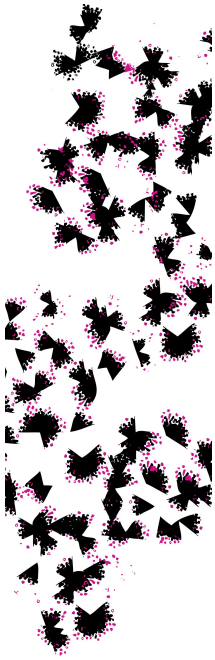


Design of an anti-slip control system of a Segway RMP 50 omni platform

R.J. (Robin) Lieftink

BSc Report

Committee:
Dr.ir. J.F. Broenink
Dr.ir. D. Dresscher
Prof.dr.ir. D.J. Schipper

July 2017

024RAM2017
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

In the i-Botics centre, founded by TNO and the University of Twente, a project is carried out focused on telerobotics. With telerobotics a robot is able to perform tasks on remote locations. For some of these tasks human expertise is needed for assessing and responding to unpredictable situations. To perform these tasks to user must be able to exactly feel what the robot is doing. With the use of sensors and haptic feedback this can be made possible. The robot used consists out of a KUKA Light-Weight Robot 4+ with a RightHand Robotics ReFlex TakkTile attached to a platform. This platform can move omni directional using the Segway RMP 50 omni, of which the velocity is controlled by a joystick.
The drawback of this system is that omnidirectional wheels can slip on smooth surfaces. This results in a deviation from the desired motion. The goal of this thesis is to build a system that corrects unwanted motion using sensors and feedback control.
The Segway RMP 50 omni is a platform which can move omni directional, due to the rotated forces which act upon the rollers of the mecanum wheels. The slip which occurs in these mecanum wheels is the consequence of a higher contact force than friction force. This slip creates a deviation from the desired direction. A slip ratio can be calculated based on theoretical wheel velocity and the actual velocity of the wheel.
The open loop system which does not take the slip in consideration is changed into a closed loop system. Different approaches of a closed loop system to cancel out the slip have been looked at. The slip controller using velocity adjustment will be used. The slip must be detected with an internal sensor as a reference point in combination with the encoders of the Segway, the extra sensor added is the IMU. The slip controller will determine the slip ratio by using the IMU measurement and the encoder measurement. Based on the calculated slip ratio the velocity scaling is determined. Due to the noise, this scaling is filtered with a moving average filter for a smooth controller. Scaling this velocity results in a lower contact force, which reduces the slip. The controller showed that the slip was detected by reducing its velocity. Due to this scaling the deviation of the platform was lower. The tests that are done were the two extremes. One surface with a high friction surface and one with a low friction surface. The high friction surface showed that both controllers worked properly, but still had a higher deviation than required due to the weight distribution. The low friction surface showed that the combination of the University floor and the wheels can not be solved, because the friction between the two is to low. Due to this low friction the minimum speed and maximum deviation was not made and the system started oscillating. These two extreme test conditions made it hard to compare the results. It is recommended to use other wheels, which can exert more friction on the ground. Also different test conditions can be recommended, to get a better results for the comparison between the open loop and closed loop system.

Samenvatting

In het i-Botics innovatiecentrum, initiatief van TNO en de University of Twente, wordt een project onderzocht gebaseerd op telerobotica. De robot kan door telerobotica taken uitvoeren op een afgelegen locatie. Een persoon is nodig voor sommige taken om onverwachte situaties te beoordelen en erop te reageren. Om deze taken uit te voeren moet de gebruiker exact voelen wat de robot doet, door het gebruik van sensoren en haptische feedback. De robot die gebruikt wordt bestaat uit een KUKA Light-Weight Robot 4++ met een RightHand Robotics ReFlex TakkTile vastgemaakt aan een platform. Dit platform kan omni directionaal bewegen door de Segway RMP 50 omni, welke is aangestuurd door een joystick.
Het nadeel van het systeem is dat de omnidirectionale wielen kunnen slippen op gladde oppervlakte. Dit resulteert in een afwijking van de gewenste beweging. Het doel van deze scriptie is het bouwen van een systeem dat ongewenste bewegingen corrigeert met het gebruik van sensoren en feedback control.
De Segway RMP 50 omni is een platform die omni directionaal kan bewegen door de krachten die de meanum wielen uitoefen op de grond. De slip in deze wielen is de consequentie van een hogere contact force dan de friction force. Deze slip creÃńert een afwijking van het gewenste pad. Een slip ratio kan berekend worden op basis van de theoretische wiel snelheid en de echte wheel snelheid.
Het openloop systeem dat geen rekening houdt met de slip wordt veranderd in een closed loop systeem. Verschillen aanpakken van een closed loop system om slip te vermijden is naar gekeken. De slip controller met gebruik van snelheid verandering zal worden gebruikt. De slip moet gedetecteerd worden doormiddel van een interne sensor als referentiepunt in combinatie met de encoders van de Segway, een IMU is hiervoor gekozen. De slip ratio wordt bepaald door de metingen van de IMU en de encoders. De verandering van de snelheid is gebaseerd op de berekende slip. De verandering van snelheid wordt gefilterd met een moving average filter, omdat er ruis is in het systeem is. De snelheid verandering zorgt voor een lager contact force, wat de slip verminderd. De controller laat zien dat slip gedetecteerd is door de snelheid aan te passen. De afwijking was minder door het veranderen van de snelheid. De testen die zijn gedaan zijn de twee uitersten. Een test met een hoge en een lage wrijvingsoppervlakte is gedaan. Het oppervlakte met de hoge wrijving liet zien dat beide controllers goed werkte, maar had nog steeds een hogere afwijking dan verplicht door de gewichtsverdeling. De oppervlakte met lage wrijving liet zien dat de combinatie van de universiteitsvloer en de wielen niet opgelost kan worden, omdat de wrijving tussen de twee oppervlakte te laag is. Door deze lage wrijving is de minimumsnelheid en maximum afwijking niet gehaald en het systeem begon te oscilleren. De twee uiterste test condities maakte het lastig om de resultaten te vergelijken. Het is aanbevolen om andere wielen te gebruiken, die meer wrijving hebben. Ook andere test omstandigheden worden aangeraden, om een beter verschil te krijgen tussen de openloop en closed loop systeem.

Contents

1 Introduction

In the i-Botics centre, founded by TNO and the University of Twente, a project is carried out focused on telerobotics. With telerobotics a robot is able to perform tasks on remote locations. For some of these tasks human expertise is needed for assessing and responding to unpredictable situations. To perform these tasks to user must be able to exactly feel what the robot is doing. With the use of sensors and haptic feedback this can be made possible. The robot used consists out of a KUKA Light-Weight Robot 4+ with a RightHand Robotics ReFlex TakkTile attached to a platform. This platform can move omni directional using the Segway RMP 50 omni, of which the velocity is controlled by a joystick.

The drawback of this system is that omnidirectional wheels can slip on smooth surfaces. This results in a deviation from the desired motion. The goal of this thesis is to build a system that corrects unwanted motion using sensors and feedback control.

In this thesis different sensors and controllers will be investigated and the currently applied open loop control will be extended with feedback control using these sensors, which will counteract the slip. The robot runs on a robotic operating system(ROS) and functionality is programmed in C++. The developed functional code will be made in a modular, ROS-independent structure; in this way these blocks can be reused in later stages or different robots.

This thesis starts with an analysis section, where mecanum wheels, slip in wheels, different controllers and sensors will be analyzed. The next section is the design and implementation, where a controller is designed based on this problem and will be implemented in ROS. Experiments will will be done on this design. In the results the outcome of these experiments will be observed and in the next section discussed. A conclusion is made on this research and different recommendations are done.

2 Analysis

In this Analysis section, a literature study is done on different parts. This study is done to develop a better understanding of the problem and to come to a conceptual solution. Starting of with the basic principles of mecanum wheels.

2.1 Mecanum wheels

Mecanum wheels are based on a normal wheel with rollers added in a certain angle. These angled rollers translate the rotational force of the wheel into a rotated force. An example of a mecanum wheel is visible in Figure 2.1.

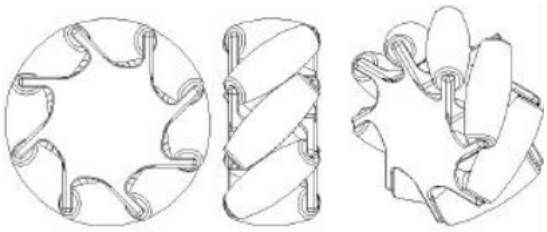


Figure 2.1: mecanum wheel [Diegel et al., 2002]

The rotated force due to the rollers is directed at a 45 degrees angle in this case. Different directions can be driven by using a setup with 4 mecanum wheels. Using equation 1 [Soni et al., 2014] the different rotational velocities of each wheel can be calculated using the velocity in the x-axis, y-axis and rotation around the z-axis.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{R} \begin{bmatrix} 1 & -1 & -(l_1+l_2) \\ 1 & 1 & l_1+l_2 \\ 1 & -1 & -(l_1+l_2) \\ 1 & 1 & l_1+l_2 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} \tag{1}$$

Where R is the radius of the wheels and l_1 is the distance between the center of the wheels and the center of the robot of the y-axis and l_2 of the x-axis. The orientation of the axis and the wheel can be seen in Figure 2.2.

