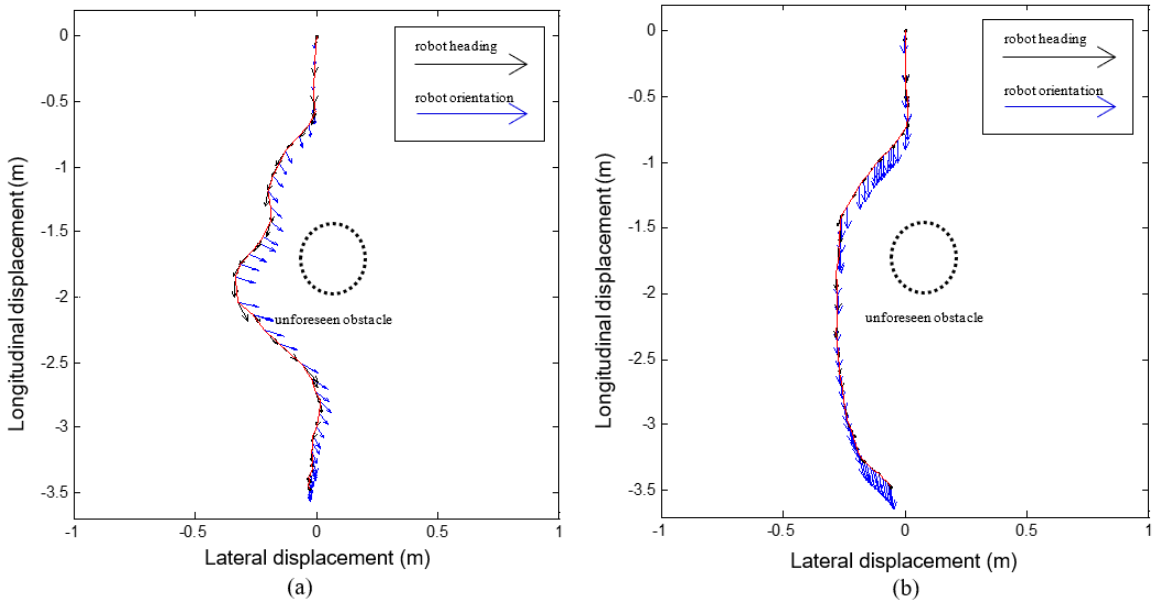


Figure 79 Trajectory of avoiding an unforeseen obstacle. (a) Energy weight ratio of 0.00%. (b) Energy weight ratio of 66.66%.

Given the reactive obstacle avoidance path in Figure 71 (c), two resulting trajectories are depicted in Figure 79 (a) and Figure 79 (b) when $\delta = 0.00\%$ and $\delta = 66.66\%$, respectively. The robot successfully avoided the unforeseen obstacle by deviating from the planned path and reached the goal pose. When $\delta = 66.66\%$, the robot performed a diagonal motion to avoid the obstacle. The original ROS dwa_local_planner tends to prefer the trajectories that make the robots face the local path goal or the obstacles. This is because some robots may only be able to observe the environment in front of them by their exteroceptive sensors. It is important to notice that both resulting trajectories with less motional power consumptions were involved with holonomic mobility of the robot such as the sideways motion and the diagonal motion. This is because the local trajectory planner concludes from the cost function that the omnidirectional motions with fewer changes in the orientation of the robot consume less motional power consumption. Thus, the proposed technique effectively takes advantage of the redundant maneuverability of holonomic mobility. In addition, the proposed technique is also able to fulfil online obstacle avoidance as the traditional DWA does.

5.5.3 Experimental Validation of Energy Reduction via the Combinational Cost Objectives

Experiments have been conducted to follow the given paths in Figure 71. Again, the ratios α, β, γ and δ are used in the experimental studies for the purpose of clear expression. To demonstrate the effectiveness of reducing the total energy via the proposed combinational cost objectives of both low motional power consumption and high speed, results are in Table VIII.

Table VIII Experimental Results of Reducing Energy Consumption

FORWARD			
weight ratio(%)*	0-100-0-0	0-12-0-88	0-60-0-40
task duration(s)	24.0	46.1	25.5
total energy(J)	1920.4	3499.2	2016.0
idle energy(J)	1743.7	3353.0	1854.7
motional energy(J)	176.7	146.2	161.3
total power(W)	80.0	75.9	79.1
idle power(W)	72.7	72.7	72.7
motional power(W)	7.4	3.2	6.3
SIDEWAY			
weight ratio(%)*	0-100-0-0	10-10-0-80	5-53-0-42
task duration(s)	22.7	30.2	18.3
total energy(J)	1861.9	2310.2	1468.8
idle energy(J)	1654.0	2194.1	1332.1
motional energy(J)	207.9	116.1	136.7
total power(W)	82.0	76.5	80.3
idle power(W)	72.7	72.7	72.7
motional power(W)	9.2	3.8	7.5
DIAGONAL			
weight ratio(%)*	0-100-0-0	10-10-0-80	5-53-0-42
task duration(s)	24.5	40.9	25.5
total energy(J)	1990.1	3179.2	2056.0
idle energy(J)	1781.1	2977.1	1857.9
motional energy(J)	209.0	202.1	198.1
total power(W)	81.2	77.7	80.6
idle power(W)	72.7	72.7	72.7
motional power(W)	8.5	4.9	7.8
CURVE			
weight ratio(%)*	17-49-17-17	6.6-6.6-6.6-80	5-45-5-45
task duration(s)	43.1	70.7	44.2
total energy(J)	3572.7	5676.0	3651.2
idle energy(J)	3132.2	5141.3	3213.9
motional energy(J)	440.5	534.7	437.3
total power(W)	82.9	80.3	82.6
idle power(W)	72.7	72.7	72.7
motional power(W)	10.2	7.6	9.9
OBSTACLE			
weight ratio(%)*	3-35-35-27	6.6-6.6-6.6-80	5-50-5-40
task duration(s)	44.9	90.8	46.2
total energy(J)	3831.5	6974.7	3935.6
idle energy(J)	3266.8	6602.5	3361.4
motional energy(J)	564.7	372.2	574.2
total power(W)	85.3	76.8	85.2
idle power(W)	72.7	72.7	72.7
motional power(W)	12.6	4.1	12.4

*weight ratio(%): α - β - γ - δ

The experimental results of varying the weight ratios of α, β, γ and δ are given in Table VIII, where it is found that the task duration is reduced due to the high speed of the motion, by utilizing a higher β. The smaller duration contributes to less idle energy consumption due to the constant idle power consumption. However, utilizing a high β does not lead to low motional energy due to the high motional power. The use of both higher β and δ reduces the total energy consumption, compared to the results in Table VII.

According to the experimental results, it is found that the idle power consumption of the robot is much higher than the motional power consumption. The idle energy consumption of the robot is also much higher than the motional energy. β makes an obvious contribution to reducing the idle energy consumption of the robot by reducing the task duration. Thus, β is significant, and a higher β can result in less total energy consumption for the experimental paths used in this study. However, it is worth mentioning that both high β and δ result in the trajectories that consume less motional energy to follow the straight-line and curved-line paths than single high β does. In particular, to follow the sideways straight-line path, high β and δ result in trajectories that consume less idle energy due to the holonomic mobility of the robot.

5.6. Summary

As a holonomic drive, the autonomous Mecanum robot can reach the same goal pose via various combinations of omnidirectional motion trajectories. Each feasible combination of trajectories consumes a different amount of energy from the robot. It is both academically interesting to achieve energy-efficient autonomous navigation via only a local planner, and economically valuable to reduce energy consumption of the Mecanum platforms by planning its omnidirectional motion trajectories. A local motion planning method that online optimizes the omnidirectional velocity profiles of the Mecanum robot for the purpose of improving energy efficiency still lacks research.

Aiming for power-minimization and energy-reduction autonomous navigation, the Dynamic Window Approach (DWA) local trajectory planning was extended by considering a new energy-related criterion, which was based on a novel energy consumption model of the four-wheeled Mecanum robot. The proposed energy consumption model from Chapter 4 was used to predict the robotic energy consumption in DWA. Based on the energy consumption model, a new energy-related criterion was utilized in the DWA cost function. A new extension of DWA was proposed to online generate the energy-optimal trajectory. Comprehensive

experiments have been performed to show the effectiveness of the proposed technique in optimizing the power consumption and reducing the energy consumption in various autonomous navigation task scenarios.

5.6.1 Results Discussion

The experimental results in Table VII show that the proposed extended DWA-based local trajectory planning method minimized the robotic power consumption in the global sense for autonomous navigation. It is also observed from Figure 78 and Figure 79 that the proposed technique took advantage of the redundant maneuverability of holonomic mobility to optimize the power consumption, and it could conduct online obstacle avoidance in the power-optimal way. Comparing the experimental results in Table VIII with the results in Table VII, the overall energy consumption in the global sense was reduced via the combination cost objectives of both low power consumption and high speed. But the overall energy consumption in the global sense was not optimized in this study.

Compared to the existing energy-efficient motion planning studies, the proposed trajectory planning is able to serve for any types of paths that the omnidirectional application mobile robot is able to move along. The proposed trajectory planning works for arbitrary initial and final velocities. Last but not the least; the proposed trajectory planning is an online approach to energy-optimal reactive behavior which can perform obstacle avoidance in an energy-optimal way. In order to online avoid unexpected obstacles for the autonomous mobile robot, the DWA-based trajectory planner continuously updates the local velocity commands for a mobile robot to execute for the next short period of the control cycle. This is because the reactive navigation system relies on the local planner that online generates a local trajectory to guide the robot to the target.

To avoid confusion, it is worth mentioning that the overall energy consumption in the global sense was not optimized in this study. It is difficult to realize absolutely energy-optimal autonomous navigation with only the effort of a local planning method without considering the global information. The accumulation of the energy optimization in each local planning does not necessarily achieve overall energy optimization in the global sense for autonomous navigation. Compared to my colleague's concurrent publication [83], which proposed a local planning method of considering both local and global energy consumption, my experiments did not utilize any global information in the local planner. Energy consumption was only optimized in each local time window. However, the DWA local trajectory planner works in

such a way that the optimal velocities of each control cycle are evaluated within each local time window that rolls forward, and the energy optimization in each constant-time local window achieves the power optimization in the local window and also the overall average power optimization in the global sense. With power optimization being a new cost objective, the overall energy consumption in the global sense was reduced via the combinational cost objectives of both low power consumption and high speed.

5.6.2 Conclusion

In this chapter, the proposed technique was based on the well-known Dynamic Window Approach, thanks to its proven flexible extension. Unlike most existing energy-optimal motion planning methods, the proposed technique is generally applicable to most mobile robots and copes with any shapes of paths. More importantly, it is compatible with online path planners and it is able to reactively avoid unforeseen obstacles. These advantages allow the DWA-based energy-optimal local trajectory planner to be conveniently fitted to autonomous navigation.

This study integrated a novel energy consumption model of the Mecanum platform into the DWA framework and introduced a new cost objective of minimizing power consumption to the DWA. Based on the energy consumption model of the Mecanum mobile robot, the omnidirectional velocities of the robot involving three degrees of freedom were then optimized in the extended DWA-based local trajectory planner by means of the proposed trajectory planning objective, minimization of power consumption, and other existing trajectory planning objectives, such as closeness to the planned path, speed and obstacle clearance. The difficulty of this contribution was to realize the minimization of the power consumption in the global sense for autonomous navigation via a local trajectory planning method. Then, based on the proposed extended DWA that could minimize power consumption, energy-reduction autonomous navigation was proposed via the combinational cost objectives of low power consumption and high speed.

For a given path, the proposed technique optimized the local velocity trajectory based on a cost function involving the energy consumption. The optimization of power consumption and the reduction of total energy consumption were made possible by taking advantage of the holonomic mobility of the Mecanum robot. In addition, the proposed technique was still able to fulfil online obstacle avoidance as the traditional DWA does. Comprehensive experiments were performed to show the effectiveness of the proposed technique in optimizing the power consumption and reducing the energy consumption in various autonomous navigation task

scenarios. The experiments also showed that the proposed method was still able to fulfil online obstacle avoidance of the traditional DWA.

Chapter 6. Energy-Optimal Offline Global Motion Planning of Omnidirectional Robots

6.1. Introduction

Industrial autonomous mobile robots are usually operated in static environments for repetitive tasks. A typical offline motion trajectory planning problem was formulated in this study for a Mecanum-wheeled industrial mobile service robot in a statically known environment. In this chapter, a globally energy-optimal motion trajectory planning problem for a holonomic omnidirectional Mecanum robot on flat ground was studied.

Given the path points including a stationary initial pose and a stationary goal pose with or without a sequence of via points, find the optimal task-space trajectories that minimize the energy consumption of the Mecanum mobile robot, under the constraints of continuous velocities and accelerations at the via points. In this study, a new application to the Mecanum mobile robot of the classic trajectory generation technique via polynomial functions was proposed to solve this problem. The polynomial function, which is commonly used in industry, was chosen for describing the omnidirectional trajectories that consist of time functions of position, velocity and acceleration for three degrees of freedom. A genetic algorithm was applied to optimize polynomial parameters and via points for the minimum-energy trajectories of the robot, based on a non-differentiable and highly nonlinear energy consumption model of the Mecanum robot from the energy consumption model [86] proposed in Chapter 4.

The simulations proved that the application of trajectory generation via polynomial functions found the energy-optimal polynomial trajectories for the Mecanum robot. It was observed that increasing the number or the order of the splines lowered the optimal energy consumption. The results have been validated via experimental testing.

6.2. Methodology – Energy-Optimal Polynomial Trajectory Generation

The Mecanum wheel trades off maneuverability against motion efficiency and its inefficient kinetic energy usage increases the energy consumption of the Mecanum robot. Some industrial mobile robots conduct repetitive transportation tasks in working areas for most of their operating time. To execute such tasks, an offline global motion planning method is needed to optimize the energy consumption of the mobile robot. In order to globally optimize the

omnidirectional motion trajectory of the Mecanum robot for the purpose of globally minimizing its energy consumption, a new application to the Mecanum robot of the classic polynomial-based trajectory generation technique is presented in this chapter. The proposed application of the technique employs polynomial functions to over generate the robotic trajectories and utilizes the genetic algorithm to find the energy-optimal trajectory.

6.2.1 Polynomial Trajectory Generation and the Genetic Algorithm

Trajectory generation based on the polynomial function has been very commonly applied to generate trajectories of mobile robots and manipulators in the industry. Paul and Zhang were the first to suggest using polynomials for robotic trajectory generation [84]. The genetic algorithm has been also applied to optimize the polynomial trajectory for fitness optimization. Shintaku used a polynomial to approximate the joint-space trajectories of an underwater manipulator and applied the genetic algorithm to search for the polynomial parameters for energy consumption optimization [85]. Tian and Collins planned the task-space trajectories of a robot manipulation using the Hermite cubic polynomial and then applied the genetic algorithm to determine the polynomial parameters for optimization purposes [86]. Yang et al. proposed a mobile robot path planning method using the Bezier curve and searching the time-optimal Bezier control points with the genetic algorithm [87].

6.2.2 Comparison with Existing Methods

The existing related energy-efficient motion planning methods have been summarized in Chapter 5. Compared to the existing works, this study has the following highlights. First, this study is concerned with the Mecanum wheel and it solves a pragmatic issue for industrial Mecanum robots. As one of the practical omnidirectional wheel designs in the industry, the energy consumption of the Mecanum robot is a pragmatic issue due to its energy inefficiency drawback and heavy-duty character. This new application of the classic method of polynomial-based trajectory generation to the Mecanum robot for the purpose of energy optimization has never been done before. Second, this study minimizes the energy consumption of holonomic robots by optimally planning the robotic motion trajectories. The research on energy-optimal motion planning for the holonomic Mecanum robot has not been fully studied. Many existing energy-optimal motion planning methods are restricted with nonholonomic constraints. The omnidirectional trajectories of holonomic drives on flat ground have three degrees of freedom. However, the feasible trajectories of nonholonomic drives have fewer degrees of freedom, e.g.

the orientations of the differential car-like drive tend to be fixed in the tangential direction of the path. Third, the proposed technique in this study optimizes path planning and motion trajectory planning simultaneously. Typically, the trajectory planning technique interpolates the polynomials between the given via points, which are separately generated from a high-level path planner. Some existing studies separate path optimization and motion trajectory optimization. The energy-minimization path planning proposed in [49, 53] did not consider robotic motion. The separation of path and motion trajectory optimization cannot guarantee overall optimization. In this study, the locations of via points and the polynomial parameters are simultaneously optimized.

6.3. Energy-Optimal Global Motion Planning

The proposed method first describes the task-space omnidirectional trajectories of the holonomic drive using polynomial functions. Based on the previously proposed energy consumption model of the Mecanum robot, a genetic algorithm is applied to optimize the decision variables including polynomial parameters and via points, for the globally energy-optimal trajectories.

6.3.1 Trajectory Generation by Polynomial Function

The holonomic Mecanum mobile robot that moves on a 2D plane ground, has three degrees of freedom. The robot is able to move in the X and Y translational directions and to spin in place in the $\emptyset$ rotational direction, independently from each other. It is enough to have position trajectory, velocity trajectory and acceleration trajectory to evaluate the robotic energy consumption according to the proposed energy consumption model from Chapter 4. Thus, the omnidirectional trajectories of the holonomic Mecanum robot consist of the nine time functions of position, velocity and acceleration for three degrees of freedom including translation X, translation Y and rotation $\emptyset$.

The polynomial spline function is chosen to describe the task-space trajectories. The first reason is that the polynomial function is very commonly applied in the industry. The second reason is because the polynomial function can be easily differentiated. The polynomial spline function that is used to describe the robotic position trajectory can be used to find velocity and acceleration trajectories by easy differentiations. According to the formulated problem in the introduction section, the path points including an initial pose $(X_0^*, Y_0^*, \emptyset_0^*)$ and a goal pose $(X_n^*, Y_n^*, \emptyset_n^*)$ with or without a sequence of via points $(X_1^*, Y_1^*, \emptyset_1^*), \dots, (X_{n-1}^*, Y_{n-1}^*, \emptyset_{n-1}^*)$ in the

task space are given. Depending on the task requirements, the via points can be optional and the number of the via points can vary. There could be no via point at all so that both path and robotic motion are determined by the polynomial trajectory generation. There could be a large number of consecutive via points, which are found by a high-level path planner. This can be regarded as spline interpolation. Or there could be some intermediate situations that several via points exist between the initial pose and the goal pose for the service robot to compulsorily pass by. Sometimes in the industrial applications, the robot must reach these path points at time $t_0, t_1, \dots, t_n$ respectively, under strict task timing constraints. Time $t_n$is also named as task duration.

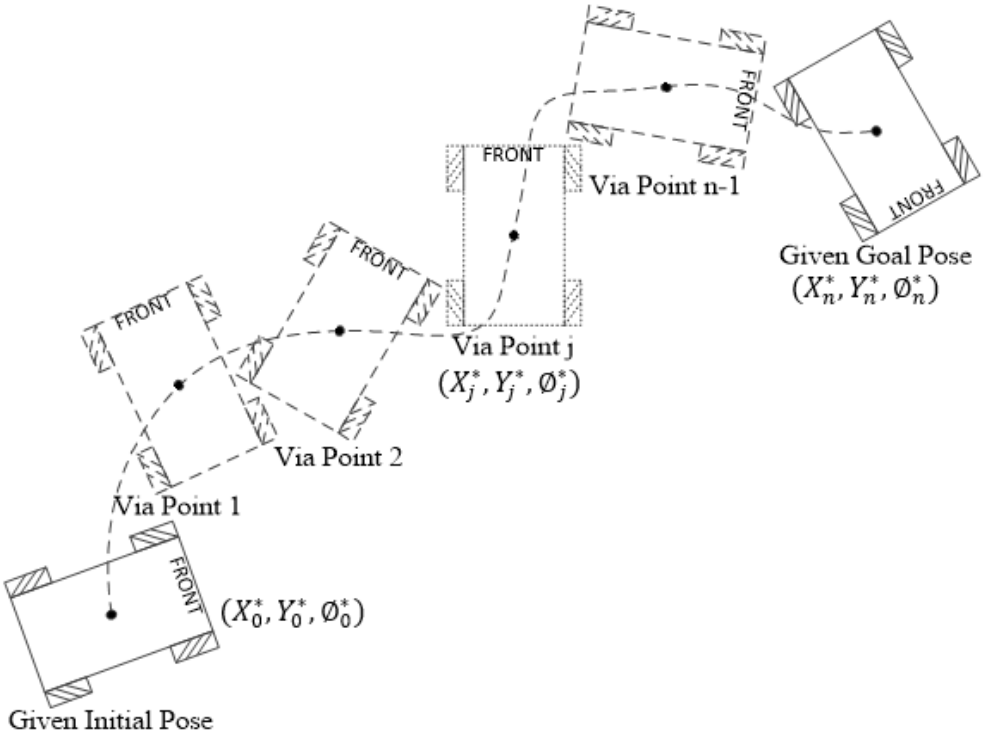


Figure 80 Path points in task space including initial pose, optional via points and goal pose.

An illustration diagram to present the formulated problem in this chapter is shown in Figure 80. All the path points including the optimal via points have been marked with the Mecanum robot icons, whose orientations are also directed. Here, $(X_j^*, Y_j^*, \emptyset_j^*)$ is the j-th via point for $j = 1,2,3, \dots, n-1$ in Figure 80. The desired pose matrices are defined in the equations below.

$$\boldsymbol{X} = \left[X_0^* \; X_1^* \; X_2^* \; \dots \; X_j^* \; \dots \; X_n^*\right]^T \tag{68}$$

$$\boldsymbol{Y} = \left[Y_0^* \; Y_1^* \; Y_2^* \; \dots \; Y_j^* \; \dots \; Y_n^*\right]^T \tag{69}$$

$$\boldsymbol{\Phi} = \left[\emptyset_0^* \; \emptyset_1^* \; \emptyset_2^* \; \dots \; \emptyset_j^* \; \dots \; \emptyset_n^*\right]^T \tag{70}$$

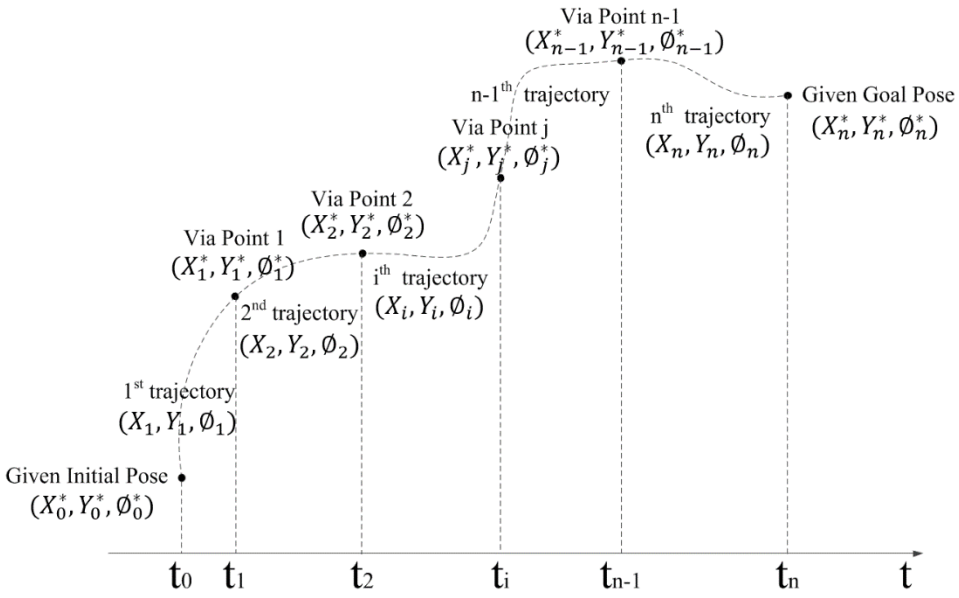


Figure 81 The position trajectories passing through the path points in time scale.

According to Figure 81, there are $n+1$ path points, so there should be n segments of polynomials, each of which links one path point to its consecutively next path point. Each path point has three degrees of freedom so three different types of time functions are employed to describe the X, Y and $\emptyset$ position trajectories. The order of the polynomials is set as k. Thus, in order to pass through a sequence of $n+1$ path points, one set of three time functions of n segments of k-order splines is employed to describe the X, Y and $\emptyset$ position trajectories.

$$\begin{aligned} X_i(t) = \; & a_{i0} + a_{i1}(t - t_{i-1}) + a_{i2}(t - t_{i-1})^2 \\ & + \cdots + a_{ik}(t - t_{i-1})^k \end{aligned} \tag{71}$$

$$\begin{aligned} Y_i(t) = \; & b_{i0} + b_{i1}(t - t_{i-1}) + b_{i2}(t - t_{i-1})^2 \\ & + \cdots + b_{ik}(t - t_{i-1})^k \end{aligned} \tag{72}$$

$$\begin{aligned} \emptyset_i(t) = \; & c_{i0} + c_{i1}(t - t_{i-1}) + c_{i2}(t - t_{i-1})^2 \\ & + \cdots + c_{ik}(t - t_{i-1})^k \end{aligned} \tag{73}$$

i determines the index of the piecewise trajectory between the path points and $i = 1,2,\dots,n$. The parameters of the splines $a_{i0}, a_{i1}, \dots, a_{ik}, b_{i0}, b_{i1}, \dots, b_{ik}, c_{i0}, c_{i1}, \dots, c_{ik}$ are to be decided under the problem constrains and optimization goals. $t \in [t_{i-1}, t_i]$ is the time since the task started. $X(t)$ is the final time function of the position trajectories that contain all $X_i(t)$. Then $\dot{X}(t)$ and $\ddot{X}(t)$ are denoted as the first and the second derivatives of $X(t)$. By analogy $Y(t)$,

$\dot{Y}(t)$, $\ddot{Y}(t)$, $\emptyset(t)$, $\dot{\emptyset}(t)$, $\ddot{\emptyset}(t)$ are defined. The rest of this chapter is only presented in terms of translation X. The trajectory in translation Y and rotation $\emptyset$ can be deduced by analogy.

