Control of Mecanum Wheels in Arbitrary Position, Angle and Number for Complete Omni-Directional Motion

by

James D. Morin

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE IN MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by: James D. Morin
Department of Mechanical Engineering
May 10, 2024

Certified by: Dr. Harrison H. Chin
Instructor/Lecturer of Mechanical Engineering, Thesis Supervisor

Accepted by: Dr. Kenneth Karmin
Associate Professor of Mechanical Engineering
Undergraduate Officer

Control of Mecanum Wheels in Arbitrary Position, Angle and Number for Complete Omni-Directional Motion

by

James D. Morin

Submitted to the Department of Mechanical Engineering
on May 10, 2024 in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE IN MECHANICAL ENGINEERING

ABSTRACT

Mecanum wheels are often used in vehicle/robotics applications where agility is a top priority, since they can be used to translate the vehicle in any direction on the ground as well as rotate around any center point. However, the standard placement and orientation of these wheels is quite rigidly adhered to in common practice, causing little to no flexibility and variation in this aspect of the vehicle's design. This thesis shows how the use of mecanum wheels need not fall under such constraints that are currently so widespread. Design considerations for arbitrary arrangements of mecanum wheels are presented, as well as some motivations for deviating from standard configurations. A system for controlling a vehicle with an unconventional arrangement of mecanum wheels is developed, proving that the wheel's rotational axes and relative placement need not align in any specific way. Furthermore, equations are derived and presented that permit for the control of any number of mecanum wheels in any arbitrary configuration on the vehicle. This control is possible for the user simply by measuring a few key details about the vehicle's wheels: for each wheel, the location of the wheel in the vehicle's coordinate frame, the positive-signal direction of the motors, and the angle of the wheel's diagonal rollers. To prove the results, a random arrangement of wheels is generated and assembled on a robot, which is successfully controlled using the framework described in this thesis. Also, an example is given of a robot which uses the methodology described to compete very successfully in an MIT robotics class competition.

Thesis supervisor: Dr. Harrison H. Chin
Title: Instructor/Lecturer of Mechanical Engineering

Acknowledgments

Thank you to my advisor Harrison Chin for his guidance and support in the completion of this thesis. Thank you to Joseph Ntaimo for offering the use of his electronic designs and software for robotics using a micro controller. Thank you also to Jinger Chong and Josh Sohn for providing the framework of DC motor control and encoder reading upon which I built my project.

Biographical Sketch

James Morin was born in Fort Campbell, Kentucky on July 30, 2001. He received his Bachelor's of Science in Mechanical Engineering with a Minor in Computer Science from MIT in 2024. He has done research in MIT's Nanophotonics and 3D Manufacturing Laboratory and Molecular Machines Laboratory, and has worked in R&D at Lenox Saw. Following graduation, he will contribute to revolutionizing electronics prototyping at Adom Industries LLC.

Contents

List of Figures

Chapter 1

Introduction

1.1 Mecanum Wheels

A mecanum wheel[1] is a type of wheel that has small rollers diagonally placed along the circumference of the wheel to enable onmi-directional motion. Since these rollers are passive and contact the ground at a specific angle, a mecanum wheel can be freely pushed at that angle relative to the axis of the wheel. Mecanum wheels are similar to omni wheels, which are a bit more intuitive. An omni wheel also has rollers along its circumference, but these rollers are oriented perpendicular to the wheel (See Fig 1.1). Due to the unique design of having rollers along the circumference, these two types of wheels are not able to transmit force in a specific direction, and as such they can be rolled freely in this direction. Perpendicular to this passive direction, the wheel can transmit force to propel the vehicle.

(a) (b)

Figure 1.1: Mecanum wheels (a) and omni wheels (b) both have passive rollers that allow the wheels to glide freely at a certain angle.

[1]Mecanum Wheels were first invented in 1973 by Swedish inventor Bengt Ilon while working as an engineer with the Swedish company Mecanum AB.

Mecanum wheels typically have rollers at a 45° angle from the axis of rotation of the wheel, meaning that a mecanum wheel transmits force at a diagonal. Note that a wheel with rollers at $+45°$ behaves differently than a wheel with rollers at $-45°$. The rollers on omni wheels are at 90° on the circumference of the wheel, meaning the wheel rolls forward normally but can be pushed sideways.

Mecanum wheels and omni wheels allow the vehicle's individual contact points with the ground to have 3 controllable degrees of freedom (DOF). For a normal wheel, the point of contact with the ground has 2 controllable DOF and 1 constrained DOF: the wheel can roll forward/backward, and the direction that the wheel faces can pivot around its point of contact with the ground. However, normal wheels do not have the ability to slide sideways (or "strafe"), constraining one DOF. The fully unconstrained 3 DOF of mecanum and omni wheels means that vehicles with several wheels will not be over-constrained when controlled properly. Or, as is common with omni wheels, an un-powered wheel on any vehicle can provide support for the vehicle's weight on the ground while not constraining motion in any way.

As mentioned, these two types of wheels are very similar, but we will be focusing on describing motion with mecanum wheels alone. In practice, an omni wheel pushes a vehicle the same way as a mecanum wheel, as long as the omni wheel is mounted at the angle of the mecanum wheel's rollers. Thus, the results of this thesis can also be applied to omni wheels with the appropriate rotation taken into account.

1.2 Common Practice

By turning the wheels in specific ways, a vehicle with mecanum wheels can move in any direction, which includes strafing in any direction and rotating around any center point. This requires that each wheel have independent control, so each wheel typically has its own motor. Because of this flexibility of motion, mecanum wheels have become a staple of robotics competitions and are used in various machinery such as automated forklifts.

In the vast majority of mecanum wheel applications, vehicles are driven by four mecanum wheels. The mecanum wheels are arranged at the corners of a rectangle around a vehicle, like the wheels of a car. They are almost always positioned with all of the wheels' axes parallel to each other, like a car that is steering directly ahead [1].

One of the most common uses of mecanum wheels is for controlling robots in robotics competitions. Competitions like FTC, FRC, and VEX Robotics require robots to be agile and to move quickly between several locations to score points as fast as possible. As such, mecanum wheels are prevalent in robotics competitions, where the ability to strafe at any angle can cut down on the time required to get tasks done and be decisive in a match's outcome.