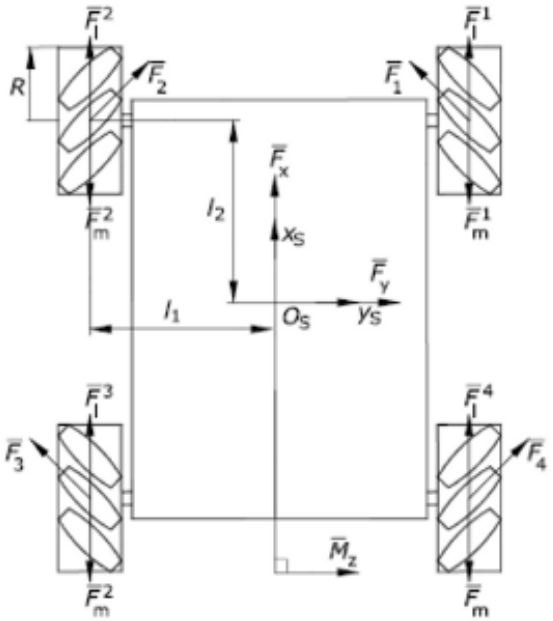


Figure 2.2: schematic of a robot chassis using mecanum wheels [Doroftei et al., 2008]

Figure 2.2 also shows the forces acting on the robot. Because of the orientation of the rollers every wheel has its force vector 45 degrees rotated. The forces $F_1 ... F_4$ are the forces on roller level, these are generated by the torque applied on the wheel. F_x and F_y are the forces applied on the center of the platform.
These forces can be used to calculate the torque that needs to be applied on the wheels. Figure 2.3 shows the vector of one wheel. Using this figure Equation 2 can be derived, which is done in the Appendix A.

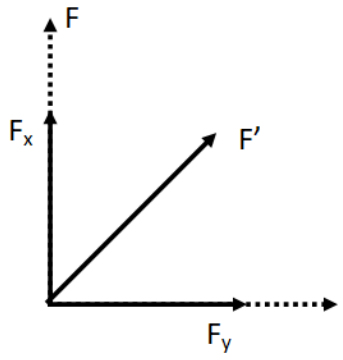


Figure 2.3: force vector of one mecanum wheel

$$F_x = F_y = \frac{1}{2R}\tau \tag{2}$$

Where F is the input force, F' is the rotated force by the roller, F_x is the x component of the rotated force and F_y the y component. The vector used in Figure 2.3 is the same as wheel two visible in Figure 2.2. Equations 14...17 show that the torque of the wheels are rotated twice, the first rotation is because of the rollers. The second rotation is the separation into the x and y component. For the rotation of around the z axis there is a torque with an arm(r). For this calculation Figure 2.4 is used.

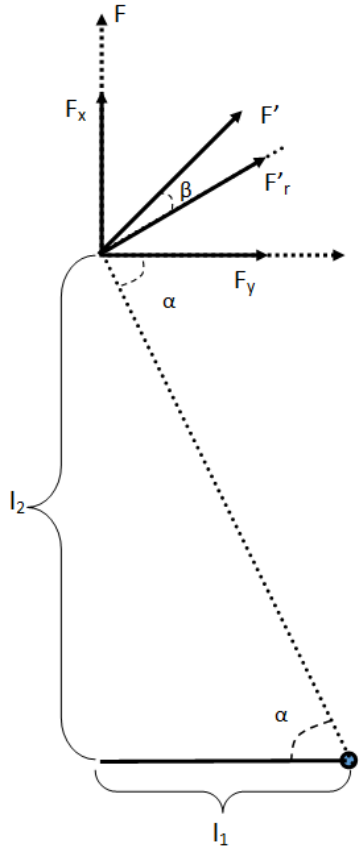


Figure 2.4: force vector of one mecanum wheel with the center of platform

Where F_r' is the rotational force applied. Using Figure 2.4 and Equation 3 the torque around the z axis can be calculated(τ_z), which can be seen in equation 4. This equation is derived in the Appendix A

$$\tau_\mathbf{z} = \|r\| \|F\| sin(\theta) \tag{3}$$

$$\tau_\mathbf{z} = \frac{1}{2R}(l_1 + l_2)\tau \tag{4}$$

Equation 4 shows the relation between the rotational torque(τ_z) by the torque applied on the wheel(τ). These equations are derived using wheel 2. Depending on the orientation of the vector of the wheel, the signs change. The F_x component of wheel one and three are negative and the τ_z component of wheel one and four are negative, due to the anticlockwise direction of the vector. Using this information for all the wheels a matrix can be made which is visible in equation 5.

$$\begin{bmatrix} F_x \\ F_y \\ \tau_z \end{bmatrix} = \frac{1}{2R} \begin{bmatrix} -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ -(l_1+l_2) & (l_1+l_2) & (l_1+l_2) & -(l_1+l_2) \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{bmatrix} \tag{5}$$

Using Equations 1 and 5 it can be seen that the platform is omni directional using different wheel velocities, which is visible in Figure 2.5.

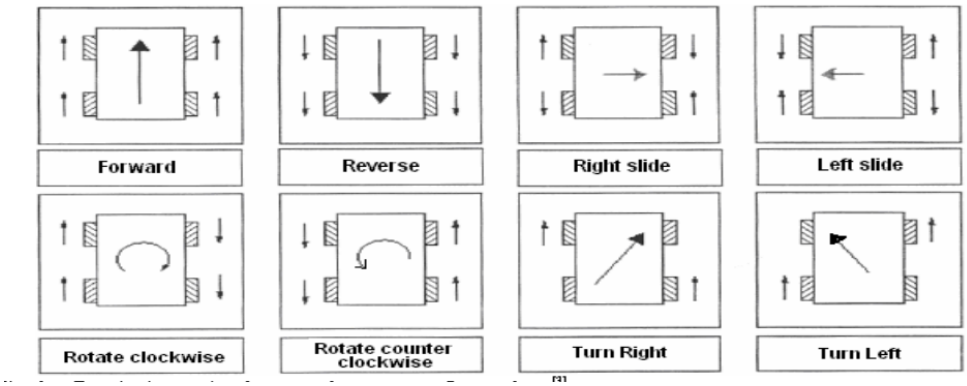


Figure 2.5: Different directions based on the motor control [Salih et al., 2006]

2.2 Slip in wheels

Slip can occur between the wheels and the ground due to insufficient friction. When there is no slip the translational velocity(v) is related to the rotational speed of the wheels(ω) given with Equation 6:

$$v = R * \omega \tag{6}$$

Where R is the radius of the wheel. This linear equation will not apply when the friction between the wheel and the surface is to low, which causes the wheel to slip. This can be expressed with equation 7

$$\lambda = \frac{R\omega_{wheel} - v_{actual}}{R\omega_{wheel}} \tag{7}$$

Where λ is the slip ratio, ω_{wheel} is the rotation of the wheel and $v_{Desired}$ is the desired velocity of the platform.
As explained above, this slip can occur due to insufficient friction between the surfaces of the

floor and the wheels. A friction coefficient is used to express the friction between two surfaces. Figure 2.6 shows that the friction coefficient against the slip ratio for different surfaces.

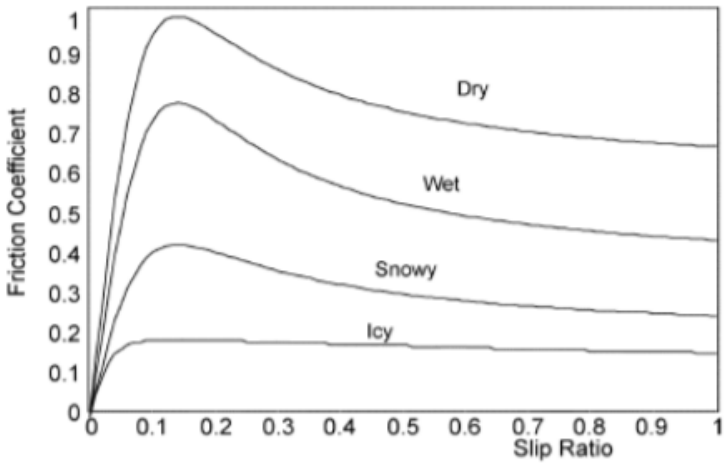


Figure 2.6: friction coefficient against the slip ratio [Junhui and Jianqiang, 2010]

In Figure 2.6 it is visible that the slip ratio depend on the friction coefficient, when a constant normal force is applied. This friction coefficient is depended on the material of the surface and the wheel. Equation 8 [Sharkawy, 2010] show the relation of the contact force with friction force when there is no slip.

$$F_{contact} \leq F_{friction} = \mu F_N \tag{8}$$

It shows that the friction force depends on the friction coefficient(μ) times the normal force(F_N) and that the contact force must be lower than this force to avoid slip. Equation 8 also shows that if the friction coefficient is high, the wheels will have a high friction force. This means that the platform can also exert a high contact force on the ground without slipping. But when the contact fore will be higher than the friction force, the wheels will slip. This can be counteracted by lowering the contact force.

2.3 Current system overview

Currently, an open loop control structure as seen in Figure 2.7 is implemented on the platform.

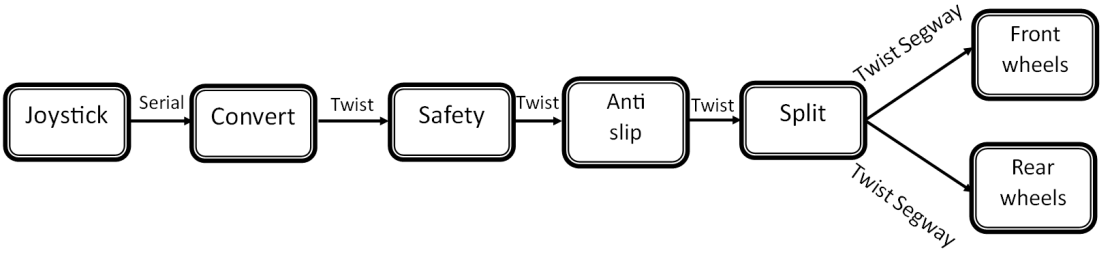


Figure 2.7: the current control scheme of the platform

The joystick is the input of the system and the convert block converts this input of the joystick into a twist message. The safety block is a dead man switch, so the platform only drives when this button is pressed. This twist is then split. This will convert the twist to 2 separate twist for the front wheels and the rear wheels of the Segway.