To generate the feasible trajectories that satisfy the requirements in the formulated motion planning problem, the following basic conditions should be imposed on the splines:

Basic Condition A:

The trajectories must pass through the given path points at the corresponding time

$$X_i(t_{i-1}) = X_{i-1}^*, x_i(t_i) = X_i^* \tag{74}$$

Basic Condition B:

The trajectories are velocity continuous at via points

$$\dot{X}_i(t_i) = \dot{X}_{i+1}(t_i) \tag{75}$$

Basic Condition C:

The trajectories are acceleration continuous at via points

$$\ddot{X}_i(t_i) = \ddot{X}_{i+1}(t_i) \tag{76}$$

The above conditions can be mathematically expressed by the following spline parameters:

$$a_{i0} = X_{i-1}^* \tag{77}$$

$$a_{i0} + a_{i1}\Delta t_i + a_{i2}\Delta t_i^2 + \cdots + a_{ik}\Delta t_i^k = X_i^* \tag{78}$$

$$a_{i1} + 2a_{i2}\Delta t_i + \cdots + k a_{ik}\Delta t_i^{k-1} = a_{(i+1)1} \tag{79}$$

$$2a_{i2} + 6a_{i3}\Delta t_i + \cdots + k(k-1)a_{ik}\Delta t_i^{k-2} = 2a_{(i+1)2} \tag{80}$$

Here, $\Delta t_i = t_i - t_{i-1}$ are the spline time variables. The spline time variable matrix $\boldsymbol{\Delta t}$ is defined as $[\Delta t_1\ \Delta t_2\ \ldots\ \Delta t_n]^T$. The spline parameter matrix $\boldsymbol{a_0}$ is defined as $[a_{10}\ a_{20}\ \ldots\ a_{n0}]^T$ and $\boldsymbol{a_1}$, $\boldsymbol{a_2}$, …, $\boldsymbol{a_k}$ are defined in a similar way. A partial set of the spline parameters is uniquely determined by the above conditions.

Basic Condition A:

$$\boldsymbol{a_0} = \boldsymbol{X} \tag{81}$$

Basic Conditions A and B:

$$a_{i2} = \frac{3(X_i^* - X_{i-1}^*)}{\Delta t_i^2} - \frac{2a_{i1} + a_{(i+1)1}}{\Delta t_i} + a_{i4}\Delta t_i^2 + 2a_{i5}\Delta t_i^3 + \cdots + (k-3)a_{ik}\Delta t_i^{k-2} \quad (82)$$

$$a_{i3} = \frac{2(X_{i-1}^* - X_i^*)}{\Delta t_i^3} + \frac{a_{i1} + a_{(i+1)1}}{\Delta t_i^2} - 2a_{i4}\Delta t_i - 3a_{i5}\Delta t_i^2 - \cdots - (k-2)a_{ik}\Delta t_i^{k-3} \quad (83)$$

Basic Condition C:

$$\frac{2a_{i1} + 4a_{(i+1)1}}{\Delta t_i} + \frac{4a_{(i+1)1} + 2a_{(i+2)1}}{\Delta t_{i+1}} = \frac{6(X_i^* - X_{i-1}^*)}{\Delta t_i^2} + \frac{6(X_{i+1}^* - X_i^*)}{\Delta t_{i+1}^2} - 2a_{i4}\Delta t_i^2 - 6a_{i5}\Delta t_i^3 - \cdots - (k^2 - 5k + 6)a_{ik}\Delta t_i^{k-2} + 2a_{(i+1)4}\Delta t_{i+1}^2 + 4a_{(i+1)5}\Delta t_{i+1}^3 + \cdots + 2(k-3)a_{(i+1)k}\Delta t_{i+1}^{k-2} \quad (84)$$

To satisfy the *Basic Conditions (A-C)*, $\boldsymbol{a_0}$, $\boldsymbol{a_1}$, $\boldsymbol{a_2}$ and $\boldsymbol{a_3}$ must be uniquely determined. The rest of the undetermined spline parameters $\boldsymbol{a_4}$, $\boldsymbol{a_5}$, …, $\boldsymbol{a_k}$ are treated as variables in this problem. Thus, the omnidirectional trajectories of the holonomic mobile robots have been described by using the polynomial spline functions. To satisfy the given conditions, the spline functions require a minimum order of three.

6.3.2 Parameter Optimization

After using the polynomial splines to describe the omnidirectional trajectories, an optimization searching process finds the energy-optimal trajectories for the Mecanum robot. The decision variables for this optimization problem consist of the spline time variables $\Delta\boldsymbol{t}$, undetermined spline parameters $\boldsymbol{a_4}$, $\boldsymbol{b_4}$, $\boldsymbol{c_4}$, $\boldsymbol{a_5}$, $\boldsymbol{b_5}$, $\boldsymbol{c_5}$, …, $\boldsymbol{a_k}$, $\boldsymbol{b_k}$, $\boldsymbol{c_k}$ and via points $(X_1^*, Y_1^*, \emptyset_1^*), \ldots, (X_{n-1}^*, Y_{n-1}^*, \emptyset_{n-1}^*)$. This also depends on the task specifications. The task is possible to put a timing constrain on the robotic trajectory planning so that $\Delta\boldsymbol{t}$ is no longer the decision variable. In addition, via points can be set free as decision variables if the task allows. These via points are named as relaxed via points in this study. Similar situations may apply to the undetermined spline parameters, for example jerk requirements.

Here, a general case is demonstrated. The total number of decision variables N_{dv} depends on the number and the order of the splines, and the number and the values of the via points. The number of the relaxed via points is denoted as N_{rvp}.

$$N_{dv} = n + 3 * n * (k - 3) + 3 * N_{rvp} \tag{85}$$

Each piecewise trajectory has one spline time variable acting as the decision variable, so the total number of the spline time variables is the same as the number of the piecewise trajectories n. The undetermined spline parameters come from the fourth-order and higher-order terms. So, the number of the undetermined spline parameters of each piecewise trajectory for each degree of freedom is $k - 3$. Every relaxed via point that has three degrees of freedom so that the values of the via point act as three decision variables.

Every combination of decision variables represents the unique candidate omnidirectional trajectories that satisfy the requirements in the formulated problem. Among all the candidate trajectories, the minimum-energy trajectories represented by an optimal combination of decision variables are to be found by optimization. The objective function is based on the energy consumption model of the robot proposed in Chapter 4. The model takes in the time history of positions, velocities and accelerations for all three degrees of freedom and estimates the robotic energy consumption.

6.4. Implementation

6.4.1 Spline Parameter Computing Matrix

Using simple variable elimination finds the equation for the determined parameters. These uniquely determined parameters can be easily computed in the following matrix equations.

$$\boldsymbol{H_1 a_1 = H_2 X + H_{3a} + H_{4a}} \tag{86}$$

$$\boldsymbol{a_2 = H_5 X + H_6 a_1 + H_{7a}} \tag{87}$$

$$\boldsymbol{a_3 = H_8 X + H_9 a_1 + H_{10a}} \tag{88}$$

The details of these matrix $\boldsymbol{H_1}$, $\boldsymbol{H_2}$, $\boldsymbol{H_{3a}}$, $\boldsymbol{H_{3b}}$, $\boldsymbol{H_{3c}}$, $\boldsymbol{H_{4a}}$, $\boldsymbol{H_{4b}}$, $\boldsymbol{H_{4c}}$, $\boldsymbol{H_5}$, $\boldsymbol{H_6}$, $\boldsymbol{H_{7a}}$, $\boldsymbol{H_{7b}}$, $\boldsymbol{H_{7c}}$, $\boldsymbol{H_8}$, $\boldsymbol{H_9}$, $\boldsymbol{H_{10a}}$, $\boldsymbol{H_{10b}}$ and $\boldsymbol{H_{10c}}$ are shown below.

$$\boldsymbol{H_1} = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \frac{2}{\Delta t_1} & \frac{4}{\Delta t_1}+\frac{4}{\Delta t_2} & \frac{2}{\Delta t_2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{2}{\Delta t_2} & \frac{4}{\Delta t_2}+\frac{4}{\Delta t_3} & \frac{2}{\Delta t_3} & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{2}{\Delta t_3} & \frac{4}{\Delta t_3}+\frac{4}{\Delta t_4} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{2}{\Delta t_{n-2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & \frac{4}{\Delta t_{n-2}}+\frac{4}{\Delta t_{n-1}} & \frac{2}{\Delta t_{n-1}} & 0 \\ 0 & 0 & 0 & 0 & \cdots & \frac{2}{\Delta t_{n-1}} & \frac{4}{\Delta t_{n-1}}+\frac{4}{\Delta t_n} & \frac{2}{\Delta t_n} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \quad (89)$$

$$\boldsymbol{H_2} = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \frac{-6}{\Delta t_1^2} & \frac{6}{\Delta t_1^2}+\frac{-6}{\Delta t_2^2} & \frac{6}{\Delta t_2^2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{-6}{\Delta t_2^2} & \frac{6}{\Delta t_2^2}+\frac{-6}{\Delta t_3^2} & \frac{6}{\Delta t_3^2} & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{-6}{\Delta t_3^2} & \frac{6}{\Delta t_3^2}+\frac{-6}{\Delta t_4^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{6}{\Delta t_{n-2}^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & \frac{6}{\Delta t_{n-2}^2}+\frac{-6}{\Delta t_{n-1}^2} & \frac{6}{\Delta t_{n-1}^2} & 0 \\ 0 & 0 & 0 & 0 & \cdots & \frac{-6}{\Delta t_{n-1}^2} & \frac{6}{\Delta t_{n-1}^2}+\frac{-6}{\Delta t_n^2} & \frac{6}{\Delta t_n^2} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \quad (90)$$

$$\boldsymbol{H_{3a}} = \begin{bmatrix} 0 \\ -2a_{14}\Delta t_1^2 - 6a_{15}\Delta t_1^3 - \cdots - (k^2-5k+6)a_{1k}\Delta t_1^{k-2} \\ -2a_{24}\Delta t_2^2 - 6a_{25}\Delta t_2^3 - \cdots - (k^2-5k+6)a_{2k}\Delta t_2^{k-2} \\ \vdots \\ -2a_{n4}\Delta t_n^2 - 6a_{n5}\Delta t_n^3 - \cdots - (k^2-5k+6)a_{nk}\Delta t_n^{k-2} \end{bmatrix} \quad (91)$$

$$\boldsymbol{H_{4a}} = \begin{bmatrix} 0 \\ 2a_{24}\Delta t_2^2 + 4a_{25}\Delta t_2^3 + \cdots + 2(k-3)a_{2k}\Delta t_2^{k-2} \\ 2a_{34}\Delta t_3^2 + 4a_{35}\Delta t_3^3 + \cdots + 2(k-3)a_{3k}\Delta t_3^{k-2} \\ \vdots \\ 2a_{n4}\Delta t_n^2 + 4a_{n5}\Delta t_n^3 + \cdots + 2(k-3)a_{nk}\Delta t_n^{k-2} \\ 0 \end{bmatrix} \quad (92)$$

$$\boldsymbol{H_5} = \begin{bmatrix} \frac{-3}{\Delta t_1^2} & \frac{3}{\Delta t_1^2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{-3}{\Delta t_2^2} & \frac{3}{\Delta t_2^2} & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{-3}{\Delta t_3^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{3}{\Delta t_{n-1}^2} & \frac{3}{\Delta t_{n-1}^2} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \frac{-3}{\Delta t_n^2} & \frac{3}{\Delta t_n^2} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \tag{93}$$

$$\boldsymbol{H_6} = \begin{bmatrix} \frac{-2}{\Delta t_1} & \frac{-1}{\Delta t_1} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{-2}{\Delta t_2} & \frac{-1}{\Delta t_2} & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{-2}{\Delta t_3} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{-2}{\Delta t_{n-1}} & \frac{-1}{\Delta t_{n-1}} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \frac{-2}{\Delta t_n} & \frac{-1}{\Delta t_n} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \tag{94}$$

$$\boldsymbol{H_{7a}} = \begin{bmatrix} a_{14}\Delta t_1^2 + 2a_{15}\Delta t_1^3 + \cdots + (k-3)a_{1k}\Delta t_1^{k-2} \\ a_{24}\Delta t_2^2 + 2a_{25}\Delta t_2^3 + \cdots + (k-3)a_{2k}\Delta t_2^{k-2} \\ \vdots \\ a_{n4}\Delta t_n^2 + 2a_{n5}\Delta t_n^3 + \cdots + (k-3)a_{nk}\Delta t_n^{k-2} \\ 0 \end{bmatrix} \tag{95}$$

$$\boldsymbol{H_8} = \begin{bmatrix} \frac{2}{\Delta t_1^3} & \frac{-2}{\Delta t_1^3} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{2}{\Delta t_2^3} & \frac{-2}{\Delta t_2^3} & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{2}{\Delta t_3^3} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{2}{\Delta t_{n-1}^3} & \frac{-2}{\Delta t_{n-1}^3} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \frac{2}{\Delta t_n^3} & \frac{-2}{\Delta t_n^3} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \tag{96}$$

$$\boldsymbol{H_9} = \begin{bmatrix} \frac{1}{\Delta t_1^2} & \frac{1}{\Delta t_1^2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{1}{\Delta t_2^2} & \frac{1}{\Delta t_2^2} & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\Delta t_3^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{\Delta t_{n-1}^2} & \frac{1}{\Delta t_{n-1}^2} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{\Delta t_n^2} & \frac{1}{\Delta t_n^2} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \tag{97}$$

$$\boldsymbol{H_{10a}} = \begin{bmatrix} -2a_{14}\Delta t_1 - 3a_{15}\Delta t_1^2 - \cdots - (k-2)a_{1k}\Delta t_1^{k-3} \\ -2a_{24}\Delta t_2 - 3a_{25}\Delta t_2^2 - \cdots - (k-2)a_{2k}\Delta t_2^{k-3} \\ \vdots \\ -2a_{n4}\Delta t_n - 3a_{n5}\Delta t_n^2 - \cdots - (k-2)a_{nk}\Delta t_n^{k-3} \\ 0 \end{bmatrix} \tag{98}$$

And $\boldsymbol{H_{3b}}$, $\boldsymbol{H_{3c}}$, $\boldsymbol{H_{4b}}$, $\boldsymbol{H_{4c}}$, $\boldsymbol{H_{7b}}$, $\boldsymbol{H_{7c}}$, $\boldsymbol{H_{10b}}$ and $\boldsymbol{H_{10c}}$ are defined in a similar way.

6.4.2 Optimization Searching Algorithm

This optimization problem is based on the evaluation of the energy consumption model, to find the least-cost combination of the decision variables. Due to the energy consumption model, the problem is non-differentiable and highly nonlinear. A genetic algorithm is applied to optimize decision variables for minimum energy consumption. A genetic algorithm solver in the MATLAB Optimization Toolbox is used to find the energy-optimal combination of decision variables, which results in minimum-energy trajectories. In this study, the default MATLAB genetic algorithm solver options are used.

6.5. Verification

To investigate the application of the classic trajectory generation technique via polynomial functions to the Mecanum robot for the purpose of finding its energy-optimal trajectories, three sets of simulations were performed. The first set of three simulations aimed to investigate the influence of increasing the polynomial spline number between the given path points on improving the resultant energy optimization. The second set of two simulations aimed to investigate the influence of increasing the spline orders on improving the resultant energy optimization. The third set of five simulations aimed to prove that the minimization of the

robotic energy consumption was achieved by considering only the energy consumption as the objective function, compared with considering both the task duration and the energy consumption as the objective function.

To implement the polynomial-based trajectory generation on the Mecanum robot, the found energy-optimal velocity trajectories from simulations were experimentally executed on the purpose-built robot. The optimal combination of the decision variables was found offline by an optimization searching algorithm and tested in the simulation. Then the polynomial functions that were parameterized by the optimal decision variables, were programmed inside the robotic control system and generated the robotic velocity commands to drive the Mecanum robot. The designed closed-loop motion control system that relied on the online robotic velocity feedback from the laser scanner and the PI controller, guaranteed the Mecanum robot to track the polynomial motion trajectories. The experiment served two purposes. First, it was aimed to experimentally validate the energy consumption model by operating the Mecanum robot to execute more sophisticated velocity trajectories. The experiments compared the current consumption and the energy consumption simulated by the model with the actual consumptions from the robot in the experiments. Second, by following the resultant energy-optimal trajectories that were found by the polynomial-based trajectory generation, the robot consumed the minimum energy consumption as simulated.

In both the simulations and experiments, the maximum robotic translational and rotational velocities were 0.1 m/s and 0.1 rad/s, respectively, and the maximum robotic translational and rotational accelerations were 0.5 m/s^2 and 0.5 rad/s^2, respectively.

6.5.1 Simulations

Single, multiple cubic polynomials and relaxed via points

In the first set of simulations, the initial pose and the goal pose were given as $(x_0^*, y_0^*, \varphi_0^*) = (0,0,-0.5\pi)$ and $(x_n^*, y_n^*, \varphi_n^*) = (2,2,-0.2\pi)$, respectively, as shown in Figure 82. Because there was no via point between the initial pose and the goal pose, there should be $n = 1$ segments of position polynomials. A set of three functions of one segment of third-order splines was able to describe the trajectories. Three functions served for three degrees of freedom including translation X, translation Y and rotation $\emptyset$. One single segment of the trajectories was enough because no via point was given and a spline order of three was the minimum requirement of passing through two poses and satisfying the stationary initial and goal states.

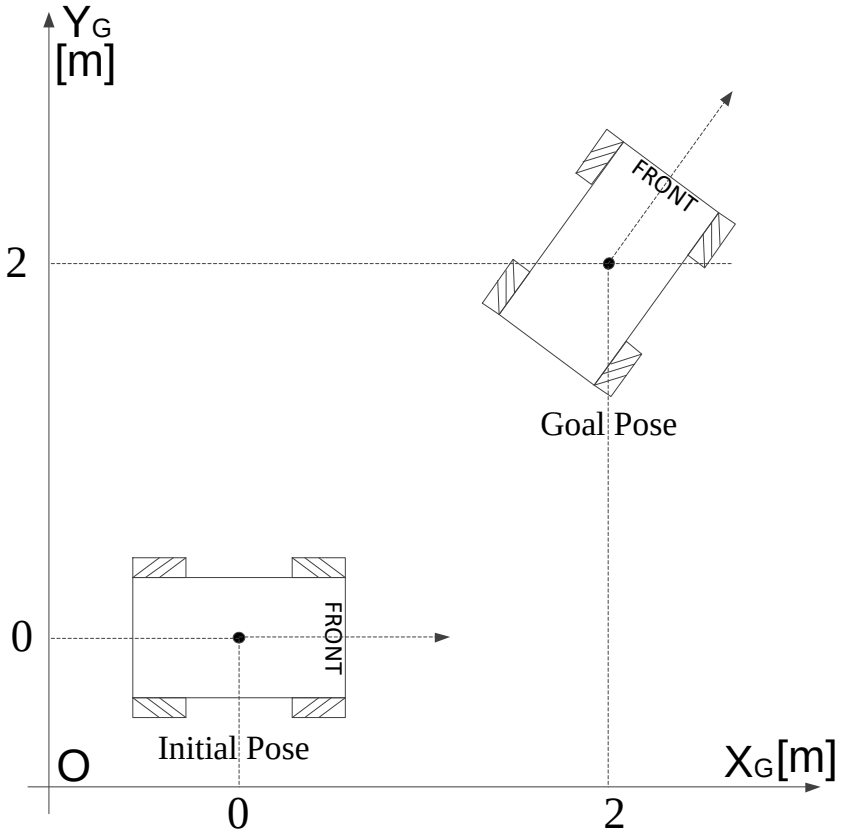


Figure 82 The initial pose and the goal pose in the first set of simulations.

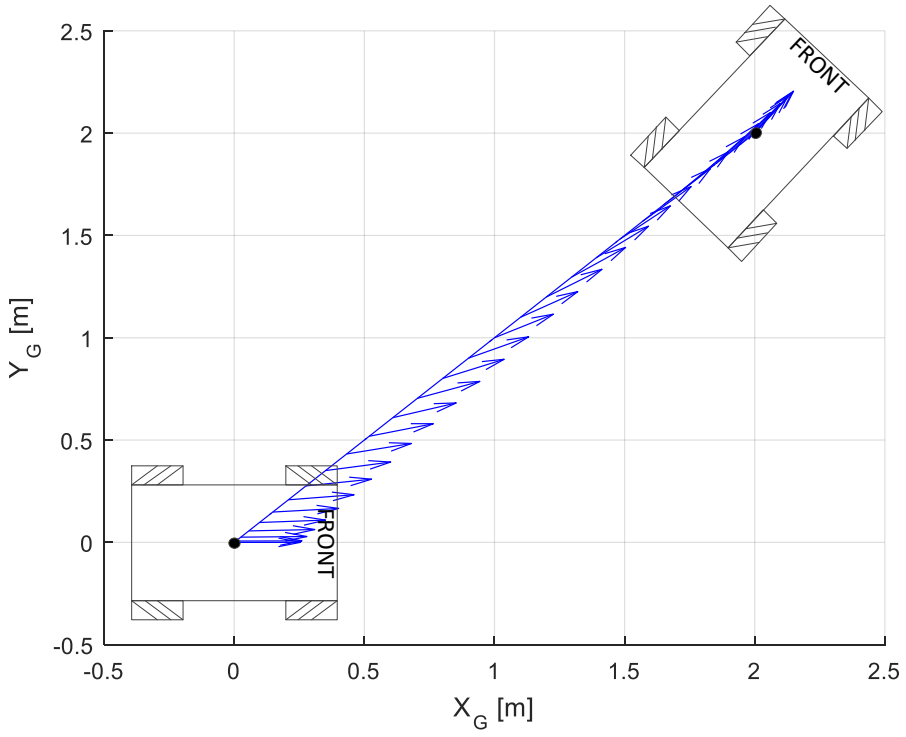
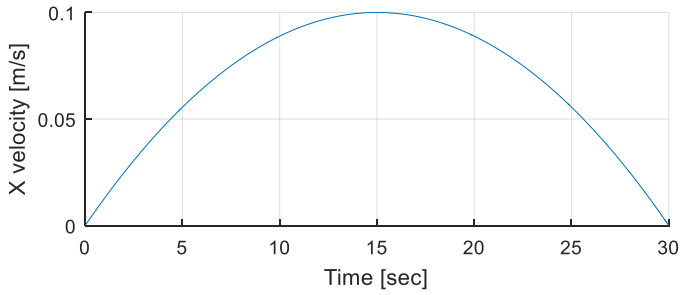
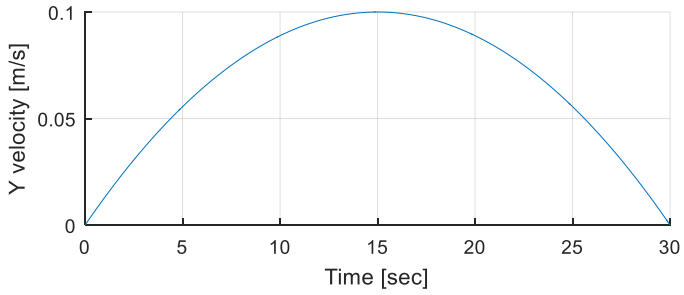


Figure 83 The energy-optimal trajectories in Simulation 1.

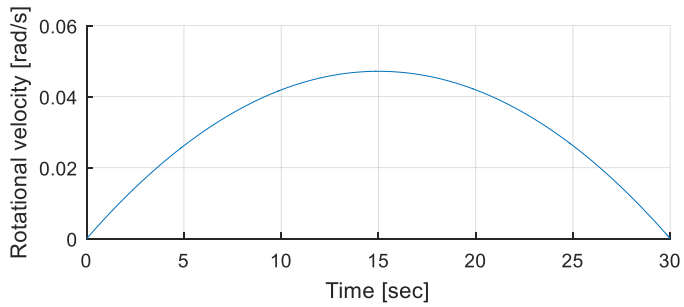
The polynomial motion trajectory in Simulation 1 was described by a set of three functions of one piece of third-order splines. All the spline parameters of the third-order splines were uniquely determined by the stationary initial pose and the stationary goal pose. Only one spline time variable that was shared by all three degrees of freedom was the decision variable for energy optimization in Simulation 1. The resultant optimal decision variable was $\Delta t = 30.1\ s$ and the resultant optimal energy consumption was 2547.7 J. The resultant energy-optimal trajectories are shown in Figure 83. In the following simulations, the paths were drawn in blue lines and the robotic orientations were represented using blue arrows. The resultant velocity trajectories of Simulation 1 are plotted in Figure 84.



(a)



(b)



(c)

Figure 84 The velocity trajectories in Simulation 1. (a) X translational velocity. (b) Y translational velocity. (c) Z rotational velocity

In order to investigate the influence of increasing the polynomial spline number on the resultant energy optimization, a set of three functions of three segments of third-order splines was utilized for comparison purposes. Simulation 2 purposely added two fixed via points with equal distances for translation X, translation Y and rotation $\emptyset$ and employed a set of three functions of three segments of third-order splines. The initial and final poses of Simulation 2 were the same as those for Simulation 1. Two additional via points were given as $(x_1^*, y_1^*, \varphi_1^*) = (0.67, 0.67, -0.4\pi)$ and$(x_2^*, y_2^*, \varphi_2^*) = (1.33, 1.33, -0.3\pi)$ in Simulation 2 and now $n = 3$. As third-order splines were utilized, all the spline parameters were again uniquely determined. The number of the decision variables increased to three spline time variables due to three piecewise trajectories. The resultant optimal decision variables were found as $\Delta \boldsymbol{t} = [9.1\ 8.3\ 9.1]\ s$ and the resultant optimal energy consumption was reduced by 10.8% to 2272.7 J. The task duration taken for Simulation 2 was 26.5 s less than 30.1 s taken for Simulation 1. The resultant energy-optimal trajectories of Simulation 2 were shown in Figure 85. The resultant velocity trajectories of Simulation 2 are plotted in Figure 86.

