Budapest University of Technology and Economics
Department of Control Engineering and Information Technology

Viktor Kálmán

On modeling and control of omnidirectional wheels

PhD. dissertation

Advisor: Dr. László Vajta

Budapest 2013.

Abstract

Mobile robots are more and more commonly used for everyday automated transportation and logistics tasks; they carry goods, parts and even people. When maneuvers in cluttered environments are executed with expensive loads the need for reliable, safe and efficient movement is paramount. This dissertation is written with the goal of solving some problems related to a special class of mobile transport robots.

Omnidirectional wheels are well known in the robotics community, their exceptional maneuvering capabilities attract a lot of robot builders and several industrial applications are known as well. To fully exploit the advantages of these special wheels sophisticated mechanics and control methods are required. Since modern engineering uses simulation wherever possible, to eliminate design errors early in the development, the first objective of my research was to create an easy to use, yet realistic model in simulation for the general omnidirectional wheel. To create this model I used empirical tire models created for automotive simulation – to make advantage of the knowledge accumulated in this domain – and modified them to generate forces like an omnidirectional wheel, in essence transforming the longitudinal force component in the direction of the rollers. I created two embodiments as an example in Dymola environment and verified their functionality using kinematic equations and simple lumped robot models.

Omnidirectional wheels have no side-force generating capabilities, this is the very attribute that allows them to generate holonomic movement. Due to this characteristic omnidirectional platforms do not keep their direction of movement when braked intensively, instead they tend to swerve around the most loaded wheel. My second goal was to try and eliminate the swerving effect when braking. The braking problem involves a nonlinear MIMO system with uncertain parameters, since for example the ground-wheel contact depends nonlinearly on the unknown wheel load, ground and wheel material characteristics. I created a sliding mode controller that takes wheel center velocity vectors as input and actuates the brakes according to the given wheels' capability to reduce unwanted platform motion i.e. swerving. I also incorporated a factor to tune braking distance. I verified the performance of the controller in simulation, it showed high tolerance against structured and unstructured uncertainties in the platform model. The control law only uses knowledge on the kinematics of the robot, yet it works well with robots with dynamic properties.

To be able to control an omnidirectional platform, feedback of the motion is essential. Due to the special movement, traditional, wheel rotation based methods are highly inaccurate; therefore I directed my research towards optical feedback, using an image detector facing the ground and correlation. The method itself is not new most people know it from optical mice. There are many similar solutions in the literature, but they are not suited to movements up to several tens of m/s. To enhance the range of the sensor I investigated the possibility of using line scan cameras. They offer high speed and resolution in a single dimension. To verify whether movements off the main axis make measurements impossible I created a simulator in Matlab and made several experiments with a virtual sensor. My results show that by careful choice of parameters and processing, a line scan camera can be made insensitive to off axis movements and high speed one dimensional measurement is possible while moving in two dimensions.

The results of my research were evaluated in simulation and an early prototype of the speed sensor was incorporated in student projects.

Kivonat

Egyre inkább mindenapossá válik mobil robotok használata automatizált szállítási, logisztikai feladatokra. Szállítanak velük csomagokat, alkatrészeket, sőt embereket is. Amikor akadályokkal zsúfolt változó környezetben kell manőverezni, esetenként drága sérülékeny teherrel, fokozottan merül fel az igény megbízható, biztonságos, hatékony helyváltoztatásra. Ez a disszertáció azzal a céllal íródott, hogy megoldást adjon a mobil szállítórobotok egy speciális osztályának néhány problémájára.
Az omnidirekcionális kerekek széles körben ismertek, kivételes manőverezési képességük sok robotépítőt vonz és számos ipari alkalmazásuk is ismert. Képességeik teljes kihasználásához kifinomult mechanikai és irányítástechnikai megoldások szükségesek. A modern mérnöki munkamenet bonyolult rendszerek tervezésekor minél több szimulációt von be, már a tervezési fázisban a költséges tévedések kiküszöbölésére. Ezért választottam egyik kutatási célomnak egy általános realisztikus omnidirekcionális szimulációs kerékmodell megalkotását. A modell megalkotásához meglévő, kipróbált empirikus járműtechnikai kerékmodelleket használtam fel és ezek erő-karaktrisztikáit módosítottam, hogy az omnidirekcionális kerekekéhez hasonló viselkedést mutassanak. Többfajta megvalósítást implementáltam Dymola környezetben, működésüket kinematikai egyenletek alapján, egyszerű dinamikus robotmodellekkel ellenőriztem.
Az omnidirekcionális kerekek nem képesek oldalirányú erőket generálni, pontosan ez teszi lehetővé a holonom mozgást. Ez azt okozza viszont, hogy ha fékezünk egy ilyen kerekekkel ellátott platformot nem tartja meg eredeti mozgásának irányát, hanem hajlamos keresztbe fordulni a legjobban terhelt kerék körül. Második kutatási célom az volt, hogy megakadályozzam a fékezéskor jelentkező keresztbefordulást. A fékezési probléma egy nemlineáris, paramétereiben bizonytalan MIMO rendszerre vezethető vissza, ugyanis például a kerék-talaj kapcsolat nemlineárisan függ az ismeretlen kerékterheléstől és a bizonytalan talaj és kerék anyag paraméterektől. Csúszó módú szabályzót készítettem amely a kerekek középpontjának sebességét használja bemenetként és az egyes kerekeket attól függően fékezi, hogy azok az adott pillanatban képesek-e a nemkívánt – keresztbe fordító – sebesség komponenseket csökkenteni. A szabályzó lehetőséget biztosít a féktáv befolyásolására is. A szabályzó elvárt működését szimulációval ellenőriztem, nagyfokú érzéketlenséget mutatott a platform modell struktúrált és struktúrálatlan bizonytalanságaival és a visszacsatolás zajával szemben. A szabályzó kizárólag a kinematikai modell ismeretében is képes volt változó dinamikus paraméterekkel rendelkező platformok irányítására.
Az irányításhoz elengedhetetlen a megfelelő visszacsatolás megléte. A speciális mozgás miatt a hagyományos, kerék-elforduláson alapuló visszacsatolás pontatlan, ezért kutatásom az optikai rendszerek felé irányult. A talaj felé néző detektor és korreláció alkalmazásának ötlete nem új, az optikai egérből mindenkinek ismerős lehet. Az irodalom számos hasonló megoldásra hoz példát, azonban ezek egyike sem alkalmas nagysebbeségű – több tíz m/s – mérés elvégzésére. A technika mérési tartományának bővítésére vonalkamerát javasoltam, mivel egy dimenzióban nagy felbontást és képfrekvenciát biztosít. Annak vizsgálatára, hogy a tengelyiránytól eltérő mozgások lehetetlenné teszik-e az egydimenziós mérést Matlab szimulációt alkalmaztam és számos szimulációs kísérletet végeztem. Az eredményeim azt mutatják, hogy megfelelő paraméterválasztással és feldolgozással a vonalkamerán alapuló szenzor érzéketlenné tehető a tengelyiránytól eltérő mozgásokra és nagysebességű egydimenziós mérés végezhető kétdimenziós mozgás esetén is.

Declaration:

I Viktor Kálmán, hereby state that this PhD Thesis is my own work wherein I only used the sources listed in the bibliography. All parts taken from other works, either as word for word citation or rewritten keeping the original meaning, have been unambiguously marked, and reference to the source was included.

Nyilatkozat:

Alulírott Kálmán Viktor kijelentem, hogy ezt a doktori értekezést magam készítettem, és abban csak az irodalmi hivatkozások listájában szereplő forrásokat használtam fel. Minden olyan részt, amelyet szó szerint, vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Budapest 2013. 04. 17.

…………………………………
Kálmán, Viktor

Table of contents

List of theses

Acknowledgements:

I would like to express my appreciation to all that helped me during the years of my PhD work. I would like to thank my supervisor Dr. László Vajta for his inspirational ideas and support. I would like to thank members of the lab and the Department for creating an exceptional working environment.

I would also like to express my gratitude to Dr. Tamás Juhász for his valuable help getting started with the Modelica language, and members of the IFF Fraunhofer Magdeburg for receiving me at their lab.

I would like to thank my colleague Tibor Takács for his great work on the speed sensor simulator and his excellent suggestions and discussions on various topics while we worked together.

1 Introduction

This dissertation is written with the aim of solving some of the problems of advanced logistics robots, namely omnidirectional transport vehicles. Although the invention of the special wheels that move them dates back to the seventies, advances in mechatronics, and control technology keep them constantly on the drawing table of engineers working with mobile robots. This family of vehicles has serious advantages in mobility compared to traditional, wheeled propulsion systems, however their more complicated design and control, and the lower quantity of scientific background research and general understanding makes them a less attractive choice, even for applications perfectly suited for them. I tried to lessen this obstacle by solving some related problems.
Omnidirectional wheels are constructed so that a vehicle equipped with them can execute true holonomic movements, in other words it can change its direction of movement without changing its orientation. Their great movement capabilities however mean that their mechanical construction is complicated, they also require independent drive and control systems for each wheel. The rolling efficiency of these wheels is worse than that of regular wheels, also they generally do not perform very well on rough surfaces, i.e. they are best suited for indoors applications.

1.1 Omnidirectional wheels in simulation

During my research I created a vehicle model for simulation purposes, in an industry standard simulation language that can be adapted and used for a wide range of omnidirectional platforms. Probably the most important part of a vehicle model is the wheel, since this is the part that makes contact with the ground and transfers forces and torques to move the vehicle. In the last few decades a great number of wheel models of different levels of complexity have been constructed, and are used regularly in automotive and heavy truck simulations. This wealth of knowledge on wheel modeling however has not been applied to other areas of vehicle simulation, such as mobile robotics, although there are a lot of common features between the two. I created a configurable omnidirectional wheel model that can be adapted to work with most of the popular empirical wheel models used in vehicle simulation today. With the help of this simulation I created configurable omnidirectional platform models and made various experiments with the most widely used configurations, the four wheeled Mecanum platform and the three wheeled omnidirectional platform, sometimes referred to as the kiwi drive platform.

1.2 Problems during braking

Due to their design omnidirectional wheels in general have a quasi one dimensional force generation capability, they can only exert substantial force parallel to the roller axes, this is the very attribute that allows omnidirectional movement. As a consequence a platform can be pushed in any direction when the wheels are rolling free. The main problem however is that they tend not to keep their orientation during braking. This is caused by slight differences between wheel forces due to uneven load distribution, or ground friction variations. These unbalanced forces create a resulting torque and thus an angular acceleration around the center of gravity during braking. I identified and solved the underlying control problem and created a nonlinear controller that makes braking of omnidirectional platforms more predictable and safe. My brake assist controller uses sliding mode control to maintain directional control of the platform during emergency braking. The control law only uses kinematics information of the platform, the relative position of the wheels and the direction of their axes and rollers. This means that the controller has a high tolerance regarding changes in the dynamic model i.e. load distribution and ground contact characteristics. Also it works with any kind of omnidirectional wheel. I demonstrated the disturbance rejection of the brake assist controller with several examples in simulation, on the two most widely used omnidirectional platforms.

1.3 Velocity feedback

Many mobile robots operate with large amounts of wheel slip, some of them have an uncertain center of rotation, some have no wheels, making traditional – wheel rotation based – dead-reckoning methods highly inaccurate. To deal with this problem I proposed a new optical speed measurement system that produces accurate two dimensional velocity measurements up to a very high speed, independent from wheel rotations, and platform kinematics. The method in itself is not new, it is similar to the working principle of a common optical mouse: snapshots of the ground are taken with a certain frequency and consecutive images are compared. The displacement of texture patterns and the snapshot frequency gives the speed of movement. My approach is new in the sense that I did not use a matrix camera but a line detector and showed that accurate one dimensional measurements can be made while moving in two dimensions. These devices feature line frequencies at the order of kHz at a very reasonable price. This enables speed measurement in the practical velocity range of most ground vehicles. Putting three of these line camera based sensors on a platform, movement along a surface can be measured in three degrees of freedom, independent from platform kinematics. This property makes the system useful for example for the class of holonomic mobile robots, independent from propulsion.

1.4 Methods and tools

Most of the work has been carried out in Modelica – Dymola and Matlab simulation environments.

Modelica is a free object-oriented modeling language with a textual definition to describe physical systems in a convenient way, by differential, algebraic and discrete equations. It is supported by the Modelica Association[1]. “It is suited for multi-domain modeling, for example, mechatronic models in robotics, automotive and aerospace applications involving mechanical, electrical, hydraulic and control subsystems, process oriented applications and generation, and distribution of electric power. Modelica is designed such that it can be utilized in a similar way as an engineer builds a real system: First trying to find standard components like motors, pumps and valves from manufacturers' catalogues with appropriate specifications and interfaces and only if there does not exist a particular subsystem, a component model would be newly constructed based on standardized interfaces.

Models in Modelica are mathematically described by differential, algebraic and discrete equations. No particular variable needs to be solved for manually. A Modelica tool will have enough information to decide that automatically. Modelica is designed such that

[1] https://modelica.org (Accessed 2012. Feb.).

available, specialized algorithms can be utilized to enable efficient handling of large models having more than hundred thousand equations. Modelica is suited (and used) for hardware-in-the-loop simulations and for embedded control systems.”[38] From my point of view the main attractiveness lies in the languages’ object oriented nature, which allows a convenient incremental development workflow. Another attractive feature is the model building philosophy of describing the systems by algebraic differential equations, thus approaching the problem from a physics point of view, as opposed to a mathematical one, which – in my experience – is less appealing to an engineer.

Matlab is a widely known computational software package, with a broad range of mathematical toolboxes in almost every domain of science and engineering. It is generally used for verifying and prototyping algorithms, but it can also be used for simulation and even real time computation. I used it to create the simulator for the speed sensor (section 4.4.) with the help of T. Takács, an MSc. student at the time. Leveraging the built in image manipulation, distance metric, and plotting functions I could concentrate on the sensor related problems. Because of this Matlab proved to be an excellent choice for the task.

2 Omnidirectional wheel simulation

2.1 Motivation

Omnidirectional wheels have been invented quite a long time ago [24] and they have a rich history in the literature. They have been used for various tasks and many different embodiments are known [41], [9]. Their use ranges from robotic soccer applications, through industrial heavy load transporters [43], and vehicle simulators [1], to educational and entertainment projects like the popular inverted pendulum, but mounted on a ball [30], [5]. Figure 1. shows some of these applications.

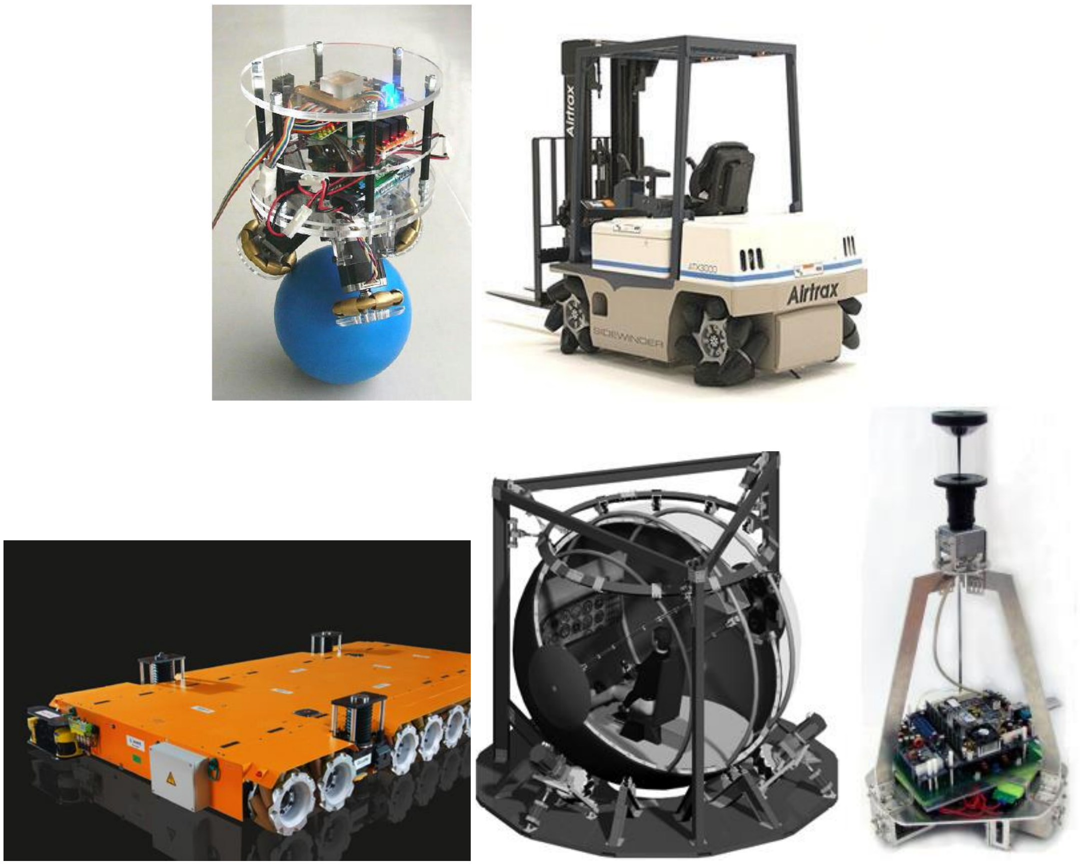


Figure 1. Examples for the use of omnidirectional wheels[2]

Modern engineering uses simulation for almost every task imaginable, to cut costs, speed up development and minimize changes late in the product life cycle. Omnidirectional platforms are no exception since they require more complex mechanical design and control, than traditional vehicles.

As I found no publicly available simulation libraries for Mecanum wheels I decided to create one myself. The rollers are often covered with rubber, especially in heavy duty applications; this means the rollers themselves behave much like a small tire with a solid carcass. The behavior of rubber under dynamic conditions is not trivial to describe, tire research is a science in itself with around 80 years of history [19]. I decided to build on the work of others and lay down the foundations of how to modify existing tire models to simulate omnidirectional wheels.

[2] [30], www.airtrax.com, www.kuka-omnimove.com, [1], [49]

2.2 Omnidirectional wheel models

This section gives a short overview on the types of omnidirectional wheels and their simulation from the literature. A great number of omnidirectional wheel users are industrial companies and users of their product, the most significant being KUKA Robotics[3], and Airtrax. The second group of people who use them are students, preparing for robotics competitions such as RoboCup and others, or scientists working on advanced mobility concepts. This second group creates the majority of publications.

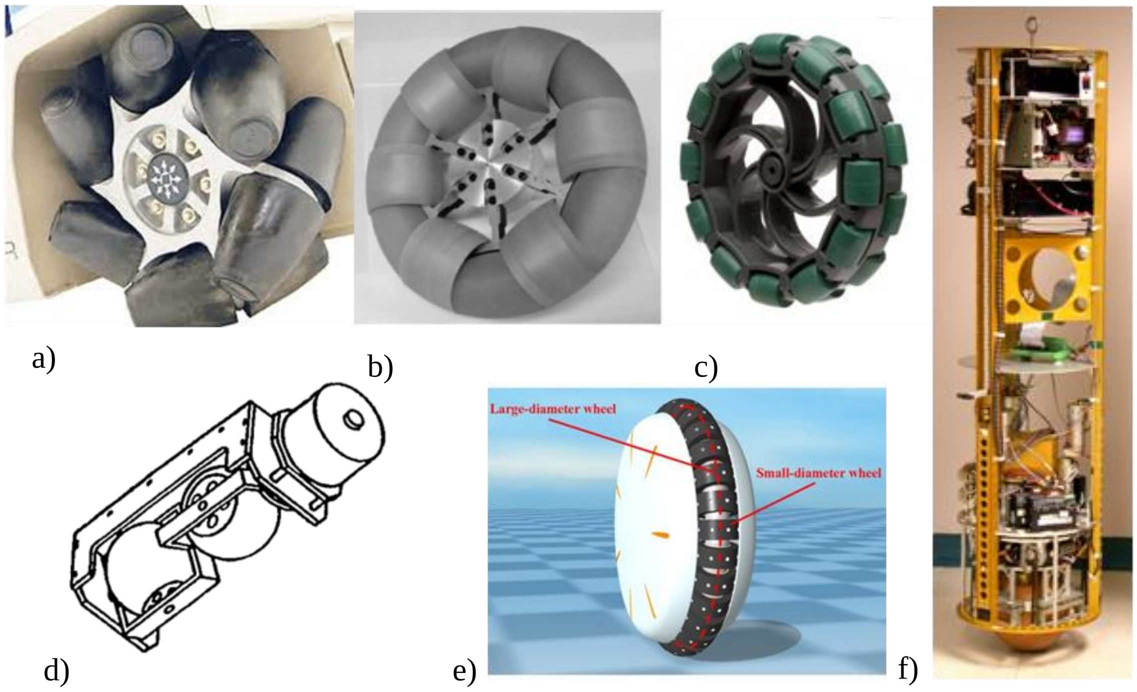


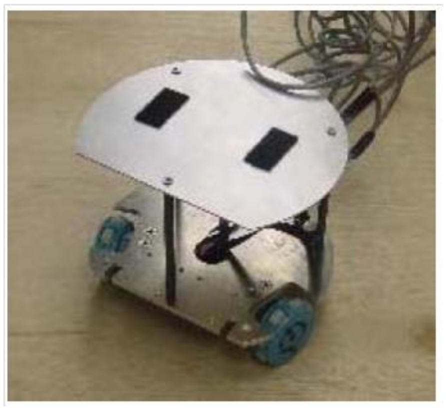
Figure 2. Different omnidirectional wheels and mobility concepts

a) Airtrax Mecanum wheel b) enhanced profile omni-wheel [10] c) multiple row wheel[4] d) Killough wheel[5] e) wheel of a Honda U3-X personal mobility platform f) Ballbot, omnidirectional balancing robot [31]

Figure 2. shows a cross section of some of the wheel designs and interesting mobility concepts. I collected them to demonstrate relevant concepts trough examples. Subfigures a), b) and c) represent the most popular wheel configurations i.e. passive rollers on the perimeter of a wheel. a) shows the Mecanum wheel used on the Airtrax forklift. It is worth noting that the rollers are shaped in an attempt to attain a round profile and a single roller touching the ground at a time. One of the problems associated with omni-wheels is the rough ride associated with changes in wheel radius when changing roller contact. Another important effect is caused by the rigid discontinuities between rollers, they cause slip especially on soft surfaces, such as a carpet [55]. b) and c) are examples of the most widely used solutions to these problems. b) uses different sized rollers, where the larger diameter rollers are shaped so that they can fit the smaller rollers inside, thus virtually eliminating the non-rolling surface on the circumference [10]. This obviously comes at the price of increased complexity. c) is a more common solution, by using multiple regular wheels mounted side by side at an angle, “bumpiness” and roller discontinuities can be minimized.