Another common use of mecanum wheels is in the control of industrial factory and warehouse vehicles such as forklifts and shuttles [2]. These vehicles can even be completely automated with reliable performance. In a factory or warehouse, maximizing the usable floor space means a more cost effective facility, enabling more productivity. A traditional vehicle with regular wheels must be steered with a non-negligible turn radius in order to change direction, so paths for these vehicles need to accommodate for this extra space. However,

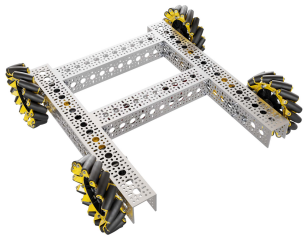


Figure 1.2: Starter kit for mecanum wheel robot, common in robotics competitions, showing a conventional rectangular wheel configuration..

with mecanum wheels, a vehicle can rotate around any point, even spinning in place, meaning that paths need not be made as wide for vehicles to successfully navigate the floor [3].

A very unique use of mecanum wheels is in Helical Robotics' magnetic climbing robot [4]. Their design of these wall climbing robots still conform to the traditional configuration of mecanum wheels, but the robot includes magnets which keep the robto in contact with a certical surface on which the robot is able to move freely.

1.3 Motivation

1.3.1 Goals

In this thesis, I propose a novel methodology that will break through each of the three common constraints for vehicles with mecanum wheels: spatial arrangement, relative angles, and wheel quantity. Given a configuration of any number of wheels, this system offers all the mathematical equations required so that the mecanum wheels can operate the described vehicle without restriction on the plane of the ground. I present the derivation of every step required to achieve this result. To prove my results, I have generated a random arrangement of wheels to implement in a real robot, and I have demonstrated how this robot drives as desired. As a byproduct of this research, I present an understanding of any configurations that fail for reasons such as under-constraint, and other important factors that one ought to consider when designing a new configuration of wheels.

1.3.2 Impact

The standard four-wheeled mecanum vehicles depend on every wheel having good traction in order to achieve all of the desired motion, so they are best used on clean, flat ground. My research will enable people to add redundancy to their motion control, so that in imperfect conditions where there may be some wheels slipping, for example, vehicles do not become stranded or fail to follow desired paths. Engineers can have the freedom to adjust the orientation of mecanum wheels to improve properties such as optimizing speed or power in certain directions, as well as optimizing steering stability. Furthermore, if one wants to

use mecanum wheels to drive a vehicle or robot but could not fit the wheels or motors in the standard configuration, there would be no problem altering the configuration and still maintaining the full range of control.

Chapter 2

Design Considerations

2.1 Coordinate Systems

Before we analyze design considerations for vehicles with mecanum wheels, it will be good to specify some coordinate systems for the vehicle.

Global Reference Frame

For controlling the vehicle, a fully defined global reference frame is not required: we don't need to define an origin or coordinate system for the global frame. It is enough to intuit that the floor is stationary and is known to be fixed in the global frame. We want to describe how the vehicle moves on the floor, so we will use the help of a reference frame defined with respect to the vehicle.

The vehicle has some pose[1] in the global frame that describes its position and orientation. We need not know the absolute values of the pose, but what matters is the rate of change in the vehicle's pose. We will describe the reference frame of the vehicle next so we can quantify the relevant directions and rates of change.

Vehicle Coordinate System

For the calculations and expressions in this thesis, we will consider the vehicle to be a rigid body in 2D, observed from top down. We will be using a vehicle coordinate system as defined by the ISO for road vehicles [5]: with respect to the vehicle, $+x$ is "forward", $+y$ is "left", and $+\theta_z$ is "up" (Fig. 2.1). θ_z represents the yaw of any object with respect to the vehicle. $\theta_z = 0$ will correspond to the direction of the positive x axis of the vehicle's coordinate system, and θ_z will increase counter clockwise, consistent with the right hand rule. This is a body-fixed frame of reference, meaning that origin and the axes of the coordinate system are "attached" to the vehicle and move in space with it. We can call the vehicle frame V.

The origin of the vehicle reference frame V needs to be defined but the location of the origin on the vehicle is not too important, and can be decided upon by the user for their

[1] A pose describes the position and orientation of an object in a coordinate system. It is often represented by a matrix T that contains a rotation matrix and a position vector, but I will simplify any poses to be represented by just the tuple of 2D coordinates (x, y, θ_z).

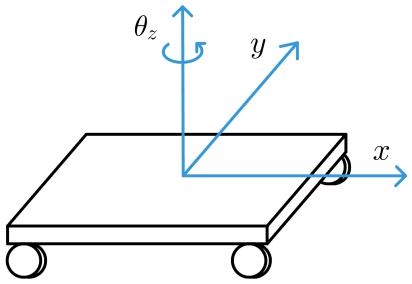


Figure 2.1: ISO Vehicle Coordinate System.

own applications. For the robots with which I am demonstrating my results, I have chosen to use an origin point that is approximately at the very center of the robot. For translating the vehicle, the location of the origin has no impact on the motor control. When rotating, the vehicle will pivot around the origin when no translation is simultaneously occurring, or when some other center of rotation is not otherwise specified. Once the location of the origin is decided, subsequent measurements should be measured relative to this origin.

Each mecanum wheel on our vehicle has its own permanent position and orientation in the vehicle's reference frame. The point of contact with the ground for a wheel i can be defined by a point $p_V^i = (x_i, y_i)$ where x_i and y_i are measured in the vehicle's reference frame. For mecanum wheels, the point of contact with the ground varies slightly as the wheel spins onto different sections of its rollers [3], but for our purposes it will be sufficient to consider the center of the wheel as the single permanent point of contact with the ground.

The wheel also has a characteristic orientation. We want to use a single angle in the frame V, ϕ_i, to describe this orientation for a wheel i. Let ϕ_i be the direction of the acceleration/force vector that wheel i propels the vehicle when its motor is given a positive voltage/signal (Fig 2.2).

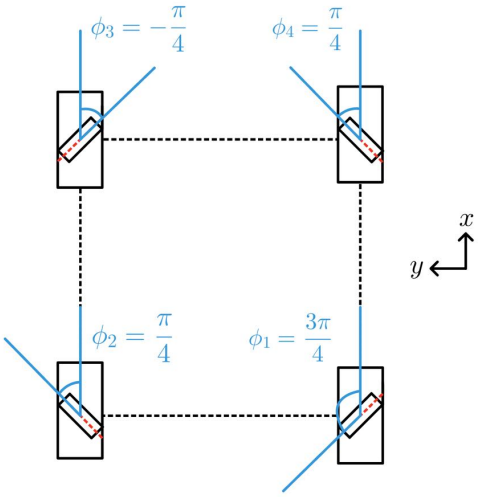


Figure 2.2: Example of possible ϕ_i values for standard mecanum wheel vehicle configuration.

In Figure 2.2, the values of ϕ_i are clearly related to the roller angle of $\pm\frac{\pi}{4}$ since the wheels are mounted traditionally, at an angle of 0° to the forward-rolling mounting angle. A deviation in the wheel's mounted angle will change ϕ_i by the same amount.