2.4 Anti slip techniques

There are different techniques to create an anti slip control system, which will be explained below. In this assignment only a control system will be looked at, not changing the surface of the floor or the wheel. The control techniques researched are commonly used in determining the position of the robot using no external reference system. This choice has been made based on the future goals of the robot, having to operate in an unknown environment.

Trajectory and heading tracking control

The first control system is trajectory and heading tracking control. A simple control scheme is visible in Figure 2.8.

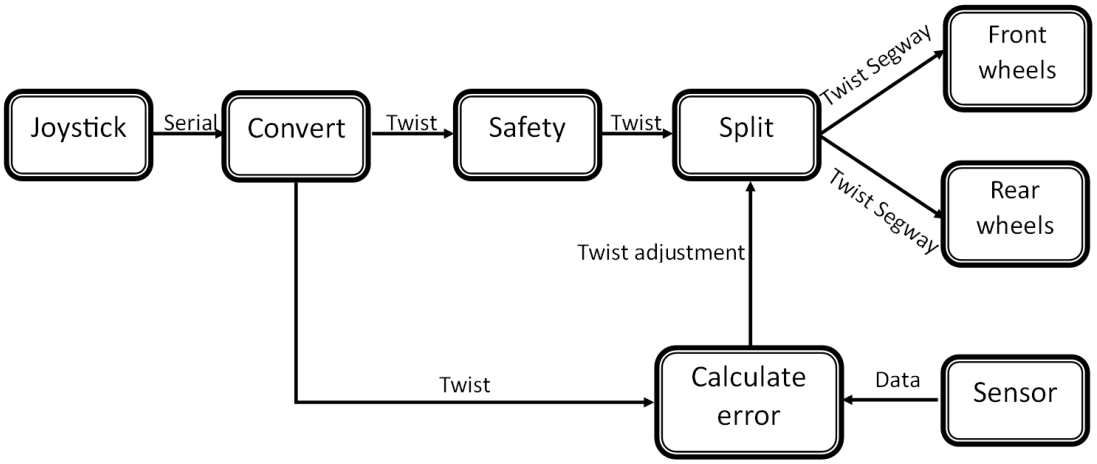


Figure 2.8: Simple scheme of the Trajectory and heading control

The platform will follow the input of the controller. Using a sensor the actual direction of the platform will be measured. An error will be calculated using the input of the controller and the measured direction. With this error a twist adjustment will be calculated. When the output direction of the platform is not as desired it will change the wheel speeds. This means that it will make some wheel velocities higher and reduce others.
The sensor used for this approach must be a sensor which measures its velocity into the x and y direction and rotation around the z direction. This can be done by one sensor. The drawback of this system is that it will not work on a floor with very low friction, then the wheels will still spin out. [Kuo et al., 2016].

Slip control using velocity adjustment

The second approach is a slip control using velocity adjustment. Like explained before, if the contact force is higher than the friction force then slip occurs. This contact force must be lowered. This contact force is expressed in Equation 9

$$F_{contact}(v) = vR + ma \tag{9}$$

Where R is the friction. It can be seen that lowering the velocity and limit the acceleration will lower the contact force. The acceleration will be limited internally in the Segway. This controller will be based on the principle that when the wheels slip, the magnitude of the velocity is scaled down in such a way that the direction is preserved. This will reduce the slip. To achieve this, the slip must be detected. To calculate the slip the velocity of the wheels must be measured and the actual velocity of the platform. The velocity of the wheels can be measured using encoders that are mounted on the wheel axis. The actual velocity can be measured using a sensor which measures the movement of the robot in the inertial frame. Using Equation 1, the corresponding velocities of the wheels can be calculated. The slip is then calculated by comparing these two velocities. Then the velocity of the wheel can be

scaled down on every wheel until no slip occurs. Using this method the velocity vector of the platform will have the same direction but the length will be scaled.

Slip control using force adjustment

The third technique has the same principle as velocity control, lowering the contact force when slip occurs. But now it will be done by adjusting the torques instead of the velocity. When the systems notices that the wheels slips then the torque input of the Segway will be scaled down to avoid this slip. The torque of the wheels can be read out of the platform. This torque will be converted to a force per wheel using Equation 10

$$F = \frac{\tau}{R} \tag{10}$$

Where F is the force that the wheel exerts on the ground and τ the torque applied with the wheel radius R.
Now that the force that every wheel exerts is calculated, it can be converted to the forces acting on the platform using Equation 5. This is the force that the platform is trying to reach(F_{theory}). Now using a accelerometer, the actual force(F_{actual}) of the platform can be calculated using Newtons second law. Now F_{theory} and F_{actual} can be compared to get the force difference. Using this force difference the torque can be lowered, resulting in a smaller contact force, which will cancel out the slip.

Combination slip controllers

The slip controllers can be combined. Two approaches can be done.in two ways. The first one is detecting the slip using the velocity and than lower the torque according to that slip Or it can be done the other way around, by detecting it with the torque and limit the wheel velocity.

Targeted approach

The chosen anti slip technique is based on what will be the future of the project and what can be implemented in the time reserved for this thesis. All systems have the advantage that only one extra sensor is needed. The first approach has as disadvantage that on a floor with low friction it will just spin harder losing more traction. The Segway has a velocity input, so this is the disadvantage for the controllers with a torque scaling. The two controllers left is the slip controller with velocity or the combined slip controller, where the slip is measured with the torque and scaled using the velocity. Both systems will work but the slip controller using velocity scaling has the advantage that only velocities are used, so no conversions have to be made. This means that the slip control using velocity adjustment will be used.

2.5 Requirements

In this subsection the requirements of the system and the sensors will be looked into and explained.

System requirements

In this assignment an anti slip system has to be made. This will be done by changing the open loop system into a closed loop system using velocity control. With this feedback the slip should be canceled. Some requirements are needed to achieve this goal:

- minimum speed of 0.5 km/h: the maximum speed of the vehicle is 3 km/h. When the speed is lower the robot will be to slow for proper use.
- deviation of max 5° in the x and y direction and rotation around the z-axis: when this is bigger the platform will deviate to much from its directed orientation. This deviation can be corrected by the user.
- Adapt to different surfaces

Sensor requirements

For this assignment a sensor is needed to make a closed loop control system. The requirements are as follows:

- Translation in x and y-axis and rotation around the z-axis in the inertial frame.
- Deviation of max 5% degrees in the x and y direction and rotation around the z-axis. To reach 0% slip there can be no deviation, but it is impossible to have a 0 degree deviation. So the max of 5 degrees is allowed. This will influence the calculation of the slip. Which will result in a small velocity deviation of the platform.

2.6 Sensors

in this subsection different usable senors will be looked at. For this robot it is necessary that the sensors are integrated into the platform. This means that not external resources can be used such as cameras or beacons. Below an analysis is done with internal sensor commonly used in robots.

Inertial measurement unit

An inertial measurement unit (IMU) exist out of a accelerometer in combination with a gyroscope, sometimes a magnetometer is added. This magnetometer is not needed in this thesis. The accelerometer of the IMU can measure linear acceleration, this can be used to calculate the translation in the x-axis and y-axis. The gyroscope is used to measure the rotation around the z-axis. This sensor can provide all required information.
The biggest drawback of this sensor is the drift. The output of the accelerometer is an acceleration. To retrieve the velocity or position of the IMU this acceleration must be integrated. The integration constant and the constant error resulting from the bias stability is neglected during integration, these errors have a large role in the error over time which is the drift in the IMU [Sukkarieh et al., 1999]. For this project this all will have an influence in the calculation of the slip.

There are advanced types of IMU's, these have internal correction methods, like a kalman filter to reduce the drift as much as possible. The Xsense is an example, this IMU has low drift due to these methods [Xsens et al., 2010]. This can increase the performance significantly and reduces the bad influences on the controller

Vision sensor

Also a vision sensor can be used. Using image processing the movements of the robot are measured. This is done by taking key points in an image and compare them in every new image. In this way the change of position can be calculated by every image. A disadvantage is the implementation time of the image recognition [Nagatani et al., 2000]. Another form of a vision sensor is a mouse sensor. This can accurately measure a position and had the image processing already implemented. The big drawback is that it will not be able to work on every surface.

Passive wheels

Passive wheels can be added to find the position of the robot. When three passive omnidirectional wheels are added to the robot the translation in the x and y direction and the rotation around the z axis can be determined precisely. The advantage is that the velocities can be detected with almost no error. The disadvantage is that the passive wheels only work on flat surfaces. With rough surface or inclines the passive wheels will not work [Tehrani et al., 2003]. Adding these extra mechanical components will bring a lot of extra work which will be to time consuming.

Sensor decision

For this project the IMU will be used because of the following reasons. The IMU is easier to implement in this robot, because it gives an acceleration and by integrating, the velocity is immediately known. Also this IMU is preferred because of its availability in the lab and the easy connection on software level. The output of the vision sensor will be images, these images needs to be processed first, which is due to time limitations not possible. The passive wheels or mouse sensor will limit the robots ability to drive everywhere. Also these wheels will take to much time to implement on the robot due the additional mechanical parts.

2.7 Conclusion

In this section the mecanum wheels and their forces acting on the body have been analyzed. How slip occurs in wheels is now know and how the slip ratio is related to this. Using this analysis different controllers are found and looked into. The slip controller with velocity scaling has been chosen. Where an IMU will be used as extra internal sensor in combination with the encoders of the Segway.

3 Design and implementation

In this section the design and implementation of the open loop system to a closed loop system will be explained. This is done by using the slip control with velocity adjustment and an IMU sensor. How this velocity control is implemented, which sensor is used and how the program works is explained below.

3.1 Controller

To make an anti slip controller for the Segway 50 RMP omni a slip control with velocity adjustment will be used.
in Figure 3.1 the control scheme is visible. This is a global block diagram of how the structure will look like. The red blocks are the new blocks in the controller compared to the open loop control.