[3] http://youbot-store.com/, http://www.kuka-omnimove.com
[4] http://www.vexrobotics.com
[5] http://www.h33.dk/opfhjul_index.en.html

The remaining three subfigures show somewhat different mobility concepts. d) shows the so-called Killough wheel, named after the inventor [41]. In this concept two quasi ball-shaped rollers are mounted in rigid brackets that are connected perpendicular to each other. The rollers are free to roll and the bracket assembly is driven by a motor. These wheels should be mounted and applied just like the omni-wheels above. (Kinematic constraints are explained in section 2.5.1.) Smooth ride and single roller contact is ensured by the shape of the rollers, the contact point however moves significantly when roller contact changes.
Subfigure e) shows the wheel of Honda U3-X[6] personal mobility platform. A seat is mounted on top of the wheel and the vehicle balances and drives on this single wheel in an omnidirectional fashion. This is achieved by having powered rollers in addition to the main drive that turns the entire wheel.
f) shows an omnidirectional platform that clearly eliminates any rolling imperfections by using a ball to ride on. It is called Ballbot and it was designed to work in areas used by people [31]. It is high enough to make eye contact yet it has a small footprint, that together with omnidirectional maneuverability enables it to get around in cluttered indoor environments.
In this dissertation I worked with the type of wheels represented by the first three subfigures. A common characteristic is that they have a relatively small width relative to their diameter and they are designed with an attempt to ensure smooth ride. In the following let us take a look at how this type of wheels has been modeled by other researchers.
Williams et al. [55] developed a wheel model motivated by the RoboCup competition. They used a small three wheeled platform with 0° rollers (Figure 3. a). The wheels they used had a single row of rollers, without any provisions to smoothen roller discontinuities, this was reflected in their results, the wheels demonstrated a strong angle dependent friction characteristics directly related to the non-rolling part touching the carpet they used for testing. It is also important to note that they found that the friction coefficient in the driven and in the free rolling direction was comparable – 3/1 and 5/3 respectively for paper and carpet – showing that the quality of the omni-wheel greatly effects the behavior of the mobile platform.


a)

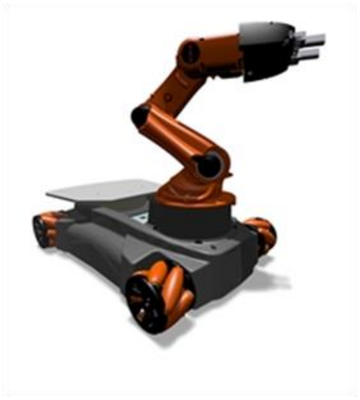

b)

Figure 3. a) Omnidirectional RoboCup player by Williams et al. [55], b) youBot by KUKA Robotics, flexible arm on a mobile base

Dresscher et al. [16] incrementally developed a model for youBot (Figure 3. b) using an energy based method and its bond graph representation. Their goal is to model this

[6] http://www.hondanews.com

rather complicated platform in a modular, reusable fashion, this also includes modeling of the Mecanum wheels. To make the model simpler they neglected dynamic behavior of the wheels and derived a kinematical model from the geometry. They neglected friction in the roller bearings and generally neglected force in the free rolling direction. They defined a transformation between drive axis movement and wheel movement in the direction parallel to the roller axis. Floor contact was modeled with a resistance and a stiffness parameter. The authors had no opportunity to validate their model on the real platform.
Tobolár et al. [51] created an object oriented library for Mecanum wheels in Modelica, unfortunately their article is very short and non-informative. As I learned from the author this is due to an NDA with KUKA Robotics.
Studying the literature the conclusion can be drawn, that in many cases omnidirectional platforms are modeled as a whole, assuming symmetrical load distribution, without having separate wheel models. However when wheels are modeled, dynamic effects are often neglected and the results are purely kinematical. This is probably justified when the platform has very low, known weight, for example a RoboCup player. Another common modeling approach is that wheel forces are assumed to be generated parallel to the direction of the rollers and forces perpendicular to the roller axis are assumed to be zero. An exception is the paper mentioned before [55], where the authors had to calculate with substantial forces in the free rolling direction, however in my opinion this was due to the disadvantageous characteristics of the omni-wheel they used.
To be able to apply a wheel model that accommodates a broad range of robotic platforms including heavy machines with uneven load distribution and various wheel designs with different roller materials, a model is needed that is easy to parameterize, includes simple dynamics, handles sliding and last but not least of all, well suited for simulation. For this I decided to build on the well proven results from the domain of regular tire modeling, while applying some of the techniques used in the literature cited above.

2.3 Regular tire models in simulation

Tire modeling in general has been an active area of research for a long time, because the behavior of a tire is a complex phenomenon, and the results can be used in countless applications, making it both a challenging and lucrative area of research. The main purpose of a tire, besides providing a smooth ride for the passengers is to transmit forces and torques in three mutually perpendicular directions to create vehicle movement and directional control. To achieve this, a tire model has to handle collision, calculate the contact patch with the ground and obstacles, and it has to generate the forces and torques that arise. Most of these calculations are nonlinear because of the characteristics of the tire material [8].
The complexity of a tire model depends on the application, more precisely the detail needed in the simulation output. The simplest models regard the tire as a rigid disk, with unchangeable radius and linear dynamic properties, the most complicated ones use finite element simulation, fine tuned to a certain rubber compound and carcass. For a tire model to be useful, a compromise between complexity and accuracy has to be found depending on the application at hand. A very good example of incremental model building in Modelica is given by [57].
Except for simple targeted experiments the model cannot be restricted to a certain driving situation. Most of the relevant cases have to be considered, such as driving with nonzero camber and sideslip angles. Another important aspect is the adaptability of tire

model characteristic parameters to real world tire behavior. A substantial number of tire models fall into the category of empirical models. These are based on measurements with real tires. Polynomial functions are fitted to the measured data points, and then the tire can be characterized by the coefficients of these polynomials. Figure 4 shows an example, the horizontal axis represents slip, the vertical axis is longitudinal friction force (in the direction of rolling). Slip is usually defined in relation to the difference between a wheel center's velocity and its circumferential velocity (see details in the next section). It is often confused with sliding, but it is important to see that slip occurs with a perfectly gripping tire as well. It can be thought of as the driven axis twisting the elastic tire material around itself.

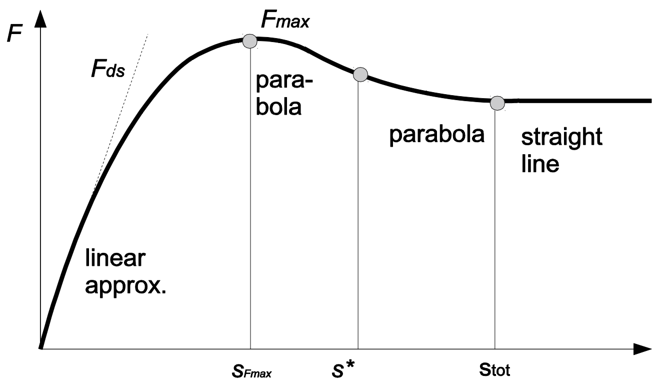


Figure 4. Longitudinal friction force generation vs. slip (after [21])

It is clear from the figure, that force is represented according to different polynomials depending on the slip value. A good example of empirical modeling is the well-known Pacejka and Rill tire models [39], [42]. This modeling approach implies that many of the model parameters have no explicit physical meaning, they only represent the coefficients of certain polynomials. This makes the use of these models complicated, one has to have real measurements of real tires to obtain useful parameters.
This is the reason why I decided choose a model for my initial experiments that was created to avoid this problem, with ease of use as a main modeling approach. The model is described in the next section (2.3.1). One can read about in detail in [21],[42], here I will only highlight some pieces of information that are needed for understanding the following sections.

2.3.1 The Rill tire model

TMEasy [21] is very user friendly and easy to use, which makes it a popular modeling choice. In order to be easy to use it takes the insufficiencies of the availability of reliable modeling data into account. It achieves this goal by using a rather limited number of parameters, with a more or less direct physical meaning. This makes it possible to make incomplete tire datasets complete by identification or educated guess. The force generation of the tire material is due to elastic deformation, this can be best described by wheel slip. In the direction of rolling (x) slip is defined by the difference of a wheels circumferential velocity and the velocity of its center, divided by the wheel's circumferential velocity. Slip in the direction parallel with the wheel axis (y) is defined differently, for this the lateral velocity of the wheel center is divided by the wheel's circumferential velocity. This is described by the following equations, from [21]:

$$s_x = -\frac{(v_x - r_D\Omega)}{r_D|\Omega|} \text{ and } s_y = -\frac{v_y}{r_D|\Omega|} \qquad (1), (2)$$

where v_x, v_y are components of the contact point velocity in lateral and longitudinal direction, respectively. Ω is the angular velocity of the wheel and r_D is the dynamic rolling radius. They can be vectorially added to obtain a generalized slip value.
The dynamic rolling radius is introduced to accommodate tire deformation due to vertical load. In this model the difference relative to the non-deformed state is calculated from the wheel load and a vertical stiffness constant [42].
Table 1. shows some parameter values for a typical tire. The combined force characteristics are directly generated via a generalized slip approach which does not need any additional fitting parameters.

Description	**Name**	$\boldsymbol{Load_1}$	**Name**	$\boldsymbol{Load_2}$
Nominal normal force	F_{znom1}	3000N	F_{znom2}	6000N
Slope at $s_x = 0$	F_{ds_x1}	50000N	F_{ds_x2}	75000N
Slip of maximum tire force	s_{max_x1}	0.15	s_{max_x2}	0.18
Maximal tire force	F_{max_x1}	3000N	F_{max_x2}	4500N
Slip where sliding begins	s_{slide_x1}	0.4	s_{slide_x2}	0.5
Force where sliding begins	F_{slide_x1}	2800N	F_{slide_x2}	4200N
Slope at $s_y = 0$	F_{ds_y1}	40000N	F_{ds_y2}	60000N
Slip of maximum tire force	s_{max_y1}	0.21	s_{max_y2}	0.24
Maximal tire force	F_{max_y1}	2750N	F_{max_y2}	4125N
Slip where sliding begins	s_{slide_y1}	0.6	s_{slide_y2}	0.8
Force where sliding begins	F_{slide_y1}	2500N	F_{slide_y2}	3750N

Table 1. Example parameters for a tire according to the Rill tire model

These values describe the static characteristic slip – force curves of the tire for a given value of road friction coefficient. Since the force is load dependent, different force curves describe the force generation of differently loaded wheels. An example is shown on Figure 5. The vertical forces F_{zi} increase in magnitude with the index i. In Table 1. two nominal load values are shown $Load_1$ is the nominal load, $Load_2$ is the maximal permissible load.

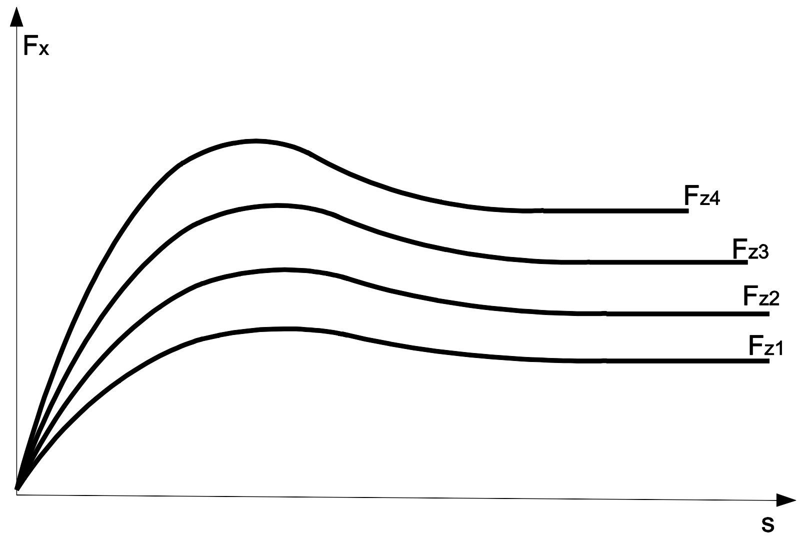


Figure 5. Load dependence of horizontal force

For values between the two nominal values a nonlinear second order interpolation is used. The quadratic interpolation for $F_z < 2F_{znom1}$ for any value is demonstrated trough the example of F_{ds_x}, the initial slope of the force – slip curve:

$$F_{ds\_x} = \left[\left(\frac{F_{dsx_2}F_{znom1}}{F_{znom2}} - \frac{F_{dsx_1}F_{znom2}}{F_{znom1}}\right)\left(\frac{F_{znom1} - F_{znom}}{F_{znom1} - F_{znom2}}\right) + \frac{F_{dsx1}F_{znom}}{F_{znom1}}\right]\frac{F_{znom}}{F_{znom1}} \tag{3}$$

All parameters are interpolated similarly.

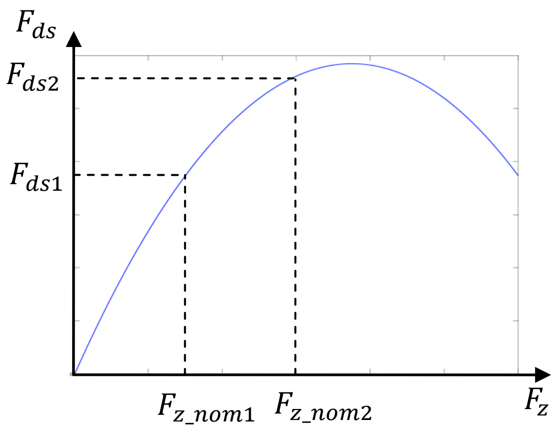


Figure 6. Quadratic interpolation of F_{ds} gradient for a given slip value

This interpolation implies that for most values there are two solutions (inverted parabola see Figure 6.), one of them is physically correct the other is invalid, meaning for example a tire overloaded outside the boundaries of the model. Figure 6. shows the interpolation curve for the slope at $s_x = 0$ i.e. the steepness of the initial, linear part of the curve.

The physical meaning of the quadratic interpolation in this case is, that with more load the tire does not get proportionally more force capability, but rather compresses and loses some traction. The values above the maximum nominal load have no physical meaning, this shows the boundaries of the model. During simulation one has to be careful not to overload the wheel and get invalid results.

The model uses first order dynamics. According to [21] the dynamic reaction of the tire forces and torques to disturbances can be approximated quite well by first order systems. Thus the dynamic behavior is modeled by a simple spring and damper model as illustrated on Figure 7.

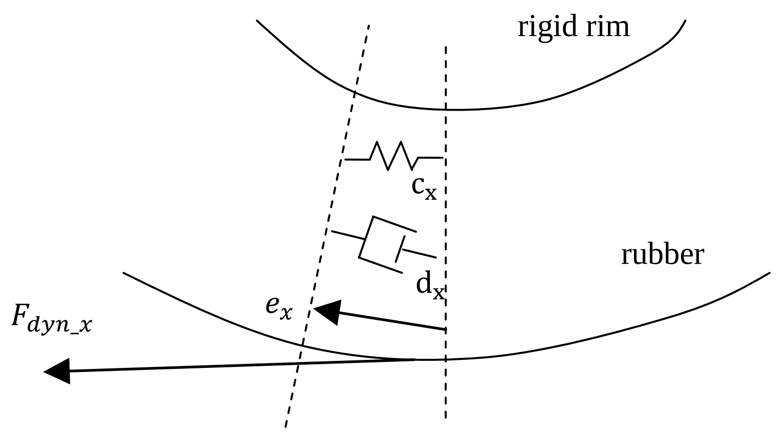


Figure 7. Tire deflection, first order tire dynamics

The dynamic force is generated from the elastic deformation e_x of the tire material:

$$F_{dyn\_x} = c_x e_x + d_x \dot{e}_x \tag{4}$$

These values may also be set intuitively, to make the wheel stiff or more compliant.

2.3.2 Contact calculation

Simple contact model of a regular tire: Forces and torques are transmitted between the ground and the tire at the contact point, the effect of contact forces can be fully described by resulting force and torque vectors at a specific point of the contact patch. The uneven ground can be described by a function of two spatial coordinates $z = z(x, y)$ and it is approximated by a local road plane. This local plane is characterized by its unit normal e_n.

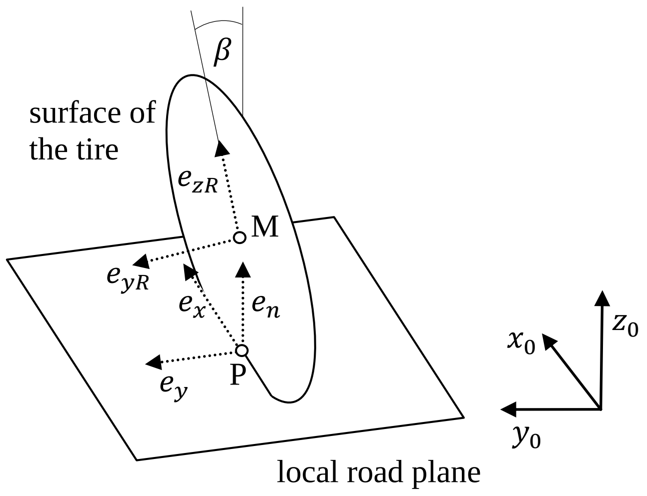


Figure 8. Contact model [21]

This normal can be computed, for example by taking four points in the ground function at a short distance forward, backward, left and right from the ground projection of the wheel centre, Q_1, Q_2, Q_3, Q_4. The vectors pointing between them are indexed by the corresponding point indexes e.g. r_{21} points from Q_1 to Q_2. Then:

$$e_n = \frac{r_{21} \times r_{43}}{|r_{21} \times r_{43}|} \tag{5}$$

this local linearization smoothes out local discontinuities and sharp bends from the surface, that occur in reality [21].
The wheel center plane is defined by the unit vector in the direction of the wheel rotation axis e_{yR}. e_x lies in the intersection of the local road plane and the wheel center plane and defines the direction of longitudinal force. The lateral force is described by the unit vector $e_y$which is perpendicular to both the track normal e_n and e_x. The camber angle $\beta = \arcsin(e_{yR}^T e_n)$ describes the inclination of the wheel in respect to the local road plane and defines the direction of e_{yR}. The geometrical contact point P lies in the intersection of the local road plane and the wheel center point M and it is the point to the shortest distance to it. Its location is described by:

$$r_{0P} = r_{0M} + r_{MP} \tag{6}$$

where r_{0M} describes the wheel center position in global coordinates and the vector from the center M to the geometric contact point P can be written as: $r_{MP} = -r_S e_{zR}$, where r_S is the static tire radius and $e_{zR} = e_x \times e_{yR}$ is the radial direction.
For a cambered tire, due to tire deformation effects it is usually more accurate to calculate the position of the so called static contact point, which estimates the point with higher pressure, the sideways deviation can be calculated from camber angle β, tire width b, and tire deflection Δz:

$$y_Q = -\frac{b^2}{12\Delta z}\frac{\tan(\beta)}{\cos(\beta)} \tag{7}$$

Kinematics of an omnidirectional wheel: Since the surface of omnidirectional wheels are more complex, than that of regular wheels because of the rollers, their contact model deserves a bit more investigation. For this derivation it is assumed that the wheel axis is parallel to the ground i.e. camber angle is zero ($\beta = 0$). Gfrerer [18] developed a geometrical model for the construction and kinematical investigation of Mecanum wheels. This section is mainly based on this article.

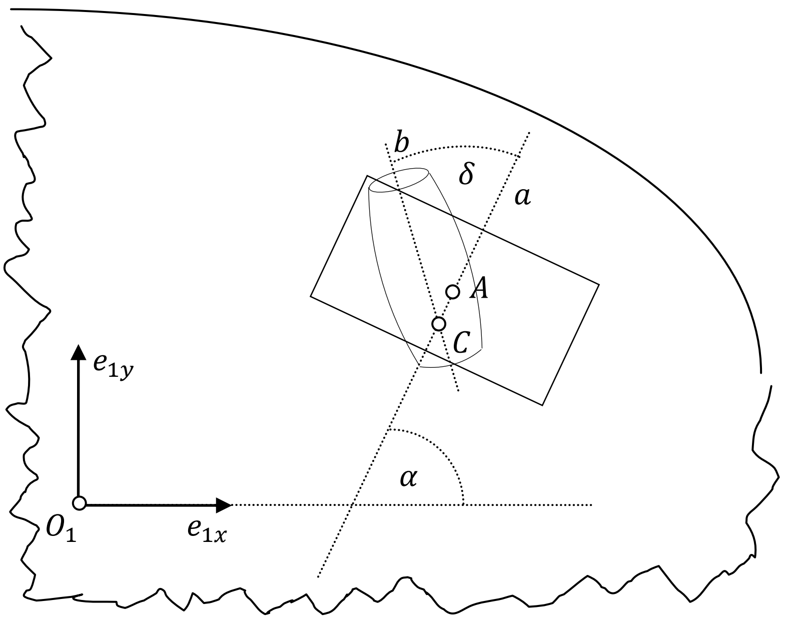


Figure 9. Kinematical representation of a general omni-wheel

When a Mecanum wheel rolls the contact point moves along the surface of a roller, Figure 9. shows the geometrical representation. It depicts a certain moment of a rolling Mecanum wheel, looking at a cut-out portion of a vehicle from above. In this situation four systems are involved: the ground, the vehicle, the wheel and the roller, which at the certain moment touches the ground at point C. This point is always under axis a of the wheel, it is at the projection of its intersection with the roller axis b to the ground. From this follows that C only lies exactly below A when axis b is horizontal. Let O_1 denote the center of the vehicle, then its coordinate system S_1 is described by $\{\boldsymbol{e}_{1x}, \boldsymbol{e}_{1y}, \boldsymbol{e}_{1z}\}$ unit vectors, the axes x and y being parallel to the ground. In this case the direction vector of the axis a:

$$\boldsymbol{a} = (\cos\alpha \quad \sin\alpha \quad 0)' = (a_x \quad a_y \quad a_z)' \tag{8}$$

The direction vector of the axis b, the roller axis depends on the rotation angle γ of the wheel:

$$\boldsymbol{b} = \begin{pmatrix} \cos\alpha\cos\delta - \sin\alpha\sin\delta\cos\gamma \\ \sin\alpha\cos\delta + \cos\alpha\sin\delta\cos\gamma \\ \sin\delta\sin\gamma \end{pmatrix} = \begin{pmatrix} \mathrm{b_x} \\ \mathrm{b_y} \\ \mathrm{b_z} \end{pmatrix} \tag{9}$$

In S_1 the x and y coordinates of the contact point are the following:

$$\left.\begin{aligned} c_x &= a_x - d\cos\alpha\cot\delta\tan\gamma \\ c_y &= a_y - d\sin\alpha\cot\delta\tan\gamma \end{aligned}\right\} \tag{10}$$

To calculate the connection between the motion of the vehicle and the rotation of the wheel, first the components of the movement need to be calculated. Since a flat ground and parallel movement is assumed, for these equations the z coordinate of the velocity vectors can be omitted, for the sake of clarity. The motion of the contact point with respect to the vehicle/ground movement can be written the following way:

$$\boldsymbol{v}_{c,gv} = \begin{pmatrix} v_x - \Omega c_y \\ v_y + \Omega c_x \end{pmatrix} \tag{11}$$

where $v_{O_1} = (v_x, v_y)'$ is the velocity vector of the vehicle and Ω is its angular velocity. The motion between the wheel and the vehicle is a rotation around axis a, the velocity vector of C for this motion is:

$$\boldsymbol{v}_{c,wv} = r\dot{\gamma}\begin{pmatrix} -\sin\alpha \\ \cos\alpha \end{pmatrix} \tag{12}$$

where $\dot{\gamma}$ is the angular velocity of the wheel.