The characteristic angle ϕ_i is very important and comes up often in our analysis and equations. An example for measuring ϕ_i is given in Appendix A. The point p_V^i and the angle ϕ_i is all the information for each wheel that we will need in order to control the vehicle however we desire.

This section is sufficient to understand the vehicle coordinate system, but if you'd like to read more on how we quantify the vehicle's motion with the body-fixed reference frame, see Appendix B.

2.2 Underconstrained Configurations

When designing a vehicle with mecanum wheels, there are a couple of orientations where the design fails to control the vehicle in all 3 DOF as desired. To analyze whether a configuration of wheels is capable of the full range of motion, we will use a graph to inspect the direction each wheel can control (i.e. apply force). I'll call this a Force Graph.

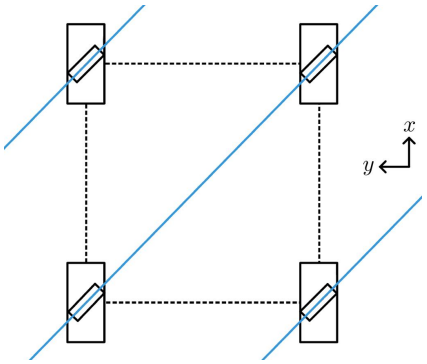
Start by drawing each wheel on a graph where the origin and axes are that of the vehicle reference frame V. Draw a line for each wheel in the direction of its rollers' axles when touching the ground. Each of these lines is at an angle of ϕ_i on the coordinate plane, passes through wheel i's ground contact point, and extends indefinitely. A wheel rolls freely on its rollers perpendicular to its corresponding line and rotates freely around its point of contact with the ground. There are two tests we can perform with this Force Graph in order to identify cases of underconstraint. Failing either test means the robot is underconstrained.

1. The first test is meant to determine whether the rollers of all of the mecanum wheels are at the same angle in the vehicle frame. This is the case if all of the lines in the Force Graph are parallel. In such an orientation, the vehicle has one direction in which it can propel itself, but perpendicular to this direction, each wheel's rollers align so that the vehicle has no control and can be pushed easily. The vehicle is also susceptible to rotation, meaning that the only control the vehicle has is one linear DOF. A vehicle with any number of wheels fails this test if all of the Force Graph lines are parallel.

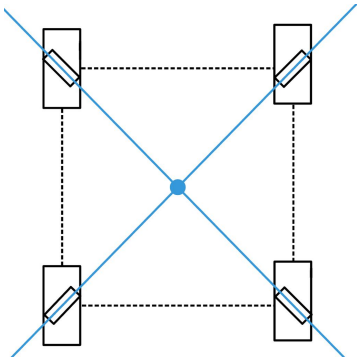
2. The second test discovers something more subtle. If every line in the Force Graph meets at one point, whether inside or outside of the vehicle, then we fail the second test. In this case, there is a rotational axis that is unconstrained. The vehicle can be freely pushed on its rollers around the point where the lines meet in the Force Graph. It is possible to find many configurations of mecanum wheels that fail this test, but the most likely is if one accidentally pointed the bottom rollers of each wheel to the center of the vehicle.

Note that any less than 3 wheels will always fail one of these tests, so the least number of mecanum wheels required to fully constrain and control a vehicle is 3. There is no upper limit to the number of mecanum wheels that can be used to control a vehicle.

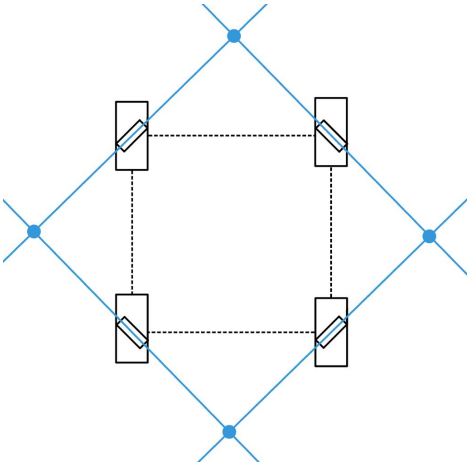
Figure 2.3 shows various examples of Force Graphs for vehicles with different wheel configurations, with some failing the Force Graph tests and others being viable configurations for omnidirectional control.



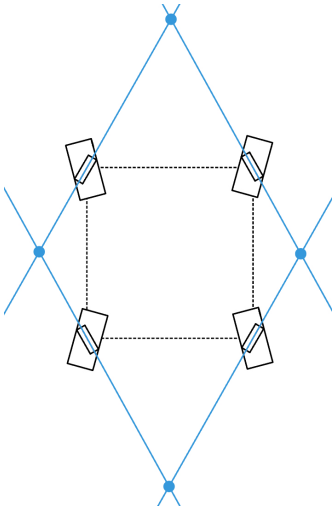
(a) All lines are parallel, failing test 1. The x and y axes are the same in the subsequent sub-figures (b-d).



(b) This configuration is a common mistake for four-wheeled vehicles. All lines meet at one point, failing test 2.



(c) Traditional configuration of 4 mecanum wheels passes both tests.



(d) Wheels angled to provide more force forward/backward at some cost to sideways force. This configuration also passes both tests.

Figure 2.3: Examples of Force Graphs for mecanum wheel vehicles.

2.3 Optimizations

2.3.1 Traction

The force that a vehicle can exert in any direction is proportional to the sum of the magnitudes of the inner product of the direction of each line in the Force Graph with the unit vector in the direction in question (Equation 2.1).

$$|F_{max}(\theta)| = \sum_{i=1}^{\text{num\_wheels}} |\overrightarrow{F_i} \cdot \hat{u}(\theta)| \tag{2.1}$$

$\overrightarrow{F_i}$ is the vector of the frictional force that wheel i can transmit on the ground. Its magnitude is dependent on the coefficient of friction of the wheel with the ground and the amount of the vehicle's weight that the wheel supports normal to the ground at its point of contact with the ground. The direction of $\overrightarrow{F_i}$ is in the direction of ϕ_i. $\hat{u}(\theta)$ is the unit vector in the direction θ that we are analyzing F_{max}.

2.3.2 Speed

The speed that any wheel can travel in the direction that its rollers spin freely (perpendicular to ϕ_i) is theoretically unbounded. The wheel's speed in the direction of ϕ_i is limited by the top speed that the motors can output for wheel i, denoted by a constant $v_{max,i} = \max(v_{desired,i})$, where $v_{desired,i}$ is the setpoint we will command the motor[2]. As such, for a single wheel, the velocity $v_{max,i}(\theta)$ that it can translate in any direction θ is restricted only by the component of that velocity in the direction of ϕ_i (See Fig. 2.4).