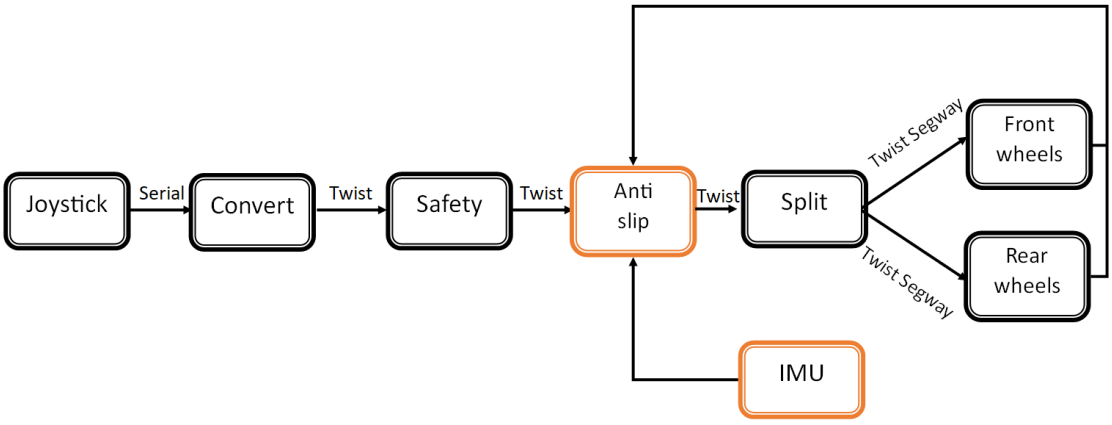


Figure 3.1: The block diagram of the velocity controller

Joystick
As explained the Joystick block converts the hardware input to serial information and sends this to the convert block.

Convert
This function converts the serial input into a twist by defining what every button means per controller (joystick, xbox), so the controllers are hard coded. This twist message is defined as follows. It gives a translational and rotational message. Both messages contain an x, y and z direction as an int64 number.

Safety
The safety is used as dead man switch. This means that the platform only has an input when

the dead man switch is pressed. This block will sent its twist to the new block anti slip.

Anti slip
The new block "anti slip" is added. This block will calculate the slip in each separate wheel using the encoders and the IMU. A detailed scheme of the block "Anti slip" is shown in Figure 3.2 to explain what functions are used.

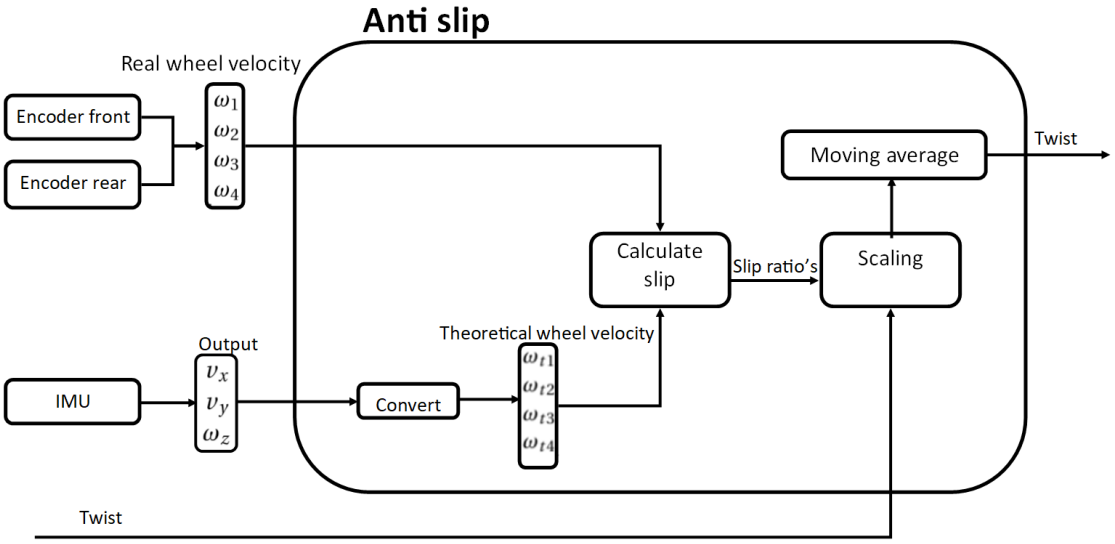


Figure 3.2: The detailed block scheme of "Anti slip"

First the sensor inputs will be discussed. The Segway has encoders which can read out the the rotational velocity of the wheels and the torque. The velocity of the wheels will be used, noted as $\omega_1 ... \omega_4$. This is also the input into the block calculate slip.
The used output of the IMU will be converted to the translational velocity of in the x and y direction and the rotational velocity around the z axis. The translational velocity of the z axis and the rotational velocity of the x and y axis will not be used because the robot is not able to make those movements. This velocity vector will be converted to the wheel velocities $\omega_{t1} ... \omega_{t4}$ using equation 1.

Now both velocities of the wheels are known, using this information the slip of each separate wheel can be calculated. This is done by Equation 11 derived from Equation 7.

$$\lambda_\omega = \frac{\omega - \omega_t}{\omega} \tag{11}$$

Where ω is the wheel velocity out of the encoder and ω_t is the wheel velocity out of the IMU The slip of each wheel is send to the block scaling.

In this block the slip ratio of every wheel is used to determine the scaling of the velocity. To do this, the wheel with the most slip is used. The slip ratio of this wheel will determine the

scaling which can be seen in Equation 12.

$$Scaling = 1 - \lambda \tag{12}$$

This scaling will used to convert the twist. The twist will be multiplied with the scaling and then send to the block moving average.
The encoders and IMU will have noise due to the shaking of the platform. The update ratio of the scaling of the velocity is done at a 100Hz. This will result in a different scaling 100 times a second, which means that the velocity also changes this much, then the platform will not drive smoothly, due to this noise.//
To solve this a simple moving average filter is used visible in equation 13.

$$s_{average} = \frac{1}{n} \sum_{i=0}^{n-1} P_{M-1} \tag{13}$$

Where M is the current data point. The advantage of this filter is that there will be a smooth transitions to different velocities and that it can be computed in real time. But this filter will add a delay in the change of the velocity. The delay will be based on the filter length of the moving average, which is $\frac{(N-1)}{2}$ [Roelandts, 2015].

The averaged scaling of the velocity is then send to the block split. This block will convert the desired twist to two separate twist message. These twist messages are then the input in the Segway front and rear which convert this message to velocities of the separate wheels.

3.2 Implemented sensors

The encoders are already implemented in the Segway, which are directly connected to the axis of the wheels. This means that the measurements of these wheels are very accurate.
However the IMU has some drift as explained above. The XSense MTi will be used in this assignment because of its availability in the lab. The MTi is a miniature, gyro enhanced Attitude and Heading Reference System (AHRS) and is mainly used as measurement tool for stabilization and control for cameras, robots, vehicles and other equipment. It provides provides drift free 3D orientation as well as calibrated 3D acceleration, 3D rate of turn (rate gyro) and 3D earth magnetic field data. The drift is very low due to internal filters. Some important specification needed for this thesis are lined up here:

- The actual alignment between the housing and output is smaller than 3 degrees
- Angular resolution of 0.05% degrees
- Static accuracy of 1 degree
- Dynamic accuracy 2 degrees RMS
- Update rate is 256 Hz
- Bias stability 0.02 m/s
- Noise of 0.002 m/s

Concluding the sensors used will not have major influences on the performance of the system due to errors. The encoder will have a low error and the IMU will have very low drift [Xsens et al., 2010].

3.3 Programming implementation

Like explained in the analysis section, the program is made in ROS using different nodes with their different functions. For the IMU a Xsense ROS node is used which was compatible with the Xsense mti. This ROS node was available from github and no changes were made. This ROS node publishes the IMU data. For the anti slip control one node is made in C++. This node works as explained above. It retrieves all the data, does it calculations and publishes the needed data for the split block. The main program of the ROS node anti slip runs at a 100Hz. This is fast enough to run the robot smoothly in real time. How to run this program is shown in the Appendix B.

4 Testing

To test the new controller different tests must be done. In this section the different circumstances of these test are discussed and why they are done.

4.1 System test

To begin the full system will be tested. This can be done by driving the platform on a surface with insufficient friction and measure the response of the system. This will show how the system reacts on the slip of the wheels. Immediately the length of the moving average filter can be tested.

The controller will adapt itself depending on what slip the platform has. This slip will change due to the direction of the robot or what kind of surface it drives on. The scaled velocity changes and a moving average filter is used to do this smoothly. The filter will introduce a delay in the response dependent on the filter length. First this filter length will be tested. Making the filter to short will result in a bad behavior in driving. Making the filter to large causes a big delay, which will eventually lead to instability.

4.2 Comparison test using OptiTrack

The difference between the open loop and closed loop control must be tested. Because of the smooth surface of the floor the wheels slip, even if the lowest output velocity is given. On this surface no accurate test can be done to evaluate the results. A carpet will be used were the robot has more grip. The wheel slip causes the biggest problems when the platform is moving sideways. To have a reliable reference in the tests, the OptiTrack system at the RaM lab will be used. The OptiTrack can precisely measure the position of the robot. Both the open loop and the closed loop system will be tested by the OptiTrack on a surface with low and high friction. The low friction surface is the Ucrete floor of the university and the high friction surface is a rubber carpet

5 Results

In this section the results of the different tests will be shown and observations will be made.

5.1 System results

First the performance of the full system is looked at. For this a normal test is done by driving the platform over a surface with low friction forwards. For clarity of the graph only the velocity and the scaling of the left front wheel is shown, the other 3 wheels are not shown in the following graphs. The result is shown in Figure 5.1.