The motion between the wheel and the roller is again a rotation, around axis b, therefore it is perpendicular to the axis. It is of the form:

$$\boldsymbol{v}_{c,rw} = k\begin{pmatrix} -b_y \\ b_x \end{pmatrix} \tag{13}$$

k being a multiplier constant.

The motion of C with respect to the roller/ground movement is zero, if rolling without slippage is assumed. By adding the previously described motions we can obtain this velocity i.e. zero:

$$\boldsymbol{v}_{c,rg} = \boldsymbol{v}_{c,gv} + \boldsymbol{v}_{c,wv} + \boldsymbol{v}_{c,rw} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{14}$$

Substituting (11), (12), (13), yields:

$$\left.\begin{aligned} r\sin\alpha\,\dot{\gamma} + b_y k &= v_x - \Omega c_y \\ r\cos\alpha\,\dot{\gamma} + b_x k &= -v_{\mathrm{y}} - \Omega c_x \end{aligned}\right\} \tag{15}$$

where r stands for wheel radius. Eliminating k yields the differential equation connecting the vehicle motion and wheel rotation:

$$r(b_x \sin\alpha - b_y \cos\alpha)\dot{\gamma} - b_x(v_x - \Omega c_y) - b_y(v_y + \Omega c_x) = 0 \tag{16}$$

The factors b_x, b_y, c_x, c_y are functions of γ according to equations (9) and (10), also $\gamma = \gamma(t)$. As it can be seen from the equations, the situation is quite complex, while a certain roller is in contact with the ground the contact point moves from one side of the wheel to the other as the wheel turns. When C reaches the wheel edge, then the next roller comes into contact with the ground – and C jumps back to the other edge, this implies that $\boldsymbol{b}(\gamma), \boldsymbol{c}(\gamma)$ are functions with jump discontinuities at the change of the rollers. In reality the situation gets more complicated because many wheels are created with roller overlap, to avoid gaps between rollers, thus having multiple contact points. In practice if we can guarantee that there is a contact point at all instants, contact point fluctuation has a negligible effect even more so when the amplitude of the fluctuation, i.e. the wheel width is significantly smaller than other dimensions of the vehicle, such as wheel span.

2.4 Omnidirectional wheel model in simulation – a practice oriented approach

As I explained in section 2.1, I decided to use readymade components, to create the omnidirectional wheel simulation. The Rill model described in section 2.3.1 is available in the academic bundle of Dymola 7.4. In order to create an omnidirectional wheel two straightforward approaches are possible:

- Put together an omni-wheel wheel from several individual wheels as rollers
- Modify an existing wheel model to behave like a Mecanum wheel

2.4.1 Individual rollers

The most straightforward method to create a usable Mecanum wheel model:

- take any tire model from the library to create a roller from the base model
- set estimated (or measured) wheel parameters and geometry
- set a certain number of rollers and arrange them according to the wheel geometry (set roller angle etc.)
- allow the rollers to spin freely along their main axis and connect them to a main axle, that can be driven by an external torque.

This is illustrated on Figure 10., for the case of the Mecanum wheel – 45° rollers – where the red cylinder in the center is the wheel hub, and the free rolling rollers are the blue cylinders around its circumference. Their axes are rotated at a 45° angle and the vectors pointing to the roller centers can be calculated by:

$$d_i = R_0 \left\{ \sin\left(\frac{2\pi i}{n}\right), 0, \cos\left(\frac{2\pi i}{n}\right) \right\} \tag{17}$$

where $i \in [1, n]$ and R_0 is the radius of the wheel without the rollers.

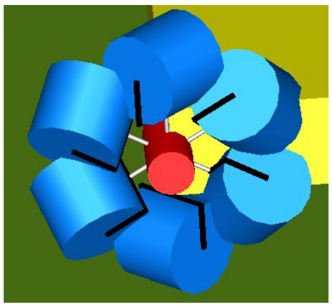


Figure 10. Mecanum wheel animated in Dymola

This approach is a clear adaptation of the mechanics of the wheel, and it does work fairly well in simulation:

- Straightforward implementation.
- Very easy to switch between different tire models.
- Implicitly handles roller inertia, and rolling resistance.
- The model incorporates discontinuities between rollers similar to a real wheel.
- If simulation time is not an issue, adding more rollers and/or a better contact geometry model could make it more realistic.

It also suffers from several disadvantages.

- Far from suitable for real time simulation. Complicated model - for a typical four wheeled six roller vehicle, collision detection and force calculation has to be carried out for 24 rollers.
- Relies on boundaries of original wheel model. The individual rollers operate at extreme situations: up to 90° sideslip and camber angles. The tire model can handle this, but it was not designed for it – loss of accuracy.
- Crude contact model, most Mecanum wheel rollers are not a simple cylinder. In order to make them ride smoother, they have a varying cross section and rounded edges. A better geometry model would add complexity (see first disadvantage).

In conclusion if I try to amend these problems I violate the principle of my original goal of creating a simple yet realistic wheel model, using available components. This approach would need a new contact model and a new roller design, from the start.

2.4.2 Single roller omni-wheel model

To overcome some of the disadvantages of the model presented in section 2.4.1, I created another one, based on a different approach. The main idea is to alter the force generation method of a single tire to behave like an omnidirectional wheel.

For a real omnidirectional wheel the number of rollers touching the ground varies between one and two, also the position of the contact point changes depending on the angular position of the wheel and the angle of the rollers, creating an angle dependent effect on the wheel behavior. (see section 2.3.2) However, in this model I assume that the force generation is continuous along the perimeter of the wheel, much like an extrapolation of the ideal case when the center of only a single roller touches the ground. This is reasonable as the rollers are usually shaped in an attempt to achieve this effect.

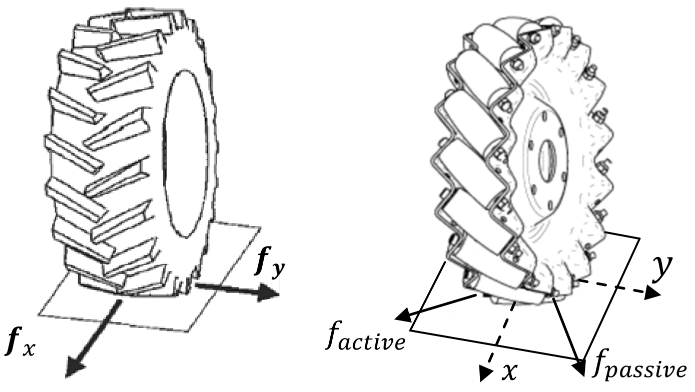


Figure 11. Basic idea of the force transformation for omni-wheels

To describe the modifications let us introduce the notation system used in the Rill model and the Modelica model for a regular wheel. They use the C (carrier) and W (wheel) coordinate systems according to the TYDEX [52] notations. (see Figure 12.)

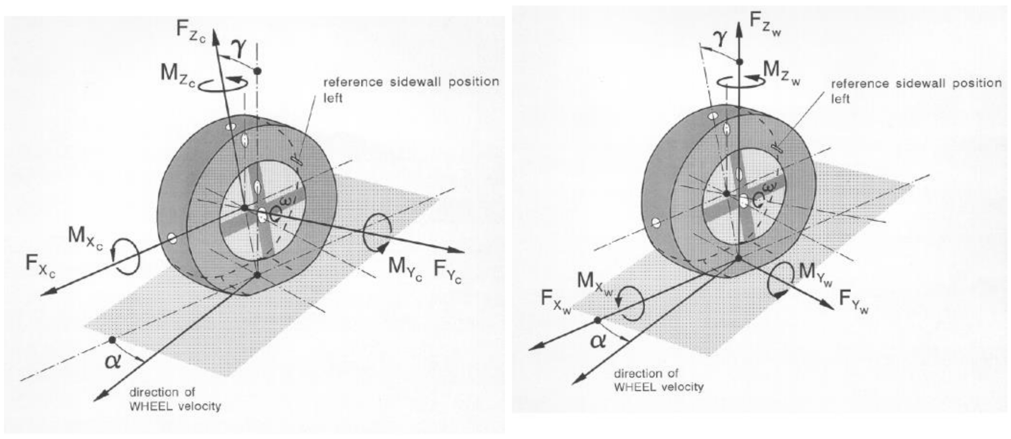


Figure 12. Tydex reference notations C – left W – right [52]

"The C-axis system is fixed to the wheel carrier with the longitudinal xc-axis parallel to the road and in the wheel plane (xc-zc-plane). The origin of the C-axis system is the wheel center. The origin of the W-axis system is the road contact-point defined by the intersection of the wheel plane, the plane through the wheel carrier, and the road tangent plane"[7].

The unit vectors Ce_x, Ce_y, Ce_z and We_x, We_y, We_z point in the direction of the C and W system axes. To accommodate the omni-wheel, I defined a We_{fw} unit vector in the direction of the rollers' axis that is the direction it can exert force – I call it active direction. To be consistent with the notations of section 2.5.1. I included the names of matching vectors on Figure 13.

$$We_{fw} = We_x \cdot Rot_\delta \qquad (18)$$

where Rot_δ is a 3x3 rotation matrix of δ.

[7] http://ti.mb.fh-osnabrueck.de/adamshelp/. web. (accessed May 2012.)

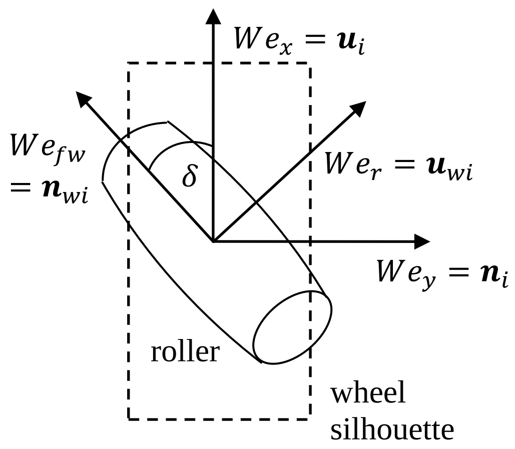

$We_x = \boldsymbol{u}_i$
$We_{fw} = \boldsymbol{n}_{wi}$
δ
$We_r = \boldsymbol{u}_{wi}$
$We_y = \boldsymbol{n}_i$
roller
wheel silhouette

Figure 13. Definition of roller vectors x is forward, y is sideways, z points "out of the paper"

To the direction of We_r – the free rolling or passive direction – the forces arising are due to bearing friction, rolling resistance and moment of inertia. From a practical point of view these forces do not amount to much and they can generally be neglected without loss of accuracy. However for the sake of completeness I include some information on them from the literature.

Rolling resistance is usually modeled similar to the coefficient of friction, as a dimensionless factor. According to [19] studies on the rolling loss characteristics of solid rubber tires led to an equation of the form:

$$f_r = \frac{R_x}{W} = C\frac{W}{D}\sqrt{\frac{h_t}{w}} \tag{19}$$

where:
R_x = rolling resistance force
W = wheel load
C = coefficient representing tire elastic characteristics
D = tire diameter
h_t = tire section height
w = tire section width

From this it is clear that rolling resistance has a linear load sensitivity. Large diameter tires decrease rolling resistance, also do low aspect ratios (h_t/w). Rolling resistance in the most simple case can be approximated by a constant for passenger car tires on concrete a good value is around 0.015. At slow speeds it behaves linearly, however on broader ranges it seems more like a speed squared relationship [19].

Bearing friction is essentially the same thing as rolling resistance however its coefficient is usually an order of magnitude smaller, than that of rubber tires, as hardened steel surfaces rolling in grease typically have very little resistance.

Moment of inertia of a solid cylinder rotating along its axis can be calculated by: $I = \frac{mr^2}{2}$. This affects the torque needed to accelerate the roller to a certain angular velocity, creating an opposing force when the wheel accelerates in the passive direction. The rollers in general are typically lightweight and small diameter compared to the wheel itself, so again this effect may be neglected.

Having made this simplifying assumption we can make a further step by defining slip for the omnidirectional wheel. In most wheel models forces are generated as a function of slip, so this is an important aspect. Slip is defined separately for the x and y directions. Since our idealized roller only generates force in the direction of its spin axis (We_{fw}) we shall only calculate slip in this direction. [42] defines slip as "total slip":

$$s = \frac{v}{R|\Omega| + v_{num}} \tag{20}$$

where

$$v = \sqrt{v_x^2 + v_y^2} \tag{21}$$

R is the wheel radius, Ω is the angular velocity, and v_{num} is a small number inserted for numerical reasons, to avoid division by zero. In my model I modify v in the slip equation:

$$v_{omni} = \sqrt{v_{fw}^2} \tag{22}$$

where v_{fw} is the projection of the velocity of the center of the wheel in the We_{fw} direction.
After redefining the slip equation, all is left to do is equate static and dynamic force equations with zero in the y direction and calculate force in the x direction according to the Rill model using the modified slip equation. The direction of this force has to be set to the direction of We_{fw}. At this point the effects of camber, rolling resistance and bore torque were not investigated.
This modeling approach greatly decreases computation time, because ground contact only needs to be calculated once per wheel. The modification is quite simple, so other wheel models could be modified if needed. If more accuracy is required roller inertia and rolling resistance could be incorporated in the model. The roller discontinuities might be modeled by modulating tire forces by an angle dependent function, similar to the one described in section 2.3.2. For my research these details were not necessary so I decided not to implement them at this point.
Due to two orders of magnitude increase in simulation speed I decided to use this model instead of the one described in section 2.4.1.

2.4.3 Scope of the models

The two modeling approaches presented above are easy to adapt to different omnidirectional wheel embodiments. The multi wheel model from section 2.4.1 might be used for various kinds of omnidirectional wheels, since the mechanics and dimensions can be readily adapted to the wheel in question. However the limitations mentioned in the corresponding section have to be observed.
The second modeling approach (section 2.4.2) is useful for the most commonly used wheels, where rollers are mounted on the circumference of the wheel, universal and Mecanum wheels. Due to the simplifications I made, the model is best used for wheels where the rollers are densely mounted and shaped to achieve a smooth rolling motion, and where the wheel is relatively thin, so that the contact point does not vary significantly when roller contact is changed. These qualitative guidelines have to be considered on an individual basis for each modeling task.

2.5 Forklift model

2.5.1 Omnidirectional platform kinematics

I decided to use simulation to verify the wheel model as it gave me great flexibility for experimenting. For this the kinematics of the platform needs to be known. In this section I describe the kinematics of omni-wheel platforms in a generalized manner. The most widely used basic configurations such as the three wheeled platform with 0° rollers (kiwi drive) and the four wheeled platform with 45° rollers (Mecanum drive) are easily derived from the general equations. This discussion builds largely on the work of Indiveri [25] notations however were changed for the sake of consistency. Figure 14. introduces the notations; for clarity a general three wheeled platform was considered. x and y of the local coordinate system lie in the ground plane z points out of the plane forming a right hand coordinate system.

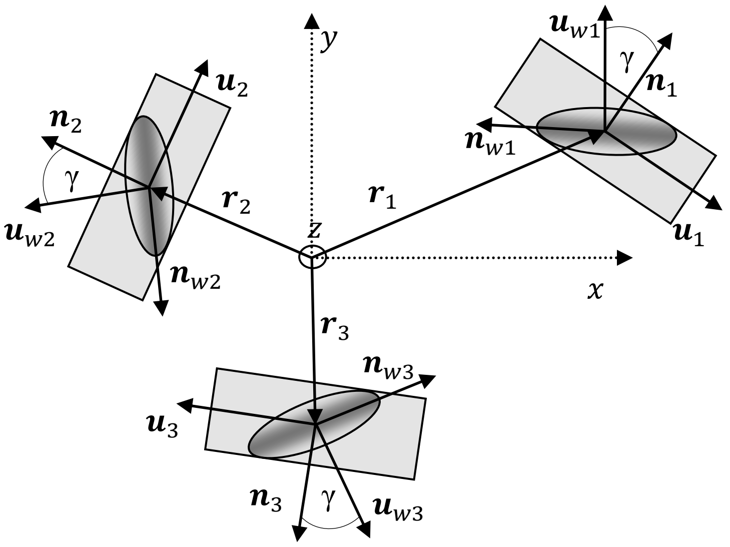


Figure 14. Notations of a general omnidirectional platform

All axes are considered parallel to the ground except for z of the local coordinate system and k, the unit ground normal. Let N stand for the number of wheels, then the wheels are indexed from 1 to N with the index i. $\boldsymbol{r}_i$ denotes the position vector of the center of the wheel contact on the ground. The rollers touching the ground are illustrated by the shaded ellipses. The unit vector $\boldsymbol{n}_{wi}$ is parallel with the roller axis, that is the active direction of the roller. $\boldsymbol{u}_{wi} = \boldsymbol{n}_{wi} \times \boldsymbol{k}$ is the tangent velocity direction of the roller – that is the passive direction – associated with the rotation around n_{wi}. All wheels are assumed to be identical with the same radius R. $\boldsymbol{n}_i$ is the unit vector along the wheels axis of rotation, $\boldsymbol{u}_i = \boldsymbol{n}_i \times \boldsymbol{k}$ is the tangent velocity direction of the wheel, as it is rotating around $\boldsymbol{n}_i$. Lets denote the platform center velocity with $\boldsymbol{v}_c$ and magnitude of angular rotation with Ω. The angular velocity points in the direction of $\boldsymbol{k}$. The velocity vector of each wheel center can be then calculated as:

$$\boldsymbol{v}_i = \boldsymbol{v}_c + \Omega\mathbf{k} \times \boldsymbol{r}_i, \quad i = 1, 2, 3, \dots N \tag{23}$$

In the case of perfect rolling the velocity $\boldsymbol{v}_i$ is a linear combination of the roller rolling around $\boldsymbol{n}_{wi}$ and the wheel rolling around $\boldsymbol{n}_i$. Assuming that $\boldsymbol{n}_{wi}$ and $\boldsymbol{n}_i$ are not aligned i.e. $\gamma \neq (2a + 1) \cdot 90°, a \in \mathbb{Z}$ then:

$$\boldsymbol{v}_i = \alpha\boldsymbol{u}_{wi} + \beta\boldsymbol{u}_i \tag{24}$$

implying

$$\boldsymbol{n}_i^T\,\boldsymbol{v}_i = \alpha(\boldsymbol{n}_i^T\,\boldsymbol{u}_{wi}) \tag{25}$$

$$\boldsymbol{n}_{wi}^T\,\boldsymbol{v}_i = \beta(\boldsymbol{n}_{wi}^T\boldsymbol{u}_i) \tag{26}$$

therefore

$$\boldsymbol{v}_i = \frac{\boldsymbol{n}_i^T\boldsymbol{v}_i}{\boldsymbol{n}_i^T\boldsymbol{u}_{wi}}\,\boldsymbol{u}_{wi} + \frac{\boldsymbol{n}_{wi}^T\boldsymbol{v}_i}{\boldsymbol{n}_{wi}^T\boldsymbol{u}_i}\,\boldsymbol{u}_i \tag{27}$$

Where the first term on the right is the passive rotation, the second is caused by wheel rotation, produced by driving motor torque. If we denote the angular speed of the motors with ω_i then, in case of perfect rolling:

$$\frac{\boldsymbol{n}_{wi}^T}{\boldsymbol{u}_i^T\boldsymbol{n}_{wi}}\boldsymbol{v}_i = \omega_i R \tag{28}$$

where the other – the passive – part of the sum:

$$\frac{\boldsymbol{n}_i^T\boldsymbol{v}_i}{\boldsymbol{n}_i^T\boldsymbol{u}_{wi}}\boldsymbol{u}_{wi} \tag{29}$$

is explicitly assumed not to play any role. Substituting in equation (23), noting that $\boldsymbol{u}_i^T\boldsymbol{n}_{wi} = -\cos(\gamma)$ we get:

$$-\frac{1}{\cos(\gamma)}\boldsymbol{n}_{wi}^T\,\boldsymbol{v}_c + \frac{1}{\cos(\gamma)}\boldsymbol{n}_{wi}^T\,\boldsymbol{r}_i \times \mathbf{k}\,\Omega = \omega_i\,R \tag{30}$$

We can simplify

$$\boldsymbol{n}_{wi}^T\boldsymbol{r}_i \times \mathbf{k} = \mathrm{n}_{\mathrm{wi}_\mathrm{x}}\mathrm{r}_{\mathrm{i}_\mathrm{y}} - \mathrm{n}_{\mathrm{wi}_\mathrm{y}}\mathrm{r}_{\mathrm{i}_\mathrm{x}} = -\mathbf{r}_\mathrm{i}^{\mathrm{T}}\mathbf{u}_{\mathrm{wi}} \tag{31}$$

because $\boldsymbol{k}$ is parallel to the z axis and $\mathbf{n}_{\mathrm{wi}}$ is perpendicular to $\mathbf{u}_{\mathrm{wi}}$. Summarizing the equations for the system we get:

$$\boldsymbol{M}\begin{pmatrix}\boldsymbol{v}_c\\ \Omega\end{pmatrix} = R\,\boldsymbol{\omega}\cos(\gamma) \tag{32}$$

where

$$\boldsymbol{M} = -\begin{pmatrix} n_{w1_x} & n_{w1_y} & r_1^T u_{w1} \\ n_{w2_x} & n_{w2_y} & r_2^T u_{w2} \\ \vdots & \vdots & \vdots \\ n_{wN_x} & n_{wN_y} & r_N^T u_{wN} \end{pmatrix} \tag{33}$$

Equations (32) and (33) represent the general kinematics model of an omni-wheeled platform. This allows the analysis of this class of omnidirectional robots as a function of the roller orientation γ and roller position $\boldsymbol{r}$. According to Indiveri [25] any desired vehicle velocity can be implemented, given proper ω_i wheel velocities if:

- $\cos(\gamma) \neq 0$ i.e. roller axes are not parallel to wheel axes, which would render the wheel incapable of generating movement
- $rank\ \boldsymbol{M} = 3$, this corresponds to controllability of the system

One of the most popular configurations is a symmetrical three wheeled platform with $\gamma = 0°$ roller orientation and wheels mounted at $120°$ to each other (Figure 15. right). For this configuration, if we take the geometrical center as our origin, using Figure 15. and equations (32) and (33), the kinematic equations turn out to be the following:

$$\begin{pmatrix}\omega_1\\ \omega_2\\ \omega_3\end{pmatrix} = \frac{1}{R}\begin{pmatrix}0.5 & -\frac{\sqrt{3}}{2} & -r\\ 0.5 & \frac{\sqrt{3}}{2} & -r\\ -1 & 0 & -r\end{pmatrix}\begin{pmatrix}v_{c_x}\\ v_{c_y}\\ \Omega\end{pmatrix} \tag{34}$$

where r stands for the radius of the platform.
Another popular configuration is a four wheeled platform with $\gamma = 45°$, more commonly known as the Mecanum platform (Figure 15. left).