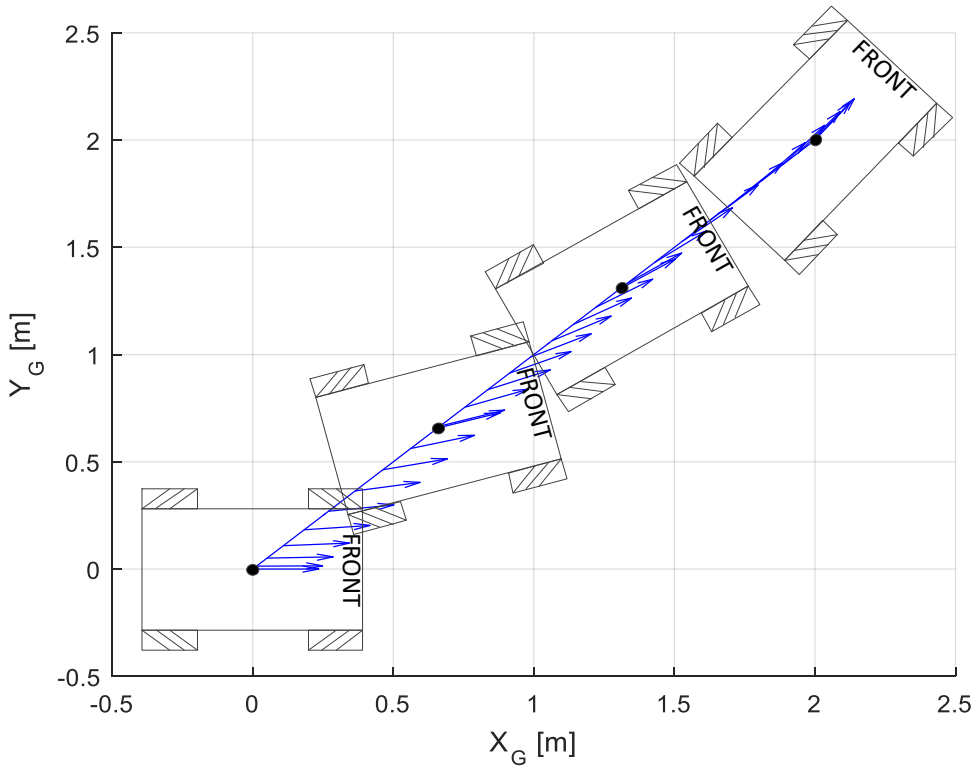
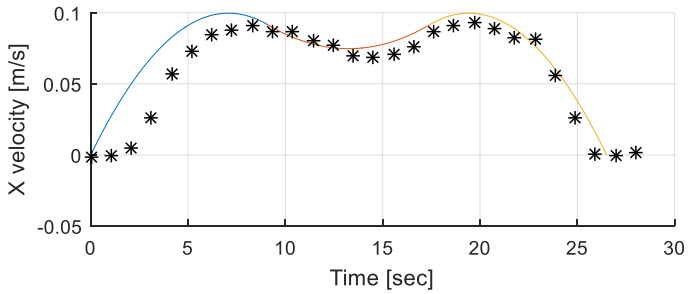
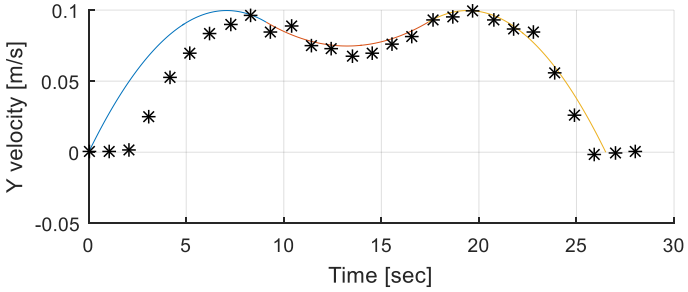


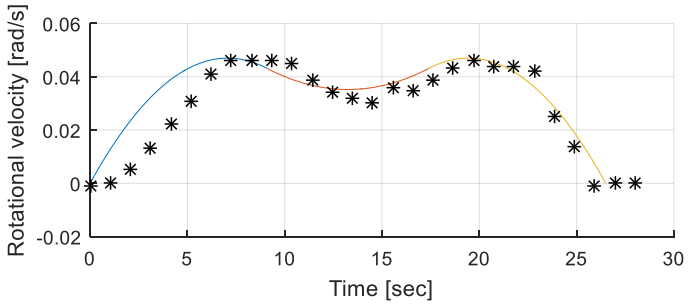
Figure 85 The energy-optimal trajectories in Simulation 2.



(a)



(b)



(c)

Figure 86 The velocity trajectories in Simulation 2 and Experiment 2. (a) X translational velocity. (b) Y translational velocity. (c) Z rotational velocity

Based on Simulation 2, Simulation 3 relaxed the added via points in Simulation 2 as the additional decision variables so that the locations of the added via points and the polynomial parameters are simultaneously optimized. The number of the decision variables increased from three to nine by adding $(x_1^*, y_1^*, \varphi_1^*)$ and $(x_2^*, y_2^*, \varphi_2^*)$ as the additional new decision variables. The resultant optimal decision variables were shown in Table IX and the optimal energy consumption was reduced by a further 3% than Simulation 2 to 2200.9 J. The task duration in Simulation 3 was 25.8 s. The resultant energy-optimal trajectories were shown in Figure 87. The resultant velocity trajectories of Simulation 3 are plotted in Figure 88. The path points, the optimal decision variables and the optimal energy consumption of the first set of simulations are shown in Table IX.

Table IX Simulation Set#1

Path Points	Optimal Decision Variables	Optimal Energy
Simulation 1		
$(x_0^*, y_0^*, \varphi_0^*) = (0,0,-0.5\pi)$	$\Delta t = 30.1$ s	
$(x_1^*, y_1^*, \varphi_1^*) = (2,2,-0.2\pi)$		2547.7 J
Simulation 2		
$(x_0^*, y_0^*, \varphi_0^*) = (0,0,-0.5\pi)$	$\Delta \boldsymbol{t} = [9.1\ 8.3\ 9.1]$ s	
$(x_1^*, y_1^*, \varphi_1^*) = (0.67,0.67,-0.4\pi)$		
$(x_2^*, y_2^*, \varphi_2^*) = (1.33,1.33,-0.3\pi)$		
$(x_3^*, y_3^*, \varphi_3^*) = (2,2,-0.2\pi)$		2272.7 J
Simulation 3		
$(x_0^*, y_0^*, \varphi_0^*) = (0,0,-0.5\pi)$	$\Delta \boldsymbol{t} = [10\ 10.5\ 5.3]$ s	
$(x_3^*, y_3^*, \varphi_3^*) = (2,2,-0.2\pi)$	$(x_1^*, y_1^*, \varphi_1^*) = (0.71,0.69,0.06)$	
	$(x_2^*, y_2^*, \varphi_2^*) = (1.66,1.67,0.56)$	2200.9 J

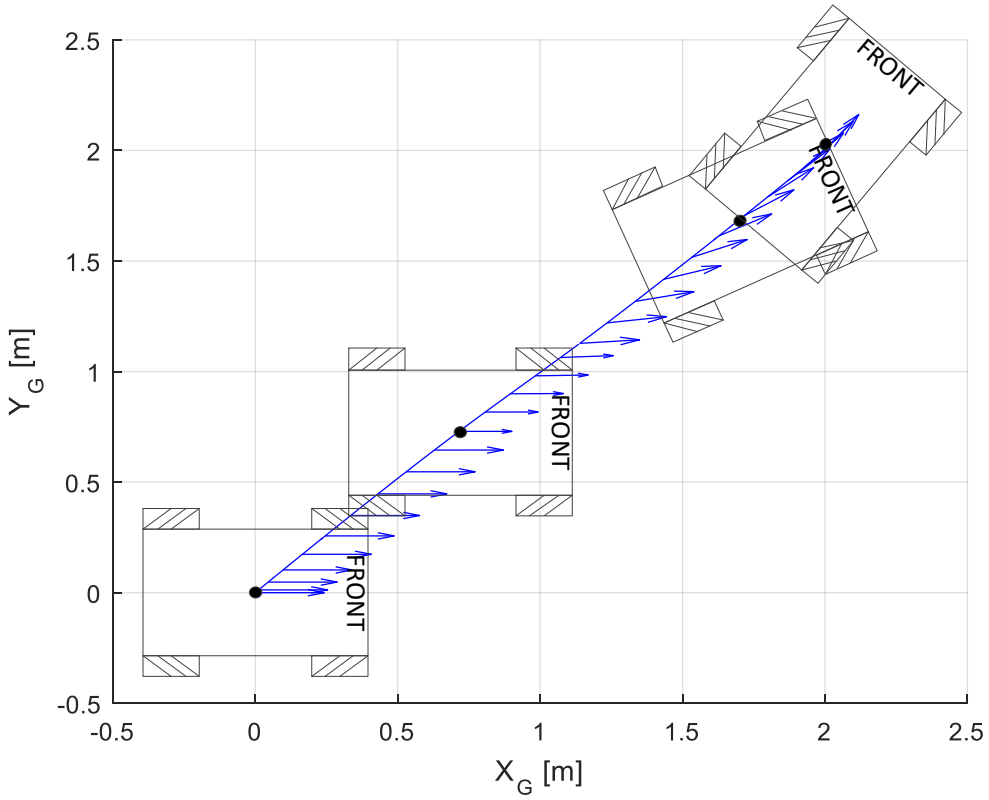
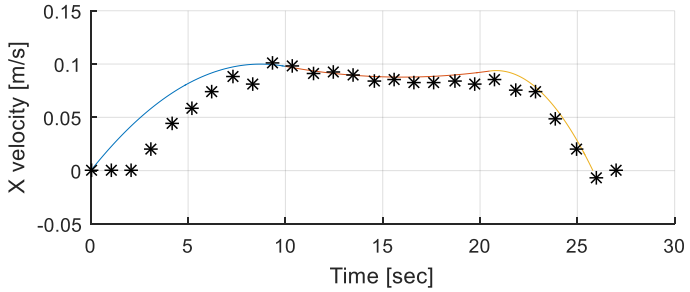
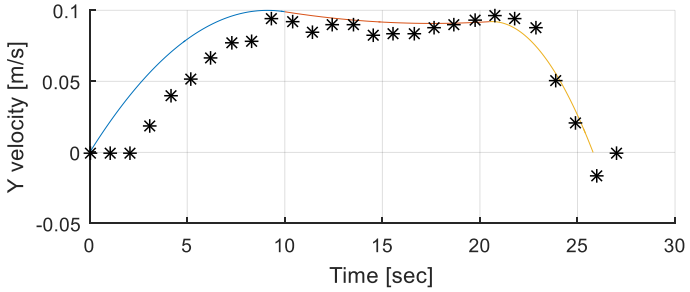


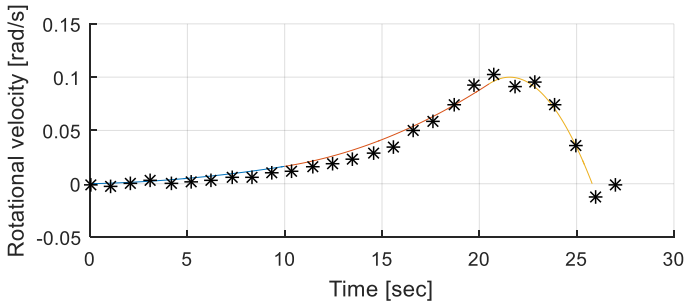
Figure 87 The energy-optimal trajectories in Simulation 3.



(a)



(b)



(c)

Figure 88 The velocity trajectories in Simulation 3 and Experiment 3. (a) X translational velocity. (b) Y translational velocity. (c) Z rotational velocity

Cubic and quartic polynomials

In the second set of simulations, the start pose, the via points and the goal pose were given as $(x_0^*, y_0^*, \varphi_0^*) = (0,0,-0.5\pi)$, $(x_1^*, y_1^*, \varphi_1^*) = (2,1,-0.5\pi)$, $(x_2^*, y_2^*, \varphi_2^*) = (1,1,-0.5\pi)$ and $(x_3^*, y_3^*, \varphi_3^*) = (2,2,-0.2\pi)$, as shown in Figure 89. Simulation 4 and 5 utilized the third-order and fourth-order splines, respectively for the purpose of investigating the influence of increasing the spline order on the resultant optimal energy. Both simulations in the second set employed a set of three functions of three segments of the splines. Because the cubic splines were utilized in Simulation 4, only three spline time variables were the decision variables. The resultant optimal decision variables were $\Delta \boldsymbol{t} = [31.2\ 21.5\ 13.8]\ s$ and the resultant optimal

energy consumption was 5497.7 J. The resultant energy-optimal trajectories of Simulation 4 are shown in Figure 90.

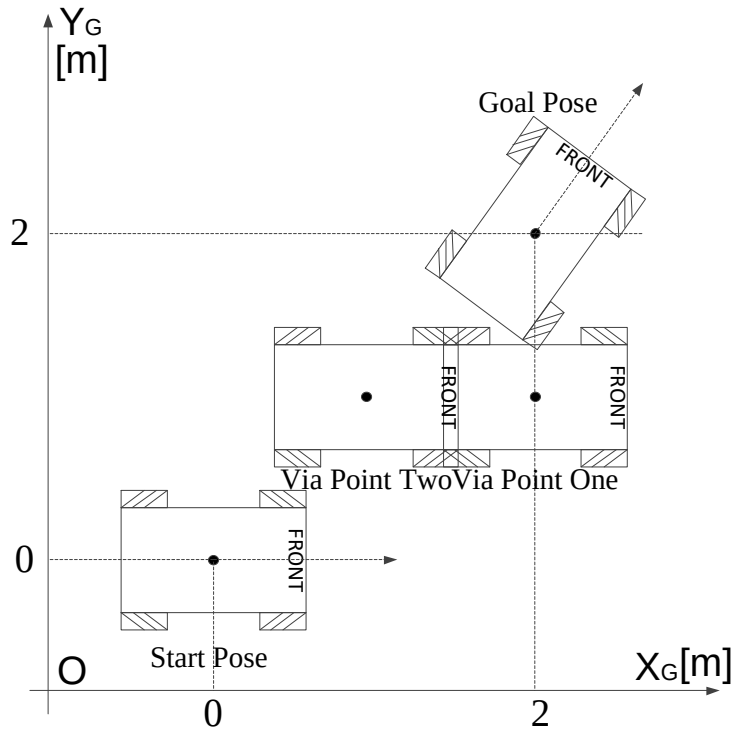


Figure 89 Path points in the second set of simulations.

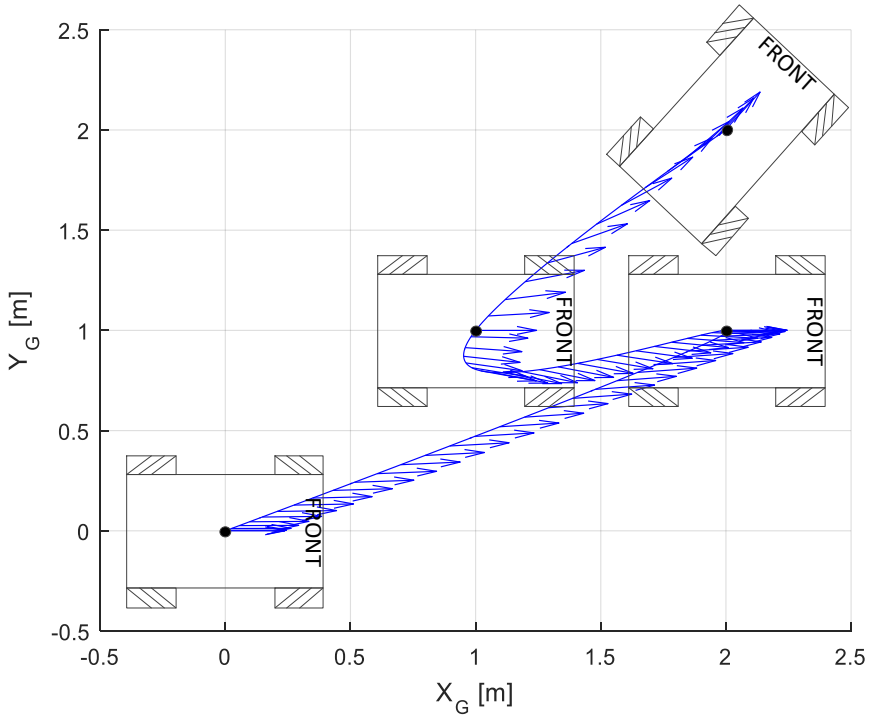
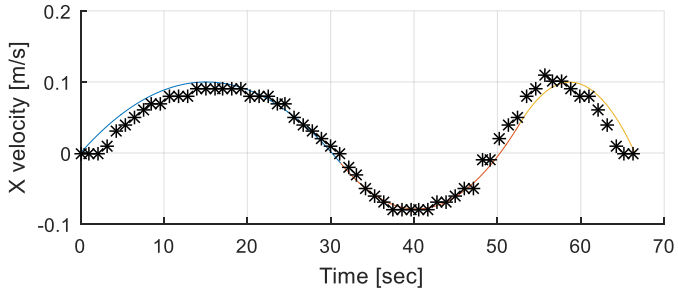
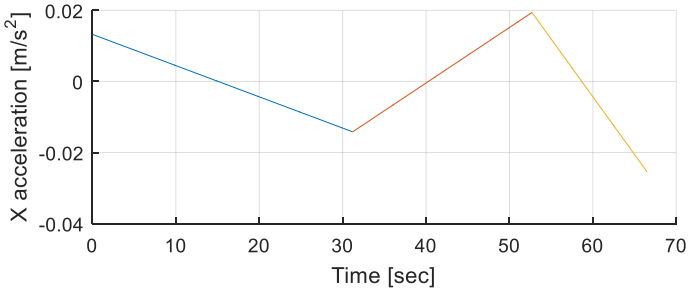


Figure 90 The energy-optimal trajectories in Simulation 4.

The resultant velocity and acceleration trajectories of Simulation 4 for translation X, translation Y and rotational Z are shown in Figure 91, Figure 92 and Figure 93, respectively.

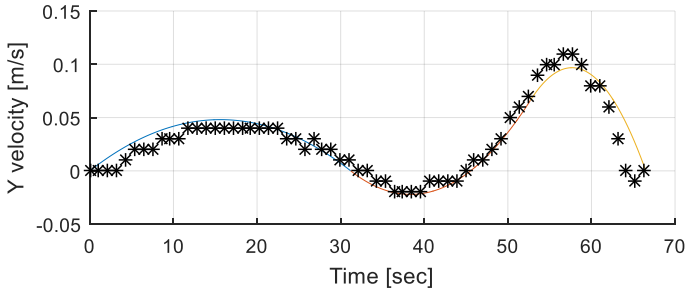


(a)

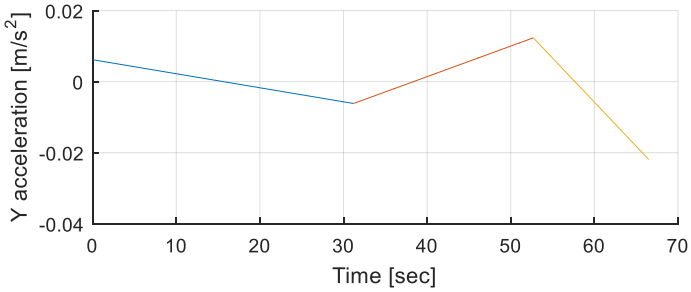


(b)

Figure 91 The X translational trajectories of Simulation and Experiment 4. (a) The X translational velocity trajectory of Simulation and Experiment 4. (b) The X translational acceleration trajectories of Simulation 4.

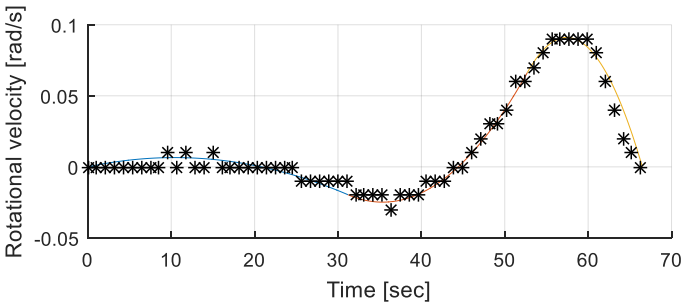


(a)

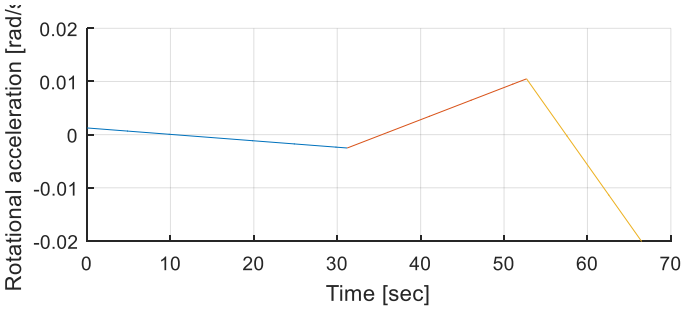


(b)

Figure 92 The Y translational trajectories of Simulation and Experiment 4. (a) The Y translational velocity trajectory of Simulation and Experiment 4. (b) The Y translational acceleration trajectories of Simulation 4.



(a)



(b)

Figure 93 The Z rotational trajectories of Simulation and Experiment 4. (a) The Z rotational velocity trajectory of Simulation and Experiment 4. (b) The Z rotational acceleration trajectories of Simulation 4.

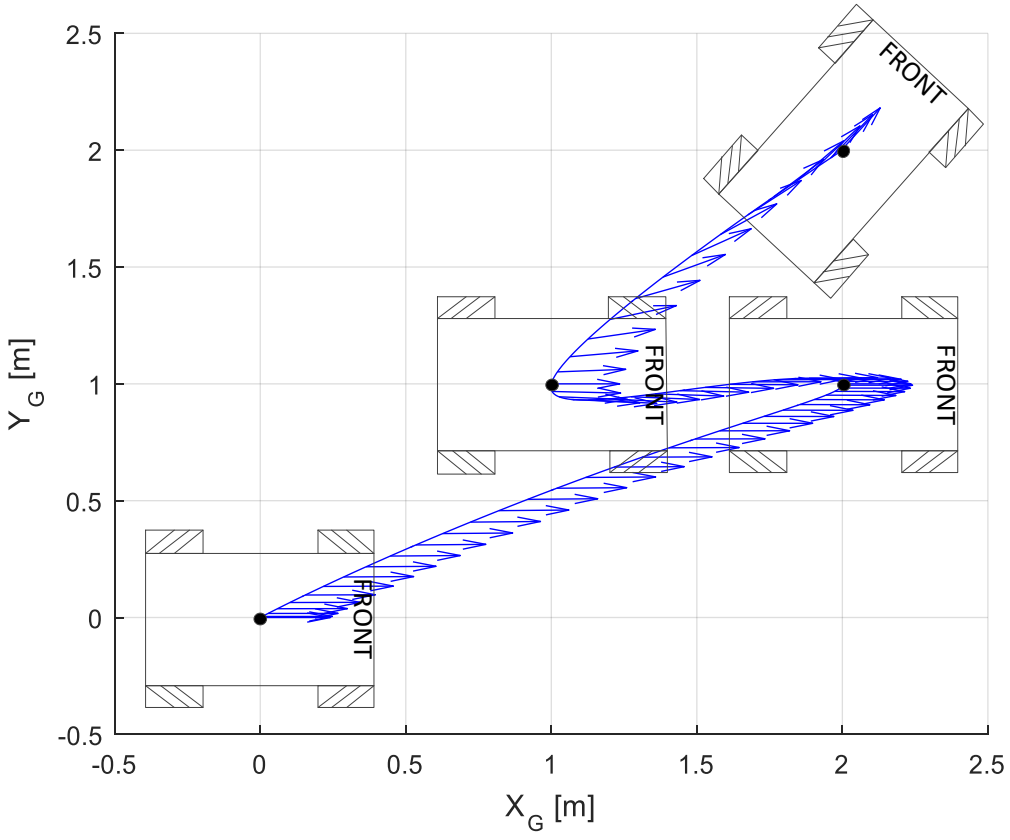
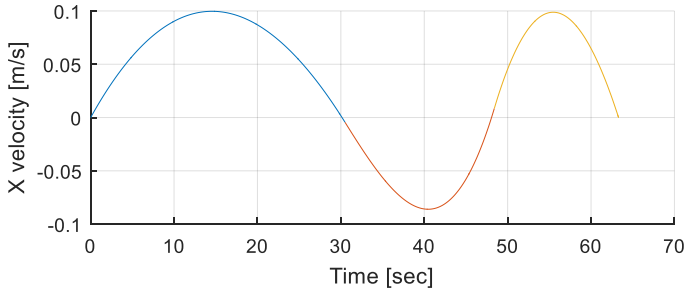
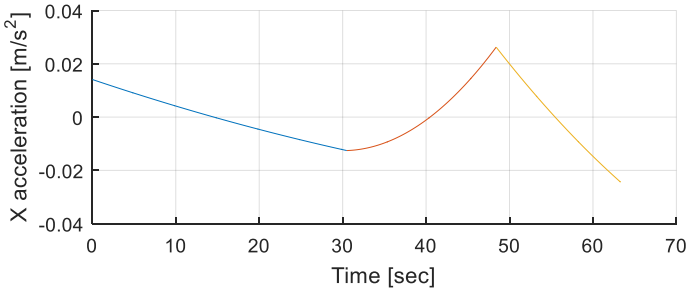


Figure 94 The energy-optimal trajectories in Simulation 5.

Simulation 5 applied the quartic splines so that both three spline time variables and the additional nine fourth-order term coefficients were applied as the decision variables. The resultant decision variables were shown in Table X and the resultant optimal energy was reduced by 4.8% to 5235.2 J. The resultant energy-optimal trajectories were shown in Figure 94. The resultant velocity and acceleration trajectories of Simulation 5 for translation X, translation Y and rotational Z were shown in Figure 95, Figure 96 and Figure 97, respectively. The results of Simulation 4-5 are indicated in Table X.