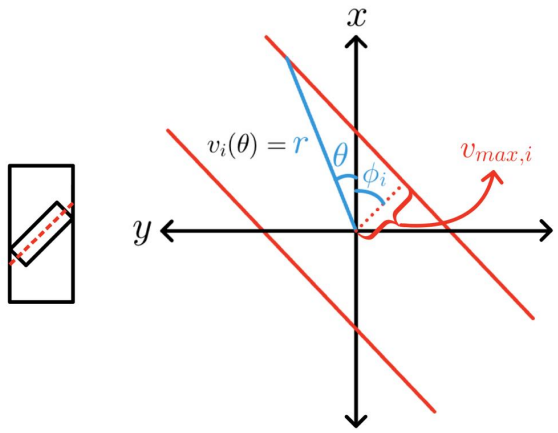


Figure 2.4: Example of $v_i(\theta)$, the max speed a wheel with characteristic angle ϕ_i can translate in the direction of θ.

The relationship between v_i and $v_{max,i}$ is given by Equation 2.2.

$$v_i(\theta) = v_{max,i}|\sec(\theta - \phi_i)| \tag{2.2}$$

With multiple wheels, the speed that the vehicle translate in the direction of θ is constrained by the smallest top speed of any wheel in that direction, as shown in Equation 2.3.

[2] $v_{max,i}$ is a constant that is either set by the user to limit the speed that the motor will turn, or the actual hard limit of speed for the motor, directly correlated with its top RPM.

$$v_{max}(\theta) = \min_{i \in [1, \text{num\_wheels}]} v_i(\theta) \tag{2.3}$$

A similar process can be done to analyze how additional rotation affects the limits of the vehicles motion. If we wish to translate at some rate and rotate at some rate proportional to the rate of translation (or just rotate), the greatest extent that we can scale the speed of this motion is constrained by the wheel whose velocity reaches its limit first in order to achieve its desired instantaneous rate of change. The fastest that the vehicle can go, v_{max} becomes a function of both the instantaneous direction of translation θ and some ratio of translation to rotation.

2.3.3 Redundancy

As mentioned in Section 1.3.2, standard four-wheeled mecanum vehicles depend on every wheel having good traction in order to achieve all of the desired motion. When it is possible that some wheels might slip in certain conditions but the agility of mecanum wheel control is still desired, consider adding more wheels for redundancy. If you design an n-wheeled vehicle for which any 3 wheels pass the Force Graph tests, then even if $n - 3$ wheels slip or are not touching the ground, the vehicle can still be controlled fully. If uneven ground is a possibility, then it is crucial that the outermost wheels pass the Force Graph tests for cases that the ground is concave. When the ground is convex, the vehicle will sit on the nearest wheels that enclose the point of the vehicle's center of gravity, so making sure such inner wheels also pass the Force Graph tests is important.

Chapter 3

Method for Generalized Mecanum Wheel Configuration

This chapter describes the Method for Generalized Mecanum Wheel Configuration (GMWC Method), including vehicle kinematics and motor control.

3.1 Kinematic Model and Control

3.1.1 Vehicle Motion

Once we know the pose of each wheel (See Sec. 2.1), we need to determine how each wheel should spin to achieve desired vehicle motion. Our vehicle is able to translate in any direction as well as rotate, so each wheel will also have a translation and rotation associated with it.

First, establish the desired behavior of the car in the form of (v_x, v_y, ω), which are the rates of change in the vehicle's forward/backward position (x), left/right position (y), and orientation (θ_z) using the directions of the axes in the vehicle frame V. If we wanted to move forward at 1m/s, to the right at 0.5m/s, and rotate clockwise at 2rad/s, our desired velocities would be $(v_x, v_y, \omega) = (1, -0.5, -2)$.

Next, consider the position of each wheel on the vehicle. Each wheel i is in a fixed position (x_i, y_i) on the vehicle. To find the motion of a wheel i when the vehicle's velocity is (v_x, v_y, ω), we must combine the linear and rotational elements of its velocity as shown in Equation 3.1.

$$\begin{bmatrix} v_{x,i} \\ v_{y,i} \end{bmatrix} = \begin{bmatrix} v_x - \omega \cdot y_i \\ v_y + \omega \cdot x_i \end{bmatrix} \tag{3.1}$$

These linear velocities are simply derived by adding the linear and rotational velocities of wheel's location on the vehicle and expressing them in the components of the x and y axes of frame V.

We can get the linear velocity in polar coordinates in V with the following equations, where the magnitude of the linear velocity is m_i and the direction is θ_i.

$$m_i = \sqrt{v_{x,i}^2 + v_{y,i}^2} \tag{3.2}$$

$$\theta_i = \text{atan2}(v_{y,i}, v_{x,i}) \tag{3.3}$$

3.2 Motor Control

Now that we know the instantaneous rate of change of the position of each wheel, we can determine what signal to send to the motors to achieve that motion. As we have seen, there is one direction that the mecanum wheel can push the vehicle, ϕ_i (See Fig. 2.2). We want the component of the wheel's velocity that is in line with ϕ_i, so we take the dot product of our linear velocity vector with the unit vector in the direction of ϕ_i.

$$v_{desired,i} = \begin{bmatrix} m_i \cos(\theta_i) \\ m_i \sin(\theta_i) \end{bmatrix} \cdot \begin{bmatrix} \cos(\phi_i) \\ \sin(\phi_i) \end{bmatrix} = m_i \cos(\theta_i)\cos(\phi_i) + m_i \sin(\theta_i)\sin(\phi_i) \tag{3.4}$$

The dot product can also be expressed as the two vectors' magnitudes multiplied by the cosine of the angle between them, so we can simplify to the following expression.

$$v_{desired,i} = m_i \cos(\theta_i - \phi_i) \tag{3.5}$$

See Figure 2.4 for a graphical representation of this calculation, where m_i is shown as $v_i(\theta)$.

This is the velocity that should be achieved by the motor's feedback controller for the wheel. The last step is to know how far in the direction of the power does the wheel travel per amount of rotation. For a normal wheel, turning by one radian results in a distance travelled equal to 1 radius of the wheel[1]. This is not exactly the case for mecanum wheels. We need an "effective radius" r_{eff} of a mecanum wheel with radius r_i as a function of the roller angle e, which ought to be known (Equation 3.6)[6]. If you cannot find any information on the angle of your mecanum wheels and find it too hard to measure the angle directly, see Appendix C for a test procedure which can be used for approximating r_{eff} for any wheel.

$$r_{eff} = r_i \sin(|e|) \tag{3.6}$$

Knowing the radius r_{eff} of the wheel, the angular velocity of the wheel should be set to ω_i as shown in equation 3.7.

$$\omega_i = \frac{v_{desired,i}}{r_{eff}} \tag{3.7}$$

Be sure to adjust the setpoint by a factor for any gear ratio between the motor and the wheel.