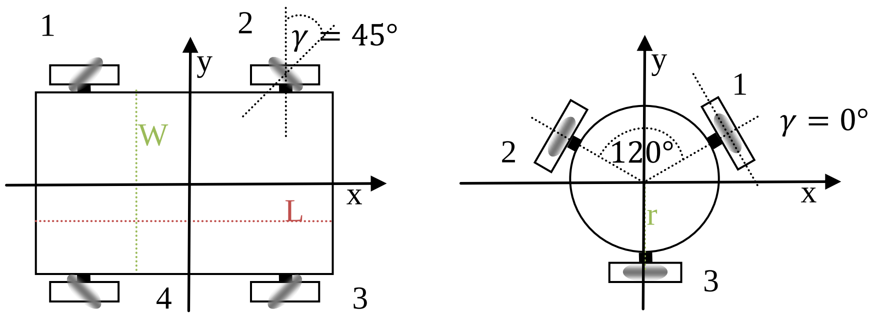


Figure 15. Most popular omnidirectional configurations

Again taking the center of platform as the origin, the kinematic equations are the following:

$$\begin{pmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{pmatrix} = \frac{\sqrt{2}}{R}\begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{L+W}{2\sqrt{2}}\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{L+W}{2\sqrt{2}}\\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{L+W}{2\sqrt{2}}\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{L+W}{2\sqrt{2}}\end{pmatrix}\begin{pmatrix}v_{c_x}\\ v_{c_y}\\ \Omega\end{pmatrix} = \frac{1}{R}\begin{pmatrix}1 & 1 & -\frac{L+W}{2}\\ 1 & -1 & -\frac{L+W}{2}\\ 1 & 1 & \frac{L+W}{2}\\ 1 & -1 & \frac{L+W}{2}\end{pmatrix}\begin{pmatrix}v_{c_x}\\ v_{c_y}\\ \Omega\end{pmatrix} \tag{35}$$

It is easy to check that the two rules mentioned above hold, namely $\cos(\gamma) \neq 0$ and $rank(\boldsymbol{M}) = 3$ for both models. Therefore the two platforms are indeed useful.

2.5.2 The Mecanum platform

From the kinematic equations derived in the previous section, to be able to exploit omnidirectional capabilities, the wheels have to be mounted in a certain pattern, shown on Figure 16. The wheels mounted across from each other are mirrored, so that $rank(\boldsymbol{M}) = 3$ (see equation (33)).

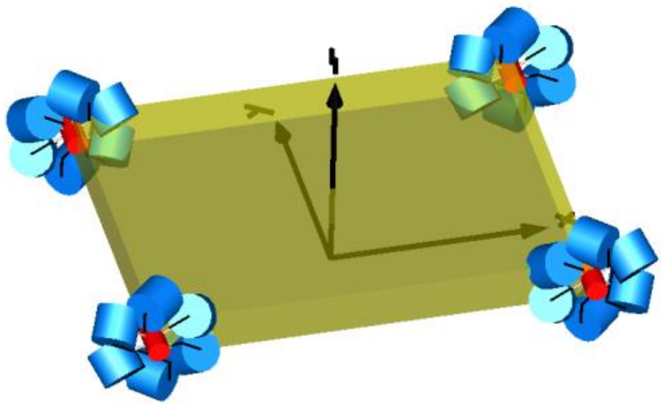


Figure 16. Wheel configuration

Figure 17. shows some of the basic movements possible with this configuration. Turning and movement in arbitrary directions can be achieved by choosing appropriate angular velocities for each wheel.

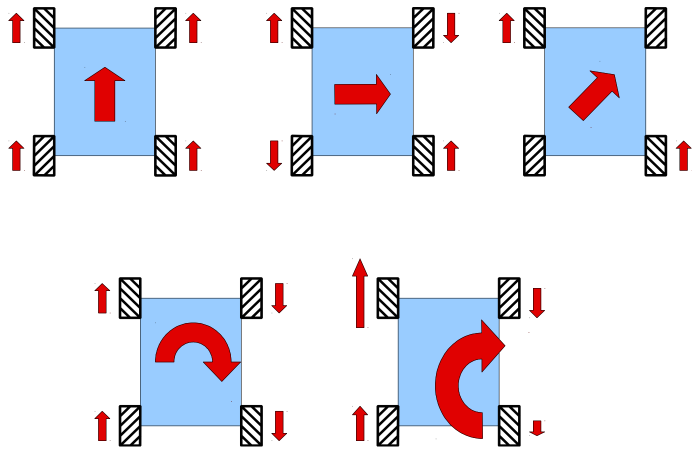


Figure 17. Basic movements of a Mecanum platform (viewed from above)

2.5.3 Empirical model validation

Having the tire model comprises the most significant portion of the simulation. To complete the model a simple chassis needs to be added. I developed two platforms in simulation. I modeled the vehicles as a single mass with inertia, with the wheels attached rigidly. Since we assume movement on flat terrain, using no suspension is a reasonable simplification. In case of the Mecanum platform, I added the payload as a point mass attached to a vertical actuator – the fork – to have a simple forklift. The platform models can be seen on Figure 18. as rendered by the Dymola software. On the left the Mecanum platform can be seen – the body is represented by the red brick, the payload by the green sphere. Wheel forces are visualized by the blue arrows. In this example the load is shifted to the front left side representing a carelessly loaded cargo. On the right the three wheeled platform can be seen. More specific descriptions of the platforms can be found in Appendix A.

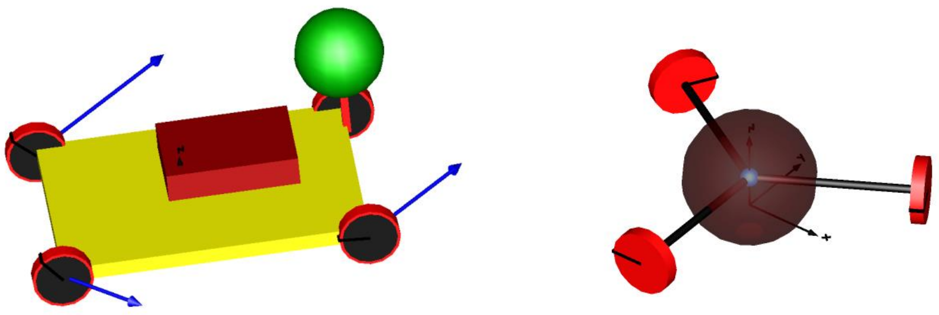


Figure 18. Simple Mecanum forklift and a three wheeled platform rendered by Dymola

To verify the model I created a controller that can drive the wheels according to the inverse kinematics model, developed in section 2.5.1. In these equations the platform velocities are known and the individual wheel angular velocities need to be calculated. The equations are well known in the literature, they can be derived by applying simple geometric calculations. Results similar to mine were obtained for example by [34] and [18].

By using the inverse kinematic equations (35) I was able to demonstrate whether the vehicle moves as expected. Figure 19. shows a spin while translate maneuver in which the Mecanum wheeled robot has a constant velocity in the global x direction a sinusoidal velocity in the global y direction, while having a constant angular velocity. The figure shows snapshots of the experiment and demonstrates that the platform moves according to expectations.

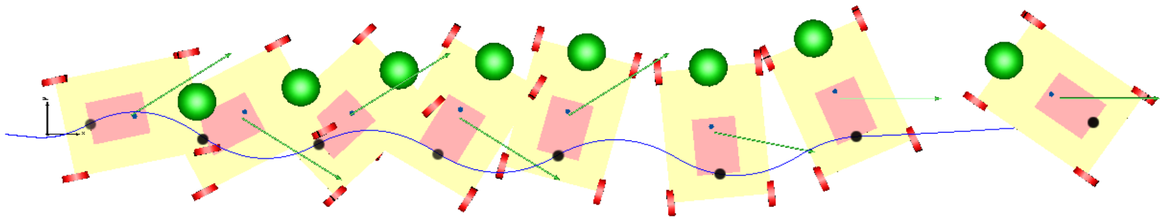


Figure 19. Spin while translate movement of the Mecanum forklift model

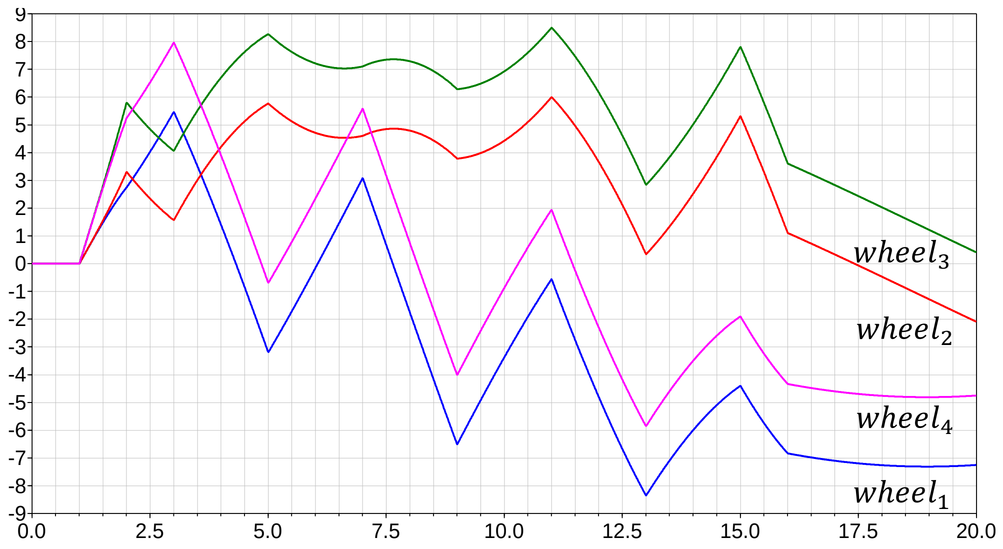


Figure 20. Angular velocity commands vs. time for the spin and translate maneuver of the Mecanum platform

The control signals – angular velocities – required to perform this movement are shown on Figure 20., the colors blue red, green, purple stand for the wheels 1, 2, 3 and 4 in this order, according to notations on Figure 15.
A similar experiment was conducted for the three wheeled platform according to equations (34). Snapshots from the simulation can be seen on Figure 21., control signals on Figure 22.

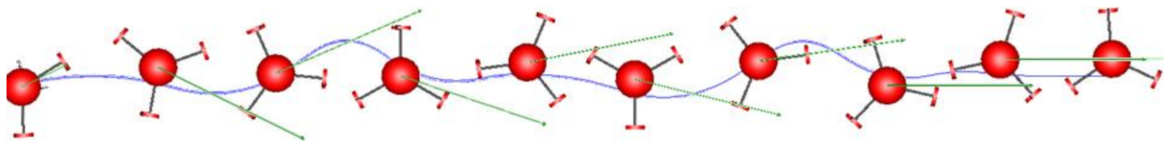


Figure 21. Spin and translate maneuver of the three wheeled platform

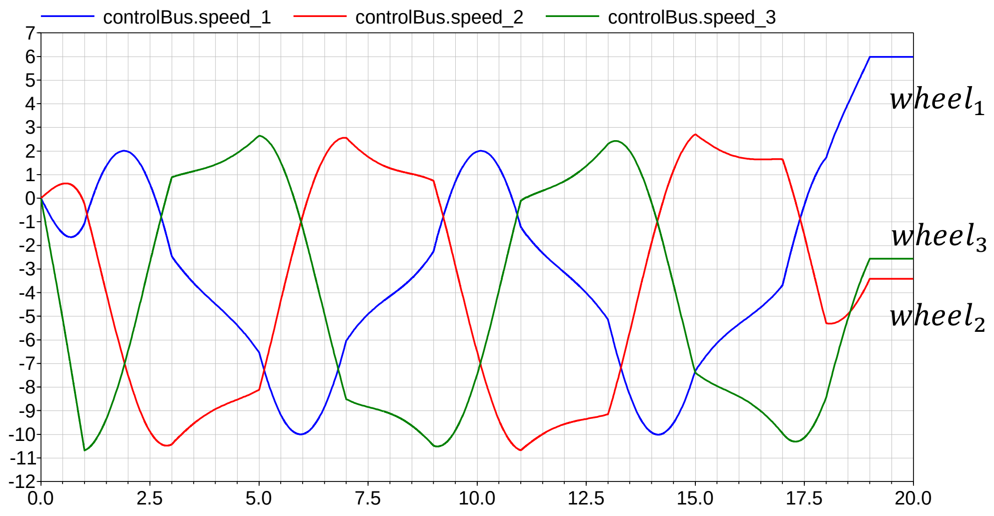


Figure 22. Control signals vs. time of the three wheeled robot for the spin and translate maneuver

Both platforms moved as expected, demonstrating the usability of the omni-wheel model for simulation that I created. To adapt it to a certain wheel material one has to tune the model parameters as one would with a normal tire.

2.6 Conclusion

The wheel model is possibly the most important part of any vehicle simulation, since the wheel is the component that connects the platform with the environment. The contact point – or patch – is determined in the context of the wheel model, also this is the part that generates forces and torques that move the platform. Tire modeling in vehicle simulation is a well researched discipline, with results that have proven themselves in countless applications. Many aspects of tire models can be adapted to be used in omnidirectional simulation. Contact point calculation, friction force generation of the rubber compound, dynamic force generation characteristics are all very similar for both wheel types. By leveraging the wealth of background information available for regular vehicle related tire research, rapid prototyping of omnidirectional wheels in simulation becomes fairly easy.

Thesis I: I created a method for adapting conventional empirical tire models to emulate the behavior of an omnidirectional wheel. The model retains the characteristics and parameter set of the adapted regular tire model, and the characteristics of the chosen omnidirectional wheel – such as roller angle – are superposed.

The behavior of the model is validated by applying inverse kinematic models of the most popular omnidirectional platforms in simulation.

Related publications:

[KJVS12], [KV12b], [VAL12]

3 Brake assistant for omnidirectional vehicles

3.1 Motivation

Due to their design omnidirectional wheels in general lack the capability to generate substantial side forces, in the direction perpendicular to the roller axis – the passive direction. This is caused by the free rolling rollers around the circumference. An interesting consequence is that an omnidirectional platform with, free rolling, sliding or locked wheels, will tend not to keep its original orientation. If, in contrast a cart with free rolling, conventional wheels is pushed with a small force[8] at an angle to its main rolling direction, the cart will nonetheless roll in the direction the wheels are pointed, perhaps with a small sideslip angle. This is illustrated on Figure 23.

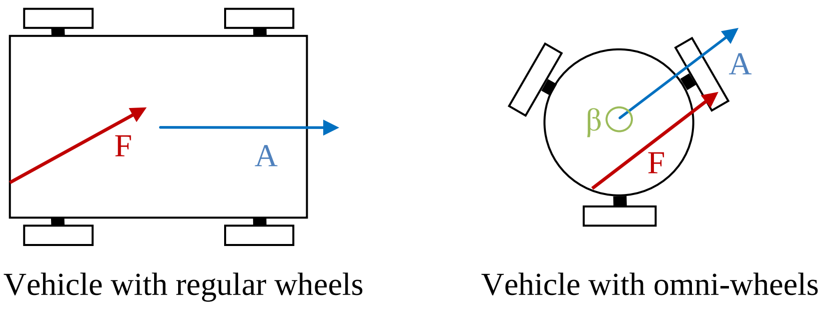


Figure 23. Illustration of the effect of side forces and pushing

This is not true for omnidirectional platforms, they move in the direction of the push. This is also true in the other way around, when the wheels are braked, even a moderate difference in wheel load – meaning a different friction force – causes the platform to turn until the torque of wheel forces reach an equilibrium. The effect is similar to a regular car swerving in a corner while braking, however in case of omnidirectional wheels this happens without excessive slip of the wheels. It is important to note that this behavior is perfectly normal, this is the very effect that allows omnidirectional movement.
A very good example can be seen on Figure 24., with a simulated braking maneuver with a Mecanum wheel platform. In the simulation the robot is unevenly loaded, most of the weight is resting on the front left wheel. The robot is accelerated to a certain velocity, going straight forward, then all the brakes are actuated at the same time. Figure 24. shows three snapshots taken after braking started. It is clearly visible that the robot turns around the most loaded wheel, because that generates the highest friction force. In this example when the active direction of the wheel with the highest load is parallel to the instantaneous velocity vector equilibrium is reached, and the robot decelerates and stops in this orientation. The blue arrows starting from the wheel center represent tire forces, the yellow arrow at the platform center is the instantaneous velocity vector. Clearly this behavior is undesirable for the operator as it is unexpected, therefore dangerous. It is easy to see this behavior leading to an accident in a narrow aisle.

[8] Obviously I am talking about a force which is not large enough to push the cart sideways or tip it over.

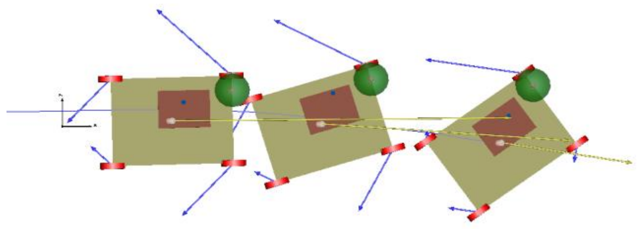


Figure 24. Brake induced swerving of a Mecanum platform

3.2 Problem description

To be able to find the most suitable control methodology for the problem at hand it is important to find an appropriate description of the system. As I already explained in section 2. the wheels exhibit nonlinear force generation characteristics, that depends strongly on wheel load and ground – tire interaction. This dependency can be described by appropriate wheel models, as accurately as needed, however the result will be only as accurate as our knowledge of instantaneous load distribution and local ground characteristics. The force vectors during a braking maneuver are illustrated in Figure 25. Friction forces $\boldsymbol{f}_i$ at the wheels are generated to oppose wheel velocity and point in the active direction of the wheel i.e.:

$$\mathrm{f}_i = \xi_i \, \boldsymbol{n}_{wi}(\boldsymbol{n}_{wi} \| {-\boldsymbol{v}_\mathrm{i}} \|) \tag{36}$$

where ξ_i is a nonlinear function of wheel load and ground characteristics, describing wheel force generation, $\boldsymbol{n}_{wi}$ is a unit vector pointing in the wheel's active direction (see Figure 14), $\| {-\boldsymbol{v}_i} \|$ is a unit vector pointing in the opposing direction of the velocity of the given wheel center.

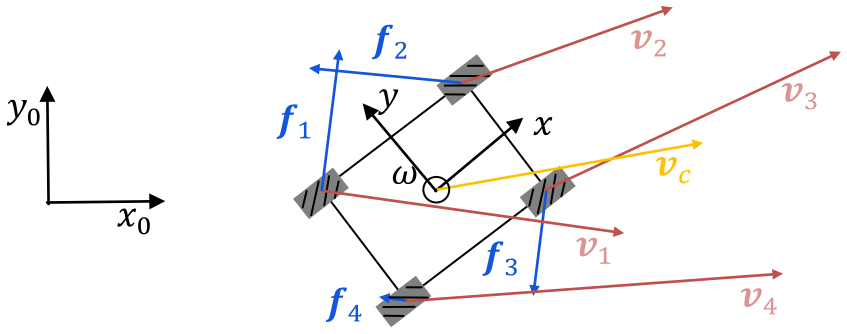


Figure 25. Forces and velocities during braking – illustration by a Mecanum platform

The deceleration of the platform depends on the direction of $\boldsymbol{f}_i$ which is a trigonometric function of $\boldsymbol{v}_c$ and $\boldsymbol{\omega}$.
As a straightforward approach one could try and linearize the equations. To linearize ξ_i knowledge of the wheel model's operating point is needed, which depends on numerous uncertain parameters.
To linearize the angle dependence of $\boldsymbol{f}_i$ the rotation of the platform would need to be constrained to a small angle for the linearization to be valid. This is too prohibitive a constraint, as for an omnidirectional platform it is normal to have high angular velocity

causing the direction of the platform orientation to change substantially during the duration of a braking maneuver.
After these considerations I decided to use a nonlinear control approach as it is clear that a parameter uncertain nonlinear MIMO system needs to be considered. Its state equation can be written in the general form:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{37}$$

where the usual notations are used – $\boldsymbol{x}$ is the state vector, $\boldsymbol{u}$ is the input and $f(\,)$ is a nonlinear function. In the case of an accelerating mechanical system, typically the state vector consists of resulting linear and angular velocity of a reference point. However for this particular case – braking omnidirectional platforms – it is more practical to also include $\boldsymbol{v}_i$ wheel center velocities: $\boldsymbol{x} = \{\,\boldsymbol{v}_i, \boldsymbol{v}_c, \boldsymbol{\omega}\}'$. These can be derived in a simple manner from linear and angular velocity of the platform center:

$$\boldsymbol{v}_i = \boldsymbol{v}_c + \boldsymbol{\omega} \times \boldsymbol{r}_{ci} \tag{38}$$

where $\boldsymbol{r}_{ci}$ is the vector pointing from the platform center to a given wheel center.
The input vector consists of $\boldsymbol{f}_i$ friction force of the wheels. System parameters do not change with time, therefore we can consider the system autonomous, or time invariant.
Dynamical equations can be derived from Newton’s law:

$$\mathrm{m}\dot{\boldsymbol{v}}_c = \sum_{\mathrm{i}} \boldsymbol{f}_i \tag{39}$$

$$\boldsymbol{J}_c\dot{\boldsymbol{\omega}} = \sum_{i} \boldsymbol{f}_i \times \boldsymbol{r}_{ci} \tag{40}$$

These equations are written for the geometrical center of the vehicle. Uncertain parameters are m mass of the vehicle, the magnitude of $\boldsymbol{f}_i$ and the $\boldsymbol{J}_c$ inertia matrix of the vehicle. State variables $\boldsymbol{v}_c$ and $\boldsymbol{\omega}$ are assumed to be measurable.
Another important property of the system, that it is strongly cross coupled, all inputs have an effect on all state variables. This property concludes from the form of the general kinematic equation of omnidirectional platforms (see section 2.5.1).
In the literature numerous examples are available for the control of inaccurately modeled nonlinear MIMO systems [15], [27], [44], [47]. To overcome the problem of model inaccuracy when controlling nonlinear systems, sliding mode control is reported to be very successful [26], [44], [45], [53].