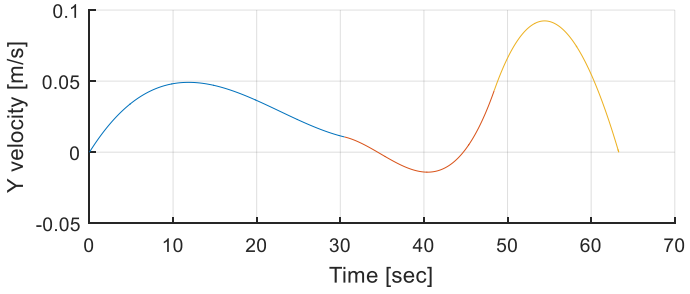


(a)

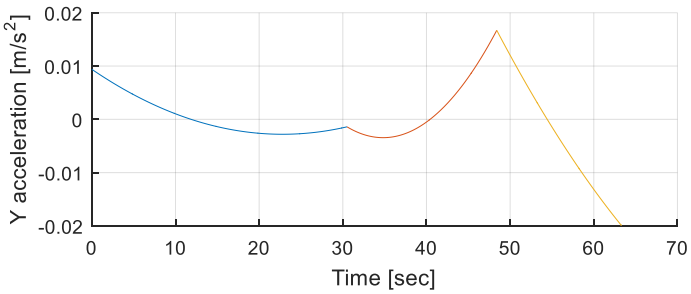


(b)

Figure 95 The X translational trajectories of Simulation and Experiment 5. (a) The X translational velocity trajectory of Simulation and Experiment 5. (b) The X translational acceleration trajectories of Simulation 5.

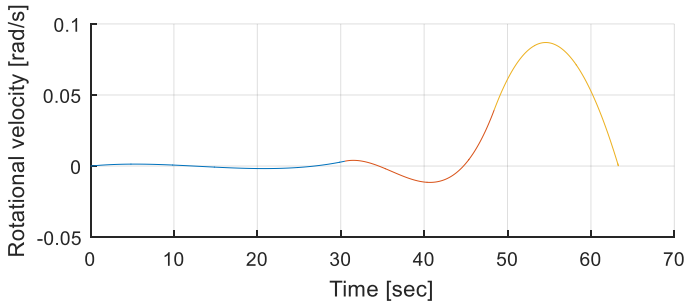


(a)

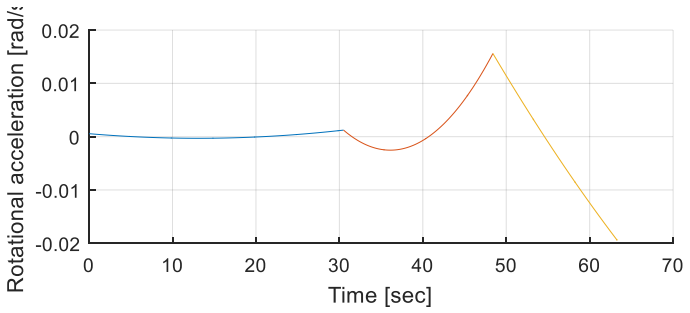


(b)

Figure 96 The Y translational trajectories of Simulation and Experiment 5. (a) The Y translational velocity trajectory of Simulation and Experiment 5. (b) The Y translational acceleration trajectories of Simulation 5.



(a)



(b)

Figure 97 The Z rotational trajectories of Simulation and Experiment 5. (a) The Z rotational velocity trajectory of Simulation and Experiment 5. (b) The Z rotational acceleration trajectories of Simulation 5.

Table X Simulation Set#2

Optimal Decision Variables	Optimal Energy
Simulation 4	
$\Delta \boldsymbol{t} = [31.2\ 21.5\ 13.8]$ s	5497.7 J
Simulation 5	
$\Delta \boldsymbol{t} = [30.5\ 17.9\ 14.9]$ s	
$\boldsymbol{a_4} = [4.94 \cdot 10^{-7}\ 9.55 \cdot 10^{-6}\ 3.39 \cdot 10^{-6}]$	
$\boldsymbol{b_4} = [1.96 \cdot 10^{-6}\ 9.10 \cdot 10^{-6}\ 2.93 \cdot 10^{-6}]$	
$\boldsymbol{c_4} = [4.21 \cdot 10^{-7}\ 9.99 \cdot 10^{-6}\ 1.63 \cdot 10^{-6}]$	5235.2 J

Time and energy optimization

In the third set of simulations, Simulations 1-5 were redone as Simulation 6-10 by utilizing the new objective function of minimizing both the task duration t_n and the energy consumption E_{total}. The objective function F_o is

$$F_o = \alpha \cdot t_n + (1-\alpha) \cdot E_{total} \tag{99}$$

The scale weight of time optimization α increased from 0 to 1. It should be noted that the maximum velocity and acceleration constraints of the robot were no longer applied in this set of simulations. As the task duration t_n was much smaller than the Mecanum robotic energy consumption E_{total} in the simulations, small α did not make any obvious difference to the results. So α starts from 0.99. The results are indicated in Table XI.

Table XI Simulation Set#3

	α=0*	α=0.990	α=0.995	α=0.999
Simulation 6	1092.2 J (6.5 s)	1173.2 J (4.3 s)	1302.1 J (3.4 s)	2048.5 J (1.8 s)
Simulation 7	1058.1 J (6.2 s)	1171.1J (3.8 s)	1215.7 J (3.5 s)	2405.6 J (1.4 s)
Simulation 8	987.1 J (6.1 s)	1065.6 J (4.7 s)	1168.2 J (3.3 s)	1980.7 J (1.9 s)
Simulation 9	1805.0 J (11.0 s)	1931.7 J (7.3 s)	2136.4 J (5.8 s)	3342.5 J (3.0 s)
Simulation 10	1788.9 J (10.7 s)	1911.2 J (7.4 s)	2136.2 J (5.8 s)	3440.6 J (3.0 s)

*Energy Optimization

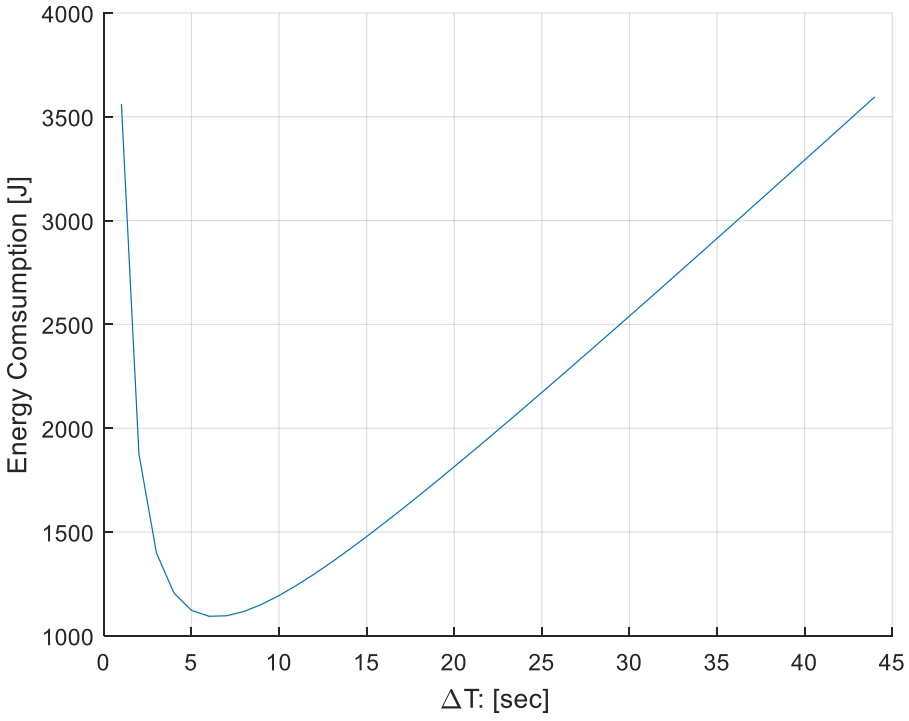


Figure 98 Energy consumption versus decision variable Δt in Simulation 6.

Simulation 6 has only one single decision variable Δt as Simulation 1 does. In order to further investigate the relationship between energy consumption and the task duration, a range of task durations (from 1 s to 44 s) is tested in Simulation 6. All the task durations are used to find the corresponding trajectories. Each trajectory is then evaluated by the energy consumption model for the energy consumption. A plot of the energy consumption versus the single decision variable Δt is illustrated in Figure 98.

6.5.2 Experiments

The energy-optimal motion trajectory via polynomial functions was implemented on the Mecanum robot. The resultant energy-optimal velocity trajectories from Simulation 2-4 were experimentally executed on the purpose-built robot. Based on the optimal decision variables that resulted from the simulations, the velocity trajectories $\dot{X}(t)$, $\dot{Y}(t)$ and $\dot{\phi}(t)$ were set as the desired velocities. Two laser scanners were utilized to detect real-time velocities of the robot in the experiments and a PI close-loop velocity controller was tuned to properly follow the desired velocity trajectories. During the experiments, the current sensors recorded the actual current of the robot's battery and the recorded battery current was used to calculate the actual energy consumption.

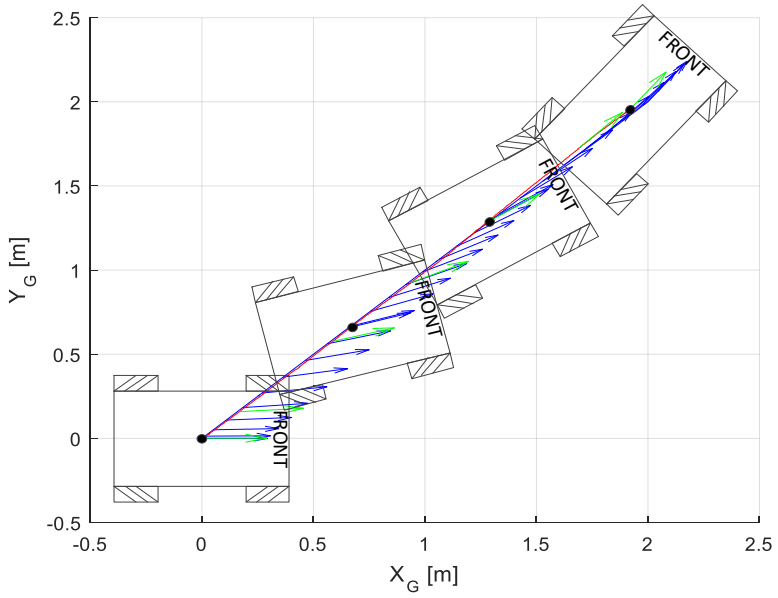


Figure 99 Desired and actual position trajectories of Simulation and Experiment 2.

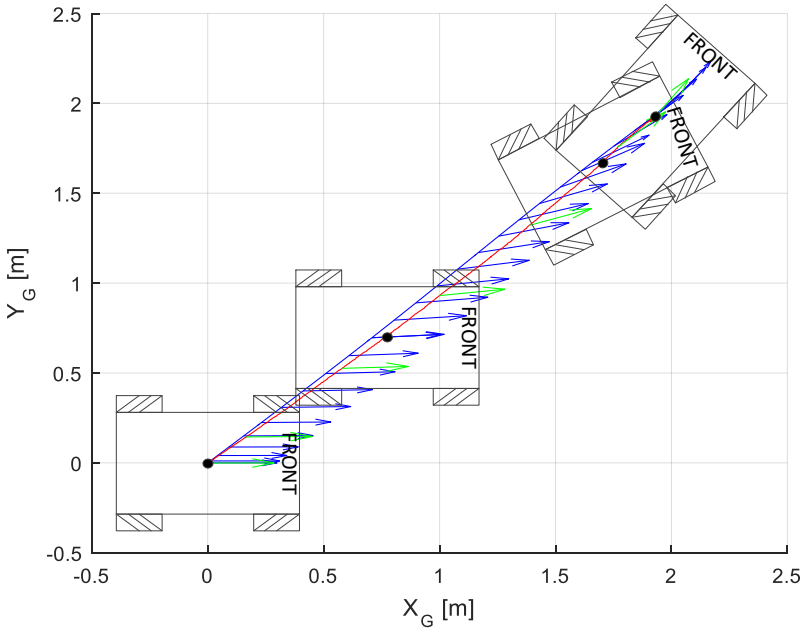


Figure 100 Desired and actual position trajectories of Simulation and Experiment 3.

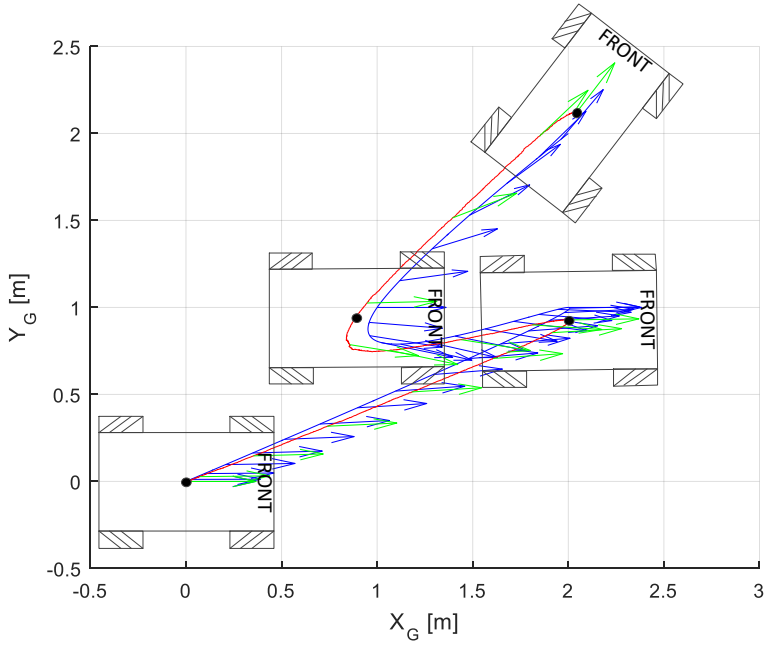
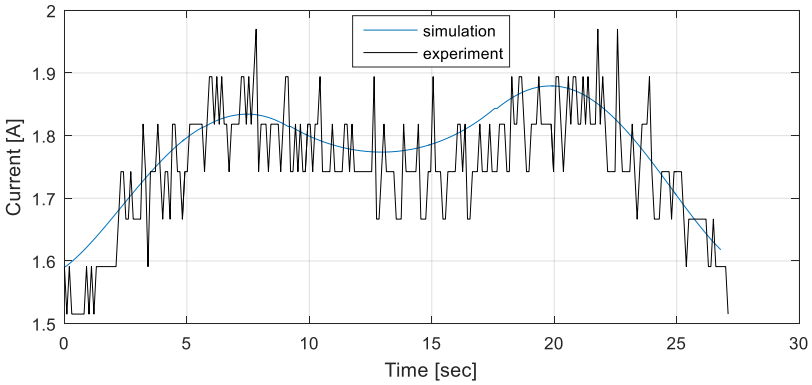


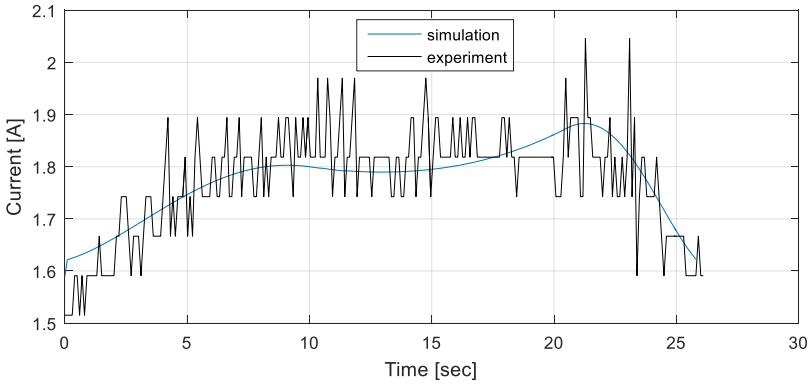
Figure 101 Desired and actual position trajectories of Simulation and Experiment 4.

The actual position trajectories in the experiments are shown in Figure 99, Figure 100 and Figure 101, to compare with the desired position trajectories. The actual paths were drawn in red lines and the actual robotic orientations were represented using green arrows. Figure 99 and Figure 100 show that the actual position trajectories matched with the desired position trajectories and Figure 101 shows that the actual position trajectories tracked the desired position trajectories within an acceptable tolerance.

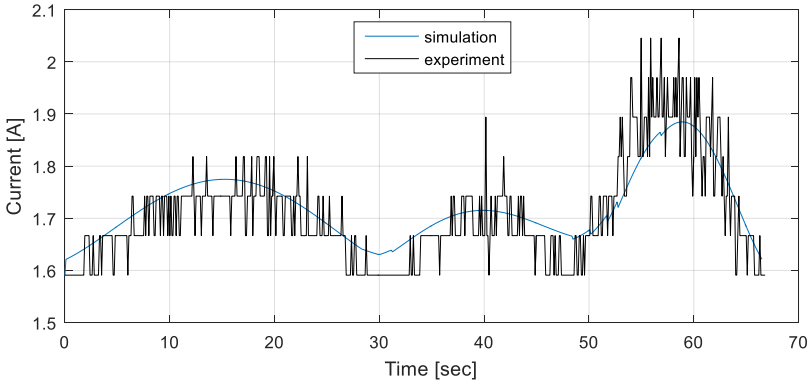
The actual velocity trajectories in the experiments are shown in Figure 86, Figure 88, Figure 91, Figure 92 and Figure 93, to compare with the desired velocity trajectories. The actual velocity was marked by black asterisks. The figures show that the actual velocity trajectories properly followed the desired velocity trajectories. There was an initial two-second delay before the robot started executing the desired velocity trajectories due to the robotic system delay.



(a)



(b)



(c)

Figure 102 Robotic battery current consumptions. (a) Simulation and Experiment 2. (b) Simulation and Experiment 3. (c) Simulation and Experiment 4.

To further validate the simulated energy consumption by the proposed energy consumption model, the actual battery current recorded in the experiments is plotted in Figure 102 to compare with the current consumption simulated by the energy consumption model. Because the robot's battery provided a stable voltage of 48 V, the actual energy consumption was calculated as 2275.9 J, 2216.4 J and 5466.8 J based on the actual current consumption in Figure 102 (a), (b) and (c), respectively.

6.5.3 Discussion

The results of Simulations 1 to 3 in Table IX show that increasing the polynomial spline number between the given path points reduced the resultant optimal energy consumption of the robot. Compared to Simulation 1, both Simulation 2 and Simulation 3 utilized more polynomial splines and lowered the resultant optimal energy by shortening the task duration while obeying the robot's maximum velocity and acceleration constraints. As shown in Figure 84, the velocity trajectory in Simulation 1 was a single segment of a second-order polynomial because one segment of third-order polynomial spline was utilized to describe a one degree-of-freedom trajectory. The velocity trajectories in Simulation 2 and Simulation 3 consisted of three segments of second-order polynomial splines. In addition, Simulation 3 reduced slightly more energy than Simulation 2 by avoiding inefficient trajectories, which was translating while rotating in such case, as shown in Figure 87. This was caused by simultaneously optimizing path planning and motion trajectory planning.

According to the results of Simulation 4-5 in Table X, the increasing spline orders reduced the resultant optimal energy consumption. As shown in the velocity and acceleration trajectory plots in Figure 95, Figure 96 and Figure 97, the quartic splines were able to manipulate the acceleration profile in a higher order for more energy optimization than the cubic splines that are shown in Figure 91, Figure 92 and Figure 93. This is because of the additional decision variables from the fourth-order spline parameters.

The results of Simulation 6-10 in Table XI demonstrated that the minimization of the robotic energy consumption was achieved by considering only the energy consumption as the objective function. The energy consumption of Simulation 6 versus its single decision variable Δt was plotted in Figure 98. Without the maximum velocity and acceleration constraints of the robot, the ideal optimal energy consumption was about 1090 J at Δt=6.5 s. This was close to the result found by the genetic algorithm in Simulation 6, which is shown in Table XI. Additionally,

when Δt was less than 6.5, the energy consumption increased dramatically because the power consumption to attain swifter robotic motion was much higher. The resultant energy consumption by multiplying the power and the task duration is still higher than the energy consumption at Δt =6.5. When Δt was more than 6.5, the energy consumption increased steadily, mostly due to the longer task duration.

Furthermore, the application of the energy-optimal trajectory generation via polynomial functions has been implemented on the purpose-built Mecanum robot. In the experiments, the purpose-built robot followed the resultant energy-optimal velocity trajectories that were found in the simulation. The actual position trajectories tracked the desired position trajectories within an acceptable tolerance. As shown in Figure 102, the simulated current and energy consumption based on the energy consumption model agreed with the actual consumption of the robot. Thus, the energy consumption model was qualified to be the energy consumption objective function for trajectory optimization purpose. The actual robotic energy consumptions were 2275.9 J, 2216.4 J and 5466.8 J, and they were close to the resultant optimal energy consumption of Simulations 2 to 4 in Table IX and Table X. Thus, the robot consumed the minimum energy consumption by following the resultant energy-optimal motion trajectories found by the polynomial-based trajectory generation technique.

In this chapter, polynomial functions were chosen to describe the robotic motion trajectory mainly because it is a common primitive function form in the industry for trajectory generation and it easily generates differentiable trajectories for robotic motion. Other differentiable primitive functions are subjective to be investigated in future works, such as trigonometric function.

6.6. Summary

Motivated by minimizing the energy consumption of the Mecanum robot via optimizing its omnidirectional motion trajectories, a new application of trajectory generation via polynomial functions was studied in this chapter. The task-space omnidirectional trajectories of the holonomic drive were described using polynomial functions. Based on the proposed energy consumption model of the robot, the MATLAB genetic algorithm solver was applied to optimize the decision variables for the minimum-energy trajectories of the robot. The simulations have shown that the application of trajectory generation via polynomial functions finds the energy-optimal trajectories for the Mecanum robot. It was observed by the simulations

that increasing the number or the order of the splines between the given path points lowers the resultant optimal energy consumption. In this technique, path planning and motion trajectory planning can be simultaneously optimized by the relaxation of via points. This contributes to reducing energy consumption and avoiding inefficient trajectories. The proposed technique has been experimentally implemented on the robot. The robot consumes the minimum energy consumption by following the energy-optimal trajectories that are found by the polynomial-based trajectory generation.

Chapter 7. Conclusion and Future Work

The study consisted of two major parts. The first part was the Mecanum robot design and the second part was the energy-efficient autonomous navigation research. The aim of this PhD research project was to first develop a heavy-duty omnidirectional Mecanum mobile robot that performs fully functional autonomous navigation for research purposes and then to realize energy-efficient autonomous navigation on the designed robot via both online and offline optimal motion planning research studies.

First, a heavy-duty omnidirectional autonomous Mecanum mobile robot was designed and developed from scratch. The Mecanum wheel locomotion mechanism is different from the conventional wheel and the Mecanum wheel has practical applications in the industry due to its special omnidirectional mobility and unique heavy-duty character. The Mecanum drive-based mobile platform is manoeuvrable, and thus more convenient to transport in a confined area. A Mecanum-based forklift working in a warehouse environment is a typical example. The design part of this PhD work has successfully developed an autonomous Mecanum robot. The design requirements and manufacturing standards of the robot were according to an autonomous warehouse forklift prototype. The designed robot can perform the fully functional autonomous navigation needed to conduct the experimental research in this project but has been also ready to be expanded by assembling a robotic manipulator.

Second, an energy-efficient autonomous navigation research study was conducted on the Mecanum robot. It is beneficial to reduce the energy consumption of omnidirectional autonomous mobile robots by optimizing the motion trajectory planning. Energy-efficient autonomous navigation via optimal trajectory planning has been researched very little for omnidirectional mobile robots. The research part of this PhD work approached the problem based on an established and experimentally validated robotic energy consumption model and achieved the goal via both online local motion trajectory planning and offline global motion trajectory planning.

7.1. Contributions

- The research proposal with the literature review was completed.
- The heavy-duty autonomous Mecanum four-wheel omnidirectional mobile robot was developed via mechatronics design. The first objective – Mechatronics Design of a

Heavy-Duty Omnidirectional Mecanum Autonomous Mobile Robot – was completed. The robot is able to perform fully functional autonomous navigation by implementing the cutting-edge mobile robotic technology from ROS. A conference paper based on the robotic system development was presented in the IEEE International Conference on Mechatronics (ICM) in 2015 [1].

- The energy consumption mode of the robot was built and experimentally validated. The completion of the second objective – Energy Consumption Model of the Four-Wheeled Omnidirectional Mecanum Robot – was achieved. After comparing the simulation results to experimental results, the energy consumption model was proved to have over 98% accuracy for primitive omnidirectional motions and over 95% accuracy for complex omnidirectional motions. A conference paper based on the proposed energy consumption model was presented in the IEEE 14th International Workshop on Advanced Motion Control (AMC) in 2016 [2, 83].

- The energy-optimal online motion planning via the Dynamic Window Approach was studied in this research. The proposed technique of extending the Dynamic Window Approach was proved to achieve the optimization of power consumption and the reduction of total energy consumption in the autonomous navigation experiments. The completion of the third objective – Energy-Optimal Online Local Trajectory Planning for Autonomous Navigation – was published in the International Journal of Advanced Robotic Systems.

- The energy-optimal motion trajectory planning for the robot was realized by using a polynomial-based trajectory generation technique to achieve the fourth objective – Energy-Optimal Offline Global Motion Planning of Omnidirectional Robots. A new application to the Mecanum mobile robot of the classic trajectory generation technique via polynomial functions was proposed to offline plan its globally energy-optimal motion trajectory. Based on the research for this objective, a journal paper is planned to be submitted to the journal, Industrial Robot.

7.2. Conclusions

- The heavy-duty omnidirectional four-wheeled Mecanum autonomous mobile robot was designed and developed from scratch in this project, for the purposes of conducting energy-optimal autonomous navigation research and for use in the future as a

warehouse forklift prototype operating in industrial environments. The entire mechatronics design of the robot includes the electromechanics design, the locomotion control system, the autonomous navigation system, and the combined robotic control system. Now, the robot is able to perform fully functional autonomous navigation.

- A novel energy consumption model specifically for the four-wheel omnidirectional Mecanum mobile robot was proposed in this study. The proposed energy consumption model of the four-wheeled Mecanum robot considers the validated influencing factors from the recently published energy model works. The proposed model has been experimentally validated by showing that the energy consumption model has over 98% accuracy for primitive omnidirectional motions and over 95% accuracy for complex omnidirectional motions. However, this model is specific to the unique dynamics of the Mecanum wheel in omnidirectional motions. Different dynamics should be accommodated in the proposed model in order to cope with other types of mobile robots.