[1]For omni wheels which have their rollers' axles at 90° to the wheel axle, this property is the same. But at the other extreme with roller axles at the same angle as the wheel axle, spinning the wheel does not power any motion

3.3 Summary of the GMWC Method

The procedure for implementing the GMWC Method can be summarized as follows:

a. For each wheel i, we must measure the position of the wheel $p_V^i = (x_i, y_i)$ and a characteristic angle of the wheel ϕ_i in the vehicle reference frame V.

b. We describe the instantaneous rate of change in the pose of the vehicle as (v_x, v_y, ω), which is the input that the reader will provide repeatedly based on how they wish their vehicle to move.

c. Each wheel has a distinct instantaneous rate of change in position, which is derived from the vehicle's change in pose and each wheel's position on the vehicle.

d. We derive each wheel's rate of change in position, represented by $v_{x,i}$ and $v_{y,i}$ in Cartesian coordinates (Equation 3.1), and represented by m_i and θ_i in polar coordinates. (Equations 3.2, 3.3)

e. Using m_i, θ_i, and ϕ_i, we find the velocity in the direction that the wheel controls, $v_{desired,i}$. (Equation 3.4 or 3.5)

f. Using the wheel's radius r_i and the roller angle e, we find the effective radius of the wheel r_{eff} in the direction of force. (Equation 3.6)

g. Knowing the effective radius of our wheel, we can find the desired angular velocity of our wheel, ω_i. (Equation 3.7)

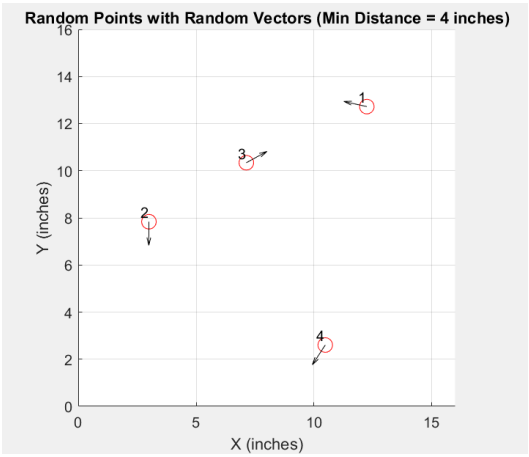
h Calculating each wheel's ω_i at each time step from our original inputs (v_x, v_y, ω) is now clearly shown, and we can control our motors by commanding them to reach the desired rate of rotation for the wheel.

Chapter 4

Experimental Verification

4.1 Randomized Wheel Configuration

To verify the proposed methodology, I generated a random set of coordinates and angles at which to mount 4 wheels on a test vehicle. This was done in MATLAB with help from ChatGPT. Each point was randomly generated from a 16 by 16 inch grid, with the constraint that each point was at least 4 inches away from each previously generated point. The angle of the wheels were also randomly generated in the range $[-\pi, \pi]$. Refer to Figure 4.1 for the result I received.



(a) Randomly generated wheel placement.

```
>> random_points

random_points =

   12.2483   12.7232
    2.9900    7.8362
    7.1294   10.3410
   10.4816    2.6018

>> random_vectors_angles

random_vectors_angles =

    2.9214
   -1.5767
    0.5002
   -2.1484
```

(b) Values of the randomly generated points.

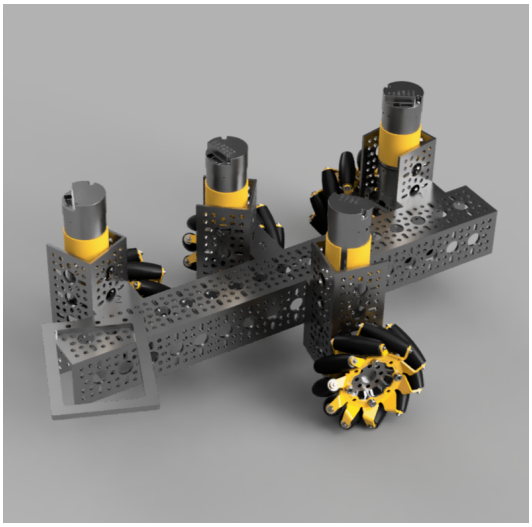
Figure 4.1: Wheel placement and angle was randomly generated in MATLAB.

To make this configuration of wheels on a robot, I used a long goBilda[1] U-channel for structure along the empty diagonal that the wheels happen to outline, and I 3D printed brackets to offset the motors/wheels to exactly the right position and orientation. For this vehicle, the origin of the vehicle coordinate frame V is the origin of the graph, and V also shares the same x and y axes as the graph. This means the default center of rotation for the

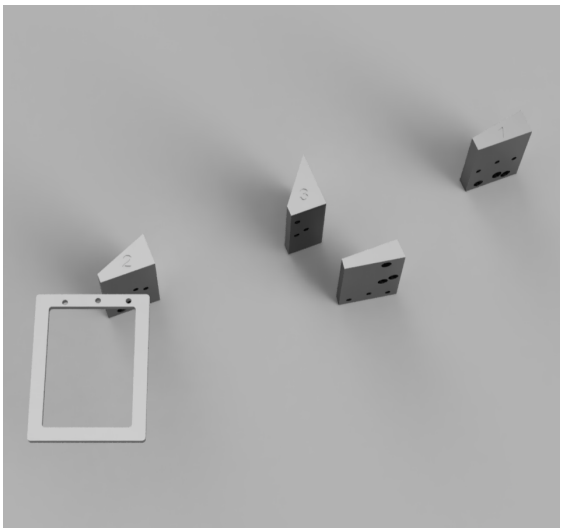
[1] goBilda is a self described "modern build system" with a large library of parts useful for robotics among other applications. https://www.gobilda.com/

vehicle on turns will not be towards the center of the vehicle but rather at this origin point which is outside the vehicle.

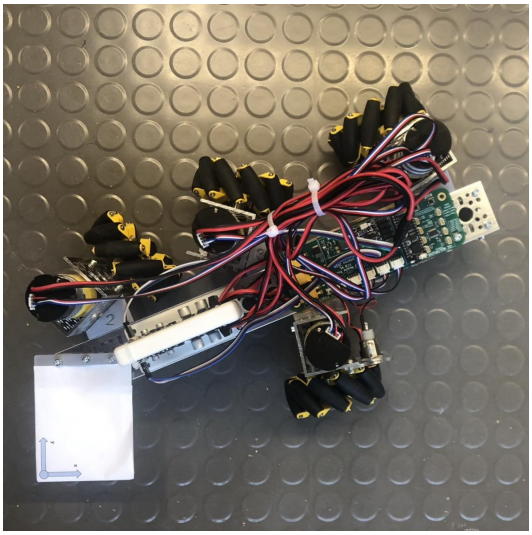
The rollers of each of the wheels make contact with the ground at the correct coordinates in V, and the orientations of the wheels are correctly designed so that the generated angle is aligned with ϕ_i. Note that this angle is for the rollers at the bottom of the wheel where it touches the ground, which are at a different orientation than the rollers of the wheel when they reach the top of the wheel. Figure 4.2 shows the resultant vehicle completely constructed with the Force Graph for this wheel configuration (see Section 2.2 regarding Force Graphs).