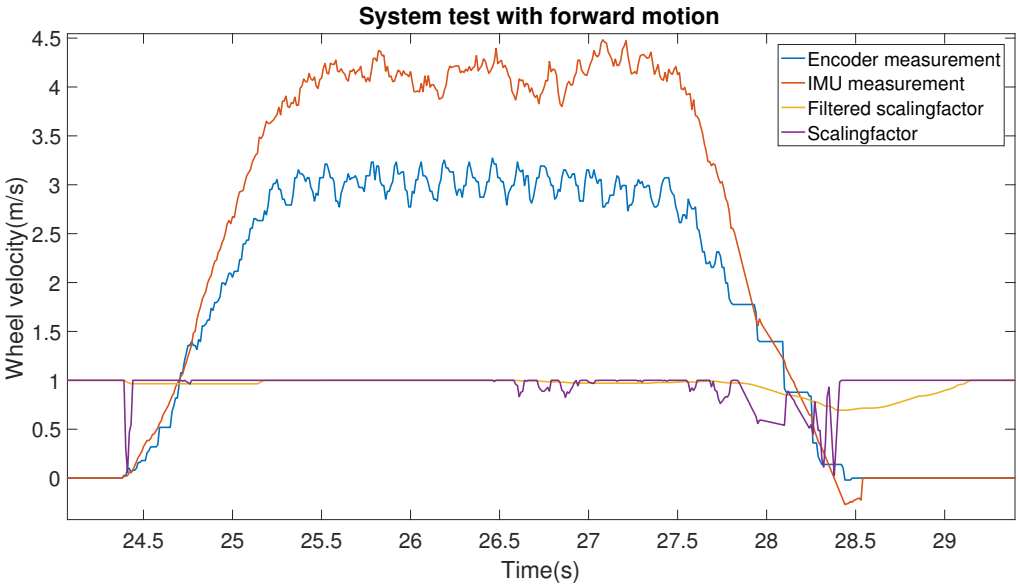


Figure 5.1: Forward motion of the Segway

The general performance of the system shows a good behavior of the system. A small oscillation can be seen, this is due to the shaking of the platform, which is caused by the rollers of the mecanum wheels. When the IMU velocity is higher or equal than the wheel velocity, the system does nothing because no slip is detected. But the figure shows that the scalingfactor changes while the IMU velocity is higher than the wheel velocity. The reason is that the figure shows the highest scalingsfactor of all the wheels. This means that another wheel detected slip which is not show in this figure.

It can be seen that the encoder does not have the same velocity as the IMU. The IMU velocity and the encoder velocity should be equal because no slip occurred in this test. Possible reasons are an IMU drift, an encoder error or a calculation error. The forward motion looking at equation 1 shows that for a forward motion no measured lengths of l_1 or l_2 is needed. This

means that it is not a calculation error. To test this a OptiTrack system is used. The velocities of the encoder,the IMU and the OptiTrack system is compared and shown in Figure 5.2.

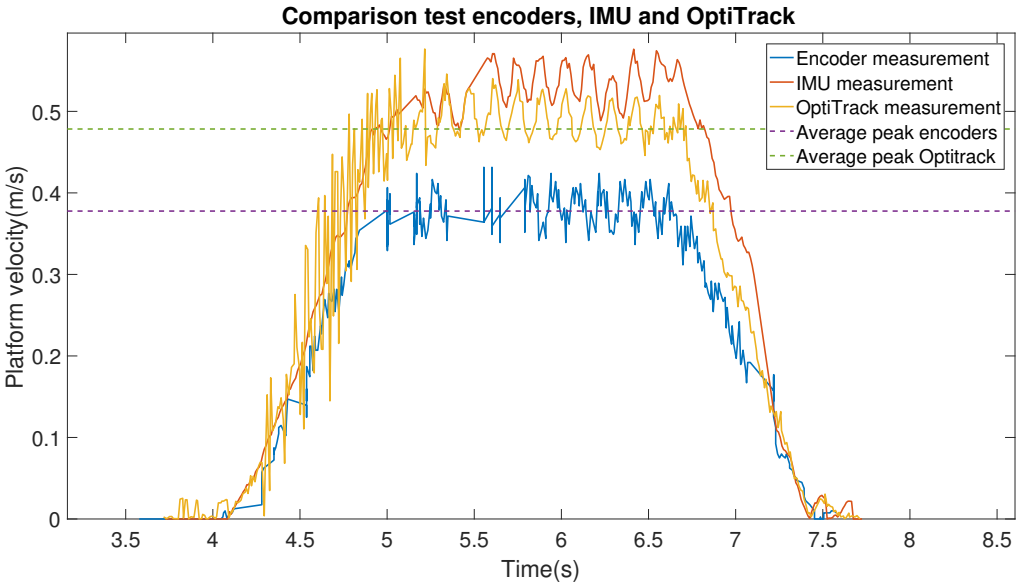
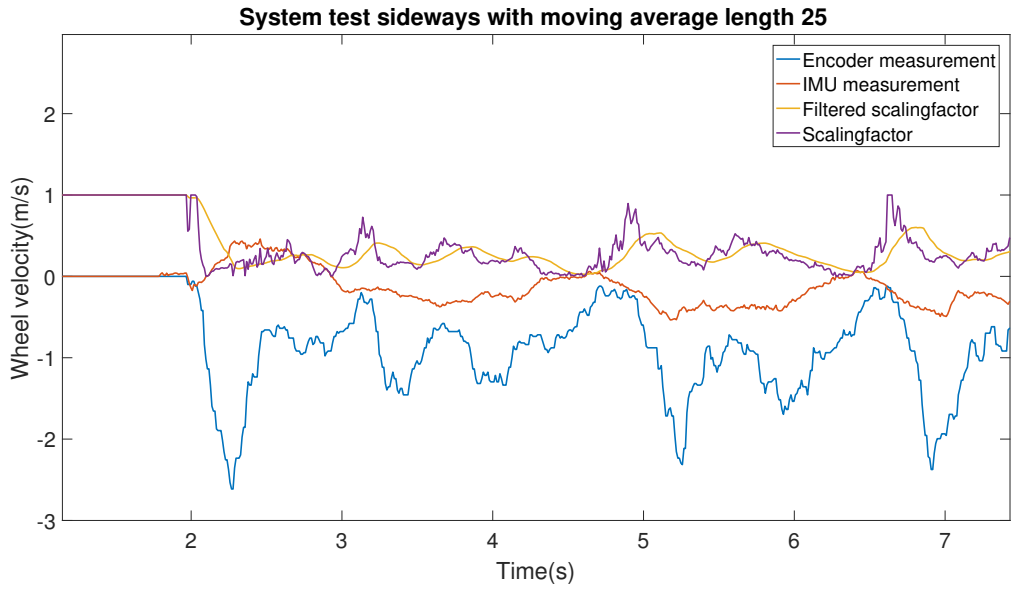


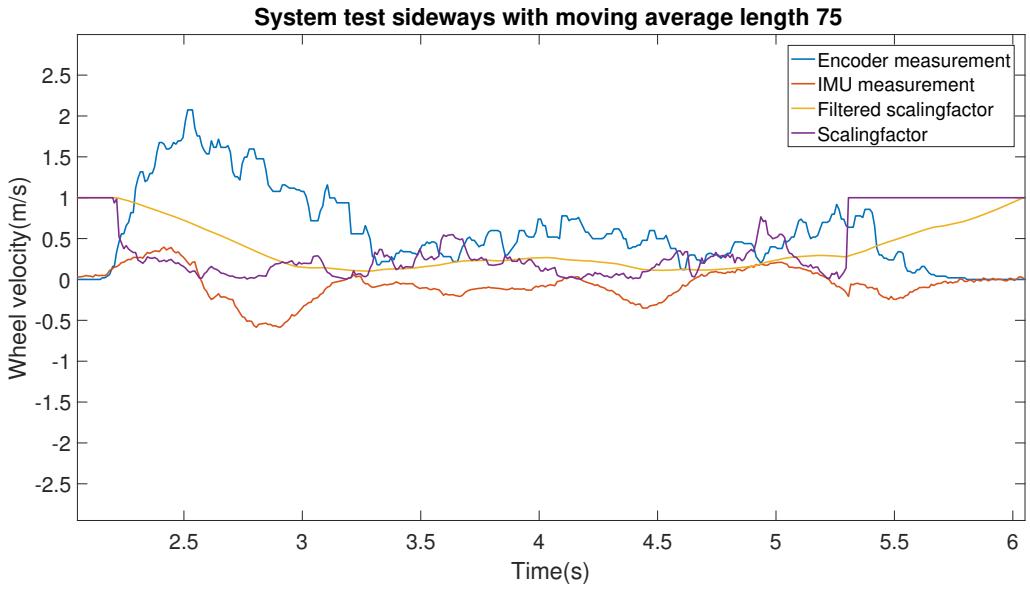
Figure 5.2: Comparison between the encoders, the IMU and the OptiTrack in a forwards motion

It is visible that there is a small difference between the IMU and the OptiTrack. The first difference is the peaks at acceleration of the OptiTrack. This instability is caused by a poor calibration of the OptiTrack. When changing to different cameras of the OptiTrack the position changes, resulting in the peaks. The next difference is at 5.5 seconds which show a bit of drift in the IMU. This is caused by a network issue, this can be seen by the fact that the encoders has at the same time a measurement error. The OptiTrack does not have a measurement error, because this system is connected to an Ethernet cable instead of WIFI.
There is a constant error in the encoder measurements. The average of the encoder and the OptiTrack have been calculated when driving at the highest velocity. These averages are shown in the graph. Based on multiple tests the difference between the averages is a factor of 0.78, which accounts for the difference seen in 5.1. This error happens in the velocity calculation of the Segway platform internally, where the velocity is based on the diameter of the wheel. The Segway has different wheels than standard used on this platform, this causes the error of the internal calculation. In the next results this error must be taken into consideration. This error will result in a lower slip ratio compared to the actual slip ratio of the wheels.

Not a lot of slip occurred in Figure 5.1. Figure 5.3 shows the result for sideways motion on a low friction surface with a moving average filter length of 25 and 75.