3.2.1 Sliding mode control

When controlling real dynamical systems uncertainties and disturbances in both the system model and the variables are a common problem. Effects of these uncertainties have to be taken into account in the controller design as they can severely impair control performance, or even cause instability.
Control of dynamical systems in the presence of heavy uncertainties has became an important subject of research. Due to the efforts made in this field many different approaches made considerable progress, such as nonlinear adaptive control, model predictive control and others. These techniques are guaranteed to reach the control objectives in spite of modeling errors and parameter uncertainties.[53]
Imprecision may come from actual uncertainty about the plant, for example because of unknown parameters, or they might come into the model on purpose by using a simplified representation of the system’s dynamics. These uncertainties are classified as

- structured - or parametric

- unstructured - or unmodeled dynamics

The first kind is actually included in the terms of the model, the second corresponds to underestimation, or oversimplification of the system order.

Sliding mode control belongs in the family of robust control, because it is extremely tolerant of parameter inaccuracies [27]. It is characterized by high simplicity. It utilizes discontinuous control laws to drive the system state trajectory onto a specified surface in the state space, the so called sliding or switching surface, and to keep the system state on this manifold after reaching it. Evidently this approach only works when the control input is designed with sufficient authority to overcome the uncertainties and disturbances acting on the system. The main advantages are, that while the system is on the manifold it behaves like a reduced order system and system dynamics are insensitive to uncertainties and disturbances.

All these great advantages however come at a price, good performance can only be achieved with extremely high control activity. This results in the so called chattering effect, which is the result of excitation of high frequency components of the system. Ideally control activity has an infinite switching frequency, which in real life applications is a high but finite frequency. High control activity also causes wear and tear of mechanical components. For further reading the reader is referred to: [15], [27], [47], [53], [56] and others.

Formally the problem can be described as follows [53]: Let's consider the following nonlinear system

$$\dot{x}(t) = f(t,x) + g(t,x)u(t) \tag{41}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, $f(t,x) \in \mathbb{R}^{n\times n}$ and $\mathrm{g}(t,x) \in \mathbb{R}^{n\times m}$. The discontinuous feedback is given by:

$$u_i = \begin{cases} u_i^+(t,x), if\ \sigma_i(x) > 0 \\ u_i^-(t,x), if\ \sigma_i(x) < 0 \end{cases} \quad i = 1,2,\dots,m \tag{42}$$

where $\sigma_i(x) = 0$ is the i-th sliding surface, and

$$\sigma(x) = [\sigma_1(x), \sigma_2(x), \dots \sigma_m(x)]^T = 0 \tag{43}$$

is the $(n - m)$ dimensional sliding manifold.

The control problem consist of developing the u_i^+, u_i^- functions and the sliding surface $\sigma(x) = 0$, so that the closed-loop system exhibits sliding mode.

A sliding mode exists if in the vicinity of the sliding surface the velocity vectors of the state trajectory are always directed towards the surface. In consequence if the state trajectory intersects the sliding surface, the value of the state trajectory remains within a neighborhood of the surface.

An ideal sliding mode exists only when the state trajectory of the controlled plant satisfies $\sigma[x(t)] = 0$ at every $t \geq t_0$ for some t_0. From time instant t_0 the system state is constrained on the discontinuity surface which is an invariant set after the sliding mode has been established. This requires infinitely fast switching. In real systems there are always imperfections, such as delays, hysteresis etc. causing the switching frequency to be a smaller, finite value. Because of this the system state oscillates in the neighborhood of the sliding surface. This oscillation is called chattering. If the switching frequency is significantly high compared to the dynamics of the system this phenomenon may have a negligible effect.

3.3 Controller design

To keep the platform on a certain trajectory, first this trajectory has to be defined. The goal is twofold:
Goal 1: the platform has to be prevented from swerving as much as possible, in the presence of feedback noise and, dynamic disturbances.
Goal 2: the platform has to be stopped, preferably in a short distance.
The goals are simple but several problems pose difficulties in reaching it:

- Since the wheels cannot be turned, the direction of force they generate only depends on the velocity direction of the respective contact patch, therefore the only option to assert control is to modulate the braking force.
- Force generation is passive in a sense that it can only act to decrease velocity, and not to increase it, making control authority unbalanced.
- Generally the friction forces of the wheels are not parallel with wheel velocities, therefore they will act to divert the platform from its original path.
- From equations (32), (38), (39), (40) it is clear that all $\boldsymbol{f}_i$ effects all $\boldsymbol{v}_i$, $\boldsymbol{v}_c$ and $\boldsymbol{\omega}$ therefore it is generally not possible to affect a single degree of freedom of the platform independently.

These considerations mean that brake modulation has to depend on the direction of the friction force, if it drives the platform towards a given trajectory, then the brake needs to be applied, otherwise the wheel needs to be rolling free. To construct the sliding surface we have to consider the nature of trajectory we would like to track. Since the magnitudes of the forces are uncertain, the trajectory cannot be defined as an exact function of time. If we assume that the platform has $\boldsymbol{v}_0$, $\boldsymbol{\omega}_0$ initial velocities at the moment when braking starts, then the trajectory needs to have the following characteristics: the platform has to gradually lose its $\boldsymbol{\omega}_0$ angular velocity and it cannot gain velocity in the direction perpendicular to $\boldsymbol{v}_0$ i.e. it cannot divert significantly from the trajectory it was following before braking. The controller has to drive the velocity components corresponding to rotation a sideways translation to zero and keep them there.
To translate these considerations into control rules the sliding surface can be written in the following form for each wheel:

$$s_{itot} = s_{i\perp} + s_{i\omega} + s_{i\parallel} \tag{44}$$

where the three components correspond to wheel velocity components parallel and perpendicular to $\boldsymbol{v}_0$ and a component due to $\boldsymbol{\omega}$. Basically the three components show whether the given wheel can decrease kinetic energy along three degrees of freedom with respect to $\boldsymbol{v}_0$.
The switching control law can be constructed so that when s_{itot} becomes negative the brake of the i-th wheel gets actuated, otherwise the wheel rolls free:

$$\boldsymbol{u}_i = \boldsymbol{f}_i(\zeta(s_{itot})) \tag{45}$$

where

$$\zeta(x) = \begin{cases} 0\, for\ \forall\, x \geq 0 \\ 1\, for\ \forall\, x < 0 \end{cases} \quad x \in \mathbb{R} \tag{46}$$

In practice this means a switchable electronic brake system, or a hydraulic cutoff valve. The control loop is shown on Figure 26.
To show that the control system is stable I use the so called passivity approach [27] (p. 436). The passivity approach means that if the components in the feedback connection

are passive in the sense that they do not generate energy on their own then it is intuitively clear that the system will be passive.

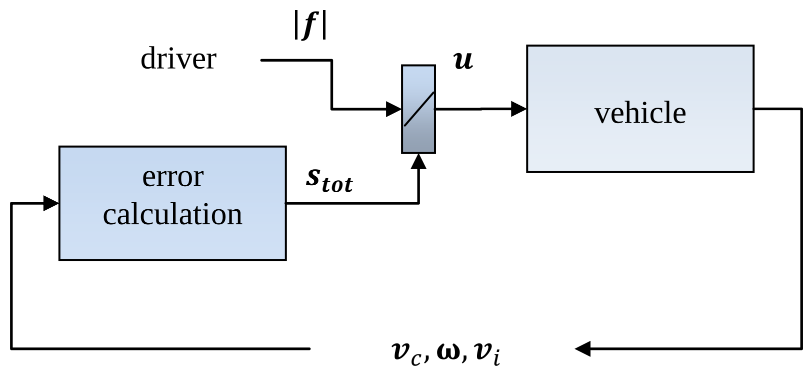


Figure 26. Brake assist control loop

Generally a system is called passive if there exists a storage function $V(t) \geq 0$, such that for all $t_0 < t_1$,

$$V(t_1) \leq V(t_0) + \int_{t_0}^{t_1} y(t)u(t)dt \tag{47}$$

where y is the system output and u is the input respectively. This equation simply states, that the energy ($V(t)$) of the system consists of the initial energy plus the supply rate yu. If the equality holds, the system is lossless, if it is a strict inequality then the system is dissipative. For example energy is lost because of friction in mechanical, because of heat exchange in thermodynamical, or because of heat generation in electrical systems.

If we set the feedback to $u = -Ky$, where K is a positive gain then it is guaranteed that the system energy remains bounded, thus the feedback system is stable.

Considering an omnidirectional platform in the context of my brake assist system, energy is stored in the form of kinetic energy. This can be written in the following form:

$$V = \frac{Mv_c^2}{2} + \frac{J\Omega^2}{2} \tag{48}$$

considering the linear and rotational kinetic energies. The “supply rate” – in this case would be more appropriately called the “drain rate” – is the product of friction force and velocity which is essentially the power dissipated by braking the wheels. The goal of the braking maneuver is to drive the kinetic energy to zero. Brake forces are always generated to oppose movement, causing negative acceleration. Since no energy is put in the system by the brake forces, it is evident that the system is passive and the inequality (47) holds true for $\forall\, t_0, t_1$ time instants. However for a braking maneuver a strict inequality is needed for the vehicle to stop in a reasonable distance, which means that if we neglect small frictional effects such as rolling resistance and bearing friction etc. a strict inequality can only be achieved if it is guaranteed that:

- at least one brake is actuated at any time instant
- with its active direction at an angle other than 90° to the direction of movement

Part of the second criterion is guaranteed by design, if the vehicle is created so that the rank of M in equation (33) is 3, so that the vehicle is in fact useful. The first criterion means that the error function of at least one wheel has to be negative at all times. This is easily guaranteed if we add a rule for the controller that turns every brake on if $\forall s_i \geq 0$.

3.4 Sliding manifold

The function defined in the previous section (eq. (44)) has to be constructed so that it has to be a sliding surface by definition as described in section 3.2.1. This implies that control action has to be applied to keep the system trajectory in the vicinity of the surface.

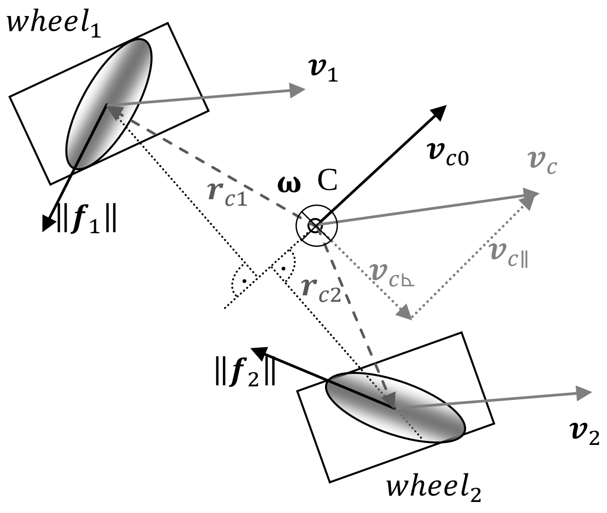


Figure 27. Velocity and force vectors of a braking omnidirectional platform

Let's discuss the three components of the sliding surface separately. The velocities and forces are illustrated on Figure 27. for two general omnidirectional wheels. C marks the geometrical center of the platform. The rectangles represent the wheels, their axis of rotation is parallel to their shorter side. The elliptical shapes are the rollers touching the ground, their active direction is parallel to the longer diameter of the ellipse.

3.4.1 Perpendicular component

The component $s_{i\perp}$ in equation (44) describes the capacity of the i-th wheel to decrease the velocity component of the wheel perpendicular to $\boldsymbol{v}_{c0}$. This velocity component is due to angular velocity of the platform and sideways translation with regards to $\boldsymbol{v}_{c0}$ as shown in equation (38). To have $s_{i\perp}$ correspond purely to the translational movement of the wheel center velocity is as follows:

$$\boldsymbol{v}_i^* = \boldsymbol{v}_i - \boldsymbol{\omega} \times \boldsymbol{r}_{ci} \qquad (49)$$

according to equation (38) obviously this equals $\boldsymbol{v}_c$ instantaneous platform velocity. Then:

$$s_{i\perp} = \|\boldsymbol{f}_i\| \boldsymbol{v}_{c\perp}^T \qquad (50)$$

where $\boldsymbol{v}_{c\perp}$ is the component of $\boldsymbol{v}_c$ perpendicular to $\boldsymbol{v}_{c0}$:

$$\boldsymbol{v}_{i\perp} = \left\|\{-v_{c0y}, v_{c0x}, v_{c0z}\}\right\| \boldsymbol{v}_i \left\|\{-v_{c0y}, v_{c0x}, v_{c0z}\}\right\|^T \qquad (51)$$

and $\|\boldsymbol{f}_i\|$ is a unit vector pointing in the direction of the friction force:

$$\|\boldsymbol{f}_i\| = -sign(\boldsymbol{v}_i \boldsymbol{n}_{wi}^{\mathrm{T}})\boldsymbol{n}_{wi} \tag{52}$$

The component described by equation (50) is a vector product, its sign depends on the cosine of the two vector's angle, if it is negative then the friction force can decrease the translational movement of the wheel with respect to $\boldsymbol{v}_{co}$. The value is scaled by the magnitude of this translational velocity. As an example on Figure 27. $wheel_2$ can decrease $\boldsymbol{v}_{c\perp}$ while $wheel_1$ would only increase it.

3.4.2 Angular component

Friction forces at the wheels cause torque around the center of rotation, and in consequence angular acceleration of the platform, if this acceleration opposes instantaneous angular velocity then the brake needs to be on otherwise it needs to be off. The vector pointing in the direction of angular acceleration, written for the geometrical center:

$$\|\boldsymbol{\beta}_{wi}\| = \|\boldsymbol{f}_i\| \times \|\boldsymbol{r}_{ci}\| \tag{53}$$

The direction of this vector is vertical. Clearly when this has the same sign as $\boldsymbol{\omega}$, then it increases it, so in that case the brake should be switched off. The corresponding component can be written as follows:

$$s_{i\omega} = \|\boldsymbol{\beta}_{wi}\|\boldsymbol{\omega}^{\mathbf{T}}|\boldsymbol{r}_{ci}| \tag{54}$$

Again this component is a vector product, in my model the angular velocity vector either points up or downwards, so $s_{i\omega}$ is negative when they point in opposite directions – brake should be actuated – and positive when they point in the same direction – brake should be off. The value is scaled by the magnitude of angular acceleration multiplied by wheel distance from the center to obtain a velocity dimension. In the example of Figure 27. $\boldsymbol{\omega}$ points downwards – in the $-z$ direction – therefore the friction force of $wheel_1$ can generate an angular acceleration that decreases it while the force arising at $wheel_2$ would increase it.

3.4.3 Parallel component

This component has to be handled differently as this is the one responsible for *Goal 2*, for stopping the platform in a short distance. Also generally the friction force of all wheels are generated to oppose this component, therefore the train of thought applied for the previous two cases is not applicable.

Solution 1: The simplest approach for the problem is to let the component be zero yielding:

$$s_{itot} = s_{i\perp} + s_{i\omega} \tag{55}$$

using equations (50) and (54). As mentioned in the beginning of section 3.3 from the input-output cross coupled nature of omnidirectional platforms, it follows that when braking any wheel all three degrees of freedom are affected, so the platform will eventually stop if some of the brakes are actuated.

Degenerate scenarios are possible, when the error function $s_{itot} \geq 0$ for $\forall\, i$. This implies that all brakes are off and the platform is not decelerating. This situation can happen for example when a Mecanum platform is moving exactly at a 45° to its local x axis and $\boldsymbol{\omega} = 0$ – this means $s_{i\omega} = 0$. In this case for two wheels, their active direction and $\boldsymbol{v}_i$ is parallel with platform velocity, for the other two their active direction is perpendicular to platform velocity, making them incapable of slowing the vehicle. Since all error functions are close to zero, it is clearly possible that all errors become positive

– due to noise or measurement errors – turning all brakes off. This scenario can be handled by different approaches. Probably the simplest approach is to activate all brakes when $s_{itot} \geq 0$ for $\forall\, i$. This guarantees divergence of system trajectory from the degenerate scenario.
When letting $s_{i\|} = 0$, only angular and perpendicular velocity components are “punished” so the controller drives the parallel component to zero only indirectly, thus allowing no control over braking distance.
To construct a better sliding surface $s_{i\|}$ has to be set to something other than zero.
Solution 2: In order to retain control over platform swerving (*Goal 1*) it is important not to let the parallel component dominate s_{tot}, therefore an angle dependent weighing function has to be defined, so that *Goal 2* only dominates the error function when the given wheel is most effective i.e. its active direction is close to parallel with its velocity ($\boldsymbol{n}_{wi} \parallel \boldsymbol{v}_i$). A cosine function of the angle between the active direction of the wheel and the velocity of its center seems to be a good candidate, as it yields 1 when the vectors are parallel and 0 when they are perpendicular.

$$\delta_i = arccos\left(\frac{\boldsymbol{n}_{wi}\boldsymbol{v}_i^T}{|\boldsymbol{v}_i|}\right) \tag{56}$$

However the derivative of cosine is small around zero, intuitively it is clear that the higher the value in the vicinity 0°, the higher the “punishment” of the parallel component. I propose a more useful weighing function for $s_{i\|}$ to be able to put an emphasis on either control goal.
Let’s define the parallel component in the form:

$$s_{i\|} = \psi(\delta_i)\|\boldsymbol{f}_i\|\boldsymbol{v}_{i\|} \tag{57}$$

where $\psi(\delta_i)$ is the weighing function dependent on the angle between $\boldsymbol{n}_{wi}$ and $\boldsymbol{v}_i$. $\psi(\gamma) \in [-1{,}1]$ and $\boldsymbol{v}_{i\|}$ is the component of $\boldsymbol{v}_i$ parallel to $\boldsymbol{v}_c$. The enhanced function I propose is a cosine function mirrored piecewise to the line connecting its extremes. The index i is omitted for clarity.

$$\psi = \left(-\left|\frac{2\delta}{\pi}\right| + 1 + \left(-\left|\frac{2\delta}{\pi}\right| + 1 - \cos(\delta)\right)\right)^k, \ \delta \epsilon [-\pi, \pi] \tag{58}$$

where k is a tuning constant, an odd positive integer, in order to keep the sign of the function. After simplification:

$$\psi = \left(-2\left|\frac{2\delta}{\pi}\right| + 2 - \cos(\delta)\right)^k \tag{59}$$

Figure 28. shows the initially envisioned cosine weighing function, and enhanced weighing functions with $k = 1$ and $k = 7$.

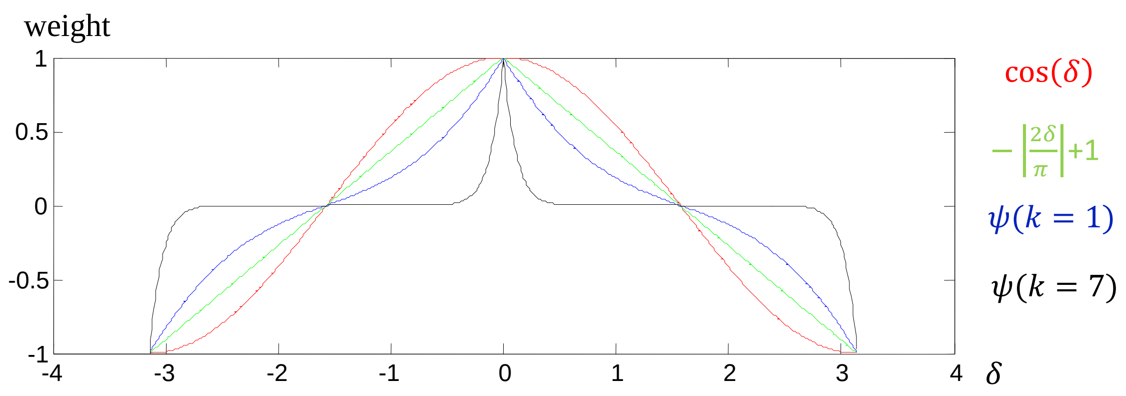


Figure 28. Enhanced weighing functions for the parallel component, larger k values yield a steeper function

The effect of the function is intuitively clear from the graph, by choosing a value for k it is possible to tune the controller and favor orientation control or stopping distance, by choosing a high k value, behavior increasingly similar to **Solution 1** can be obtained, a lower value yields shorter stopping distance. The differences are demonstrated trough experiments with a Mecanum platform in section 3.6.

3.5 Disturbance rejection

As Khalil [27] explains in his book uncertainties of a nonlinear system may appear in a structured or in an unstructured form. Unstructured uncertainties are, for example incorrectly identified model parameters, or external noises, while structured uncertainties are a result of modeling choice, i.e. of deliberate model simplification.
The philosophy behind the controller ensures high immunity against structured disturbances, because it is created with the assumption that only the kinematic structure of the vehicle is known, and requires no information on system dynamics. Since the control goals are not defined as an exact function of time, but rather as constraints on the velocities in three degrees of freedom, these goals can be reached even with a changing dynamic model.
It is intuitively clear that the control rule is robust against unstructured disturbances, because it only uses the direction of force and velocity vectors to determine a binary on off rule for the brakes, making the magnitude of the vectors of secondary importance, thus effectively be immune against feedback noise. The controller is also immune against small changes in the model, for example the changing ground contact position of a real omnidirectional wheel (see section 2.3.2.) makes small deviations in the direction of the ground force however the controller only uses the sign of its projection on velocity components, making it robust against small directional changes.
The following chapter shows the working controller in simulation and demonstrates its capabilities, under various conditions.

3.6 Demonstrative experiments

To demonstrate the capabilities of the system, I realized the controller in the Modelica language, and integrated it into the omnidirectional vehicle simulation described in section 2.5. I ran experiments with the two most popular platforms.
The **Mecanum** platform has four wheels, the vehicle body is represented by a point mass with inertia, and the load is represented by a simple point mass. Both the position and value of the masses, as well as vehicle dimensions are configurable. The platform is

shown on Figure 29. The body is represented by the red brick and the load is the green sphere. It is easy to see, that the center of gravity is offset to the front left of the platform simulating a carelessly loaded heavy cargo, and an obese driver.