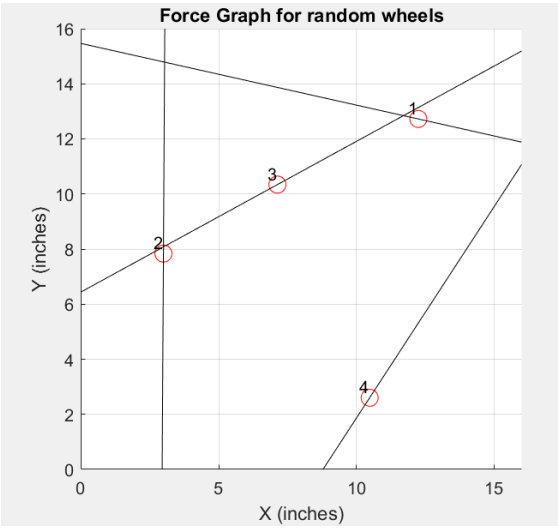
(a) Robot with random wheel configuration.



(b) Offsets to achieve the desired wheel placement.



(c) Robot with the generated configuration with the vehicle frame origin and axes displayed.



(d) Force graph of this wheel configuration shows a design where any 3 wheels pass both tests.

Figure 4.2: Creation of a robot with the randomly generated wheel configuration.

4.2 Robot Components and Electronics

The following components and electronics were used for my experiments:

- Set of four goBilda 96mm Mecanum Wheels
- Four goBilda 312rpm yellow jacket planetary gear brushed DC motors with encoders
- Two Cytron MDD10A DC motor drivers
- 11.1V 3A NiMh Battery
- Buck converter on a custom PCB
- Two ESP32-S3 microcontrollers
- Two Parallax 2-axis joysticks
- Appropriate jumper wires and power wires

4.3 Code

I used the methodology in this paper to program a microcontroller to control the vehicle, commanding the angular velocity setpoints of each wheel for a given vehicle velocity input. I programmed another microcontroller to read two joysticks and send that data to the vehicle so the user can drive the vehicle remotely, using ESP-NOW to communicate. For smooth motor control, first a low pass filter was given to the setpoints calculated for the wheels, then a PID controller sent the signal to the motor drivers which powered the motors using encoder feedback. To manage packages and separate environments for the different microcontrollers more effectively, this was implemented using Visual Studio Code with the PlatformIO plugin.

All relevant samples of the code are shown in Appendix D, as well as a link to a full GitHub repository.

4.4 Results

Randomized Configuration Robot

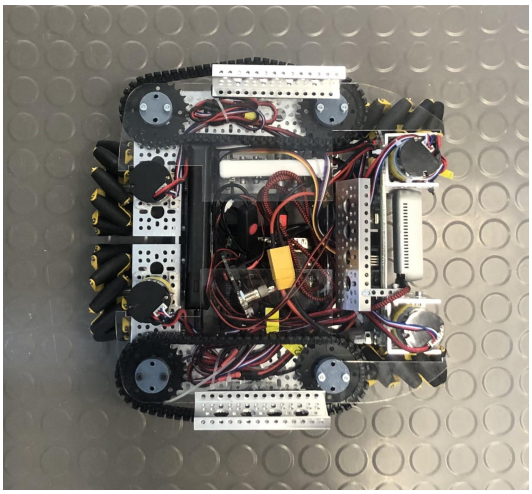
The robot with the wheel configuration presented above operated as expected. It was capable of strafing in any direction and rotated as expected around the origin point which I denoted with a printout of the x and y axes in the appropriate place on the robot. As we can see from the Force Graph (Fig. 4.2d), any 3 of these wheels can fully control the vehicle, so I also tested the robot with one motor removed. With enough space on the vehicle, it is evident that any number of wheels can be controlled using the GMWC Method. A brief video that shows the robot's performance is available on YouTube: https://www.youtube.com/watch?v=YZ9cJ1N7c4Y

The success of this vehicle with random wheels proves that the methodology of this thesis accurately allows for the control of mecanum wheels in any configuration. This methodology has also already proven successful in operating a robot in a competition setting.

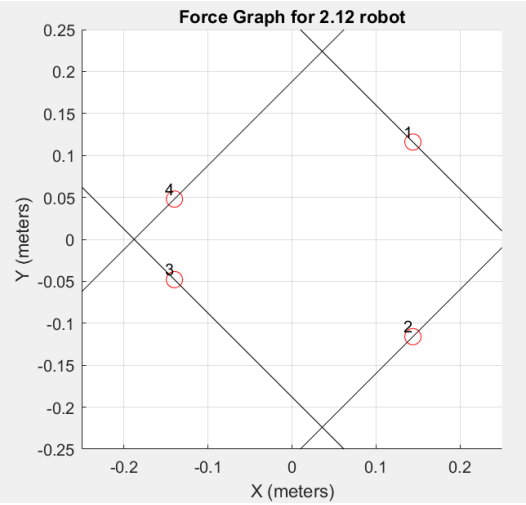
MIT 2.12 Term Project Mobile Robot

While developing this methodology, I also made another robot which used the concepts of the GMWC Method. In the class 2.12 Introduction to Robotics at MIT, the final team challenge requires a mobile robot to cooperate with a UR5 robotic arm to perform a search and rescue operation. The mobile robot must find its way to the disaster zone from the starting zone, then receive the victim from the UR5 and carry the victim back to safety. The shortest path to the disaster site went though a narrow tunnel, so the robot needed to be quite compact. The shortest return path involves climbing and descending ramps, so our robot also needed to be able to navigate these to complete the challenge in the shortest amount of time.

For this challenge, I created a robot with mecanum wheels in another novel configuration. The front two wheels are in a typical configuration, at the front corners of the robot and mounted in a standard orientation. However, the two back wheels in my design are not on the sides of the vehicle like the front wheels. Rather, they are turned 90° and mounted on the back face of the robot (see Fig. 4.3).



(a) 2.12 Competition Robot.



(b) Force graph for competition robot.