- Aiming for power-minimization and energy-reduction autonomous navigation, a new extension of the Dynamic Window Approach (DWA), was proposed to online generate the locally energy-optimal trajectory for mobile robots, by adding a new energy-related criterion. Unlike most existing energy-optimal motion planning methods, the proposed technique is generally applicable to most mobile robots and copes with any shape of path. More importantly, it is compatible with online path planners and is able to reactively avoid unforeseen obstacles. These advantages allow the DWA-based energy-optimal local trajectory planner to be conveniently fitted to autonomous navigation. Comprehensive experiments have been performed to show the effectiveness of the proposed technique in various autonomous navigation task scenarios. The optimization of power consumption and the reduction of total energy consumption were made possible by taking advantage of the holonomic mobility of the Mecanum robot.

- A new application to the Mecanum mobile robot of the classic trajectory generation technique via polynomial functions was proposed to offline plan its globally energy-optimal motion trajectory. Simulations proved that the application of trajectory generation via polynomial functions could find the energy-optimal polynomial trajectories for the Mecanum robot. It was observed that increasing the number or the

order of the splines lowers the optimal energy consumption. The simulation results have been validated by experiments.

7.3. Future Work

- In the future, the designed Mecanum robot should be further developed to be an omnidirectional Mecanum warehouse forklift prototype. Future hardware design and development of the robot should consider installing a robotic manipulator, and the control of the robotic manipulator should be fused into the current robotic control system.

- The future sensor trial study could extend to 3D-LiDAR (Light Detection and Ranging). 3D-LiDAR-based on SLAM and 3D-LiDAR-based navigation are both interesting topics for the future.

- A future energy-efficient autonomous navigation research study should additionally optimize planning of the robotic manipulator arm to pick up an object at the goal pose. Thus, the energy-optimal planning problem is defined to be optimally planning both the robotic motion trajectory and the robotic arm's trajectory for the robot to pick up the goal object based on its current pose.

References

1. Xie, L., et al. *Heavy-duty omni-directional Mecanum-wheeled robot for autonomous navigation: System development and simulation realization*. in *Mechatronics (ICM), 2015 IEEE International Conference on*. 2015. IEEE.
2. Xie, L., et al. *Experimental validation of energy consumption model for the four-wheeled omnidirectional Mecanum robots for energy-optimal motion control*. in *2016 IEEE 14th International Workshop on Advanced Motion Control (AMC)*. 2016. IEEE.
3. Xie, L., et al., *Power-minimization and energy-reduction autonomous navigation of an omnidirectional Mecanum robot via the dynamic window approach local trajectory planning.* International Journal of Advanced Robotic Systems, 2018. **15**(1): p. 1729881418754563.
4. Siegwart, R., I.R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. 2011: MIT press.
5. Doroftei, I., V. Grosu, and V. Spinu, *Omnidirectional mobile robot–design and implementation*. 2007: INTECH Open Access Publisher.
6. Adăscăliţei, F. and I. Doroftei, *Practical applications for mobile robots based on mecanum wheels-a systematic survey.* The Romanian Review Precision Mechanics, Optics & Mechatronics, 2011. **40**: p. 21-29.
7. Ilon, B.E., *Wheels for a course stable selfpropelling vehicle movable in any desired direction on the ground or some other base*. 1975, Google Patents.
8. Gfrerrer, A., *Geometry and kinematics of the Mecanum wheel.* Computer Aided Geometric Design, 2008. **25**(9): p. 784-791.
9. Diegel, O., et al. *Improved mecanum wheel design for omni-directional robots*. in *Proc. 2002 Australasian Conference on Robotics and Automation, Auckland*. 2002.
10. Phillips, J.G., *Mechatronic design and construction of an intelligent mobile robot for educational purposes.* Master of Technology Thesis, Massey University, Palmerston North, New Zealand, 2000. **150**.
11. Lippit, T. and W. Jones, *OmniBot Mobile Base.* KSC Re-search and Technology Report, NASA, USA, 1998.
12. Houshangi, N. and T. Lippitt. *Omnibot mobile base for hazardous environment*. in *Electrical and Computer Engineering, 1999 IEEE Canadian Conference on*. 1999. IEEE.
13. Vogler, A., et al., *MarsCruiserOne*. 2007, SAE Technical Paper.
14. Bischoff, R., U. Huggenberger, and E. Prassler. *Kuka youbot-a mobile manipulator for research and education*. in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. 2011. IEEE.
15. Schulze, L., S. Behling, and S. Buhrs. *Development of a micro drive-under tractor-research and application*. in *Proceedings of the International MultiConference of Engineers and Computer Scientists*. 2011.
16. Hoyer, H., U. Borgolte, and A. Jochheim. *The OMNI-Wheelchair-State of the art*. in *Proceedings of Conference on Technology and Persons with Disabilities, Los Angeles*. 1999.
17. How, T.-V., *Development of an anti-collision and navigation system for powered wheelchairs*. 2010, University of Toronto (Canada).
18. Hsu, P.-E., Y.-L. Hsu, and J.-M. Lu, *iRW-An Intelligent Robotic Wheelchair Integrated with Advanced Robotic and Telehealth Solutions.* 2011.
19. Muir, P.F. and C.P. Neuman, *Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot*, in *Autonomous robot vehicles*. 1990, Springer. p. 25-31.

20. Goller, M., et al. *Setup and control architecture for an interactive shopping cart in human all day environments*. in *Advanced Robotics, 2009. ICAR 2009. International Conference on*. 2009. IEEE.
21. De Villiers, M. and N. Tlale, *Development of a control model for a four wheel mecanum vehicle*. Journal of Dynamic Systems, Measurement, and Control, 2012. **134**(1): p. 011007.
22. Muir, P.F. and C.P. Neuman, *Kinematic modeling of wheeled mobile robots.* Journal of robotic systems, 1987. **4**(2): p. 281-340.
23. Han, K.-L., H. Kim, and J.S. Lee. *The sources of position errors of omni-directional mobile robot with Mecanum wheel*. in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*. 2010. IEEE.
24. Shimada, A., et al. *Mecanum-wheel vehicle systems based on position corrective control*. in *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*. 2005. IEEE.
25. Lin, L.-C. and H.-Y. Shih, *Modeling and Adaptive Control of an Omni-Mecanum-Wheeled Robot.* Intelligent Control and Automation, 2013. **4**(02): p. 166.
26. Viboonchaicheep, P., A. Shimada, and Y. Kosaka. *Position rectification control for Mecanum wheeled omni-directional vehicles*. in *Industrial Electronics Society, 2003. IECON'03. The 29th Annual Conference of the IEEE*. 2003. IEEE.
27. Tsai, C.-C. and H.-L. Wu. *Nonsingular terminal sliding control using fuzzy wavelet networks for Mecanum wheeled omni-directional vehicles*. in *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*. 2010. IEEE.
28. Tlale, N. and M. de Villiers. *Kinematics and dynamics modelling of a mecanum wheeled mobile platform*. in *Mechatronics and Machine Vision in Practice, 2008. M2VIP 2008. 15th International Conference on*. 2008. IEEE.
29. Zimmermann, K., I. Zeidis, and M. Abdelrahman, *Dynamics of Mechanical Systems with Mecanum Wheels*, in *Applied Non-Linear Dynamical Systems*. 2014, Springer. p. 269-279.
30. Matsinos, E., *Modelling of the motion of a Mecanum-wheeled vehicle.* arXiv preprint arXiv:1211.2323, 2012.
31. Han, K.-L., et al. *Design and control of mobile robot with Mecanum wheel*. in *ICCAS-SICE, 2009*. 2009. IEEE.
32. Kumra, S., R. Saxena, and S. Mehta, *Navigation System for Omni-directional Automatic Guided Vehicle with Mecanum Wheel.* IOSR Journal of Electrical and Electronics Engineering (IOSRJEEE) ISSN: 2278-1676 Volume. **2**: p. 35-39.
33. Park, J., et al. *Driving control of mobile robot with Mecanum wheel using fuzzy inference system*. in *Control Automation and Systems (ICCAS), 2010 International Conference on*. 2010. IEEE.
34. Dickerson, S.L. and B.D. Lapin. *Control of an omni-directional robotic vehicle with Mecanum wheels*. in *Telesystems Conference, 1991. Proceedings. Vol. 1., NTC'91., National*. 1991. IEEE.
35. Kamiuchi, S. and S. Maeyama. *A novel human interface of an omni-directional wheelchair*. in *Robot and Human Interactive Communication, 2004. ROMAN 2004. 13th IEEE International Workshop on*. 2004. IEEE.
36. Tsai, C.-C., F.-C. Tai, and Y.-R. Lee. *Motion controller design and embedded realization for mecanum wheeled omnidirectional robots*. in *Intelligent Control and Automation (WCICA), 2011 9th World Congress on*. 2011. IEEE.
37. Cooney, J., W. Xu, and G. Bright, *Visual dead-reckoning for motion control of a Mecanum-wheeled mobile robot.* Mechatronics, 2004. **14**(6): p. 623-637.

38. Morimoto, S., et al., *Servo drive system and control characteristics of salient pole permanent magnet synchronous motor.* IEEE Transactions on Industry Applications, 1993. **29**(2): p. 338-343.
39. *TAPAS Robotics-enabled Logistics and Assistive Services for the Transformable Factory of the Future.*
40. Miller, K., *Introduction to OmniRob 2.1*. 2013, KUKA Laboratories GmbH.
41. Marder-Eppstein, E., et al. *The Office Marathon: Robust Navigation in an Indoor Office Environment*. in *International Conference on Robotics and Automation*. 2010.
42. Shiller, Z., *Off-Line and On-Line Trajectory Planning*, in *Motion and Operation Planning of Robotic Systems*. 2015, Springer. p. 29-62.
43. Xu, W., *A virtual target approach for resolving the limit cycle problem in navigation of a fuzzy behaviour-based mobile robot.* Robotics and Autonomous Systems, 2000. **30**(4): p. 315-324.
44. Fox, D., W. Burgard, and S. Thrun, *The dynamic window approach to collision avoidance.* IEEE Robotics & Automation Magazine, 1997. **4**(1): p. 23-33.
45. Brock, O. and O. Khatib. *High-speed navigation using the global dynamic window approach*. in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. 1999. IEEE.
46. Ögren, P. and N.E. Leonard, *A convergent dynamic window approach to obstacle avoidance.* Robotics, IEEE Transactions on, 2005. **21**(2): p. 188-195.
47. Berti, H., A. Sappa, and O. Agamennoni, *Improved dynamic window approach by using Lyapunov stability criteria.* Latin American applied research, 2008. **38**(4): p. 289-298.
48. Kiss, D. and G. Tevesz. *Advanced dynamic window based navigation approach using model predictive control*. in *Methods and Models in Automation and Robotics (MMAR), 2012 17th International Conference on*. 2012. IEEE.
49. Liu, S. and D. Sun, *Minimizing energy consumption of wheeled mobile robots via optimal motion planning.* Mechatronics, IEEE/ASME Transactions on, 2014. **19**(2): p. 401-411.
50. Kim, C.H. and B.K. Kim, *Minimum-energy translational trajectory generation for differential-driven wheeled mobile robots.* Journal of Intelligent and Robotic Systems, 2007. **49**(4): p. 367-383.
51. Kim, H. and B.K. Kim, *Online minimum-energy trajectory planning and control on a straight-line path for three-wheeled omnidirectional mobile robots.* Industrial Electronics, IEEE Transactions on, 2014. **61**(9): p. 4771-4779.
52. Mei, Y., et al. *Energy-efficient motion planning for mobile robots*. in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. 2004. IEEE.
53. Sun, Z. and J.H. Reif, *On finding energy-minimizing paths on terrains.* Robotics, IEEE Transactions on, 2005. **21**(1): p. 102-114.
54. Bose, B.K., *Global warming: Energy, environmental pollution, and the impact of power electronics.* Industrial Electronics Magazine, IEEE, 2010. **4**(1): p. 6-17.
55. Bose, B.K. *Energy, environment, and advances in power electronics*. in *Industrial Electronics, 2000. ISIE 2000. Proceedings of the 2000 IEEE International Symposium on*. 2000. IEEE.
56. Bose, B.K., *Power electronics and motor drives recent progress and perspective.* Industrial Electronics, IEEE Transactions on, 2009. **56**(2): p. 581-588.
57. Mei, Y., et al. *A case study of mobile robot's energy consumption and conservation techniques*. in *Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on*. 2005. IEEE.

58. Wang, T., et al. *Staying-alive and energy-efficient path planning for mobile robots*. in *American Control Conference, 2008*. 2008. IEEE.
59. Rowe, N.C. and R.S. Ross, *Optimal grid-free path planning across arbitrarily contoured terrain with anisotropic friction and gravity effects.* Robotics and Automation, IEEE Transactions on, 1990. **6**(5): p. 540-553.
60. Hart, P.E., N.J. Nilsson, and B. Raphael, *A formal basis for the heuristic determination of minimum cost paths.* Systems Science and Cybernetics, IEEE Transactions on, 1968. **4**(2): p. 100-107.
61. Beard, R.W., et al., *Coordinated target assignment and intercept for unmanned air vehicles.* Robotics and Automation, IEEE Transactions on, 2002. **18**(6): p. 911-922.
62. Pettersson, P.O. and P. Doherty, *Probabilistic roadmap based path planning for an autonomous unmanned aerial vehicle.* Sensors, 2004. **200**: p. 66Hz.
63. Kim, K.B. and B.K. Kim, *Minimum-time trajectory for three-wheeled omnidirectional mobile robots following a bounded-curvature path with a referenced heading profile.* Robotics, IEEE Transactions on, 2011. **27**(4): p. 800-808.
64. Yang, K. and S. Sukkarieh, *An analytical continuous-curvature path-smoothing algorithm.* Robotics, IEEE Transactions on, 2010. **26**(3): p. 561-568.
65. Ray, L.E., et al., *Design and power management of a solar‐powered "cool robot" for polar instrument networks.* Journal of Field Robotics, 2007. **24**(7): p. 581-599.
66. Michaud, S., et al. *SOLERO: Solar powered exploration rover*. in *None*. 2002.
67. Lever, J. and L. Ray, *Revised solar-power budget for cool robot polar science campaigns.* Cold Regions Science and Technology, 2008. **52**(2): p. 177-190.
68. Ray, L., et al. *The design of a mobile robot for instrument network deployment in Antarctica*. in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. 2005. IEEE.
69. Yang, J., et al., *Comparison of optimal solutions to real-time path planning for a mobile vehicle.* Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 2010. **40**(4): p. 721-731.
70. Hussein, A.M. and A. Elnagar. *On smooth and safe trajectory planning in 2d environments*. in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*. 1997. IEEE.
71. Duleba, I. and J.Z. Sasiadek, *Nonholonomic motion planning based on Newton algorithm with energy optimization.* Control Systems Technology, IEEE Transactions on, 2003. **11**(3): p. 355-363.
72. Kim, H. and B.K. Kim. *Minimum-energy trajectory planning and control on a straight line with rotation for three-wheeled omni-directional mobile robots*. in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. 2012. IEEE.
73. Kim, H. and B.-K. Kim. *Minimum-energy translational trajectory planning for battery-powered three-wheeled omni-directional mobile robots*. in *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*. 2008. IEEE.
74. Kim, H.J. and B.K. Kim. *Minimum-energy trajectory planning on a tangent for battery-powered three-wheeled omni-directional mobile robots*. in *Control Automation and Systems (ICCAS), 2010 International Conference on*. 2010. IEEE.
75. How, T.-V., *Development of an Anti-Collision and Navigation System for Powered Wheelchairs*. 2011.
76. Bräunl, T., *Embedded robotics: mobile robot design and applications with embedded systems*. 2008: Springer Science & Business Media.
77. Xie, L. and C. Hu, *Mechatronics Design and Heavy-Duty Omni-Directional Mecanum Mobile Robot for Research Purposes.* UoAA, 2013: p. 55.

78. Quigley, M., et al. *ROS: an open-source Robot Operating System*. in *ICRA workshop on open source software*. 2009.
79. Chua, T.K., *Sensing for Localization of an Omnidirectional Mecanum Wheeled Robot.* Master of Thesis, The University of Auckland, Auckland, New Zealand, 2015.
80. TSENG, C.-T., *RGB-D SLAM AND REACTIVE NAVIGATION OF AN OMNI-DIRECTIONAL MOBILE ROBOT*. UoA, 2014: p. 1.
81. Scott, J., J. McLeish, and W.H. Round, *Speed control with low armature loss for very small sensorless brushed DC motors.* Industrial Electronics, IEEE Transactions on, 2009. **56**(4): p. 1223-1229.
82. Fox, D., W. Burgard, and S. Thrun, *The dynamic window approach to collision avoidance.* Robotics & Automation Magazine, IEEE, 1997. **4**(1): p. 23-33.
83. Henkel, C., A. Bubeck, and W. Xu, *Energy Efficient Dynamic Window Approach for Local Path Planning in Mobile Service Robotics.* IFAC-PapersOnLine, 2016. **49**(15): p. 32-37.
84. Paul, R.P. and H. Zhang. *Robot motion trajectory specification and generation*. in *Robotics Research: The Second International Symposium*. 1984.
85. Shintaku, E., *Minimum energy trajectory for an underwater manipulator and its simple planning method by using a genetic algorithm.* Advanced robotics, 1998. **13**(2): p. 115-138.
86. Tian, L. and C. Collins, *An effective robot trajectory planning method using a genetic algorithm.* Mechatronics, 2004. **14**(5): p. 455-470.
87. Linquan, Y., et al. *Path planning algorithm for mobile robot obstacle avoidance adopting Bezier curve based on genetic algorithm*. in *Control and Decision Conference, 2008. CCDC 2008. Chinese*. 2008. IEEE.

Appendix A. Instructions of Bringing Up and Running AuckBot

1. Connect Control-PC to the Beckhoff-PLC.

    On the Control-PC (located on the robot) open a terminator and execute:

    source start.sh

    Then in the same terminal execute

    roslaunch auckbot_bringup auckbot.launch

(Note: Robot must be turned on and Beckhoff-PLC must be in run-mode)

2. Connect Remote Control PC to the Control PC

    Now on the Remote Control PC, which is connected to the MecanumRobot Wi-Fi (password: **mecanum2013**) open the terminator and execute (now referred to as initial step):

    source startonremote.sh

    You will get asked for the password of the Control-PC (as we are now remote controlling it), type it in.

(Note: you have to execute this initial step for every new terminator window you open)

3. Start the Keyboard Controls

    Then execute

    roslaunch auckbot_teleop keyboard_teleop.launch

    Now you can remote control the robot.

(Note: for remote control only, the next steps are not necessary, you can stop here)

4. Start and setup the Path Planning Algorithms and Localization. For using the path planning algorithms follow the next steps:

    In a new terminal execute (after the *initial step* mentioned above)

    source setup_planner.sh

You can choose the setup for the global and the local planner you wish to use by typing in first one number for the global and then the second time the number for the local planner. Now the robot is ready to run with the chosen planner settings.

5. Start Visualization rviz

Finally, start the visualization execute in a new terminal

ROS_IP ROS_MASTER_URI

roslaunch auckbot_navigation rviz.launch

(Note: don't forget the initial step)

Additional Notes:

- You can also run all those steps only on the Control-PC, only use **source start.sh** instead of **source startonremote.sh** as initial step. Step 2 becomes obviously obsolete then.
- Remote control (Step 3) and Visualization (Step 5) are not necessary to use the robot and its path planning algorithms but recommended.
- Steps 3 and 4 can be switched.
- Always try to start up the robot on the same position and pose where the zero points on the map are (more information in Setting up a map) so it easily finds its localization.
- Additional information can be found on the ROS webpages (www.ros.org and www.ros.org/wiki).
- Two important files that save and update the parameters:
    - auckbot/eodwa_local_planner/cfg/EODWAPlanner.cfg
    - auckbot/auckbot_navigation/param/base_local_planner_params_eodwa.yaml

Appendix B. Instructions of Building Environment Map

1. Start up Auckbot

To start and run Auckbot please follow the steps in **Appendix C. Instructions of Bringing Up and Running AuckBot**, before moving on here. Don't start the Path Planning Algorithm though (Step 4). In the visualization choose base_link as Fixed Frame in the Global Options

2. Mapping

In a new terminal (never forget the initial steps) execute:

roslaunch auckbot_navigation gmapping.launch

The mapping process has now started. Use the keyboard controls to move the robot around in the workspace and check in the visualization the mapping progress. Once the map is finished use in a new terminal to save the map. It is now saved in two files (map.pgm and map.yaml) in the current directory.

rosrun map_server map_saver

3. Change name and move it

To change the name of the map, rename the two files and also replace "map" in the yaml file (open it in gedit) with the new name. Then move the two files into the following directory:

[your workspace]\src\auckbot\auckbot_navigation\maps

You also must change the launch file (respectively create a new one) for the *amcl_move_base* command. You find them in the following folder:

[your workspace]\src\auckbot\auckbot_navigation\launch

In the amcl_move_base_**XX**.launch file that you will use in step 4 in **Appendix C. Instructions of Bringing Up and Running AuckBot** (which is set in the setup_planner.sh file), you have to insert the file name of your new map, in the line where the map file is called. Now you can run the amcl_move_base_**XX**.launch file,

respectively start source the setup_planner.sh file where this command is already implemented.

Appendix C. More Experimental Results of Extended Dynamic Window Approach

There are in total three different navigation scenarios that are designed in these experiments. Five types of common test paths including straight-line forward path, straight-line sideways path, straight-line diagonal path, curve-line path and reactive obstacle avoidance path, are proposed to validate the proposed local trajectory planner that is based on the extended Dynamic Window Approach. More experimental results are presented in this section Appendix D.

- Scenario One: open ground without obstacles

  In the open ground without obstacles, from start to goal, it is always a straight primitive path. Depending on the initial robotic orientation and the final robotic orientation, the following primitive straight path segments are tested:

  - Straight-Line Forward
  - Straight-Line Sideways
  - Straight-Line Diagonal

- Scenario Two: ground with known obstacles

  When the map show obstacles existing in the way, the generated path will be a curve path on the fringe of the obstacle.

  - Curve-Line

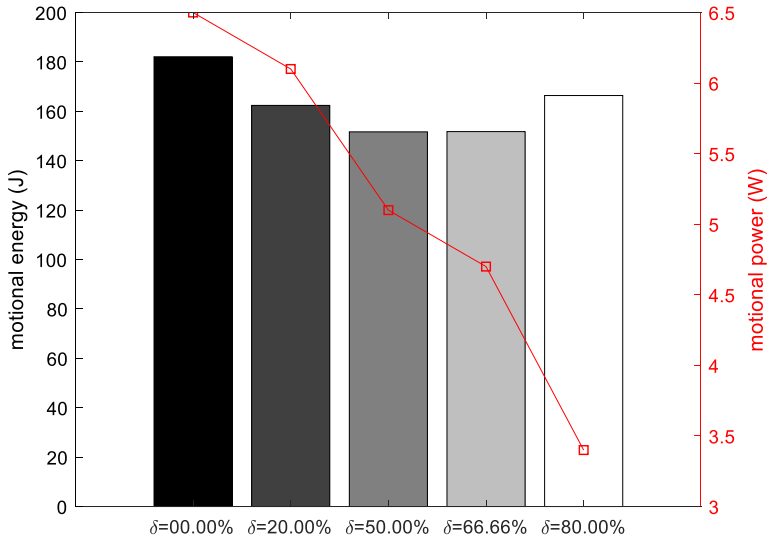
- Open ground with unforeseen obstacles

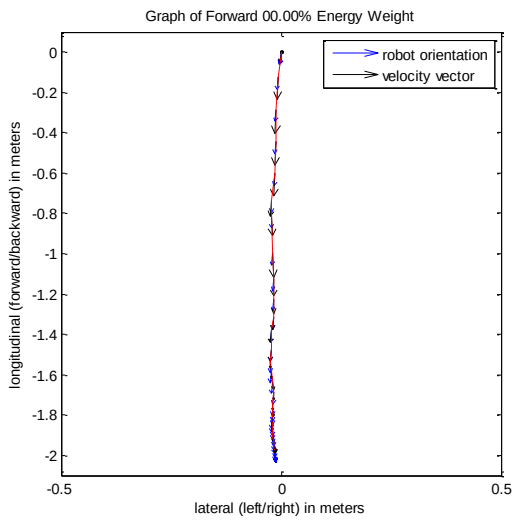
  There is an unforeseen obstacle existing along the path. When the local perception of the robot detects the unforeseen obstacle, both the global path planner and the local trajectory planner will reactively deviate from the obstacle.