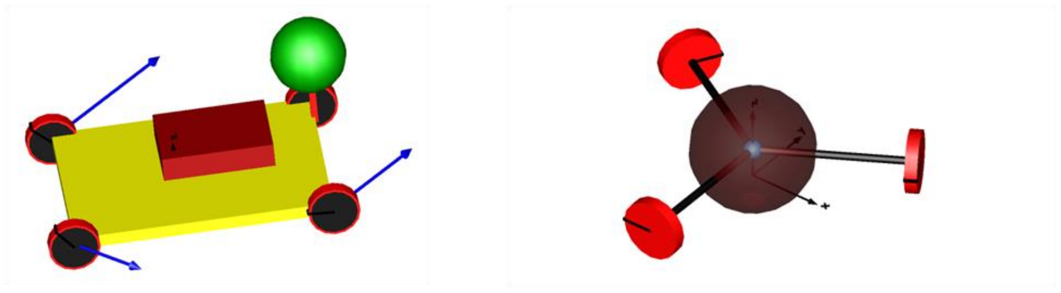


Figure 29. The most popular omnidirectional platforms in simulation, rendered by Dymola

I also built a model of a **three wheeled robot**, with 0° rollers. Its body is represented by a point mass with inertia.
Speed measurement feedback can be realized by any convenient method, however I propose an optical speed sensor mounted on the platform, described later in this dissertation, in section 4. The sensor measures true ground speed, and angular velocity, and works much like a conventional optical mouse. This method provides better speed and precision than similar works from the literature [28], [29]. The sensor is included in the simulation by quantizing the simulated speed values with a zero order hold, and adding random noise.
The control loop works the following way: When the driver – or autopilot – pushes the brake pedal, the brake assist gets activated and the value of $\boldsymbol{v}_{c0}$ is stored. Based on the velocity values $\boldsymbol{v}_c$ and $\boldsymbol{\omega}$, received from the optical feedback unit, instantaneous wheel velocities are calculated according to equation (38). During the braking maneuver the controller continuously evaluates the four s_{itot} error functions for the four wheels. Based on the sign of the i-th error function the actuator at the i-th wheel lets brake pressure trough, – the brake force asserted by the driver gets applied – or inhibits braking – lets the wheel roll freely. For practical reasons, to avoid low speed drift, the brake assist is only active above a certain velocity threshold, below that it is switched off, and all the brakes are activated normally.

3.6.1 Braking in a straight line

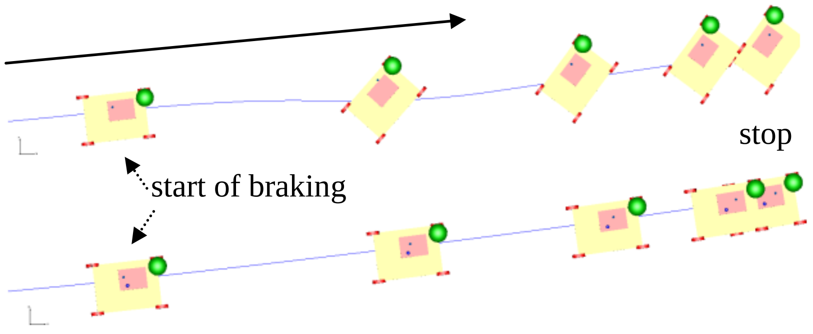


Figure 30. Demonstration of brake assist, slowing down from 10m/s

Figure 30. shows a demonstration of the brake assistant. The upper part shows snapshots of an emergency braking trajectory at regular intervals, with the brake assist deactivated, the lower part shows the same experiment with the brake assist activated. The initial velocity is 10m/s.

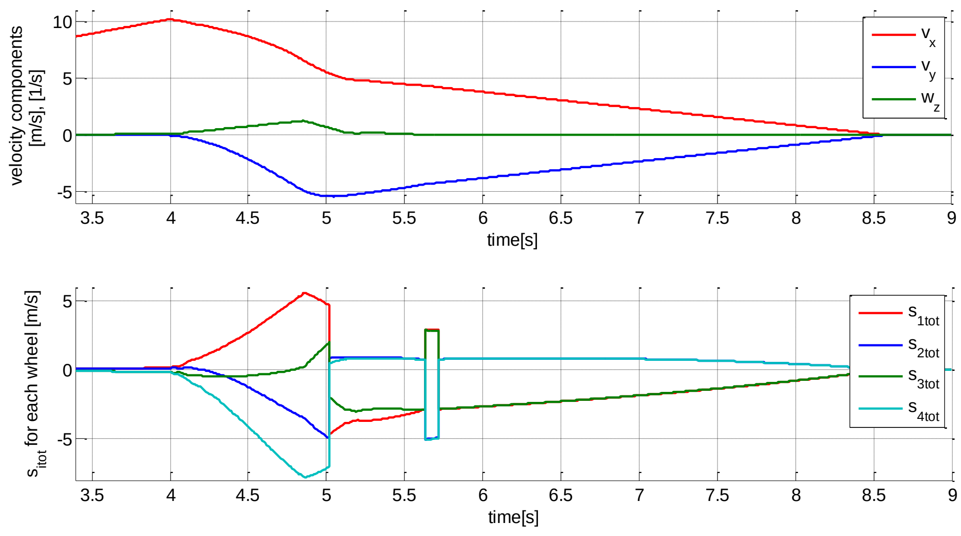


Figure 31. Platform velocity components in body reference and values of s_{itot} vs. time for the straight line braking maneuver. Mecanum platform, unassisted case.

The difference between the two trajectories is obvious, with the brake assist activated, the vehicle does not swerve, but keeps its initial direction of movement.

Figure 31. shows platform linear and angular velocities in local coordinates on the upper graph and the error function for each wheel on the lower. For both graphs the horizontal axis is time. On the error graph, the red, blue, green, cyan colors show the error for wheels 1, 2, 3, 4 respectively. The brakes were engaged at 4s.

Figure 32. shows the same quantities for the same experiment, this time with the brake assist activated.

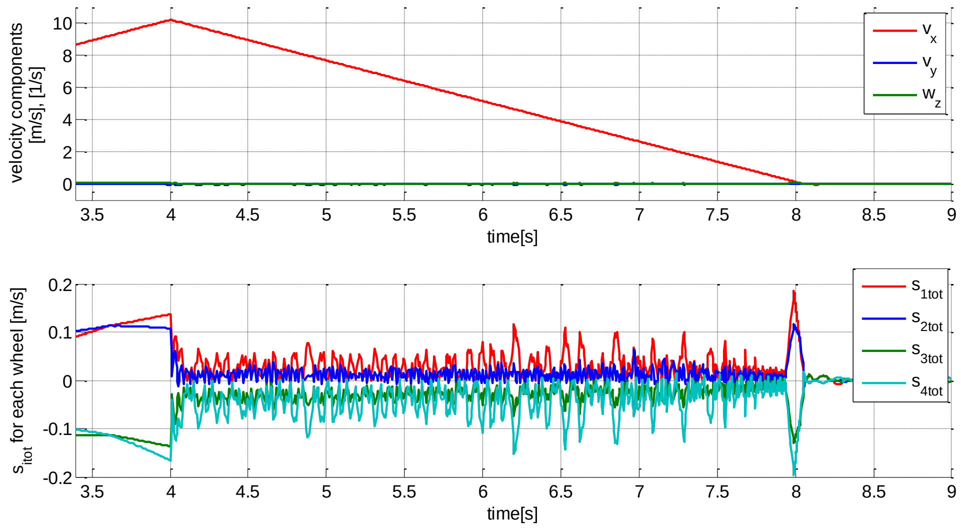


Figure 32. Platform velocity components in body reference and values of s_{itot} vs. time for the straight line braking maneuver. Mecanum platform, assisted case.

It is clear that the error function (the distance from the sliding manifold) is an order of magnitude smaller in the assisted case, also sideways velocity and angular velocity of the platform are negligible.

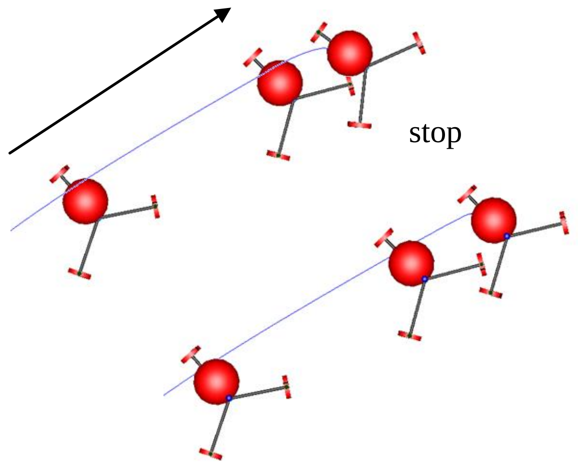


Figure 33. Brake assist on the three wheeled robot. Upper left unassisted, right below assisted.

I ran a similar same experiment for the three wheeled omnidirectional platform to show how the general control rule works for a different platform. Note that the only difference in the control equations comes from the difference in the kinematics of the two robots. The three wheeled robot has a homogeneous spherical body of 500kgs with plastic-like density. It is mounted on the platform with an offset relative to the geometrical center to create imbalanced wheel load. The body is represented by the red sphere on Figure 33. The first snapshot is taken while moving with constant speed, the second shortly after braking and the third at the final position. The effectiveness of the controller is demonstrated on a short distance stop, there is significant difference between the orientation in the resting position for the unassisted (left) and the assisted (right) case.

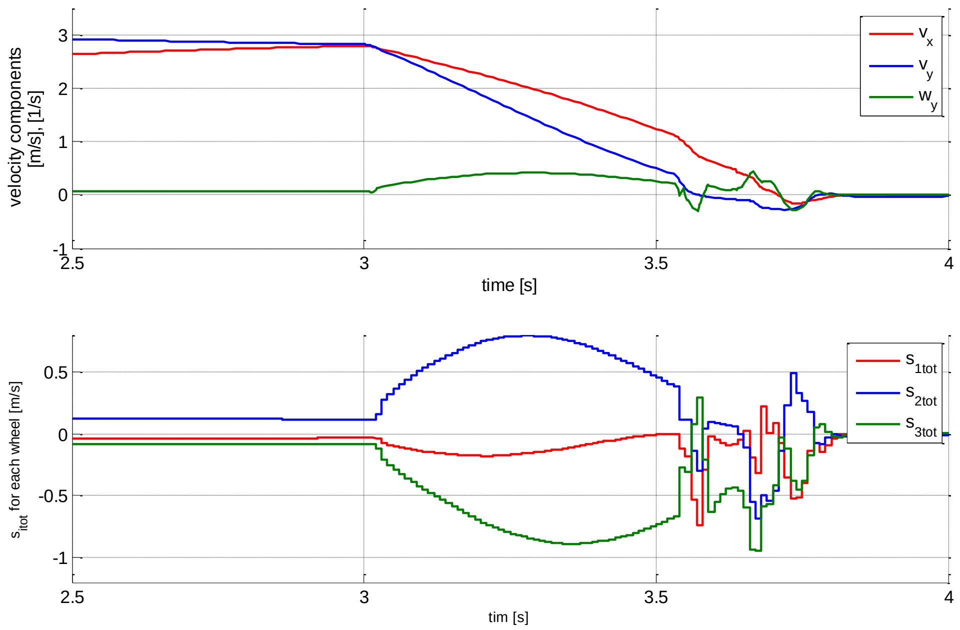


Figure 34. Platform velocity components in body reference and values of s_{itot} vs. time for the straight line braking maneuver. 3-wheel platform, unassisted case.

Figure 34. upper graph shows linear and angular velocities in the vehicle frame. It is clear that braking was started at an approximately 45° movement ($v_x = v_y$) however it does not stay that way, and the platform also gains angular velocity. The lower graph shows the error functions for the three wheels, in this experiment the vehicle comes to a complete halt within less than a second.

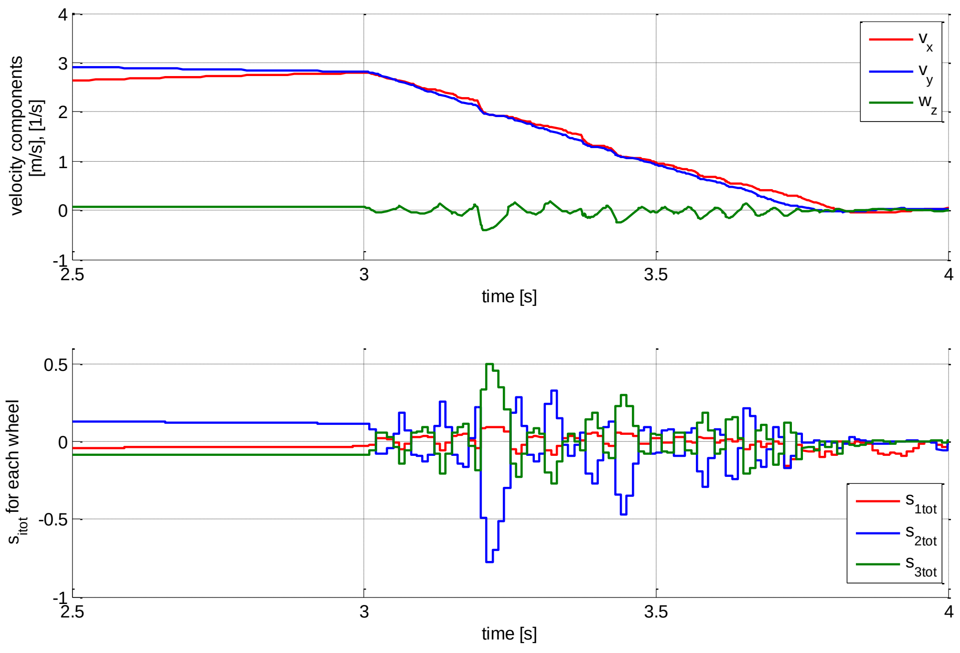


Figure 35. Platform velocity components in body reference and values of s_{itot} vs. time for the straight line braking maneuver. 3-wheel platform, assisted case.

Figure 35. shows the error and velocity functions for the same experiment. The errors decline more rapidly and the linear velocity components “stay together” showing that the initial 45° direction of movement is kept.

3.6.2 Demonstration of disturbance rejection

The controller behavior in the presence of structured uncertainties is, unfortunately difficult to test in simulation, however my approach on that was to create a fairly complex physical model in the simulator, while the controller is based on the kinematic model of the vehicle, which is clearly a crude representation of reality. The model in simulation describes the vehicle with a physics based approach, the platform is built from elements with mass, inertia and complex dynamic properties, where applicable. This approach ensures that there is considerable difference between the simulated model and the model the controller is based on, giving a fairly good idea on structured disturbance rejection capabilities, without actually testing the controller on real hardware. Unfortunately at the time of writing I had no such opportunity.
The situation is a lot better when it comes to demonstrating the capabilities of the controller facing unstructured uncertainties. I investigated several kinds of disturbances. I quantized the feedback signal in time, thus creating a time dependent error in the feedback, also getting closer to the behavior of a real sensor. The sampling time of the simulated velocity sensor was 0.01s and a zero order hold supplied the feedback between sampling instants. All the simulations of section 3.6. were made with this kind of feedback, the effect of the zero order hold is clearly visible on some of the signals (see Figure 34. and Figure 35. for example).

Sensor noise is simulated by a uniform distribution[9] (η) centered around the real value. It can be set independently for the linear and the angular velocities. Also an offset can be incorporated in the signal.

$$\tilde{\boldsymbol{v}}_c = \boldsymbol{v}_c + \eta(\boldsymbol{v}_c) + \boldsymbol{v}_0 \tag{60}$$

where $\tilde{\boldsymbol{v}}_c$ is the noisy linear velocity signal and $\boldsymbol{v}_o = [v_{ox}, v_{oy}]$ is an offset. Similarly

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \eta_\omega(\boldsymbol{\omega}) + \boldsymbol{\omega}_0 \tag{61}$$

where $\tilde{\boldsymbol{\omega}}$ is the noisy angular velocity measurement and $\boldsymbol{\omega}_o$ is an offset.
To introduce another kind of disturbance I created a patch on the ground with smaller friction coefficient and made the vehicle brake with some of its wheels on the "icy patch" and the others on regular ground. The maneuver is shown on Figure 36. The bright red colored platform is the assisted and the dark red is the unassisted vehicle.

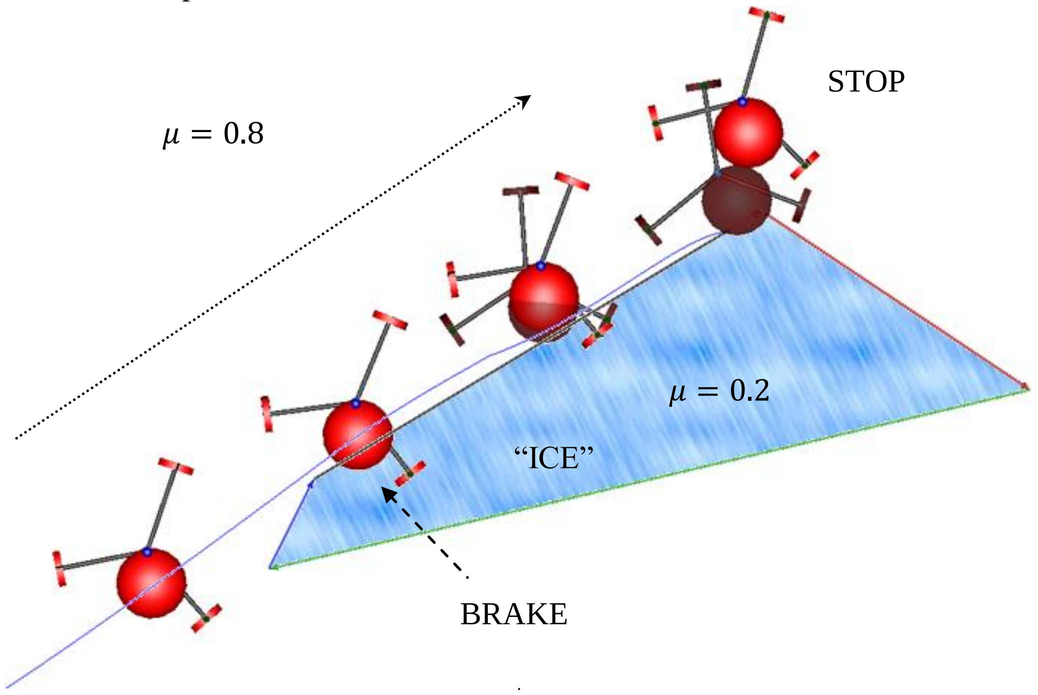


Figure 36. Braking on an ice patch. Showing snapshots of two experiments overlaid – bright assisted, dark unassisted platform.

Parameters of the platform (mass inertia load distribution, wheel characteristics etc.) are the same as in the experiment of section 3.6.1. Braking is started when one of the wheels is already on the ice patch, at 3s. The unassisted robot quickly gains some angular velocity due to the wheel on the ice loosing traction and stops after a considerable amount of swerving. The assisted robot however continues on without any significant change in orientation, or direction of movement. The most significant difference is that the assisted robot stops almost half a platform width farther. This is obviously caused by having to release the brakes very often on the wheels on normal ground to even their torques with the wheel on ice. Error functions and state variables can be seen on Figure 37. and Figure 38., in this case the differences speak for themselves.

[9] Uniform distribution might not be the best simulation of sensor noise, however my purpose here is to demonstrate disturbance rejection of the controller.

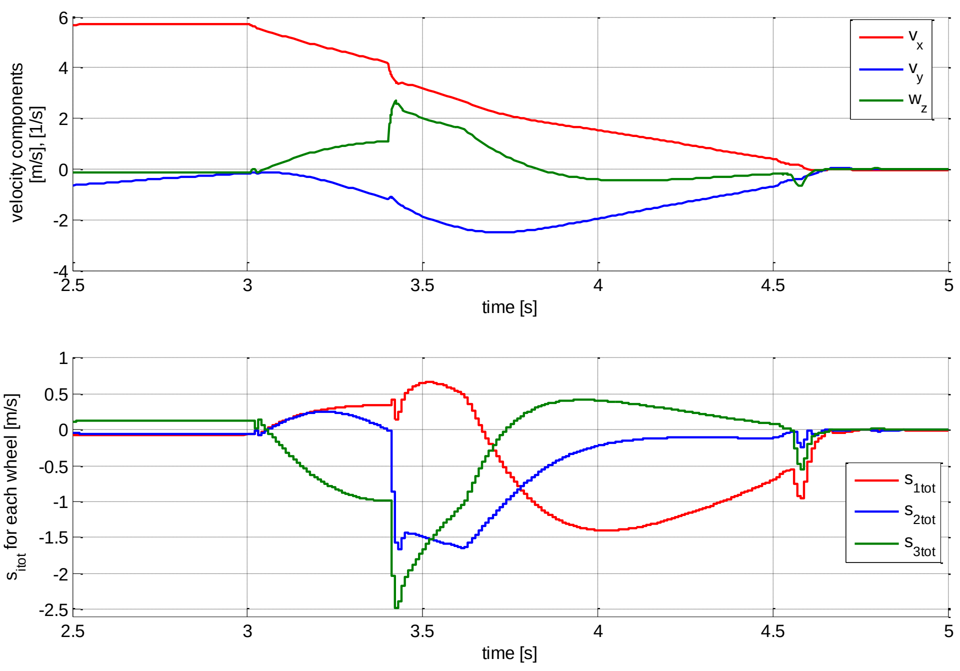


Figure 37. Braking on an ice patch – Velocities and errors, unassisted

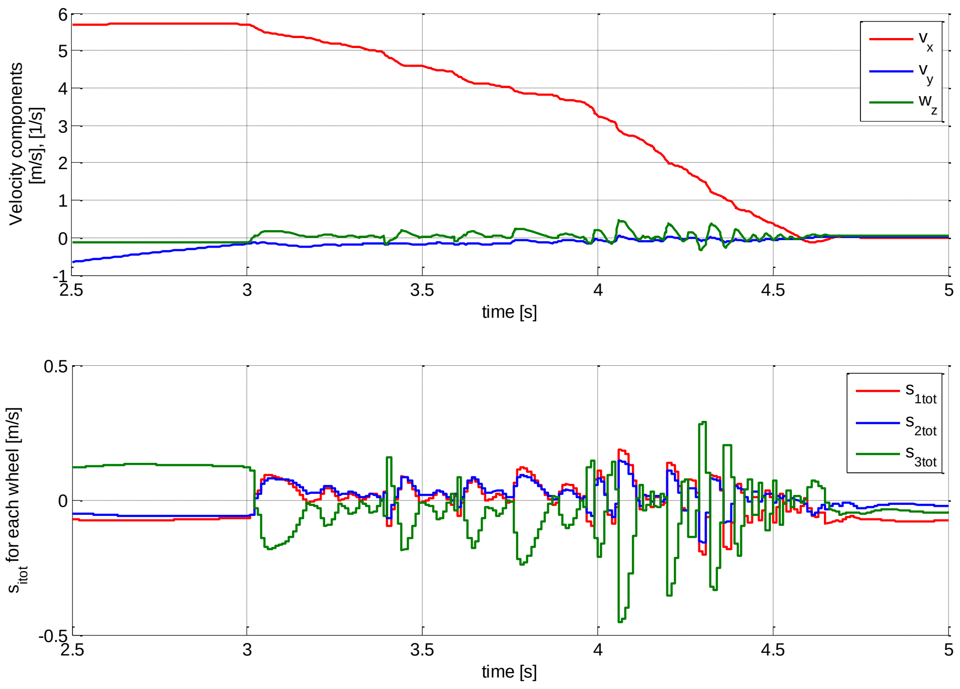


Figure 38. Braking on an ice patch – Velocities and errors, assisted

I also added some noise to the feedback. Figure 39. shows the final stopped state of the previous ice patch maneuver with a noisy feedback signal, where $\boldsymbol{v}_o = \{0.01, 0.01, 0\}$, $\boldsymbol{\omega}_o = \{0,0,0.05\}$, with the noisy values being linearly distributed in the range of $\pm 15\%$ around the real value. The platform marked with a yellow circle was operated with a noisy feedback, the dark red platform and the bright red platform – partly occluded by the one with noisy feedback – are exactly the same as in the final position of the previous experiment (Figure 36). The difference is small in spite of the amount of noise and offset injected into the system.