Figure 4.3: Mecanum wheel configuration for 2.12 mobile robot.

Part of the inspiration for this configuration of mecanum wheels was that while designing the mobile robot, the large motors for each mecanum wheel were not all able to fit in places that would permit the typical configuration of wheels because of other design choices. It was easiest to move two of the wheels to be along the back side of the robot, but I still needed to determine how to control all the wheels correctly[2].

This configuration was also practical and enabled our robot to do more than the standard configuration would allow. Mecanum wheels are well suited for flat surfaces but are less suited for uneven terrain. At the top of ramps where some wheels lose contact with the surface, the traditional configuration of wheels would not be able to handle this situation sufficiently. Getting high-centered by the top edge of the ramp was also an issue.

[2]This general configuration was the first new configuration for mecanum wheels that I ever thought of, and the inspiration for this topic as my undergraduate thesis topic.

The novel configuration of wheels that we used for our mobile robot fixed this issue without sacrificing any mobility. By crossing over the top of ramps sideways, the two close wheels on the "back" of the robot are now facing sideways to the direction of motion and provide a narrower wheelbase for crossing a ramp's protruding top edge. This means that the robot will not be high-centered at the top edges of ramps, which would not have been possible for such a compact mobile robot were it not for the novel wheel configuration.

This configuration has wheels at symmetrical but non-standard positions, and at comfortable but varied angles, so tuning the control for this configuration assisted my understanding and derivation of the methodology I have presented. It employs the GMWC Method to successfully compete in the class Final Challenge.

The robot was very agile and without restriction in its motion, and the control was responsive and accurate. This design helped my team complete the challenge in the fastest time out of any team. To understand better how this robot behaved, and to see how well this robot performed in competition, a recording of the competition is available on YouTube: https://www.youtube.com/watch?v=fRTGsvEkgaY

Chapter 5

Conclusion

This thesis is intended to free designers from the traditional constraints of mecanum wheel configuration. With the Method for Generalized Mecanum Wheel Configuration (GMWC), mecanum wheels can be used to operate vehicles with any arbitrary number and configuration. There are certain properties of some configurations that result in a failure to fully control a vehicle, but apart from these the user has the means to design and operate a vehicle whose wheels are oriented to optimize whatever property of the vehicle that they desire. Such properties include speed and traction in certain directions, or simply the layout of the components on a vehicle. To prove the methodology, one randomized and one designed robot were successfully operated using the GMWC Method. A full repository of the code that was used to operate the randomized vehicle is available as a reference for those who wish to program a vehicle using this methodology. Hopefully, the information presented will enable designers to utilize mecanum wheels with more flexibility and creativity, resulting in even greater possibilities for many applications.

Appendix A

Determining ϕ_i

This system depends on finding a parameter ϕ_i for each wheel. We have defined ϕ_i as the direction of the acceleration/force vector that wheel i propels the vehicle when its motor is given a positive voltage/signal. Here is an example of how I determined these directions for my test vehicle.

1. **Determine a way to apply a positive voltage/signal to a single motor**
   This is simple for DC motors, since you can just apply a positive voltage directly to the wires of the motor. If you are using a motor driver board, it may have buttons for this purpose. (Be sure to check the direction a button sends the current if it is not explicitly stated.) For other types of motors, servos, or any other actuator for the wheels, it may be easiest to do this step in software. Write code to send a positive direction signal to only one wheel at a time, and you should be good to go.

2. **Observe the general direction of force**
   Let this wheel contact a surface and observe what direction the wheel propels with this positive signal. I made this easy for me by setting my robot on the ground and putting a piece of paper under the wheel that will be tested. The single moving wheel pushed the paper under it in a direction. The direction that the wheel pushes the vehicle is π away from the direction that the paper is pushed.

3. **Calculate ϕ_i using known values**
   Knowing the general direction of ϕ_i (opposite the direction the paper was pushed), we can find its exact value based on the design of the vehicle. The angle that the wheel is mounted relative to the vehicle and the angle of the rollers on the mecanum wheel should be known values. The sum of these values should be a multiple of π plus ϕ_i. The angle of the wheel mount is $0°$ when the wheel is traditionally mounted, and the angle of the rollers is $0°$ when they are oriented like an omni wheel. Again, note that $-\frac{\pi}{4}$ and $\frac{\pi}{4}$ are different roller angles that will come in the same set of mecanum wheels.

4. **Repeat for each wheel**
   With each wheel tested and calculated, we will have determined ϕ_i for each wheel $i \in [1, \text{num\_wheels}]$.

Example

1. For a wheel controlled by a micro controller, we can program a positive signal to be sent to the actuator for this wheel whenever a certain button is pressed. Otherwise the wheel is stationary.

2. We observe that the paper is pushed to the top left of the vehicle, which is an angle of between $\frac{\pi}{2}$ and 0 in our vehicle reference frame. The direction that the vehicle is propelled is π away from this, so ϕ_i is between $\frac{-\pi}{2}$ and $-\pi$.

3. From the design of the vehicle, we know that the wheel is mounted at an angle to the left of $+x$, and this angle is $\frac{\pi}{3}$. The angle of our mecanum wheel rollers has magnitude $\frac{\pi}{4}$ and at the point of contact to the ground, they are diagonally to the right, so the angle of the rollers is $\frac{-\pi}{4}$ (right is counterclockwise, negative). Note that in a typical set of four mecanum wheels, assuming the angle of their rollers is $\frac{\pi}{4}$, two of the wheels will have an angle of $\frac{\pi}{4}$ and two will have an angle of $-\frac{\pi}{4}$.
   We add our two known angles as described in Step 3.
   $\frac{\pi}{3} + \frac{-\pi}{4} + \pi n = \frac{\pi}{12} + \pi n = \phi_i$, where n is some integer. Since ϕ_i was determined experimentally to be between $\frac{-\pi}{2}$ and $-\pi$, we can see that n is -1 and the value of ϕ_i is exactly $\frac{-11\pi}{12}$.

Appendix B

Discrete Time Calculations for Understanding Instantaneous Rate of Change

Instantaneous Coordinate Frames

To derive our motion, we will be considering the instantaneous rate of change of the vehicle's position and orientation relative to the ground. Knowing the origin of the global reference frame is not necessary to do this, but we do want to measure using a reference frame that is stationary relative to the ground. It would also be easiest to use the same direction for this stationary frame's axes as the vehicle reference frame V so that forward/backward motion remains on the x axis and sideways motion remains on the y axis. Let's combine these two features.