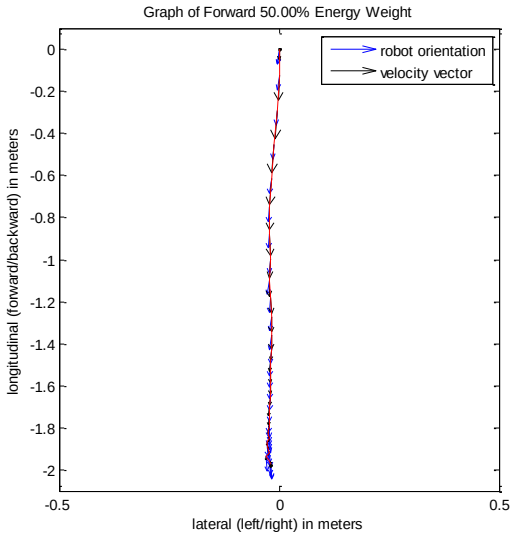
  - Reactive Obstacle Avoidance

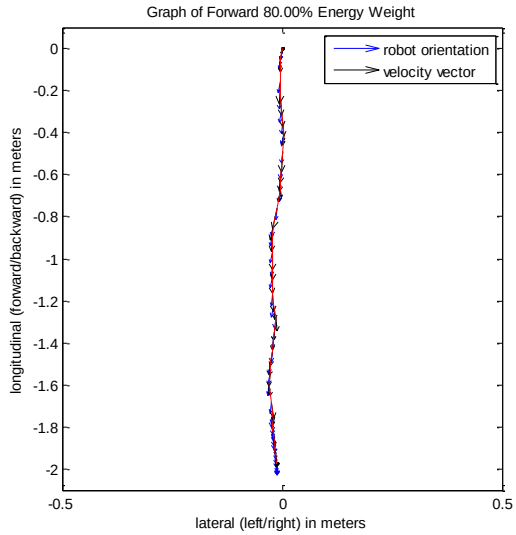
Straight-Line Forward

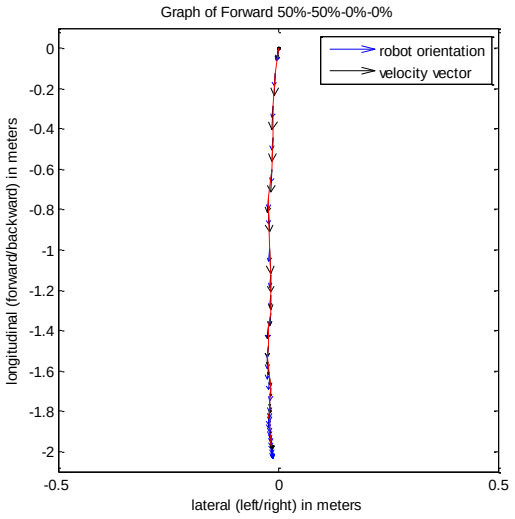
δ(%)	0.00%	20.00%	50.00%	66.66%	80.00%
		FORWARD			
task duration(s)	27.9	26.8	29.9	32.4	49.2
total energy(J)	2209.9	2110.2	2323.2	2510.7	3745.6
idle energy(J)	2027.8	1947.8	2171.5	2358.9	3579.2
motional energy(J)	182.1	162.4	151.7	151.8	166.4
total power(W)	79.2	78.7	77.7	77.5	76.1
idle power(W)	72.7	72.7	72.7	72.7	72.7
motional power(W)	6.5	6.1	5.1	4.7	3.4

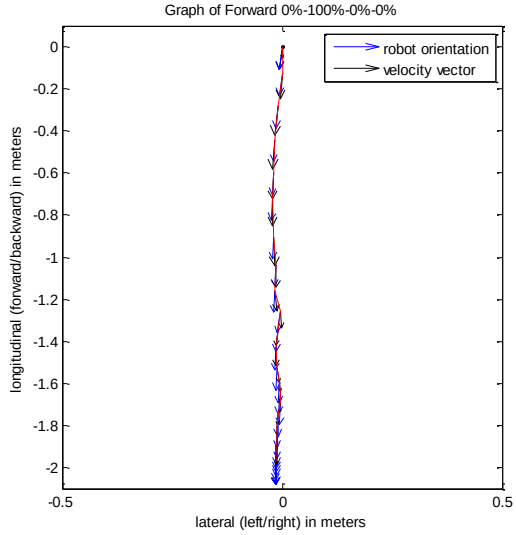


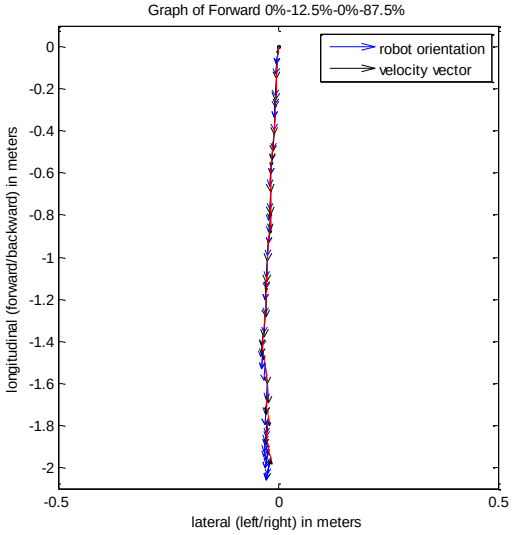


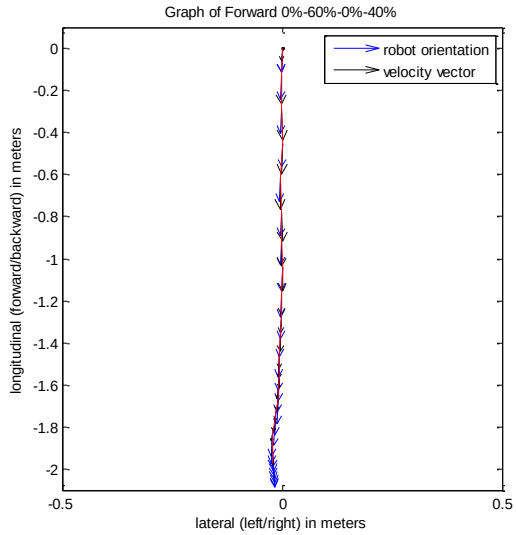



Graph of Forward 80.00% Energy Weight
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-0.5
0
0.5


Graph of Forward 50%-50%-0%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-0.5
0
0.5

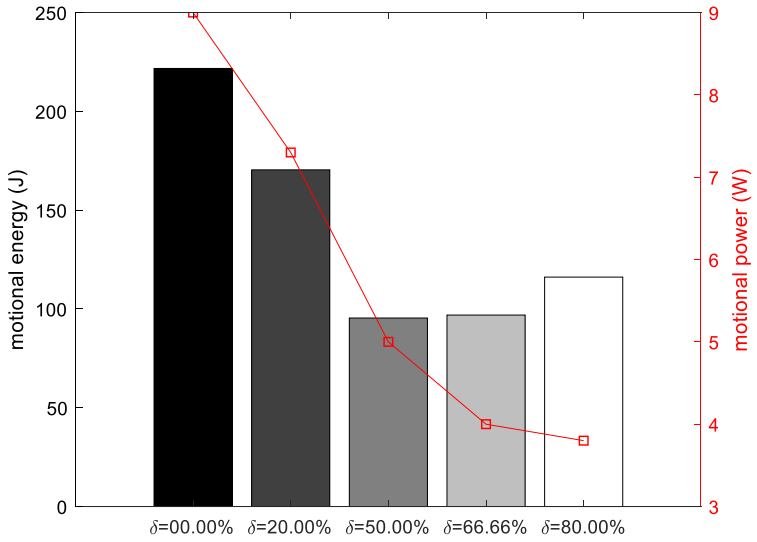

Graph of Forward 0%-100%-0%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-0.5
0
0.5

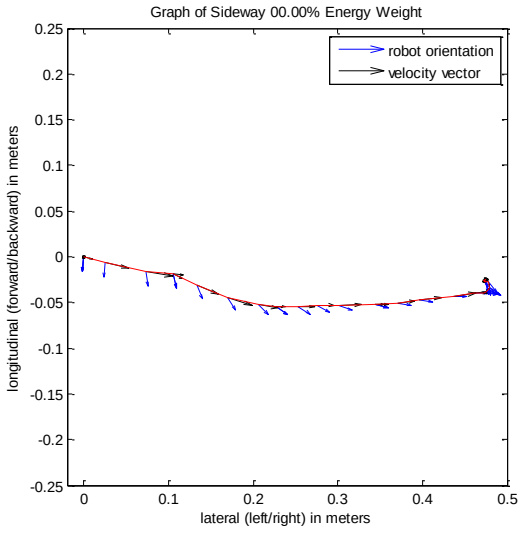

Graph of Forward 0%-12.5%-0%-87.5%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-0.5
0
0.5

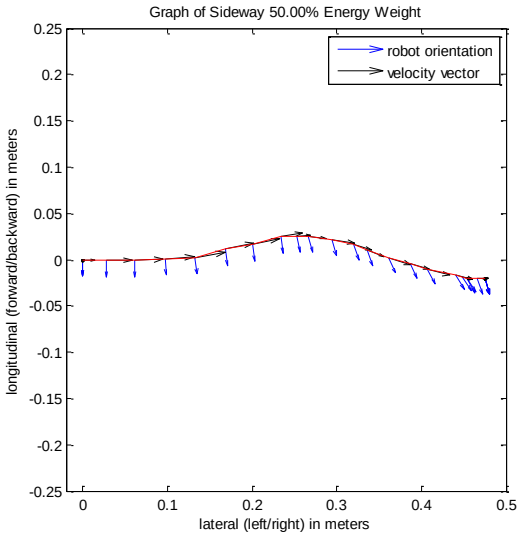

Graph of Forward 0%-60%-0%-40%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-0.5
0
0.5

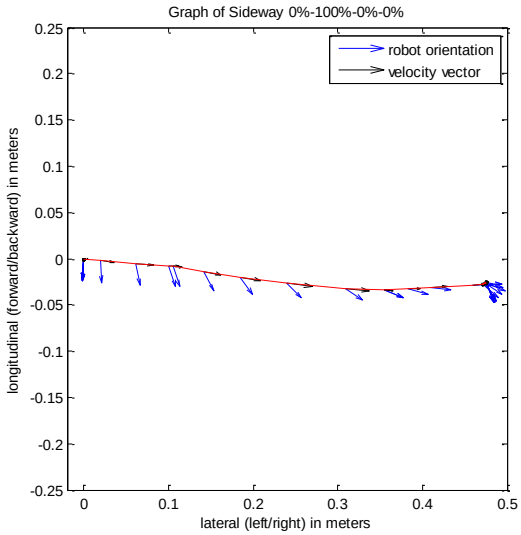
Straight-Line Sideways

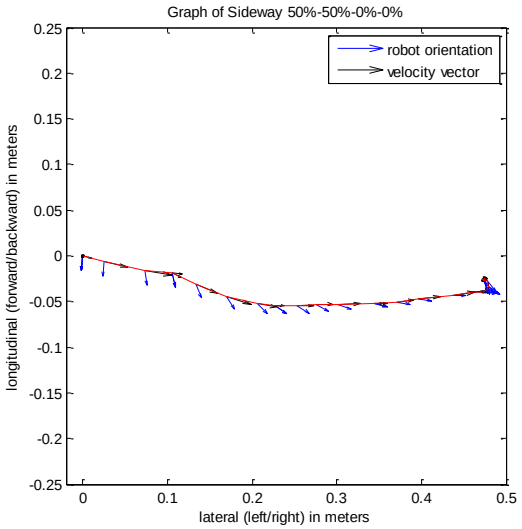
δ(%)	0.00%	20.00%	50.00%	66.66%	80.00%
			SIDEWAY		
task duration(s)	24.5	23.2	19.2	24.5	30.2
total energy(J)	2004.8	1860.7	1490.1	1879.8	2310.2
idle energy(J)	1783.1	1690.3	1394.7	1782.9	2194.1
motional energy(J)	221.7	170.4	95.4	96.9	116.1
total power(W)	81.8	80.2	77.6	76.7	76.5
idle power(W)	72.7	72.7	72.7	72.7	72.7
motional power(W)	9.0	7.3	5.0	4.0	3.8

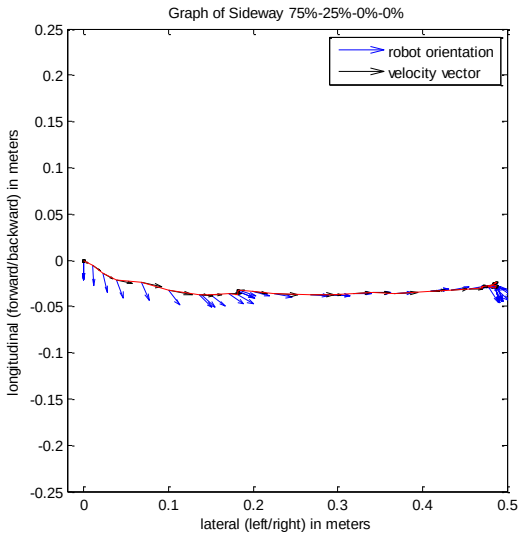


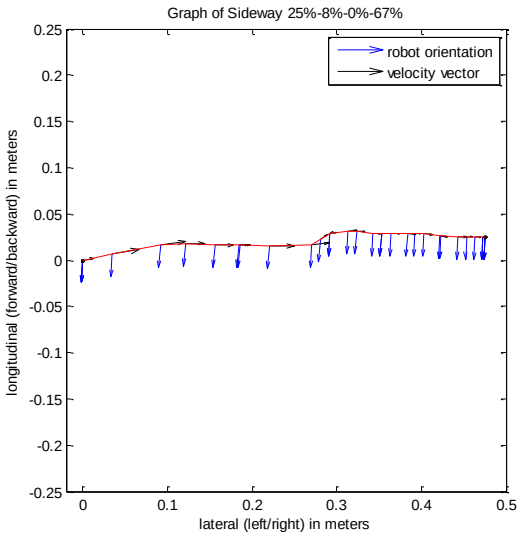


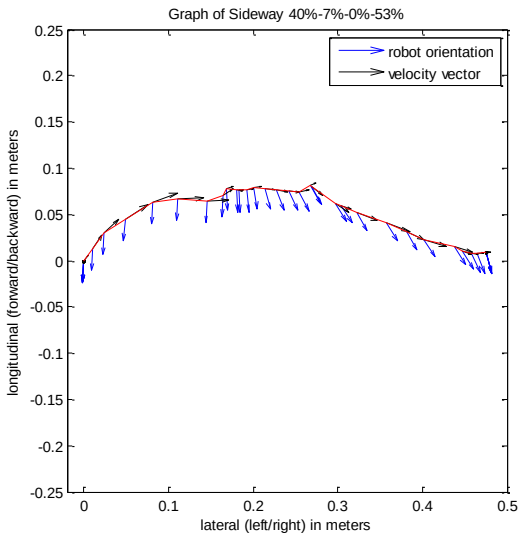


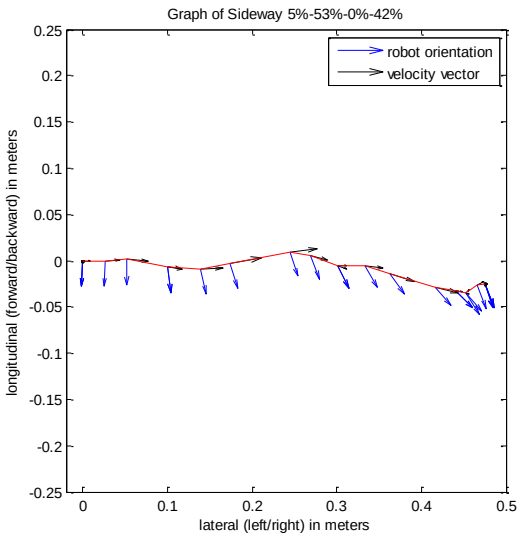

Graph of Sideway 0%-100%-0%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters


Graph of Sideway 50%-50%-0%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters


Graph of Sideway 75%-25%-0%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

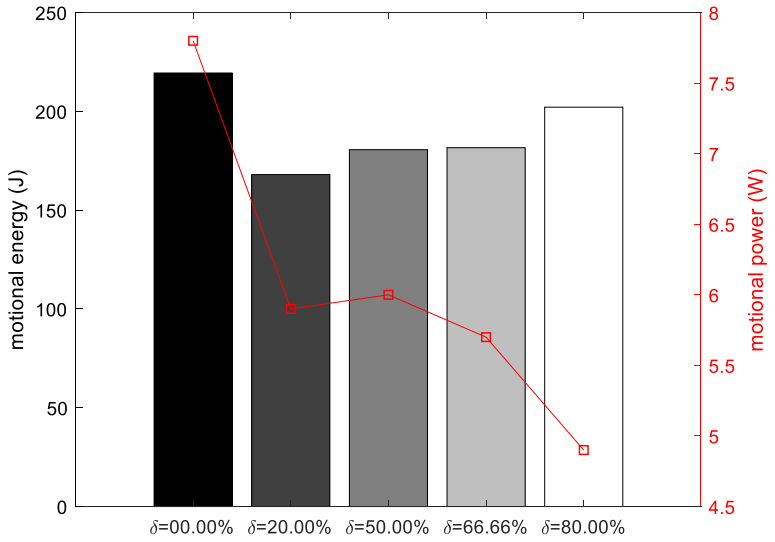

Graph of Sideway 25%-8%-0%-67%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

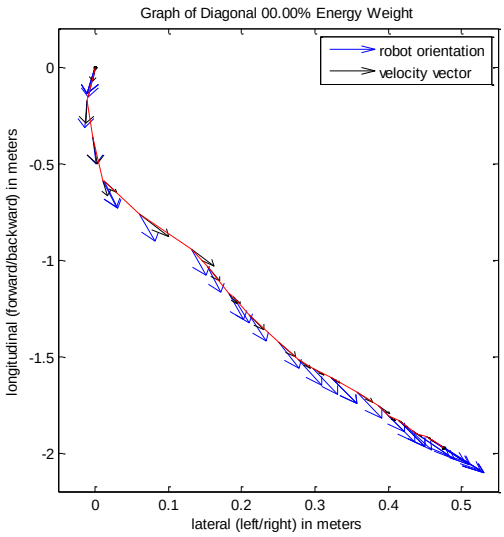

Graph of Sideway 40%-7%-0%-53%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

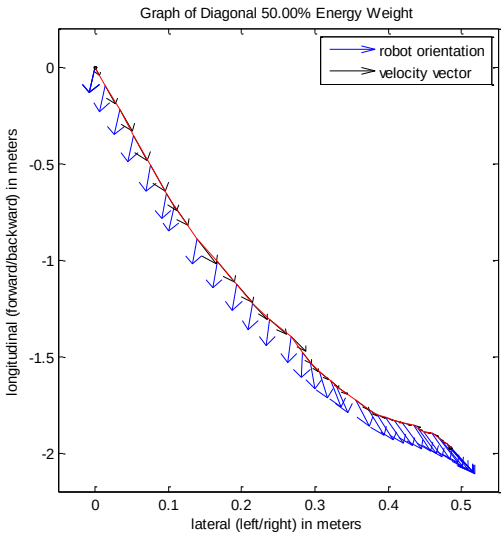

Graph of Sideway 5%-53%-0%-42%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

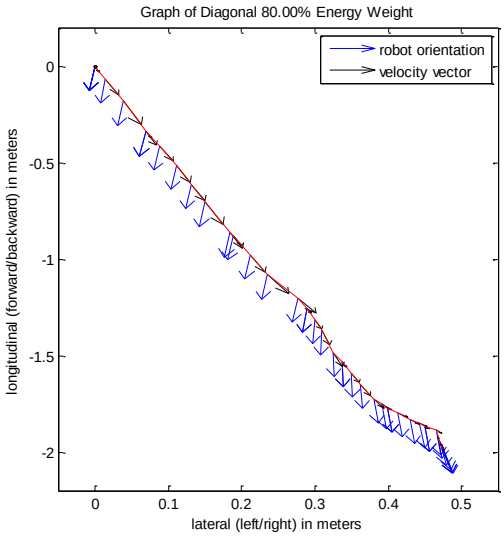
Straight-Line Diagonal

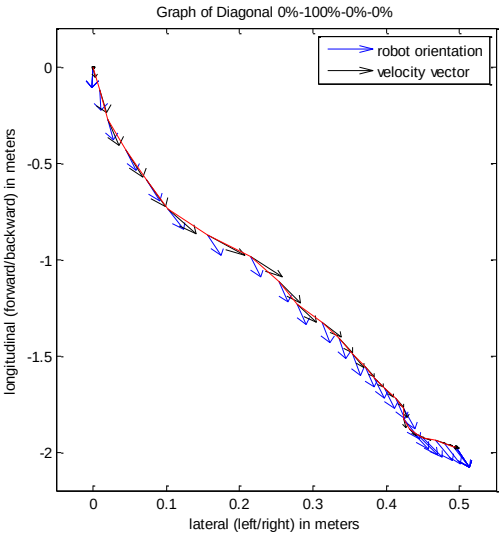
δ(%)	0.00%	20.00%	50.00%	66.66%	80.00%
		DIAGONAL			
task duration(s)	28.2	28.6	30.2	31.6	40.9
total energy(J)	2271.2	2250.5	2380.3	2481.8	3179.2
idle energy(J)	2051.8	2082.5	2199.7	2300.2	2977.1
motional energy(J)	219.4	168.0	180.6	181.6	202.1
total power(W)	80.5	78.7	78.8	78.5	77.7
idle power(W)	72.7	72.7	72.7	72.7	72.7
motional power(W)	7.8	5.9	6.0	5.7	4.9

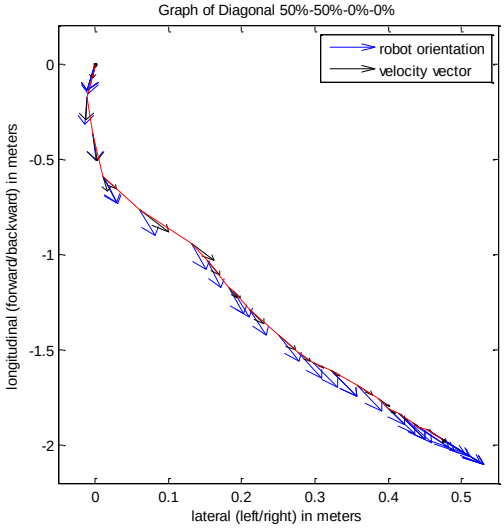


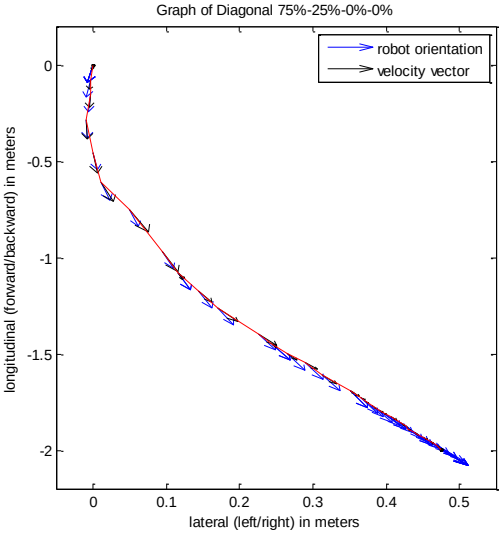


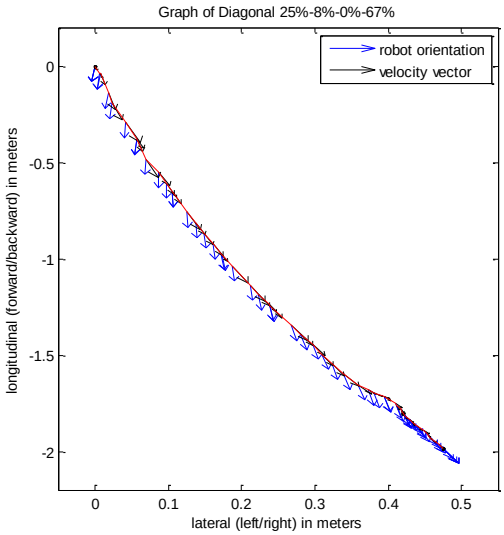


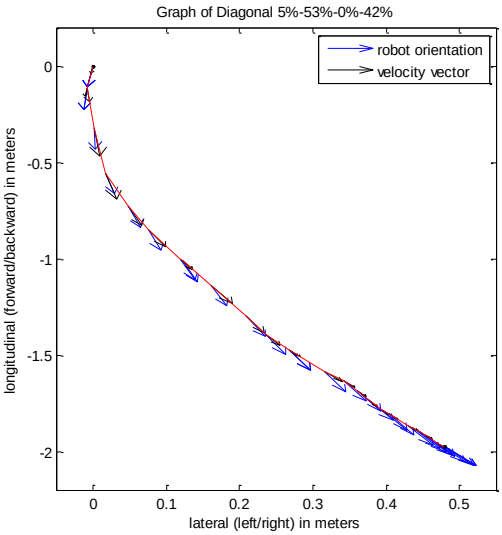

Graph of Diagonal 80.00% Energy Weight
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters


Graph of Diagonal 0%-100%-0%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters


Graph of Diagonal 50%-50%-0%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

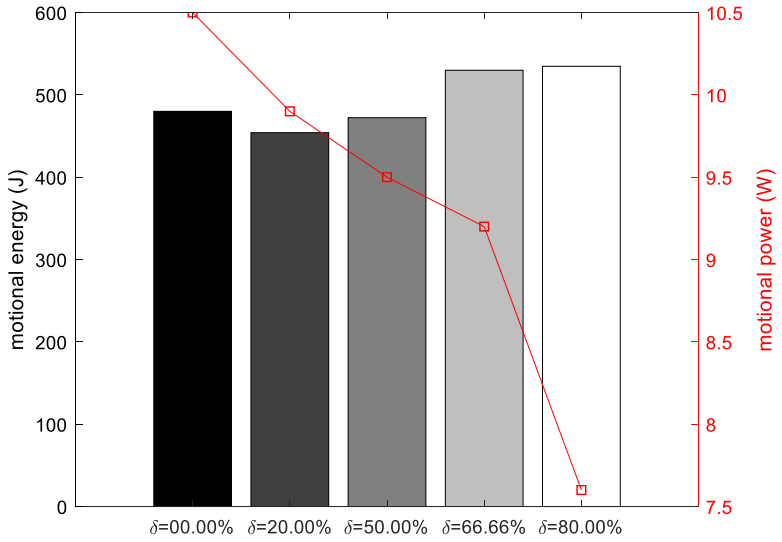

Graph of Diagonal 75%-25%-0%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

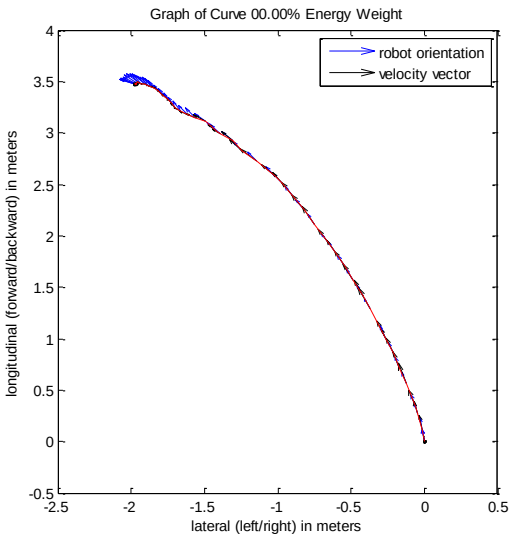

Graph of Diagonal 25%-8%-0%-67%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

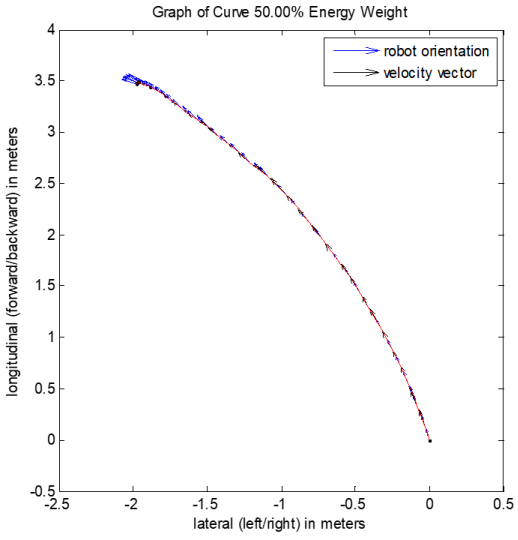

Graph of Diagonal 5%-53%-0%-42%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