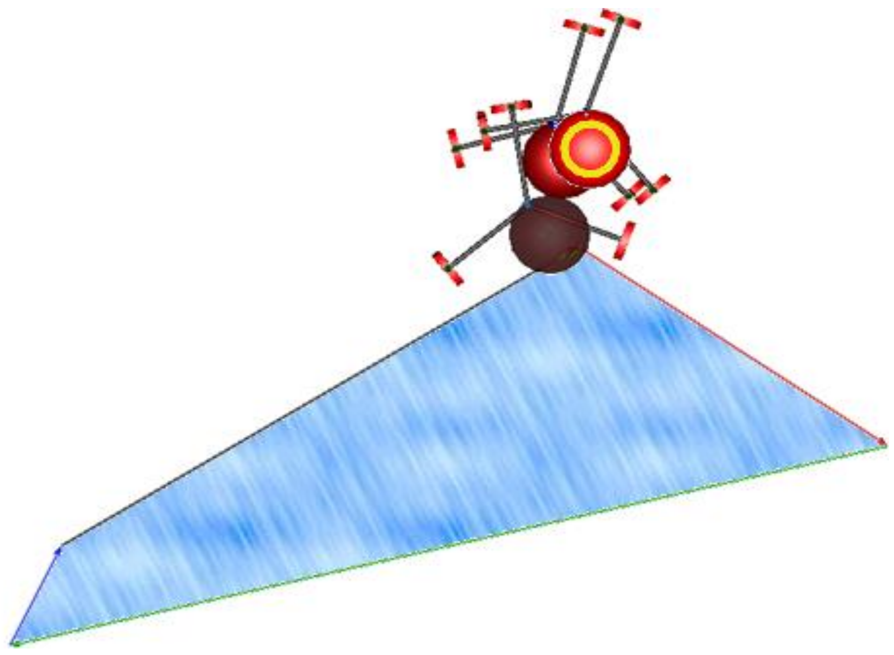


Figure 39. The effect of noisy feedback – noisy platform marked with yellow circle

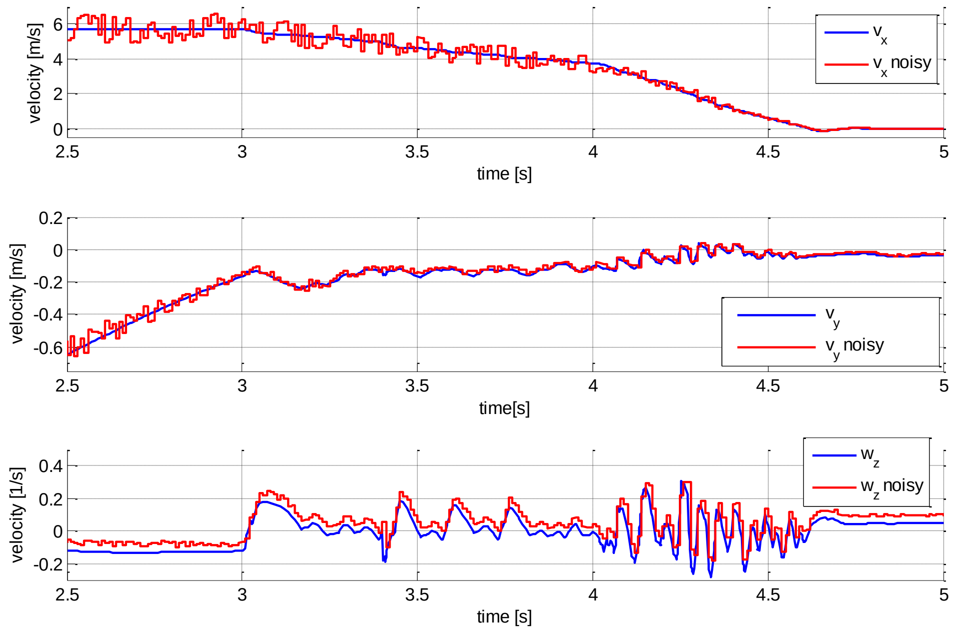


Figure 40. Simulated noise on the feedback signals – blue original, red noisy

Figure 40. shows the noise applied to velocity signals, measured noisy signals are marked with red, the original signals are in blue. The graphs show – from top to bottom – x and y component of linear velocity and angular velocity.

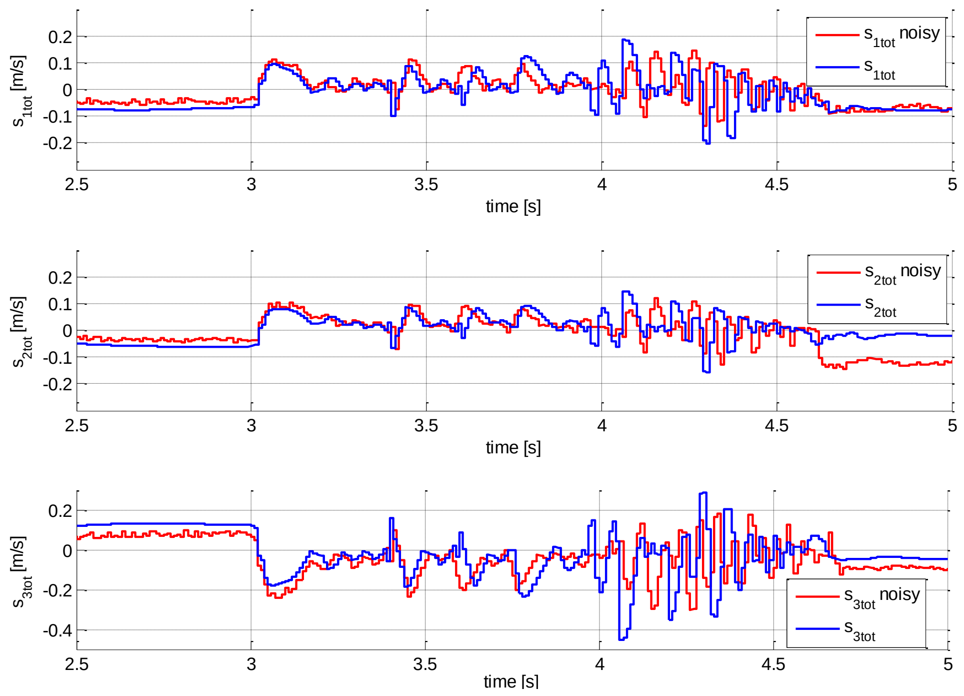


Figure 41. Comparison of error functions – blue no noise – red noisy feedback

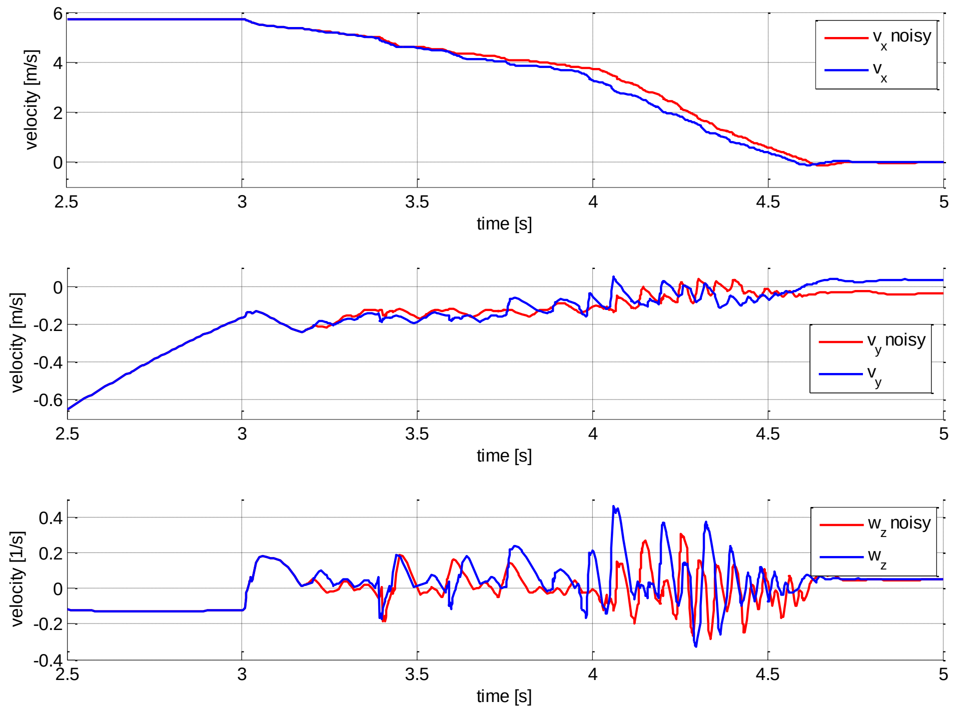


Figure 42. Comparison of velocities – blue no noise – red noisy feedback

Figure 41. shows the error signals for wheels 1, 2, 3 (top to bottom) for the noisy experiment in red and for the experiment with no noise in blue. Figure 42. shows velocities for the two experiments, again noisy in red and blue with no noise. x, y linear velocities and angular velocity, top to bottom. It is clearly visible that in this experiment the noise had most effect on the stopping distance, it had minimal effect on the angular velocity, and as it can be seen in the final position from the animation the controller still prevented the platform from swerving.

3.6.3 Braking in a turn while translate movement

Braking in a straight line, the brake assist controller demonstrated that it can prevent unwanted swerving. Also high disturbance rejection capabilities were shown. When slowing from straight movement the most important components of the s_{itot} error function are $s_{i\perp}$ and $s_{i\omega}$, as these are responsible for keeping the trajectory straight. When there is substantial angular velocity the situation is more complicated. I created demonstrative experiments to show the effect of different realizations of the $s_{i\parallel}$ component as explained in section 3.4.3.

The example on Figure 43. shows a braking trajectory from a turn-while-translate maneuver, again snapshots from the experiment were captured. The robot starts from the bottom of the figure, the first snapshot is at the beginning of braking the last is a complete standstill. The ice patch is added as a challenge to the controller, also to emphasize the swerving effect. Sub-figure a) shows the original trajectory, the unassisted case. As explained in section 3.4.3 the controller prevents swerving as intended with letting $s_{i\parallel} = 0$, however this implies longer stopping distance as shown in sub-figure b), where the experiment was executed under the same conditions, but with an active controller implemented with $\psi(\delta) = 0$. Sub-plot c) shows the same experiment with a cosine weighing function for $s_{i\parallel}$, $\psi(\delta) = \cos(\delta)$. The experiments support my theory that the vehicle stops in a shorter distance, but due to the relatively high value of the cosine function around zero, directional control is worse.

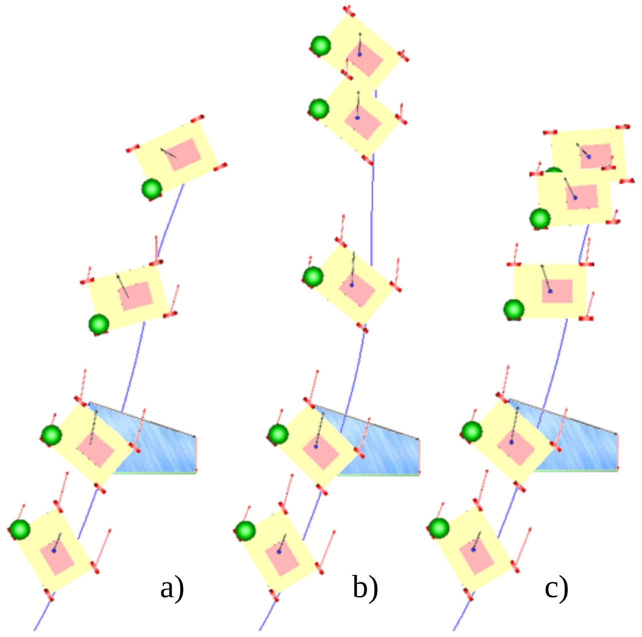


Figure 43. Snapshots from a braking maneuver a) no assistant b) 0 weighing function c) cosine weighing function

Figure 44., Figure 45. and Figure 46. show the state variables v_{cx}, v_{cy} and ω_z for the three experiments, together with the four s_{itot} error functions.

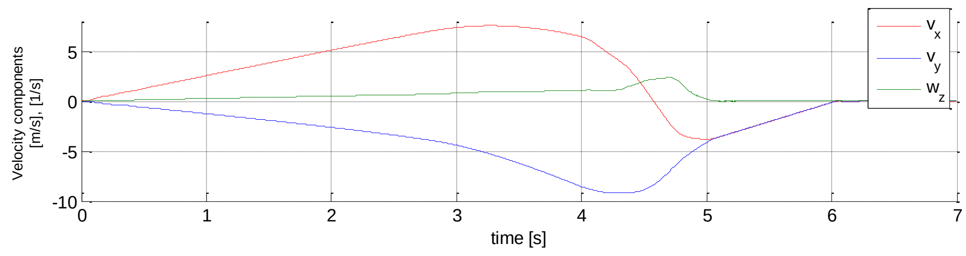


Figure 44. Linear and angular velocities of the unassisted vehicle (Figure 43. a))

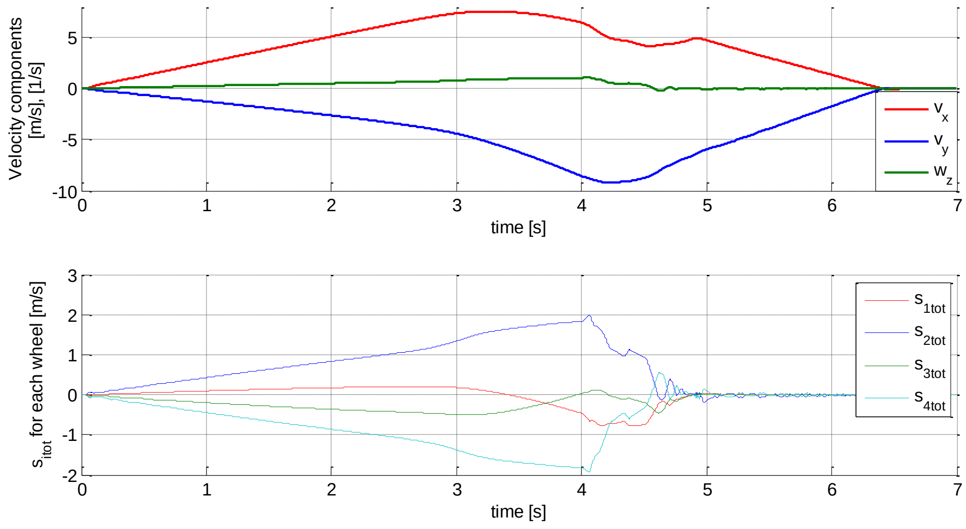


Figure 45. Velocities and error functions for the assisted maneuver – 0 weighing function

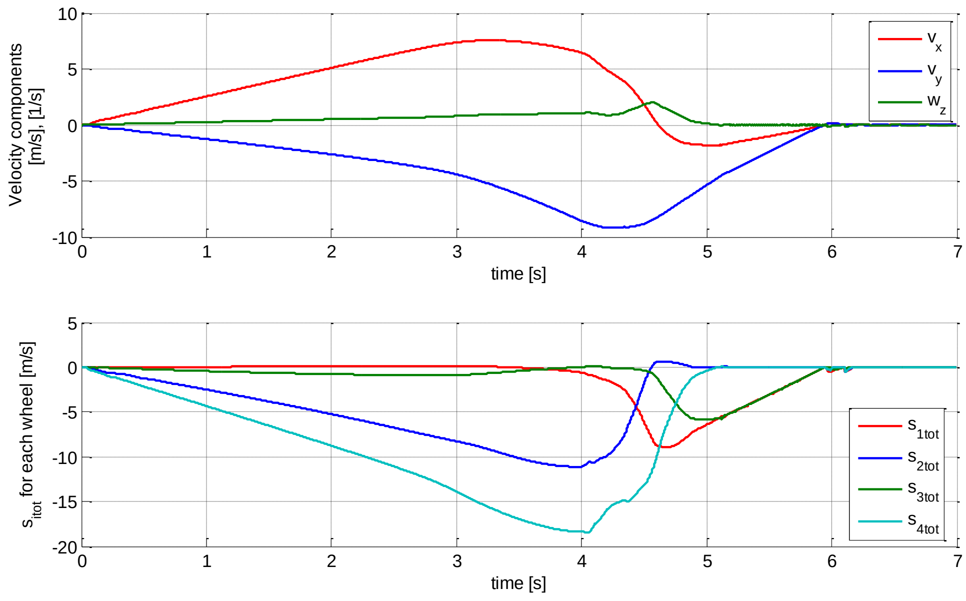


Figure 46. Velocities and error functions for the assisted maneuver – cosine weighing function

Results of the experiments related to the enhanced weighing function (section 3.4.3) can be seen on Figure 47. As a reference sub-figure a) shows the experiment with a cosine weighing function, the same as Figure 43. c). Sub-figure b) shows the result obtained with $k = 1$ and c) with $k = 7$. The merit of the enhanced ψ function lies in the ability to choose a preference of keeping platform orientation, or stopping in a short distance. Figure 48. shows state variables and error functions for the experiment with $k = 1$ and Table 2. sums up the orientation changes and stopping distances for the experiments in this section, for easy comparison. Both orientation changes and stopping distances are measured between the start of braking and the final position.

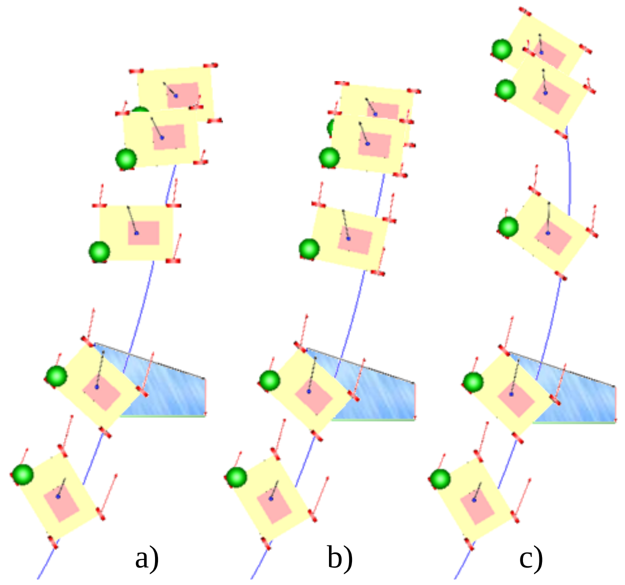


Figure 47. Comparison of weighing functions a) cosine (same as Figure 43 c)) b) enhanced function *k* = 1 c) *k* = 7

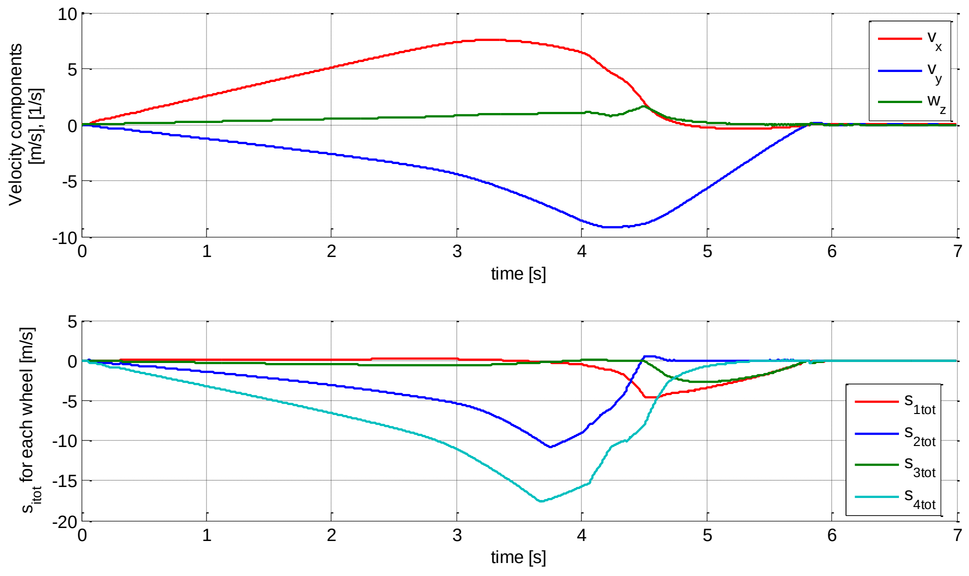


Figure 48. Velocities and error functions for the assisted maneuver – enhanced weighing function *k* = 1

	unassisted	$\psi = 0$	$\psi = \cos(\gamma)$	enhanced ψ $k = 1$	enhanced ψ $k = 7$
orientation change	83.8°	18.9°	63.2°	52°	26.3°
stopping distance	10m	12.2m	9.7m	9.3m	10.4m
figures	Figure 43. a) Figure 44.	Figure 43. b) Figure 45.	Figure 43.c) Figure 46.	Figure 47. b) Figure 48.	Figure 47. c)

Table 2. Comparison of controller implementations

3.7 Conclusion

> **Thesis II: I created a brake assist controller for a class of mobile robots, that force the vehicle to keep its orientation on a well behaved trajectory during braking. The control method is nonlinear sliding mode control and it is generally applicable to the class of mobile robots with omnidirectional wheels. The advantage of the control law is that it is robust against uncertainties and disturbances to the extent that it achieves the control goal by using only the kinematic model of the platform, with no information on load distribution and tire dynamic characteristics.**

Omnidirectional wheels are an attractive choice for mobile robotic applications requiring high maneuverability. However their advantages come at a price, they also have some aspects in which they are inferior to regular tires. My research focused on a disadvantage that stems from the fact that this type of wheels have no force generation capabilities in the direction perpendicular to their rollers axes – the very reason they enable omnidirectional movement. This unidirectional force generation makes omnidirectional platforms change their orientation when undergoing heavy deceleration. Uneven wheel load, locally different ground characteristics, differences in wheel material or angle have an effect on the force generation capabilities of omni-wheels. The unbalanced friction forces with no opposing side forces cause the vehicle to turn and/or translate while braking. This behavior obviously may come as a surprise to the operator, causing panic and/or loss of control resulting in an accident. As an example, slowing down an omnidirectional forklift in a tight warehouse aisle may result in the vehicle bumping into the shelves due to its higher width when turning.