(a) System test moving sideways with moving average length of 25



(b) System test moving sideways with moving average length of 75

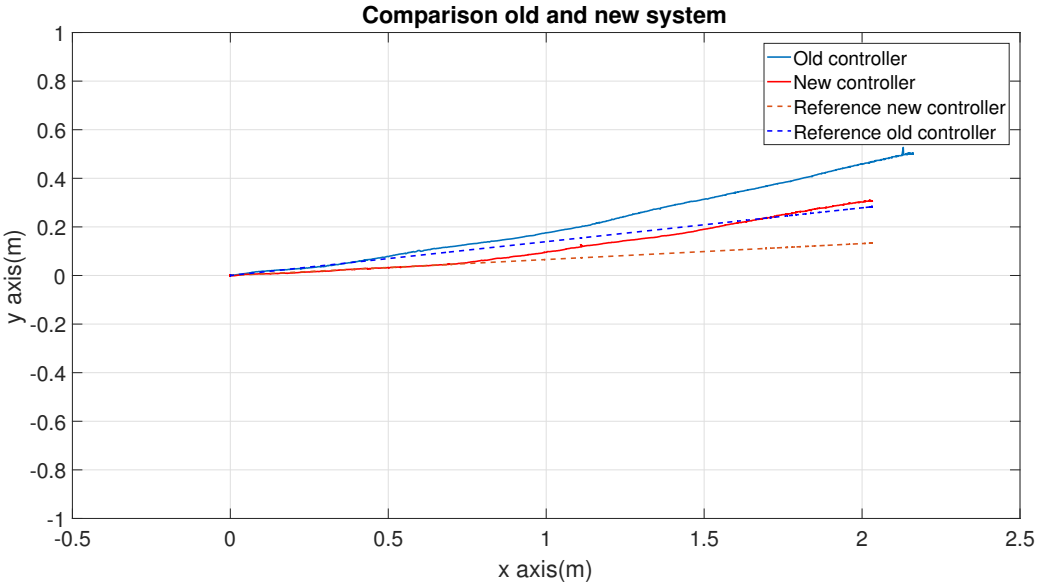
Figure 5.3: System test moving sideways with different moving average lengths

Figure 5.3 shows that both systems immediately reacts to the detected slip by looking at the scalingfactor. The scalingfactor shows the change based on the slip ratio, calculated using the velocity difference of the encoders and the IMU. This scalingfactor is filtered with a moving average filter. it can be seen that the response of Figure 5.3b is much more smooth and converges to one velocity, due to the longer filter length. The reaction time of this response

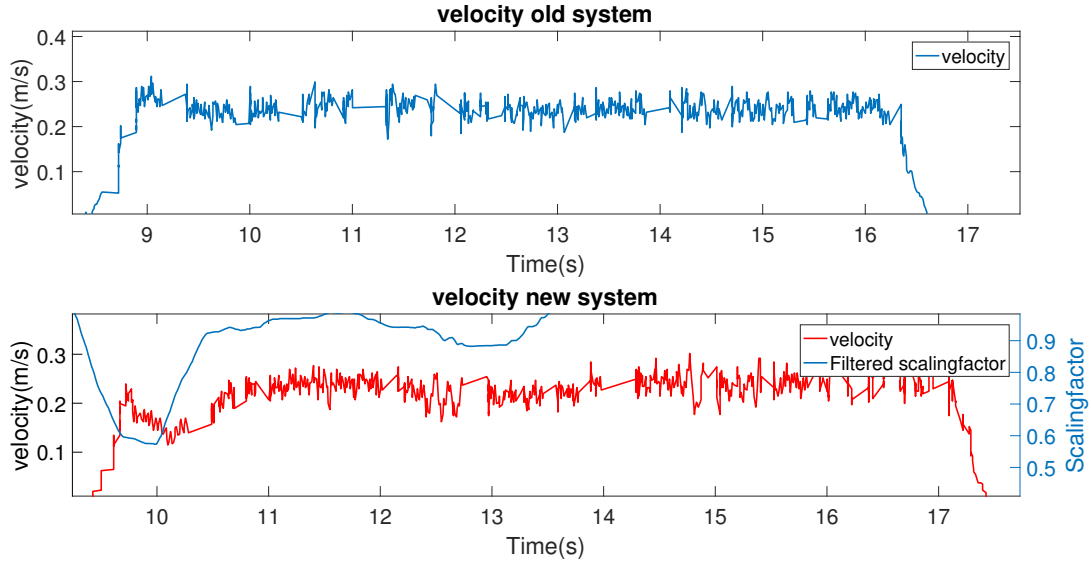
is within 1 second. Figure 5.3a shows a faster reaction time within 0.5 seconds but results in unwanted oscillation.
It is visible that the wheels have a much higher rotational velocity then is expected based on the IMU measurements. Resulting in a big scaling of the velocity in both systems. This scaling results in a low velocity of the platform.

5.2 Comparison results using OptiTrack

To compare the new closed loop system with the old open loop system the OptiTrack system is used. Two different test were performed. One with a surface with insufficient friction and one with sufficient friction. First the test with sufficient friction is done using a carpet. In Figure 5.4 the position and velocity of the platform is show moving to the right.



(a) Position of the old and new system



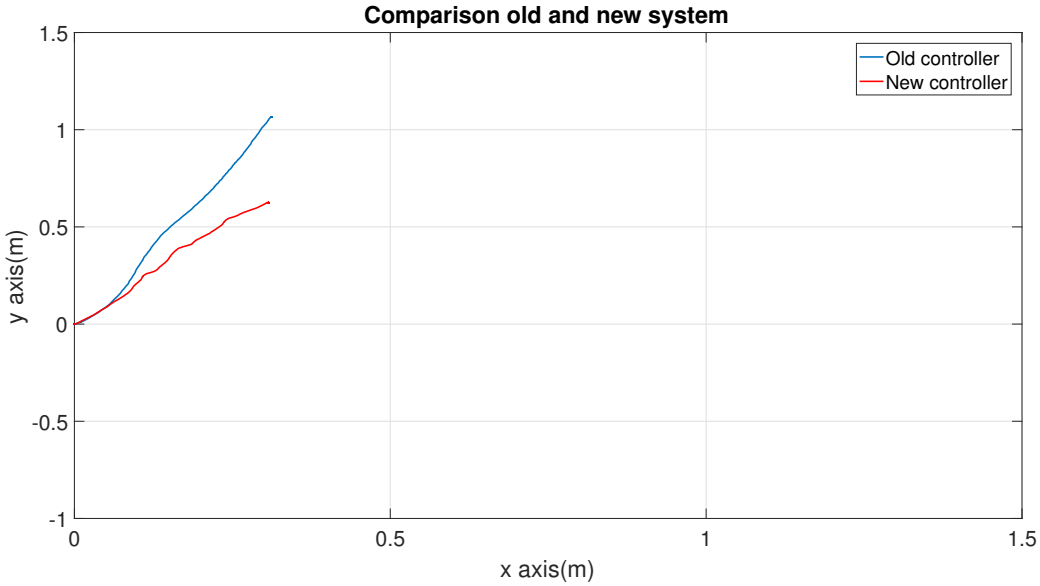
(b) velocity of the old and new system

Figure 5.4: comparing of the two systems on a carpet with sufficient friction

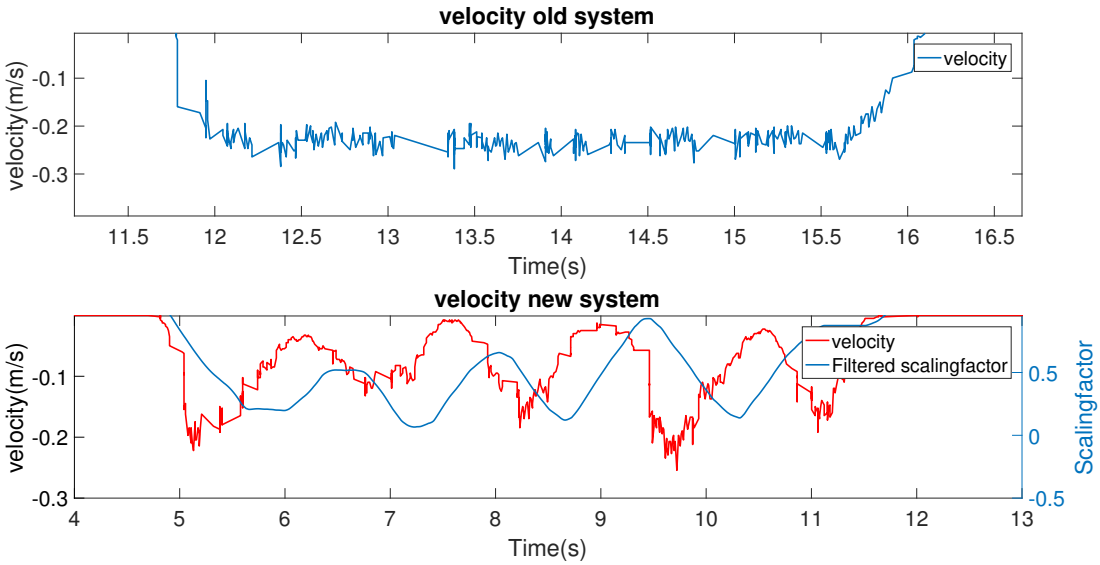
The input of the controller was a straight movement following the x axis. Due to alignment errors there is a small inputs difference between the two controllers and the x axis. The real trajectory of the new controller is determined by the first 0.5 seconds of its path, due to the straight line it moves. The difference between the trajectory of the new and old controller is known by the OptiTrack, this is used to plot the second trajectory. It can be seen in Figure 5.4a

that both controllers drive in the beginning in a straight line and later it deviates. The new controller follows it trajectory better but the differences are minimal. Figure 5.4b shows that the old system applies a constant velocity. The new system has in the beginning a scaling of the velocity due to detected slip, but still has a average velocity above the required minimum of 0.5km/h. Due to this detection of slip the trajectory of the new controller is more straight compared to the old controller. Figure 5.4 also shows that the robot always has a deviation into the same direction. In this case it is into the positive y direction, which is calculated and is more than the required five degrees. This is caused by the weight distribution of the robot.