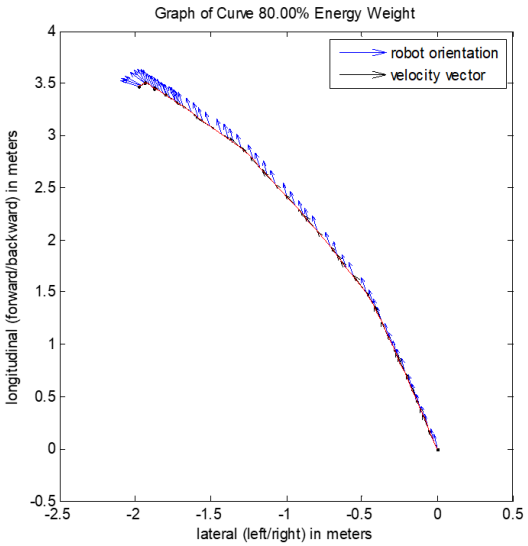
Curve-Line

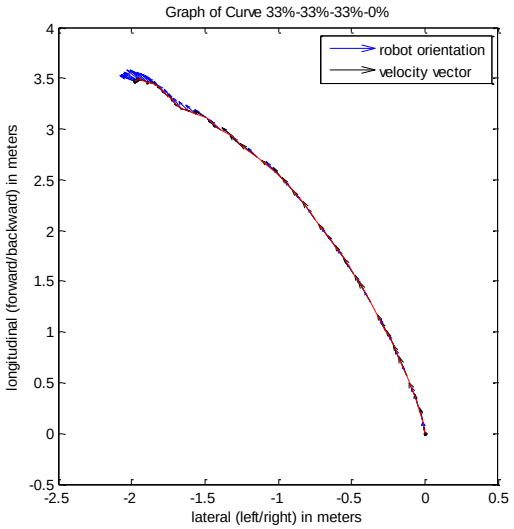
δ(%)	0.00%	20.00%	50.00%	66.66%	80.00%
		CURVE			
task duration(s)	45.8	46.1	49.7	57.4	70.7
total energy(J)	3813.4	3803.8	4084.8	4701.6	5676.0
idle energy(J)	3333.4	3349.7	3612.5	4171.7	5141.3
motional energy(J)	480.0	454.1	472.3	529.9	534.7
total power(W)	83.3	82.5	82.2	81.9	80.3
idle power(W)	72.7	72.7	72.7	72.7	72.7
motional power(W)	10.5	9.9	9.5	9.2	7.6

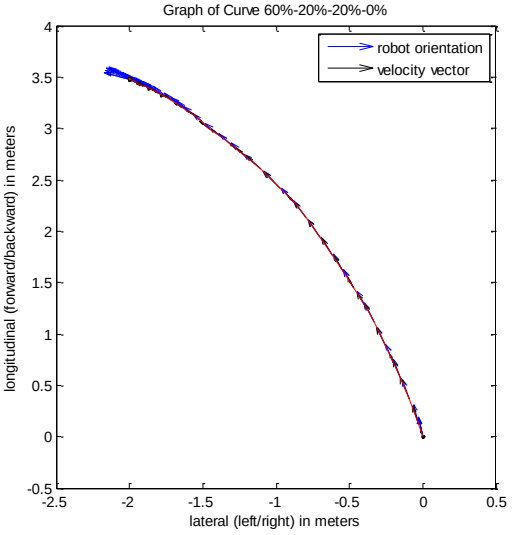


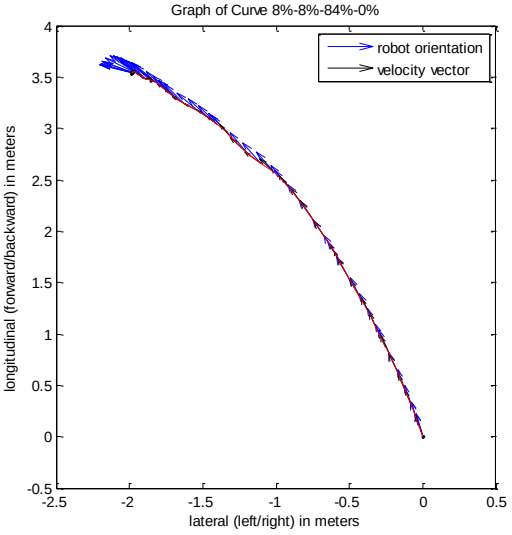


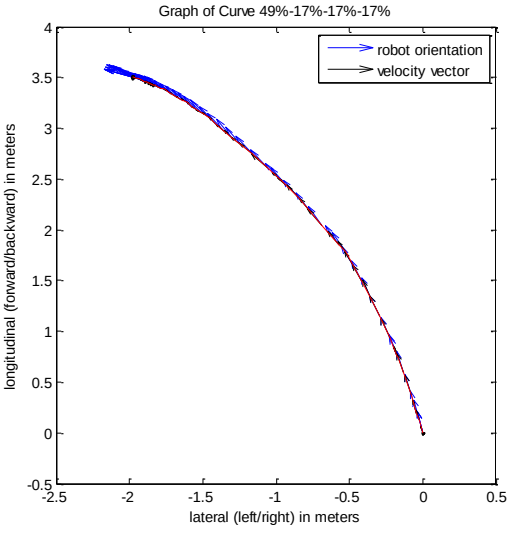


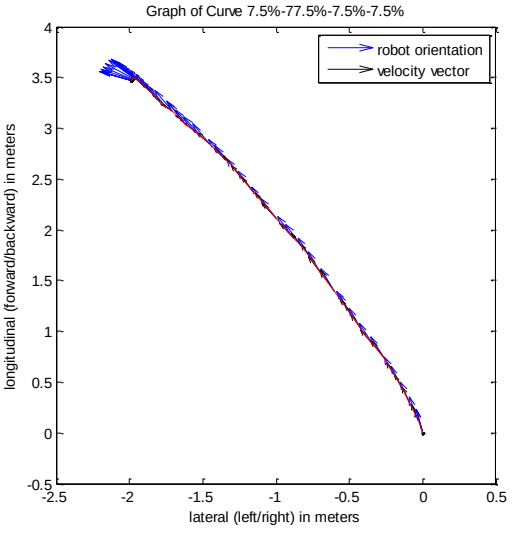

Graph of Curve 80.00% Energy Weight
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

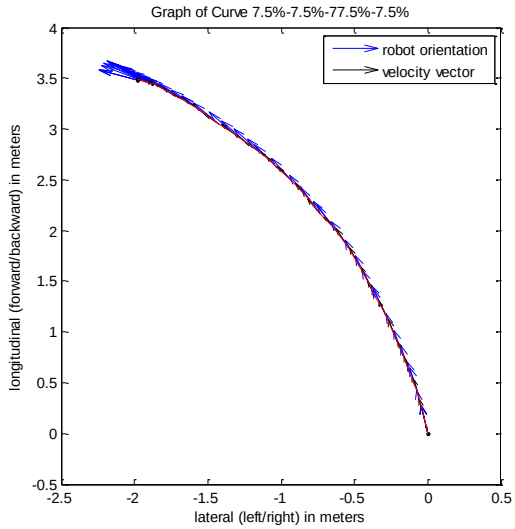

Graph of Curve 33%-33%-33%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

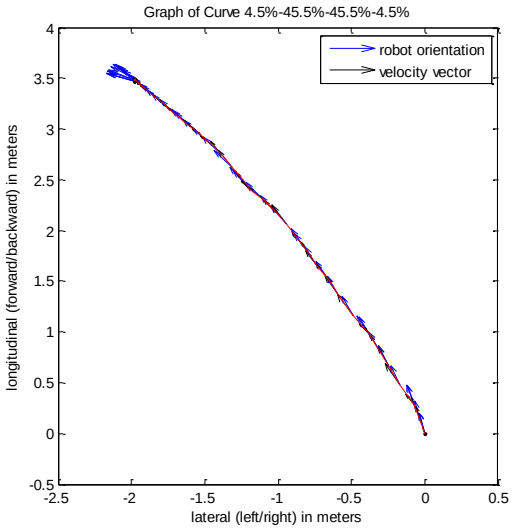

Graph of Curve 60%-20%-20%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters


Graph of Curve 8%-8%-84%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters


Graph of Curve 49%-17%-17%-17%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

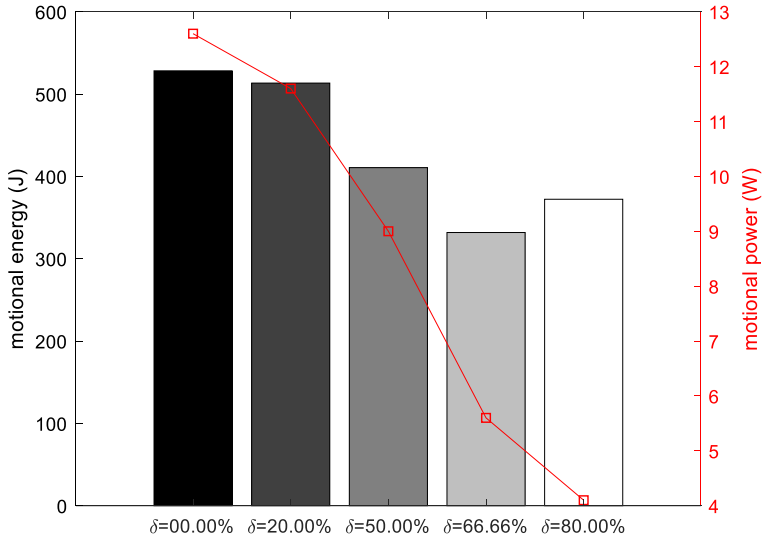

Graph of Curve 7.5%-77.5%-7.5%-7.5%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

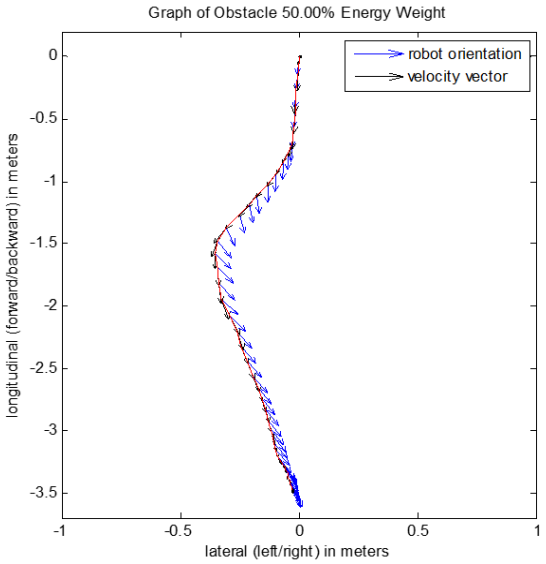


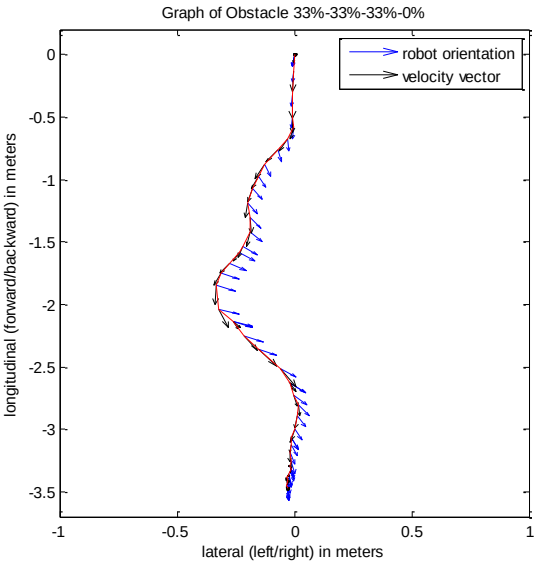


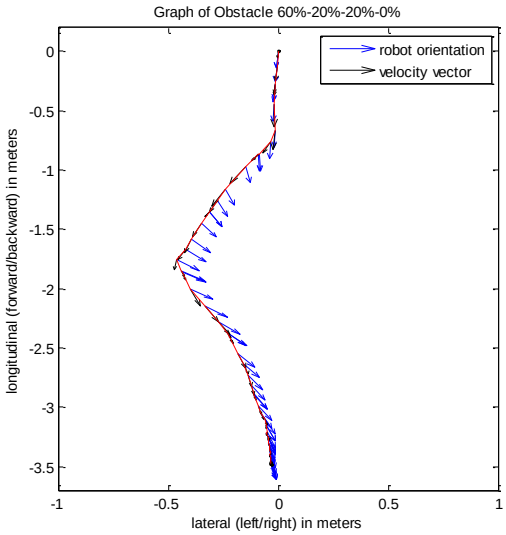
Reactive Obstacle Avoidance

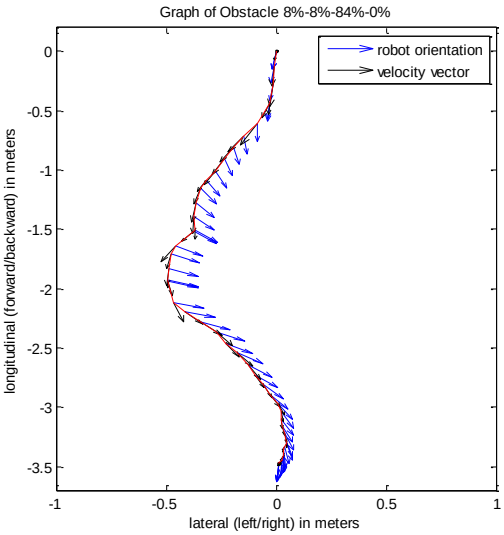
δ(%)	0.00%	20.00%	50.00%	66.66%	80.00%
		OBSTACLE			
task duration(s)	41.8	44.4	45.6	59.1	90.8
total energy(J)	3570.7	3740.2	3728.7	4631.0	6974.7
idle energy(J)	3042.5	3226.9	3318.1	4299.2	6602.5
motional energy(J)	528.2	513.3	410.6	331.8	372.2
total power(W)	85.4	84.2	81.8	78.4	76.8
idle power(W)	72.7	72.7	72.7	72.7	72.7
Motional power(W)	12.6	11.6	9.0	5.6	4.1

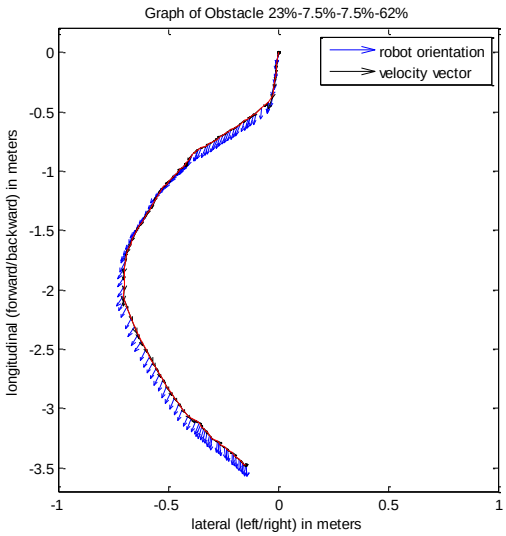


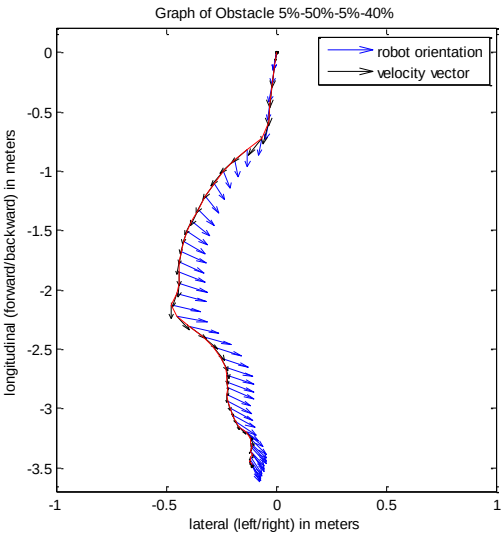

Graph of Obstacle 50.00% Energy Weight
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

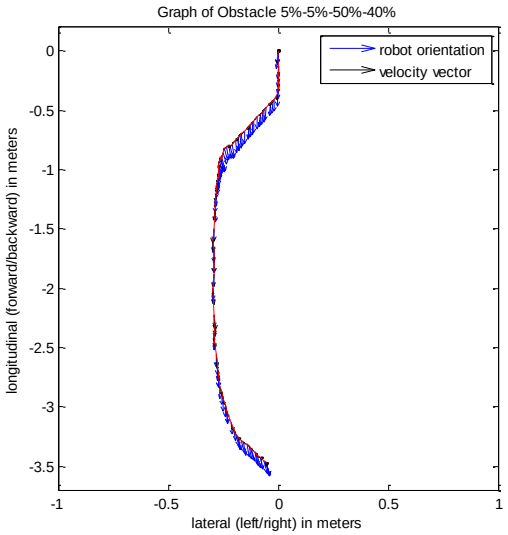

Graph of Obstacle 33%-33%-33%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters

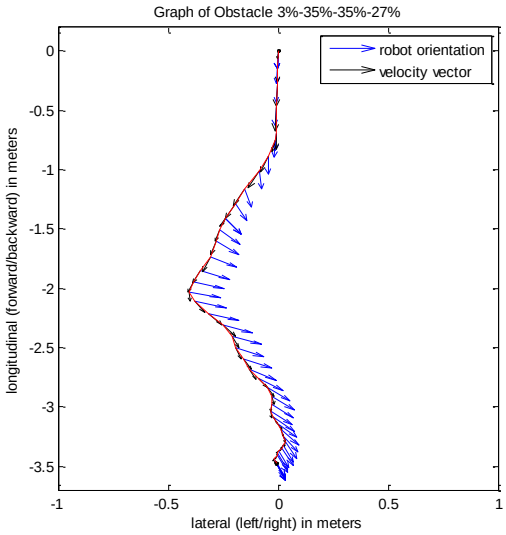

Graph of Obstacle 60%-20%-20%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters


Graph of Obstacle 8%-8%-84%-0%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters


Graph of Obstacle 23%-7.5%-7.5%-62%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters


Graph of Obstacle 5%-50%-5%-40%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters


Graph of Obstacle 5%-5%-50%-40%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-1
-0.5
0
0.5
1


Graph of Obstacle 3%-35%-35%-27%
robot orientation
velocity vector
longitudinal (forward/backward) in meters
lateral (left/right) in meters
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-1
-0.5
0
0.5
1

Appendix D. More Experimental Results of Energy-Optimal Polynomial Trajectory Generation

Obstacle Avoidance at Static Environments

When there are obstacles existing in the environment, the map stores the locations of the obstacles. As this is an offline planning problem, all environment information should be known. For these locations that are occupied by the obstacles, a simple elimination program that illegalise the trajectories invading the obstacle regions can be put into the cost function. A case is shown below:

$(x_0^*, y_0^*, \varphi_0^*) = (0,0,0)$.

$(x_1^*, y_1^*, \varphi_1^*)$ decision variables.

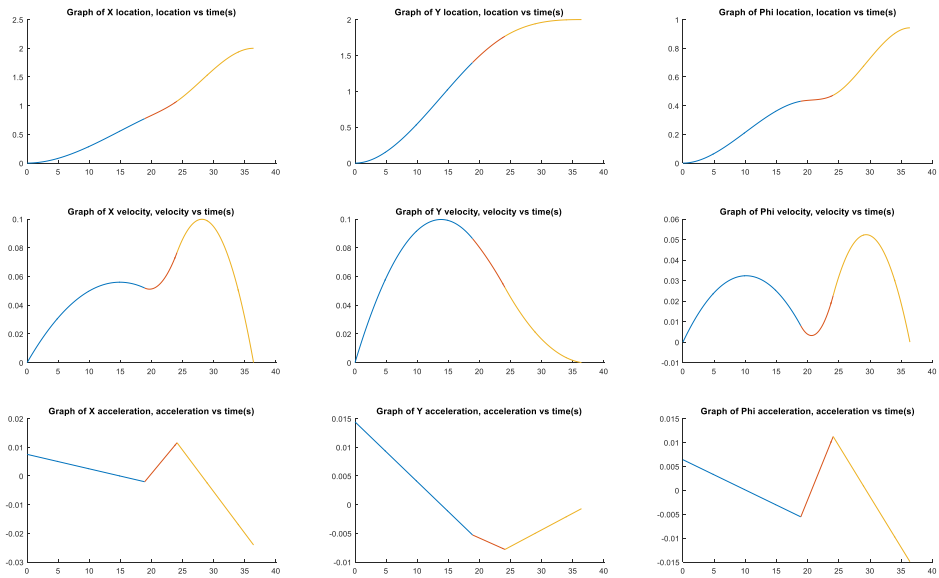
$(x_2^*, y_2^*, \varphi_2^*)$ decision variables.

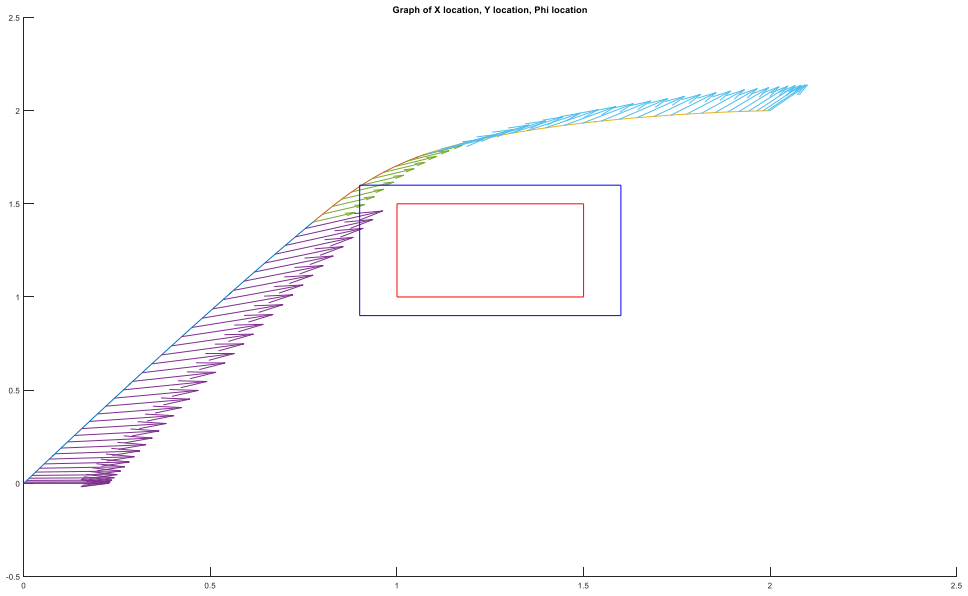
$(x_0^*, y_0^*, \varphi_0^*) = (2,2,-0.3\pi)$.

Optimal delta time $\Delta \boldsymbol{t} = [18.9\ 5.2\ 12.3]\ s$:

Optimal via points: [0.775, 1.402, 0.431], [1.079, 1.768, 0.473]

Optimal energy: 3019 J





More Comparison Simulations

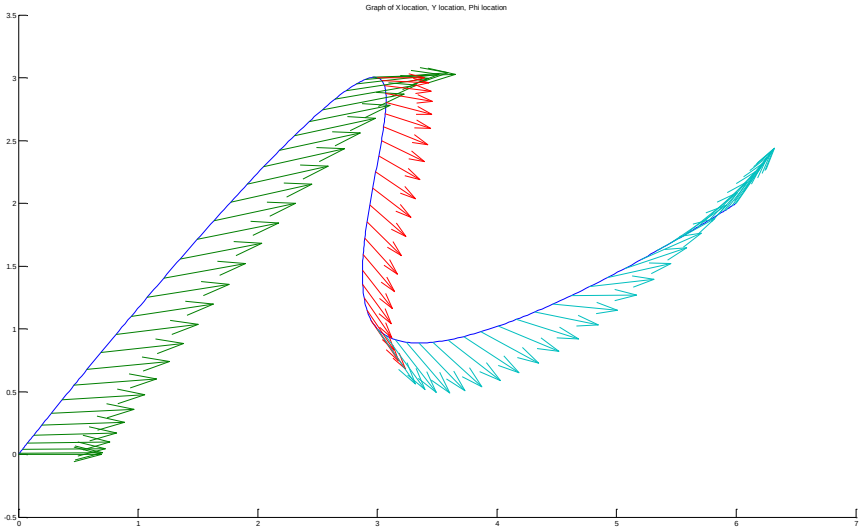
There are three time intervals $\Delta \boldsymbol{t}$ as decision variables. Stationary at initial final states - zero initial and final velocity. The via points are $(x_0^*, y_0^*, \varphi_0^*) = (0,0,0)$, $(x_1^*, y_1^*, \varphi_1^*) = (3,3,0)$, $(x_2^*, y_2^*, \varphi_2^*) = (3,1,0.3\pi)$ and $(x_0^*, y_0^*, \varphi_0^*) = (6,2,-0.3\pi)$.