Let V' be a stationary reference frame that we will employ to derive the instantaneous rate of change of the vehicle's position and orientation. Imagine we want to find these rates of change at some time $t = t_0$. We will set up V'_{t_0} such that its origin and the direction of its axes are the same as those of the moving vehicle frame V at $t = t_0$. Then we imagine a small time step δt during which the vehicle might move by a small amount. The reference frame V moves with the vehicle during that time step, but V'_{t_0} has not moved, and we can compare the position and orientation of the vehicle at $t = t_0$ and $t = t_0 + \delta t$.

By definition, the vehicle's origin starts at $(x_{V,t_0}, y_{V,t_0}, \theta_{V,t_0}) = (0, 0, 0)$ in frame V'_{t_0}. After δt time, the vehicle moves relative to its starting position to the point $(x_{V,t_0+\delta t}, y_{V,t_0+\delta t}, \theta_{V,t_0+\delta t}) = (\delta x_V, \delta y_V, \delta\theta_V)$. We can use the limit as δt approaches 0 to find:

$$(v_x, v_y, \omega) = (\frac{dx_V}{dt}, \frac{dy_V}{dt}, \frac{d\theta_V}{dt})$$

Appendix C

Determining r_{eff}

This test can approximate r_{eff} for a mecanum wheel with rollers at any angle, particularly if you are unsure of the roller angle. We have defined r_{eff} as the distance that the wheel travels in the powered direction when the wheel rotates one radian. I would suggest doing your best to find the information on the angles of your wheels' rollers and using Equation 3.6, but I've also listed this procedure below to find r_{eff} experimentally.

1 **Determine a way to measure the rotation of the wheel**
For servos in default (non-continuous) mode, the angle of the servo is a linear function of the duty cycle of the PWM signal to the servo since servos typically use a potentiometer as an encoder and take care of the closed loop control themselves. For other motors I would suggest using an encoder and reading the encoder count as the motor moves. Without encoders, I would suggest marking one roller on a mecanum wheel and rotating the wheel by one full rotation so that the marked roller returns to its original position.

2 **Perform a known rotation and measure the distance travelled**
We are specifically interested in the distance travelled in the direction of ϕ_i. The motion of a mecanum wheel perpendicular to this direction is unconstrained and could be any arbitrary value. Make sure you do not rotate the orientation of the wheel during this experiment and only measure the distance travelled in the direction of the wheel's bottom roller's axle. The wheel should spin the fastest when travelling exactly in the direction of ϕ_i.

3 **Calculate r_{eff} using the experimental results**
Divide the distance travelled by the number of radians the wheel rotated.
This value is r_{eff}.

Appendix D

Example Code

A repository with the full code that I used to operate the randomly generated configuration is available at https://github.com/JamesDMorin/mecanum_random. The following is the section of the code that I used which was derived directly from the GMWC Method.

```
#define NUM_MOTORS 4
#define R_EFF 0.067882251 // 0.096m * sin(45), Equation 3.6
#define M_ALPHA 0.1

double setpoints[NUM_MOTORS] = {0, 0, 0, 0};       // angular velocity to turn the motor
double v_desired[NUM_MOTORS] = {0, 0, 0, 0};       // spped of the wheel in the direction of ϕ_i

// x (m), y (m), ϕ_i (rad). Conversion included from inches to meters for x and y
double motor_poses[NUM_MOTORS][3] = { {12.2483*25.4/1000, 12.7232*25.4/1000,  2.9214},
                                      { 2.9900*25.4/1000,  7.8362*25.4/1000, -1.5767},
                                      { 7.1298*25.4/1000, 10.3410*25.4/1000,  0.5002},
                                      {10.4816*25.4/1000,  2.6018*25.4/1000, -2.1484} };

/**
 * Updates the setpoints for the angular velocity of each wheel's motor given a set of input target velocities.
 *
 * @param forward, sideways Vehicle velocity along vehicle coordinate frame axes in m/s.
 * @param rotation Vehicle angular velocity in rad/s
 */
void updateSetpoints(double forward, double sideways, double rotation) {

    double motor_magnitude_angle[NUM_MOTORS][2];
    for (uint8_t i = 0; i < NUM_MOTORS; i++) {

        // Equation 3.1
        double motor_forward = forward - motor_poses[i][1] * rotation;
        double motor_sideways = sideways + motor_poses[i][0] * rotation;

        // Equation 3.2
        motor_magnitude_angle[i][0] = sqrt(pow(motor_forward, 2) + pow(motor_sideways, 2));
        // Equation 3.3
        motor_magnitude_angle[i][1] = atan2(motor_sideways, motor_forward);

        // Equation 3.5
        v_desired[i] = motor_magnitude_angle[i][0] * cos(motor_magnitude_angle[i][1] - motor_poses[i][2]);

        // Equation 3.7 with an alpha low pass filter
        setpoints[i] = M_ALPHA * v_desired[i]/R_EFF + (1-M_ALPHA) * setpoints[i];

        Serial.printf("Motor %u: Setpoint: %.2f, ", i, setpoints[i]);
    }
    Serial.println();
}
```

Figure D.1: Code snippet from the control of the randomized wheel configuration.

The array `setpoints` is updated with the target rate of rotation for each motor every time `updateSetpoints` is called. The array `motor_poses` is the measured pose information of each wheel on the vehicle. Another function can use `setpoints` at any time to command the motors.

There are a couple of libraries that I used to drive the motors, including libraries for the encoders, the motor drivers, and PID control. I also used more libraries for the controller signal. Check out the GitHub repository linked above for all of the information on the libraries used.

References

[1] B. Q. H. Taheri and N. Ghaeminezhad, "Kinematic model of a four mecanum wheeled mobile robot," *International Journal of Computer Applications*, vol. 113, no. 3, pp. 6–9, Mar. 2015. DOI: 10.5120/19804-1586.

[2] F. Adascalitei and I. Doroftei, "Practical applications for mobile robots based on mecanum wheels - a systematic survey," *Romanian Review Precision Mechanics, Optics and Mechatronics*, 2011.

[3] S. L. Dickerson and B. D. Lapin, "Control of an omni-directional robotic vehicle with mecanum wheels," *IEEE Xplore*, Mar. 1991.

[4] S. D. Kamdar, "Design and manufacturing of a mecanum wheel for the magnetic climbing robot," M.S.M.E. dissertation, Embry-Riddle Aeronautical University, Daytona Beach, FL, May 2015.

[5] I. S. O. "Road vehicles — vehicle dynamics and road-holding ability — vocabulary (iso standard no. 8855:2011)." (2011), URL: https://www.iso.org/obp/ui/#iso:std:iso:8855:ed-2:v1:en.

[6] A. Gfrerrer, "Geometry and kinematics of the mecanum wheel," *Computer Aided Geometric Design*, vol. 25, no. 9, pp. 784–791, Dec. 2008. DOI: 10.1016/j.cagd.2008.07.008.