To compare the difference between the controllers when a lot of slip occurs, a surface with insufficient friction has to be used. This can be seen in Figure 5.5. A velocity into the negative y axis is the input.



(a) Position of the old and new system



(b) velocity of the old and new system

Figure 5.5: comparing of the two systems on a surface with insufficient friction moving to the right

Figure 5.5a shows that with both systems the platform goes forwards instead of sideways and deviates way more than the required five degrees. The new controller has half of the deviation compared to the old controller, while running twice as long. This indicates that the system detects the slip and counteracts it. Looking at Figure 5.5b it can be seen that the velocity

oscillates and does not converge to one value. Also the velocity does not remain above the required minimum velocity
The big deviation and oscillation of the new system is due to the low friction between the floor and the wheels. This frictions is so low that it cant counteract all the slip.

The platform shows that it adapts to different surfaces as required, This can be seen by in the Figures 5.4b and 5.5b by looking at the different wheel velocities of the tests with high and low friction.

6 Discussion

In this section some observations during testing of the system are discussed.

6.1 Friction mecanum wheels

This thesis was based on removing the slip in mecanum wheels using a slip control with velocity adjustment. The big drawback were the wheels in this case. The wheels are made of a hard plastic with a very low friction coefficient. This resulted in slip, even when the lowest input velocity was given on a floor with low friction(University floors). This means that the new anti-slip control is not sufficient on this floor. The sideways motion seems most prone to slip.

6.2 Testing conditions

For the comparison tests two test conditions were made. On a floor with low friction, this was the university floor and a carpet with sufficient friction. The test were performed on both surfaces. Like explained in the section above the university floor caused the platform to slip using the smallest velocities. The results of the new control system showed better behavior than the old system, but due to this constant slip they were not exact enough to compare.
The carpet was used as the second condition. On this carpet both systems had enough grip. This resulted in minimum difference between the 2 systems.

6.3 Oscillation

in Figure 5.5b it can be seen that the new system oscillates when a lot of slip occurs. This oscillation showed that the system is unstable. there was no solution found, due to the time limitations. To solve this instability, the nonlinear system must be analyzed. This can be done in the future.

7 Conclusion and Recommendations

7.1 Conclusion

The Segway RMP 50 omni is a platform which can move omni directional, due to the forces which act upon the mecanum wheels. Slip can occur in these mecanum wheels, which is the consequence of a higher contact force than friction force. This slip creates a deviation in the desired direction.
The previous control is an open loop controller which does not take this slip into account. Different controllers have been looked into to counteract this slip, a slip controller using velocity scaling is used. For this controller a sensor is needed as reference point in combination with the encoders of the Segway to detect the slip, the extra sensor added is an IMU. The slip controller using velocity scaling is designed and implemented in the platform.
The closed loop controller showed that the controller responded to slip adequately, because the velocity was scaled when slip occurred. This reduced the slip significantly. The tests that are done were the two extremes. One surface with a high friction and one with a low friction. The high friction surface showed that both controllers worked properly, but still had a higher deviation than required due to the weight distribution. The low friction surface showed that the combination of the University floor and the wheels can not be solved, because the friction between the two is to low. Due to this low friction the minimum speed and maximum deviation was not made and the system started oscillating. These two extreme test conditions made it hard to compare the results.

7.2 Recommendations

Wheels

New wheels can be recommended, due to the low friction coefficient of the current wheels. When tests on the carpet were done it could be noticed that the plastic wheels have enough friction. This means that the wheels can be changed and add rubber like wheels introducing more friction.

Other testing conditions

Two testing conditions were done. One with the university floor and one with a carpet, due to the availability in the OptiTrack lab. This were the two extreme conditions. To test the system it can be recommended to use a kind of carpet were the old system would slip and the new system can, with the help of the slip feedback, maintain its grip on this carpet. This testing condition would show a better validation of the system.

Appendix

A Derivations

The derivations of the torques of the x-axis, y-axis and z-axis.

Derivation x-axis and y-axis

$$F = \frac{1}{R}\tau \tag{14}$$

$$F' = \frac{\sqrt{2}}{2}F \tag{15}$$

$$F_x = F_y = \frac{\sqrt{2}}{2}F' \tag{16}$$

$$F_x = F_y = \frac{1}{2R}\tau \tag{17}$$

Derivation z-axis

$$r = \sqrt{l_1^2 + l_2^2} \tag{18}$$

$$\tau_{\mathbf{z}} = \|r\| \, \|F\| \, sin(\theta) \tag{19}$$

$$\tau_{\mathbf{z}} = \sqrt{l_1^2 + l_2^2}F_r' \tag{20}$$

$$\tau_{\mathbf{z}} = \sqrt{l_1^2 + l_2^2}F' sin(\beta) \tag{21}$$

$$\tau_{\mathbf{z}} = \sqrt{l_1^2 + l_2^2}F' sin(\alpha + 45^\circ) \tag{22}$$

$$\tau_{\mathbf{z}} = \sqrt{l_1^2 + l_2^2}F' sin(tan^{-1}(\frac{l_2}{l_1} + 45^\circ) \tag{23}$$

$$\tau_{\mathbf{z}} = F'\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{\frac{l_2}{l_1} + 1}} + \frac{\frac{l_2}{l_1}}{\sqrt{\frac{l_2}{l_1} + 1}}) \tag{24}$$

$$\tau_{\mathbf{z}} = F'\frac{1}{\sqrt{2}}(l_1 + l_2) \tag{25}$$

Using equation 14 and 15:

$$\tau_{\mathbf{z}} = \frac{1}{2R}(l_1 + l_2)\tau \tag{26}$$

B Setup anti slip controller

The anti slip controller can be set up by using the following steps. It has been expected that the platform package is already installed i on the PC

1. Power the platform
2. Connect the controller to your own laptop
3. Create an antislip package in the Catkin workspace as well and place the code from Appendix C in the src folder and call it segread. Also make sure the CMake file of the interface is updated to the following:

```
cmake_minimum_required(VERSION 2.8.3)
project(antislip)

find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
)

catkin_package(

)

include_directories(
  ${catkin_INCLUDE_DIRS}
)
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(segread src/segread.cpp)
target_link_libraries(segread ${catkin_LIBRARIES})
```

4. Create an launch file in the platform package called antislip with the following code:

```
cmake_minimum_required(VERSION 2.8.3)
project(antislip)

find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
)

catkin_package(

)

include_directories(
  ${catkin_INCLUDE_DIRS}
)
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(segread src/segread.cpp)
target_link_libraries(segread ${catkin_LIBRARIES})
```

5. Go to the terminal of the platform PC

```
ssh platform@IP_MASTER
```

6. start a different segway controller.

```
roslaunch platform robinseg.launch
```

7. Open another terminal of the platform pc and start the xsense driver

```
roslaunch xsense xsensedriver.launch
```

8. Every terminal now used needs to know that the platform pc is the master. This is done by the following command:

```
export ROS_MASTER_URI=http://IP_MASTER:11311
```