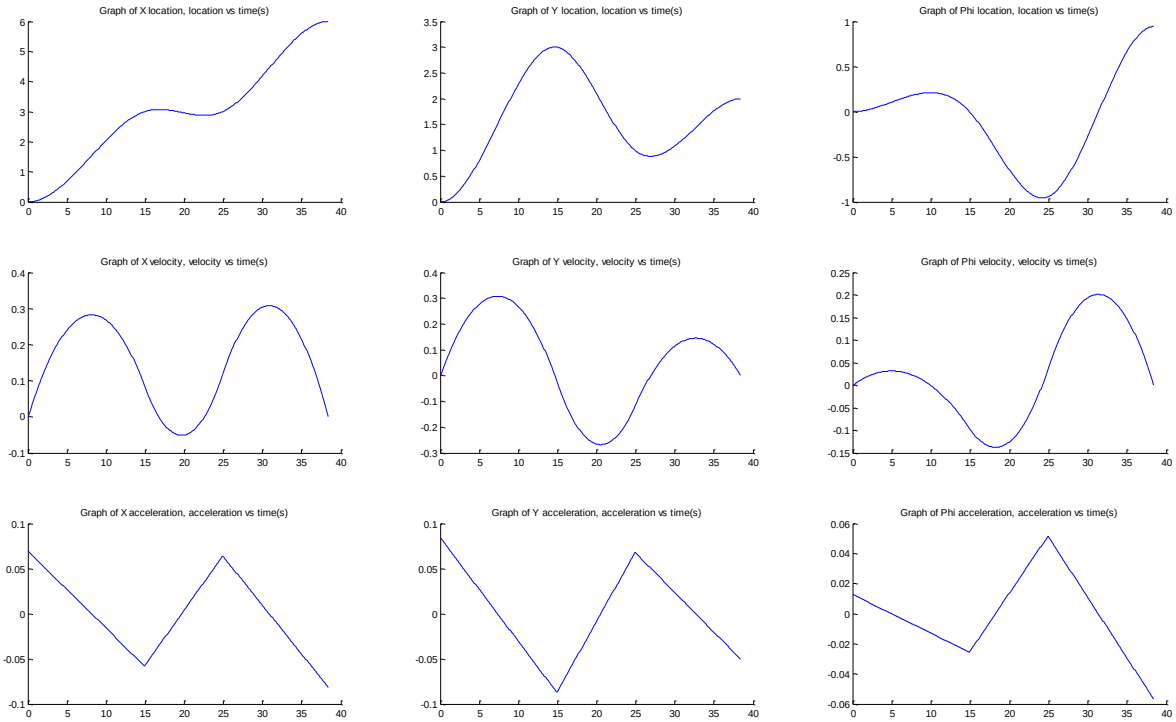
1. Energy optimization

Optimal time intervals $\Delta \boldsymbol{t} = [14.9\ 10.0\ 13.5]\ s = 38.4\ s$

Path length 9.63m, 1.04π.

Energy consumption: 7335 J.



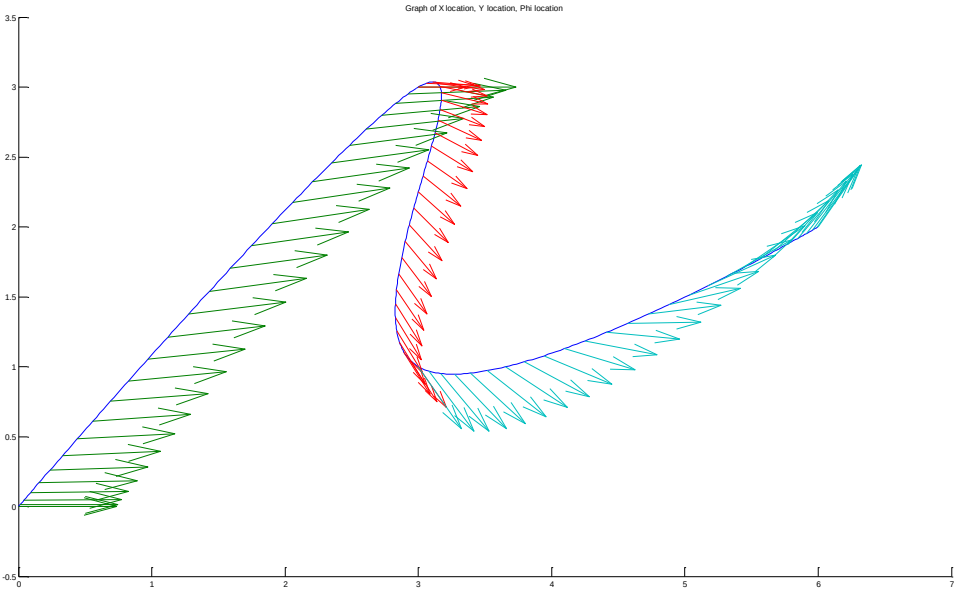


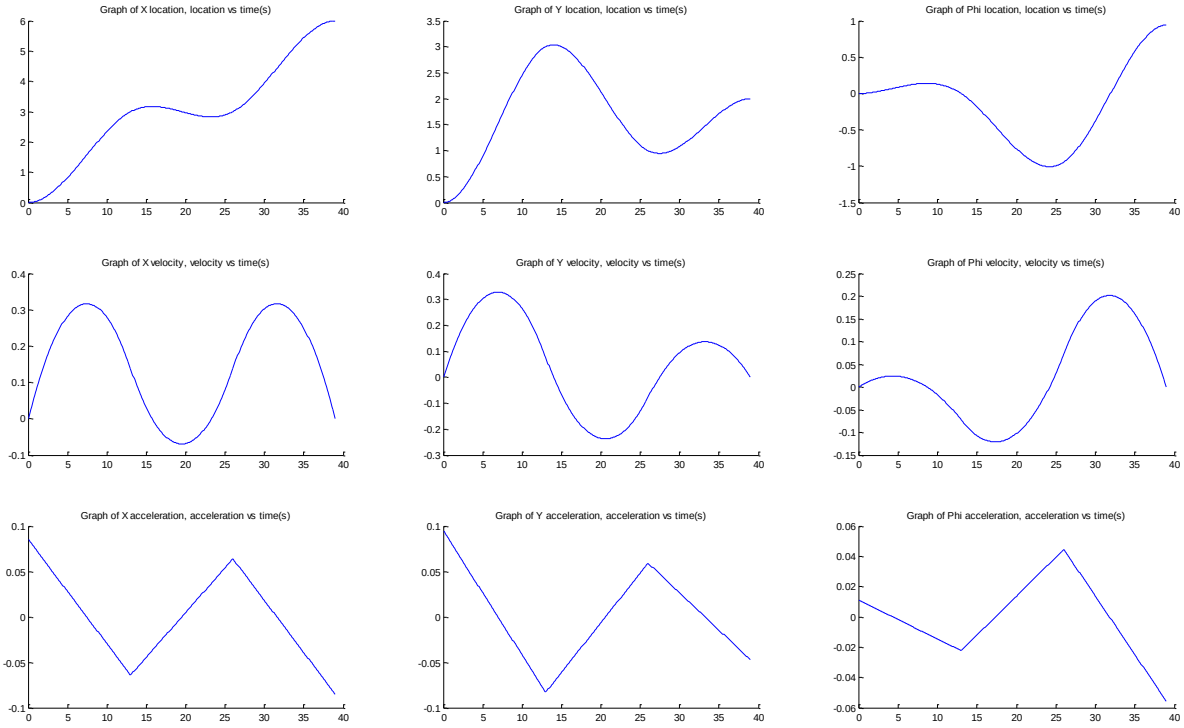
2. Additional constraint added into case 1 – equal time intervals

Optimal time intervals $\Delta \boldsymbol{t} = [13.0\ 13.0\ 13.0]\ s = 39.0\ s$

Path length 9.76m, 1.04π.

Energy consumption: 7500 J.



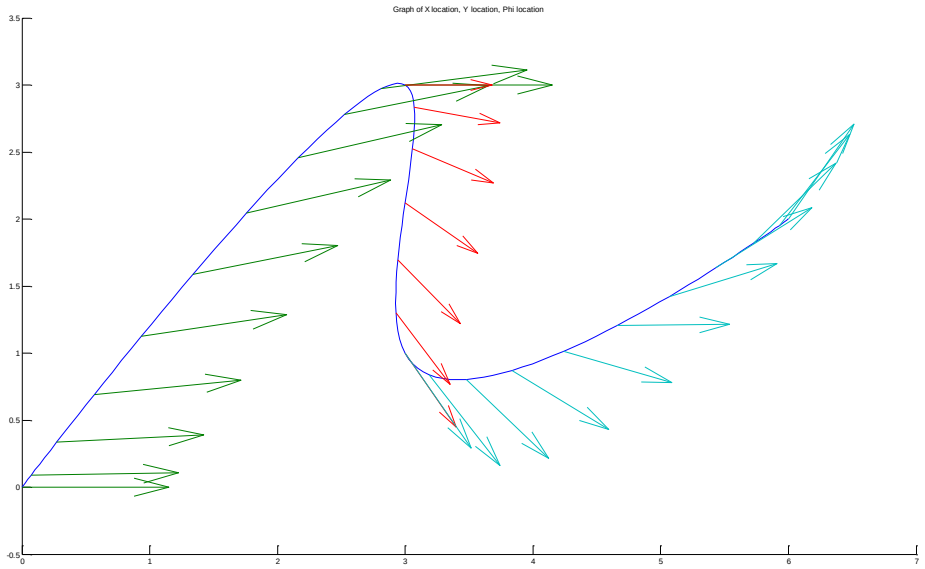


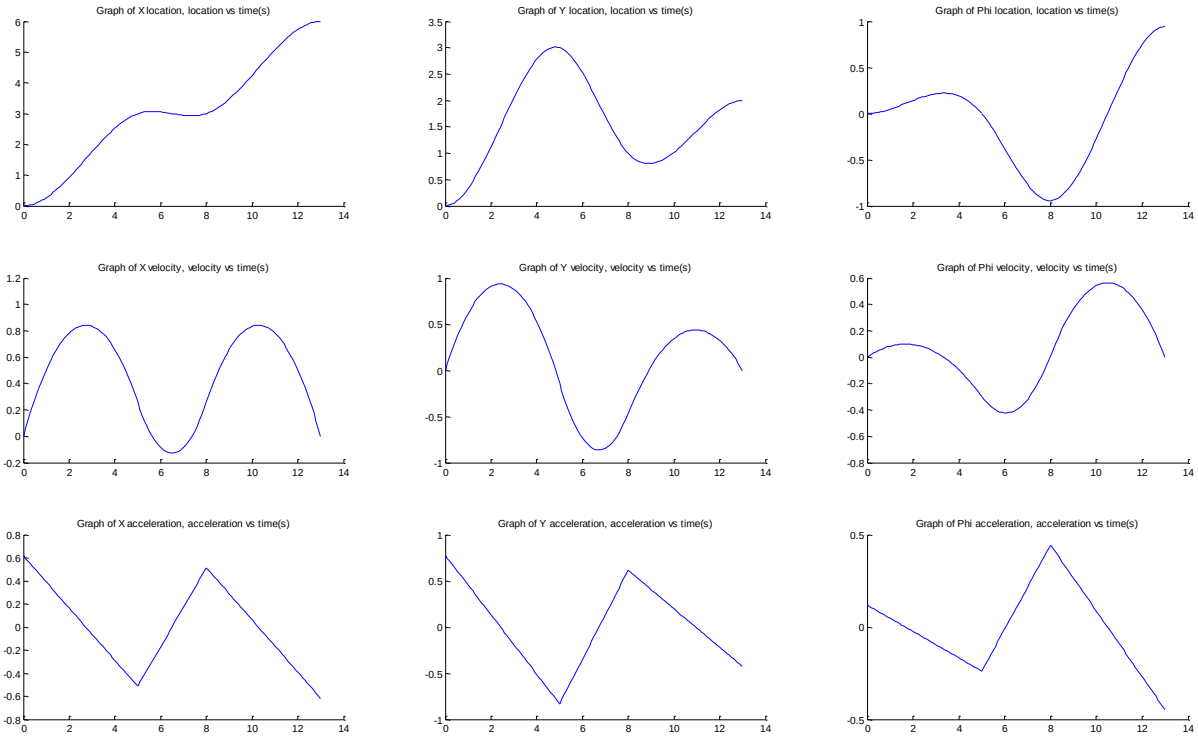
3. Time optimization (velocity limit 1 m/s and 2 rad/s, acceleration limit 5 m/s^2 and 5 rad/s^2)

Optimal time intervals $\Delta \boldsymbol{t} = [5.0\ 3.0\ 5.0]\ s = 13.0\ s$

Path length 9.71m, 1.04π.

Energy consumption: 11529 J.



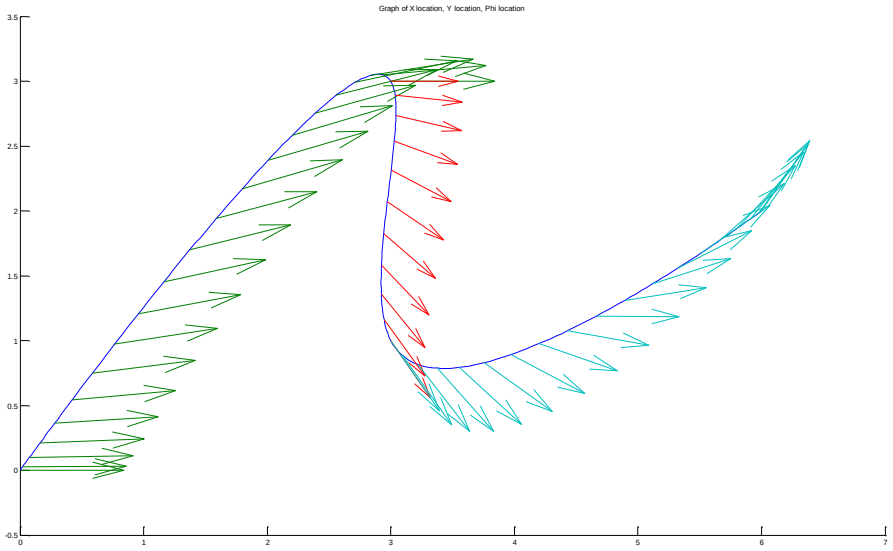


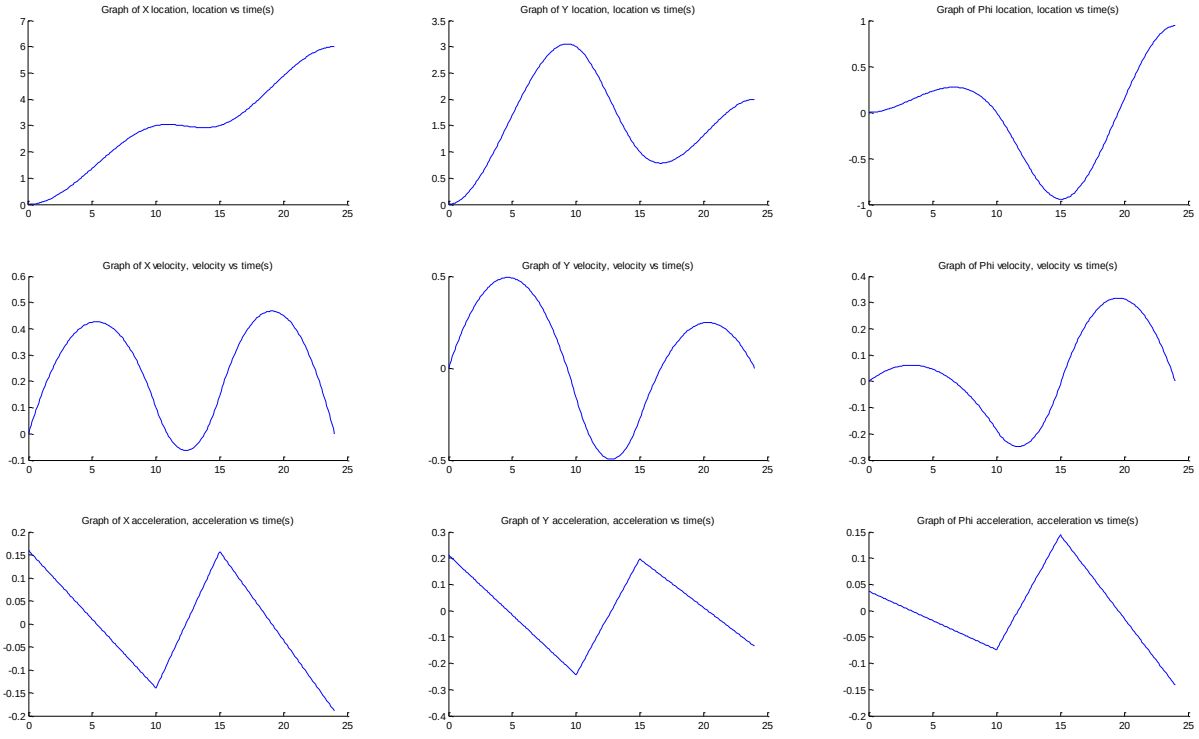
3. Time optimization (velocity limit 0.5 m/s and 1 rad/s, acceleration limit 2.5 m/s^2 and 2.5 rad/s^2)

Optimal time intervals $\Delta \boldsymbol{t} = [10.0\ 5.0\ 9.0]\ s = 24.0\ s$

Path length 9.77m, 1.07π.

Energy consumption: 8173 J.



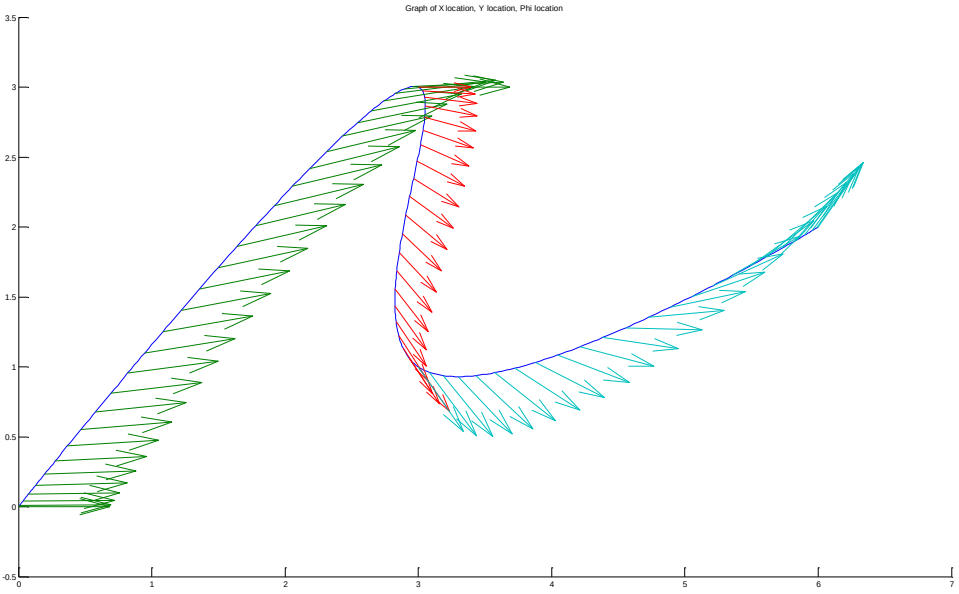


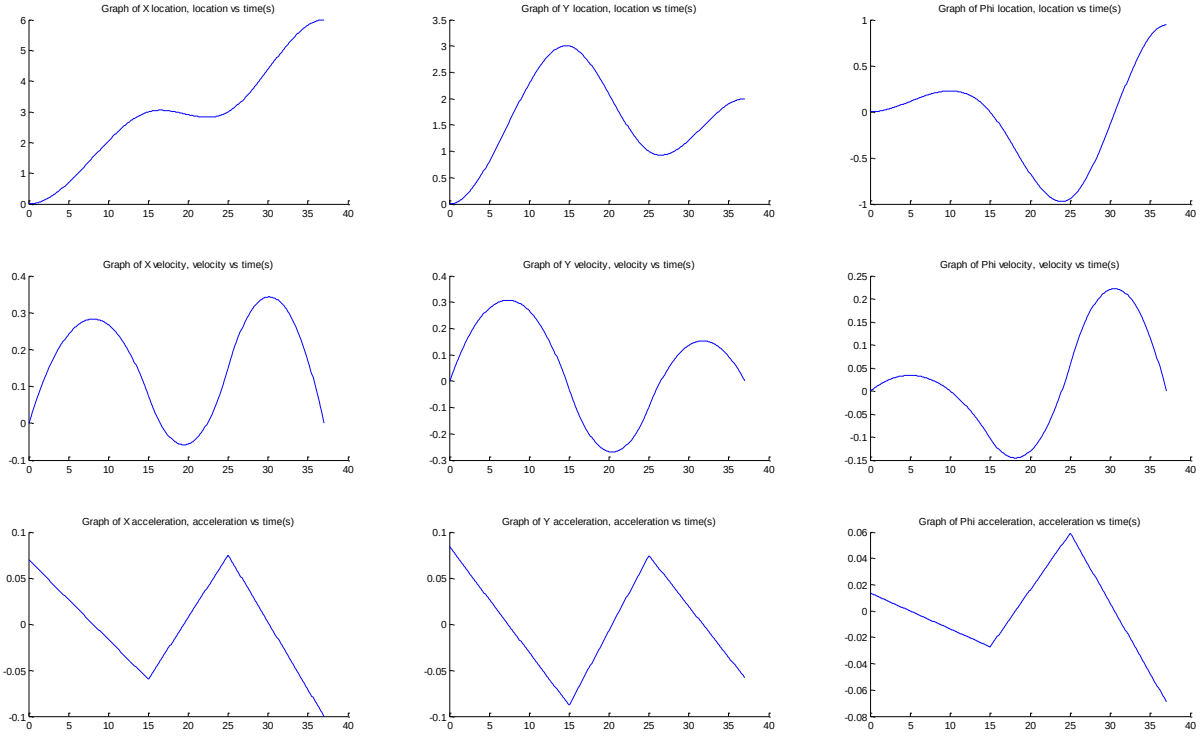
4. Shortest path length

Optimal time intervals $\Delta \boldsymbol{t} = [15.1\ 10.3\ 12.1]\ s = 37.5\ s$

Path length 9.61m, 1.03π.

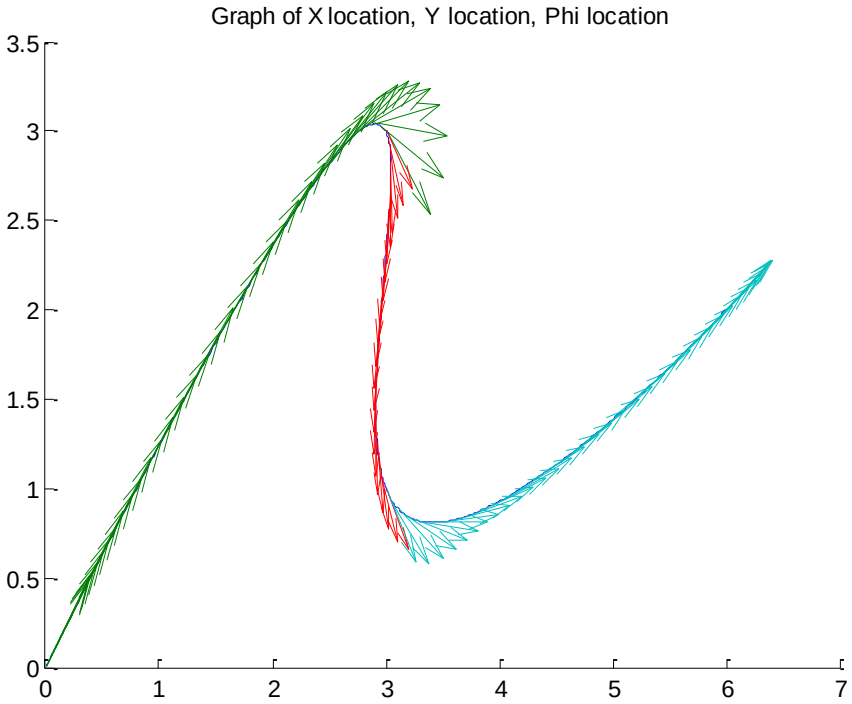
Energy consumption: 7358 J.

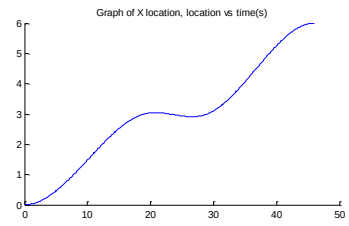


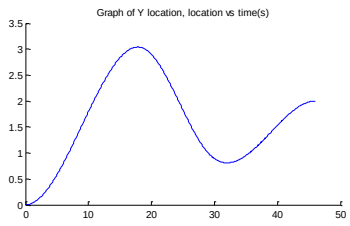


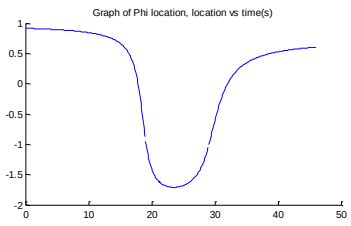
5. Nonholonomic constraints – The trajectory is based on differential drive, instead of omnidirectional drive

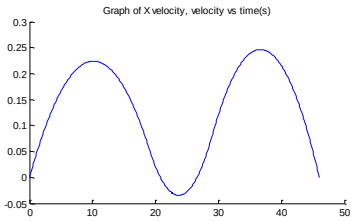
Energy consumption: 8983 J

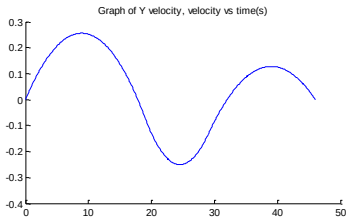

Graph of X location, Y location, Phi location

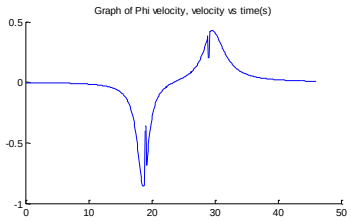

Graph of X location, location vs time(s)

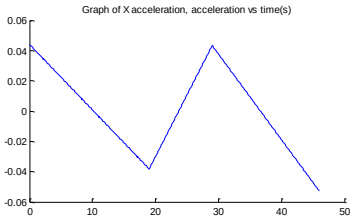

Graph of Y location, location vs time(s)

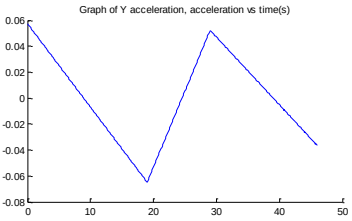

Graph of Phi location, location vs time(s)

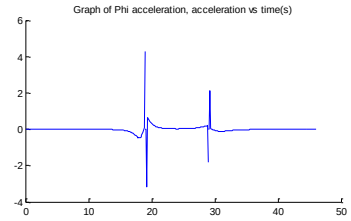

Graph of X velocity, velocity vs time(s)


Graph of Y velocity, velocity vs time(s)


Graph of Phi velocity, velocity vs time(s)


Graph of X acceleration, acceleration vs time(s)


Graph of Y acceleration, acceleration vs time(s)


Graph of Phi acceleration, acceleration vs time(s)