9. Now launch the antislip file

```
roslaunch platform antislip.launch
```

10. launch the segread file

```
rosrun antislip segread
```

C Code

```
#include "ros/ros.h"
#include "math.h"
// #include "std_msgs/String.h"
#include "segway_rmp/SegwayStatusStamped.h"
#include "sensor_msgs/Imu.h"
#include "geometry_msgs/Twist.h"
#include "std_msgs/Float64.h"
#include<tf/LinearMath/Matrix3x3.h>
#include<tf/LinearMath/Quaternion.h>
#include <algorithm>
#include <iostream>
#include <vector>

#define MOVING_AVERAGE_LENGTH 75
float scaling, scaling1,scaling2,scaling3,scaling4;
float lfs,lrs,rfs,rrs; //slip per wheel(l = left f = front s = slip)
float lfw,lrw,rfw,rrw;//  wheel speeds of the encoder(w= wheel)
float lfwI,rfwI,lrwI,rrwI; // wheel speeds of the IMU(I = IMU)
float accX,accY;
float velX,velY,velZ;
float velOldX,velOldY;
float joyX,joyY,joyZ;
float movingAverage[MOVING_AVERAGE_LENGTH];
float average;
const float R = 0.125; // wheel radius
const float lx = 0.30019; // distances centre of the wheel and centre of the platform
const float ly = 0.13945;
bool still;
ros::Publisher vel_data;
ros::Publisher cmd;
geometry_msgs::Twist test;
geometry_msgs::Twist cmd_vel;

// %Tag(CALLBACK)%
void CallbackFront(const segway_rmp::SegwayStatusStamped::ConstPtr& msg) //writes the wheel
 ↪  velocities to variables
{
  lfw = msg->segway.left_wheel_velocity*(1/R);
  rfw = msg->segway.right_wheel_velocity*(1/R);
}
void CallbackRear(const segway_rmp::SegwayStatusStamped::ConstPtr& msg) //writes the wheel
 ↪  velocities to variables
{
  lrw = msg->segway.left_wheel_velocity*(1/R);
  rrw = msg->segway.right_wheel_velocity*(1/R);
}

void CallbackImu(const sensor_msgs::Imu::ConstPtr& msg) // writes the acceleration to a
 ↪  variable while canceling out the gravity using pitch and roll
{

  tf::Quaternion
   ↪  bq(msg->orientation.x,msg->orientation.y,msg->orientation.z,msg->orientation.w);
  double roll,pitch,yaw;
  tf::Matrix3x3(bq).getRPY(roll,pitch,yaw);
  accX = (msg->linear_acceleration.x + 9.81 * sin(pitch)) * cos(pitch);
  accY = (msg->linear_acceleration.y - 9.81 * sin(roll))  * cos(roll);
  velZ =  msg->angular_velocity.z;
```

```
55  }
56
57  void CallbackJoy(const geometry_msgs::Twist::ConstPtr& msg) // writes the twist from the
     ↪  controller to the variables
58  {
59    joyX=msg->linear.x;
60    joyY=msg->linear.y;
61    joyZ=msg->angular.z;
62  }
63  // %EndTag(CALLBACK)%
64
65  void checkJoy() // checks if the robot is driving by looking at the wheels velocities
66  {
67     if ((lfw==0) and (rfw==0) and (lrw==0) and (rrw==0)){
68       still=true;
69     }
70     else{
71      still=false;
72     }
73
74  }
75  void getVel() // calculates the velocity when driving from the IMU data, by integrating. When
     ↪  standing still everyting is set to 0.
76  {
77    if (still == true){
78      velX = 0;
79      velY = 0;
80      velOldX = 0;
81      velOldY = 0;
82      velZ = 0;
83      iets = 2;
84
85    }
86    else if (still == false){
87        velX = velOldX + accX*0.01;
88      velOldX = velX;
89        velY = velOldY + accY*0.01;
90      velOldY = velY;
91      iets = 1;
92    }
93  }
94
95  void getWheelVelocityImu() // calulates the wheel velocities bas on the intgerated data from
     ↪  the IMU
96  {
97    lfwI = (1/R)*(velX - velY - (lx+ly)*velZ);
98    rfwI = (1/R)*(velX + velY + (lx+ly)*velZ);
99    lrwI = (1/R)*(velX + velY - (lx+ly)*velZ);
100   rrwI = (1/R)*(velX - velY + (lx+ly)*velZ);
101 }
102
103 void calculateSlip()   //calculates the slip from the data of encoders and IMu, only when the
     ↪  IMU data is smaller than the encoder data
104 {
105
106   if (still == false){
107     if (fabs(lfwI) <= fabs(lfw)){
108       lfs = 1 - fabs(lfwI)/fabs(lfw);
109       ROS_INFO("false");
110     }
```

```
111       else{
112         lfs = 0;
113       }
114       if (fabs(rfwI) <= fabs(rfw)){
115         rfs = 1 - fabs(rfwI)/fabs(rfw);
116          ROS_INFO("false1");
117       }
118       else{
119         rfs = 0;
120       }
121       if (fabs(lrwI) <= fabs(lrw)){
122         lrs = 1 - fabs(lrwI)/(lrw);
123          ROS_INFO("false2");
124       }
125       else{
126         lrs = 0;
127       }
128       if (fabs(rrwI) <= fabs(rrw)){
129         rrs = 1 - fabs(rrwI)/fabs(rrw);
130          ROS_INFO("false3");
131       }
132       else{
133         rrs = 0;
134       }
135     }
136     else if (still==true){
137       lfs = 0;
138       rfs = 0;
139       lrs = 0;
140       rrs = 0;
141     }
142
143
144   }
145   void getScalingsFactor() // calculates the scaling of the velocity using the slip in the
        ↪  wheels.
146   {
147     if (fabs(lfs) < 1){
148       scaling1 = 1 - fabs(lfs);
149     }
150     else{
151       scaling1 = 1;
152     }
153     if (fabs(rfs) < 1){
154       scaling2 = 1 - fabs(rfs);
155     }
156     else{
157       scaling2 = 1;
158     }
159     if (fabs(lrs) < 1){
160       scaling3 = 1 - fabs(lrs);
161     }
162     else{
163       scaling3 = 1;
164     }
165     if (fabs(rrs) < 1){
166       scaling4 = 1 - fabs(rrs);
167     }
168     else{
169       scaling4 = 1;
```

```
170     }
171
172     scaling = fmin(scaling1, fmin(scaling2,fmin(scaling3,scaling4)));
173   }
174
175   void getMovingAverage() // moving average filter
176   {
177     for (int i = MOVING_AVERAGE_LENGTH; i > 1; i--) {
178       movingAverage[i-1] = movingAverage[i-2];
179     }
180     movingAverage[0] = scaling;
181     average = 0.0;
182     for (int i = 0; i < MOVING_AVERAGE_LENGTH; i++) {
183       average += movingAverage[i];
184     }
185     average /= MOVING_AVERAGE_LENGTH;
186   }
187
188
189   void scaleCmd() // scaling of the controller twist
190   {
191     if (abs(average) <= 1){
192       cmd_vel.linear.x = joyX* fabs(average);
193       cmd_vel.linear.y = joyY* fabs(average);
194       cmd_vel.angular.z = joyZ * fabs(average);
195     }
196     else{
197       cmd_vel.linear.x = 0;
198       cmd_vel.linear.y = 0;
199       cmd_vel.angular.z = 0;
200     }
201   }
202   void publish() // publishes all data for testing purposes.
203   {
204     test.linear.y = rfwI;
205       test.linear.z = average;
206     test.linear.x = rfw;
207     test.angular.y = scaling;
208       test.angular.z = scaling2;
209     test.angular.x = scaling3;
210
211   }
212
213   int main(int argc, char **argv){
214     ros::init(argc, argv, "segread");
215     ros::NodeHandle n;
216
217     ros::Subscriber sub_front = n.subscribe("/segway_rmp_node_front/segway_status", 1000,
        ↪  CallbackFront); // all subscribers
218     ros::Subscriber sub_rear = n.subscribe("/segway_rmp_node_rear/segway_status", 1000,
        ↪  CallbackRear);
219     ros::Subscriber sub_Imu = n.subscribe("imu/data", 1000, CallbackImu);
220     ros::Subscriber sub_Joy = n.subscribe("joy_vel", 1000, CallbackJoy);
221
222     ros::Publisher vel_data = n.advertise<geometry_msgs::Twist>("test", 1000); // all
        ↪  publishers
223     ros::Publisher cmd = n.advertise<geometry_msgs::Twist>("cmd_vel", 1000);
224
225
226     ros::Rate r(100);
```

```
227
228     while(ros::ok){ // main loop of functions
229       getVel();
230       checkJoy();
231       publish();
232       getWheelVelocityImu();
233       calculateSlip();
234       getScalingsFactor();
235       scaleCmd();
236       getMovingAverage();
237       ROS_INFO("vel x:: [%f]", velX); // all debugging
238       ROS_INFO("acc x:: [%f]", accX);
239       ROS_INFO("scaling :: [%f]", scaling1);
240       ROS_INFO("scaling :: [%f]", scaling2);
241       ROS_INFO("scaling :: [%f]", scaling3);
242       ROS_INFO("scaling :: [%f]", scaling4);
243       ROS_INFO("lfs :: [%f]", lfs);
244       ROS_INFO("lfs :: [%f]", lrs);
245       ROS_INFO("lfs :: [%f]", rfs);
246       ROS_INFO("lfs :: [%f]", rrs);
247       vel_data.publish(test);
248       cmd.publish(cmd_vel);
249       ros::spinOnce();
250       r.sleep();
251     }
252
253     // %Tag(SPIN)%
254     ros::spin();
255     // %EndTag(SPIN)%
256   return 0;
257   }
258   // %EndTag(FULLTEXT)%
```

References

[Diegel et al., 2002] Diegel, O., Badve, A., Bright, G., Potgieter, J., and Tlale, S. (2002). Improved mecanum wheel design for omni-directional robots. In *Proc. 2002 Australasian Conference on Robotics and Automation, Auckland*, pages 117–121.

[Doroftei et al., 2008] Doroftei, I., Grosu, V., and Spinu, V. (2008). Design and control of an omni-directional mobile robot. In *Novel Algorithms and Techniques in Telecommunications, Automation and Industrial Electronics*, pages 105–110. Springer.

[Junhui and Jianqiang, 2010] Junhui, L. and Jianqiang, W. (2010). Road surface condition detection based on road surface temperature and solar radiation. In *Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on*, volume 4, pages 4–7. IEEE.

[Kuo et al., 2016] Kuo, C.-H. et al. (2016). Trajectory and heading tracking of a mecanum wheeled robot using fuzzy logic control. In *Instrumentation, Control and Automation (ICA), 2016 International Conference on*, pages 54–59. IEEE.

[Nagatani et al., 2000] Nagatani, K., Tachibana, S., Sofne, M., and Tanaka, Y. (2000). Improvement of odometry for omnidirectional vehicle using optical flow information. In *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 1, pages 468–473. IEEE.

[Roelandts, 2015] Roelandts, T. (2015). The moving average as a filter.

[Salih et al., 2006] Salih, J. E. M., Rizon, M., Yaacob, S., Adom, A. H., and Mamat, M. R. (2006). Designing omni-directional mobile robot with mecanum wheel. *American Journal of Applied Sciences*, 3(5):1831–1835.

[Sharkawy, 2010] Sharkawy, A. B. (2010). Genetic fuzzy self-tuning pid controllers for antilock braking systems. *Engineering Applications of Artificial Intelligence*, 23(7):1041–1052.

[Soni et al., 2014] Soni, S., Mistry, T., and Hanath, J. (2014). Experimental analysis of mecanum wheel and omni wheel. *International Journal of Innovative Science, Engineering & Technology*, 1(3):292–295.

[Sukkarieh et al., 1999] Sukkarieh, S., Nebot, E. M., and Durrant-Whyte, H. F. (1999). A high integrity imu/gps navigation loop for autonomous land vehicle applications. *IEEE Transactions on Robotics and Automation*, 15(3):572–578.

[Tehrani et al., 2003] Tehrani, A. F., Doosthosseini, A. M., Moballegh, H. R., Amini, P., and DaneshPanah, M. M. (2003). A new odometry system to reduce asymmetric errors for omnidirectional mobile robots. In *Robot Soccer World Cup*, pages 600–610. Springer.

[Xsens et al., 2010] Xsens, N. et al. (2010). Mti and mtx user manual and technical documentation.