To help with this situation, I created a brake assist controller. As a first step I investigated the dynamic model of the braking platform, and concluded that it contains nonlinearities and both structured and unstructured uncertainties. As a second step I researched viable control methods for uncertain nonlinear systems and settled with sliding mode control. Linear and angular velocities of the platform are measured and fed back, and brakes are actuated in a discontinuous manner, depending on their ability to decrease unwanted velocity components of the platform. Finally I proposed a method to tune the control law to favor smaller orientation change or shorter stopping distance. The advantage of this approach is that the closed control loop drives the state trajectory of the vehicle to zero without having information on dynamic properties of the platform. This results in exceptional disturbance and parameter uncertainty rejection properties.

Related publications:

[Kal13], [KV12a], [VAL12]

4 Optical velocity measurement

Velocity of mobile robots can be measured in many ways. The best method to use depends on the application. Motion can be measured by sensors exterior or interior to the robot. The use of external sensors has the advantage of the robot not having to carry the sensor thus introducing fewer limitations on size, durability and power supply. On the other hand a great disadvantage is that the motion is constrained to the range of the sensor. Sensors mounted on the robot itself usually do not limit the area of movement, however they need to be designed with the conditions of usage in mind i.e. vibrations, limited power and computational capacity etc. The type of sensor that can be used may also depend on the propulsion system. For example incremental encoders are great for wheeled robots, however they have little use for legged, or flying locomotion.
In general, methods depending on the measurement of wheel revolutions have a limited use in measuring movement speed of robots, where high values of slip are common for normal operation. A good example are tracked and omnidirectional vehicles. For these, alternative methods have been conceived. One popular approach uses inertial sensors. These gained popularity with the adoption of MEMS technology. A serious drawback is that inertial sensors usually measure acceleration, so when integrating their output to acquire velocity measurements one introduces cumulative errors due to noise.
GPS evolved to the level where it is both affordable and accurate enough to mount on robots for speed measurement. However its dependence on direct satellite visibility makes it ineffective in environments with tall obstacles, or overhead covers, for example forests, natural and urban canyons and indoor locations.
Optical sensors supply by far the most information, and as greater and greater processing capabilities become readily available, their use becomes more widespread. Their main advantage is that they can supply detailed movement information, independent from the locomotion system. The main challenge is that they are sensitive to dirt, lighting conditions, calibration and they are only as good as the algorithm behind them.
In general the conclusion is that no single best solution exists, one has to make a compromise for the current task at hand, or use multiple methods and sensor fusion.
Since there is a great potential in optical sensors in general, my work focuses on the family of optical motion sensors. I created a sensor concept, that features high resolution without any compromise in speed.

4.1 Introduction to optical ego movement sensors – related work

When applying optical speed measurement usually two main approaches can be found in the literature. The first method employs a forward looking camera, or cameras and extracts velocity information from the movement of the surroundings, by optical flow, or by 3D from motion or stereo techniques. The forward looking camera is subject to changing lighting conditions and usually a detailed image is needed to extract information.
The other method uses an imaging sensor facing the ground, a very good example is the well known optical mouse. This method requires simpler devices and algorithms as lighting conditions can be handled better and the variety of the image is smaller, in most cases the ground has a simple texture, which moves as a single object.
Optical speed measurement is an emerging discipline, with existing commercial solutions and research activity in the academic sector, however many problems remain

unsolved. As for commercial technologies the most widely known example is the optical mouse, which on the other hand has generated a fair amount of academic research. The optical mouse uses two distinct but essentially similar techniques for displacement calculation. The classical method uses angled LED illumination and relies on the micro texture of the surface. The more advanced method is laser speckle pattern technology. Laser speckle patterns can be observed when a rough surface (rough, relative to the wavelength) is illuminated with a coherent light and the interference of the reflected light waves creates a surface dependent random intensity map on the detector [13]. (Figure 49.) When the detector is moved relative to the surface, the speckle pattern changes accordingly and optical flow can be calculated. The advantage over surface texture based methods is its accuracy and ability to function properly on relatively textureless smooth surfaces.

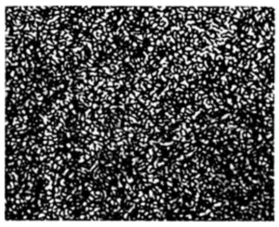


Figure 49. A laser speckle pattern (from [13])

Frequency analysis is a less frequently used method. The light reflected from the surface travels through an optical grating, and is focused on a pair of photo-detectors. The surface elements, passing in front of the grating generate a certain signal frequency in the detectors depending on the sampling frequency, ground speed, grid graduation, ratio of the image, size of the surface elements and the size of the picture on the grating. The difference of the two signals is computed and the frequency of the difference signals corresponds to true ground speed [12].

Indoor dead reckoning solutions for small mobile robots using optical mice were suggested by several authors ([40], [6], [48]) T.W. Ng investigated the usability and accuracy of optical mice for scientific measurements in several articles ([35], [36]) with good results. It was found that the readings possessed low levels of error and high degrees of linearity. The mean square error for measurements in the x-axis increased significantly when the distance between the surface and the detector was increased possibly caused by the illumination direction of the mouse. Several researchers proposed the use of optical mice as a dead reckoning sensor for small indoor mobile robots in one and two sensor configurations. By using one sensor and kinematical constraints from the model of the platform, a slip free dead reckoning system can be realized. The kinematic constraint originates from the sensors inability to calculate rotation. By using two sensors the constraint can be removed and the measurements become independent of the platforms kinematics. Systematic errors originate from measurement errors, alignment errors and change of distance from the ground. Bonarini [6] achieved results comparable to other dead reckoning systems up to a speed of 0.3 m/s by using the UMB benchmark test [7]. Sorensen found that the error of the two mice system was smallest when the sensors were as far as possible from the centre of rotation, and when good care were taken of maintaining constant height. He found that when these constraints were met, the system performed significantly better than other dead reckoning systems [48]. In their work Palacin et al. [40] found that if measurements from an array of sensors were averaged the error became independent from the distance traveled. They also found that the sensor needed a different calibration when moving in an arc, possibly due to the sideways illumination used in computer mice. Another problem was the extreme height dependence of the sensor, which made it impossible for them to use it on carpet. They proposed a modified sensor for they found mice to be unfit for mobile robot navigation. My results were similar to other researchers, they found that one way to make mouse sensors useful for navigation is to equip them with telecentric lens, to avoid magnification changes, to use homogeneous illumination, to avoid directional problems and to use at least two sensors to get rid of kinematic constraints [TK07]. By using different magnification larger portions of the ground is projected on the sensor making higher speeds possible, but this is limited by ground texture size.

Mouse sensors are cheap and readily available, and with certain modifications they can be used for low speed mobile robot dead reckoning. However they are limited by their low resolution and speed, also their algorithm can only be changed by the factory.

Horn et al. [23] aimed at developing a sensor system for automobiles. They used a fusion approach with two cameras and a Kalman filter. One of the cameras is a forward looking stereo camera to estimate yaw rate and forward velocity, the other camera is facing the ground and used to estimate two dimensional velocity. It was found that the camera facing the ground gave better results for lateral and longitudinal velocity than the stereo camera. The fusion approach provided good results even when one of the sensors was failing. The system was tested at slow (< 1 m/s) speeds on a towed cart in a lab. Chhaniyara et al. [11] followed a somewhat similar approach and used a matrix camera facing the ground to estimate speed over ground. They used a mechanism that moved the camera over sand and compared optical flow speed estimates with measurements from an encoder attached to the mechanism. They used Matlab and the Lukas and Kanade algorithm to compute optical flow. They obtained good results at low speeds (0-50 mm/s), however the suitability of the algorithm they used is questionable. Nourani-Vatani et. al. [37] also used a similar method with a ground-facing matrix camera mounted on a passenger car. They made experiments arount 1m/s and exploited the fact that the car had Ackermann steering, since they used a single sensor. The velocity measurements had errors comparable to the wheel speed sensor and the cumulative odometry errors were at around 5% after 100m traveled.

This technology has already found its way to the transportation industry as well. Corrsys - Datron has a one-of-a-kind optical speed sensor [12] used for testing the dynamics of passenger vehicles before mass production. The sensor is claimed to be working on any surface, including water and snow, but it is priced for the big automotive manufacturers. It uses the frequency analysis method. OSMES by Siemens is an optical speed measurement system for automated trains [46]. It uses the principle of laser speckle interferometry mentioned above, and “looks” directly on the rails to measure the trains speed.

It is clear that much work has been done in the field of optical speed measurement and navigation, however several issues remain open for research. Current industrial solutions are somewhat bulky and definitely not priced for the average mobile robot. Solutions by academic researchers have not matured to the level of really useful applications, they are usually very slow. Mouse chips are mostly the sensors of choice. With some modifications their problems of ground distance dependence, lighting and calibration can be helped, but their current speed and resolution is simply not enough for high speed (the order of ten m/s) applications.

Most of these sensors work by using the principle of optical flow. This concept has many uses in the field of robotics, from the flow vector field it is possible to obtain distance information, avoid collision with obstacles, track patterns on the image etc. [14]. It is worthwhile to take a deeper look into this field of image processing.

4.2 Optical flow

Visual movements are caused by the relative displacement of the observer (eye, camera) and the objects of the world. The measurement of these motions can be used in several areas of robotics, like object tracking and segmentation, navigation, and optical speed measurement etc. Most techniques of visual motion measurement are based on the well researched discipline called "optical flow". The basic idea is to compare consecutive images of a scene produced by camera and calculate a vector field for each image which shows the displacements of the pixels to get the next image of the scene. This vector field is often called optical flow or optical flow field (Figure 50.).

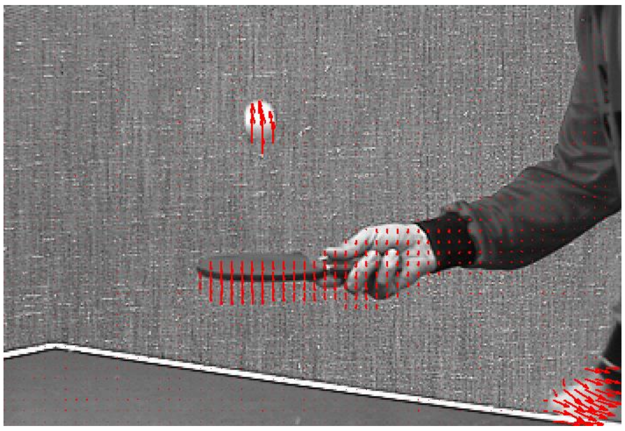


Figure 50. The principle of optical flow, demonstrated on a ping-pong player, local movements

Since the first algorithm presented by Horn and Schunck [22] several techniques have been published to determine optical flow field. The common base of these techniques is the optical flow constraint from Horn & Schunck which presumes that the related points in the consecutive images have the same intensity value. Putting it in another way, a spatial point projected in the image plane has constant (time-invariant or projection invariant) intensity value:

$$E(x(t+\Delta t), y(t+\Delta t), t+\Delta t) = E(x(t), y(t), t) \qquad (62)$$

where $\frac{dE}{dt} = 0$, $E(x, y, t)$ is the intensity of the (x, y) point in time t.

From a Taylor expansion of (62) or from the dependencies between the total and partial derivative using the time invariant property, the general form of constrains is easily derived:

$$\frac{\partial E}{\partial x}\frac{\partial x}{dt} + \frac{\partial E}{\partial y}\frac{\partial y}{dt} + \frac{\partial E}{\partial x} = 0 \qquad (63)$$

where $\frac{\partial x}{dt}$ and $\frac{\partial y}{dt}$ refer to the coordinates of the velocity vector (the two unknowns of the equation), $\frac{\partial E}{dt}$ denotes change of the intensity value in time $\frac{\partial E}{\partial x}$ and $\frac{\partial E}{\partial y}$ denote components of the spatial gradient vector of intensity field.

This constraint is not sufficient to determine both components of the velocity vector, only the component in the direction of local gradient can be estimated. As a consequence, to compute the optical flow field it is necessary to introduce additional constraints.

The method of Horn and Schunck starts from the observation that the points of the image plane do not move independently, if we view opaque objects of finite size undergoing rigid motion or deformation. Therefore the neighboring points of moving objects have quite similar velocities and the vectors of the optical flow field vary smoothly almost everywhere. The following formula represents this smoothness constraint:

$$min\left\{\left(\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial u}{\partial y}\right)^2+\left(\frac{\partial v}{\partial x}\right)^2+\left(\frac{\partial v}{\partial y}\right)^2\right\} \tag{64}$$

where u and v are the coordinates of the velocity vector.
Therefore the goal is to determine a velocity vector field which minimizes the optical flow and the smoothness constrain together:

$$min\left\{\iint_D \left(\frac{\partial E}{\partial x}\frac{\partial x}{dt}+\frac{\partial E}{\partial y}\frac{\partial y}{dt}+\frac{\partial E}{\partial x}\right)+\alpha^2\left(\left(\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial u}{\partial y}\right)^2+\left(\frac{\partial v}{\partial x}\right)^2+\left(\frac{\partial v}{\partial y}\right)^2\right)dxdy\right\} \tag{65}$$

It seems that to compute individual velocity vectors it is necessary to take the whole image into consideration, because every vector depends on every other vector. Therefore this method is classified as a global technique [3].
Another approach presented by Lucas and Kanade assumes the velocities are the same in a small local area (local techniques) [2]. Therefore to calculate the velocity vector of a point it is possible to write more than one optical flow constraint because the points in the small region have the same velocity:

$$\boldsymbol{WA}\mathbf{v}=\boldsymbol{W}\mathbf{b} \tag{66}$$

where

$$\boldsymbol{A}=\begin{bmatrix}\nabla E(x_1,y_1)\\ \nabla E(x_2,y_2)\\ \dots\\ \nabla E(x_{mxm},y_{mxm})\end{bmatrix} \tag{67}$$

$$\mathbf{v}=\begin{bmatrix}u\\ v\end{bmatrix} \tag{68}$$

$$\mathbf{b}=-\begin{bmatrix}E_t(x_1,y_1)\\ E_t(x_2,y_2)\\ \dots\\ E_t(x_{mxm},y_{mxm})\end{bmatrix} \tag{69}$$

$$\boldsymbol{W}=\begin{bmatrix}\omega_1 & 0 & 0 & 0\\ 0 & \omega_2 & 0 & 0\\ \dots & \dots & \dots & \dots\\ 0 & 0 & 0 & \omega_{mxm}\end{bmatrix} \tag{70}$$

In this case the local region has mxm points and $\boldsymbol{W}$ is a weight matrix.
Because the equation system is overconstrained, and in general it has no solution, therefore the velocity estimates are computed by minimizing

$$\sum_{\mathbf{x}\in\Omega(\mathrm{mxm})}\boldsymbol{W}^2(\mathbf{x})[\nabla E(\mathbf{x})\cdot\mathbf{v}+E_t(\mathbf{x})]^2 \tag{71}$$

After using the least mean squares method, the solution is the following:

$$\mathbf{v} = (\boldsymbol{A}^T\boldsymbol{W}^2\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{W}^2\mathbf{b} \tag{72}$$

This method can only measure relatively small displacements, therefore it is often called the iterative Lucas-Kanade algorithm.

The previous two algorithms are directly based on the gradients of scenes, therefore these techniques are often called differential methods. Unfortunately these techniques suffer from a serious disadvantage: accurate numerical differentiation is sometimes impractical because of small temporal support (only a few frames) or poor signal-to-noise ratio [2]. Region-based techniques define velocity as the shift d that yields the best fit between image regions at different times. Finding the best match amounts to maximizing (or minimizing) a similarity measure (over d), such as the sum of square distances (SSD), normalized cross correlation, etc. The optical flow constraint (the related points in consequent images have the same intensity value) can also be found in these techniques indirectly because the best match tries to minimize the difference of the intensity values of the points.

One of the well-known techniques belonging to this group was published by Anandan in 1987 [2] which combines the Laplace-pyramid (to decrease the correlation between the pixels of the images) and the "coarse-to-fine" SSD matching method. Another region-based algorithm presented by Singh, is also built on the SSD metric but uses three consequent images from the scene to calculate the displacement of the regions in the second image. Therefore the inaccuracy caused by noise and periodical texture is decreased [3].

A third class of optical flow techniques is based on the frequency domain of the image sequence. One of the advantages brought by these methods is that motion-sensitive mechanisms operating on spatiotemporally oriented energy in Fourier space can estimate motion in image signals, for which matching approaches would fail. A good example is the motion of random dot patterns, which are difficult to capture with region-based or differential methods, whereas in frequency domain, the resulting oriented energy may be rapidly extracted to determine optical flow field [3].

These methods can be classified in two groups: energy-based approaches are built on the amplitude, phase-based techniques use the phases of the Fourier space to determine the optical flow field. The method developed by Heeger [20], formulated as a least square fit of spatiotemporal energy to a plane in frequency space belongs to the first group. An example for the phase-based methods is the algorithm by Fleet and Jepson [17].

Another approach for the classification of optical flow algorithms is based on image movements. According to this approach two groups can be defined. The first class is called local image movement. In this case several objects of various sizes and velocities are moving in the visual field of the camera, in different directions. Therefore the motion in the image plane can be described with vectors corresponding to individual pixels. With this vector field the motion, shape etc. of the different objects can be estimated. This is the general case, when the camera is facing forward, any obstacle can get into view and even stationary objects seem to move differently due to motion parallax.

In the other case when the camera is facing the ground, we measure movement relative to a single object. The class of global image movement is introduced. In this case the motion of all pixels of the image corresponds to the relative movement of the camera and exactly one object with smooth surface, covering the whole field of view. The constraint about covering the whole field of view causes a very close relationship between the motion vectors: they have the same length and direction, and they can only change smoothly. This is the reason for the name “global”. The condition of smooth surface guarantees that the distance between the camera and every point of the object are quite the same, therefore the effect of motion parallax cannot cause sharp differences between velocity vectors (Figure 51.).

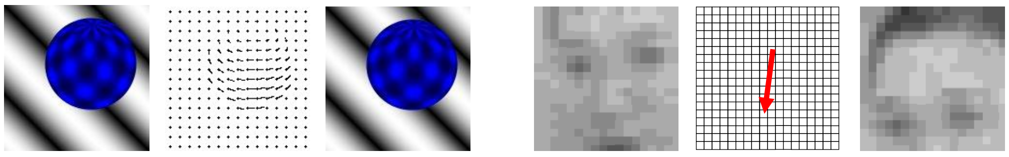


Figure 51. a.) Local image movements[10] **b.) Global image movements [TK07]**

These two strict constraints of global image movement can be approximated by a camera facing the ground and taking pictures of it periodically. If a general mobile platform like a car or mobile robot is assumed, and the camera has a sufficiently high frame rate, it is possible to disregard the orientation change between successive images as the arc travelled can be approximated with a straight line, and therefore all vectors in the optical flow field have the same length and direction. The great advantage of this approach is that there is no need to determine the motion of each pixel, because they are all the same; therefore the calculation of optical flow is simpler, faster and more accurate. From the field calculation techniques presented previously, region-based methods fit this application best. In this case the window of the region contains the whole image and the comparison is between the two consecutive images. Other solutions which calculate the velocity vectors in pixel level and try to determine the camera movement from the heterogeneous motion vector field, naturally cannot make use of this very important piece of apriori information. Therefore the application of these techniques in the class of global image movement is not efficient.

4.3 Optical correlation sensor

In this section I outline the basics of the motion measurement system. First I show the basic problems and some assumptions on which the investigations are based. Then I describe a multisensor setup that is capable of providing two dimensional velocity measurements independent of the platform. Finally I present the simulator which we created together with my colleague Tibor Takács to verify the feasibility of different sensor embodiments, and the validity of my basic assumptions.

4.3.1 Basics

The working principle is quite simple: an optical sensor (photo-detector, camera, etc.) is attached to the mobile platform, facing the ground. From the visual information - captured periodically - it is possible to estimate the real velocity of the vehicle relative to the ground. (Figure 52.)

[10] a.) http://of-eval.sourceforge.net/

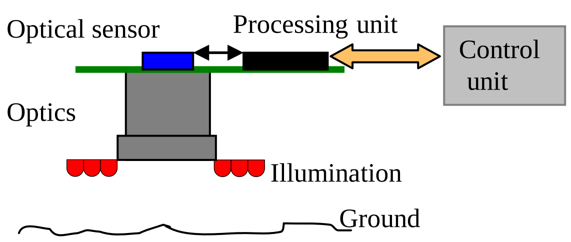


Figure 52. Working principle of the optical speed sensor

Changing ground distance: The distance between the sensor and the ground is continuously changing because of the macroscopic unevenness of the surface and the movement of the suspension of the platform, resulting in a variable field of view. This causes miscalibration, which can be a serious source of errors in speed measurement. Magnification of a conventional lens can be made invariant to defocus by simply adding an aperture at an analytically derived location. The resulting optical configuration is called "telecentric." Most commercially available lenses can be turned into telecentric ones [54]. In this range the image features seen by the camera does not change their size. This approach does not solve the problem of the change in depth of field but blurriness – a known weakness of telecentric optics [50] – only causes loss of accuracy while change of magnification causes miscalibration.
Maximal speed: Two important parameters of the sensor are sampling rate and the size of the image seen by the camera – the field of view. Frame rate and field of view determine the maximal measurable velocity. If the speed of the mobile agent is higher than this limit, there is no similarity between the consequent images as they do not overlap. This is illustrated on Figure 53.

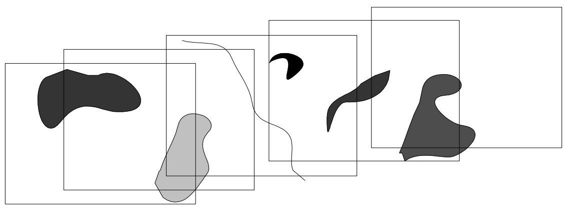


Figure 53. Overlapping snapshots of a ground pattern

This can cause false readings, or no similarity whatsoever, thus estimation of the real velocity is impossible. Fortunately a concrete mobile robot, or car has a well-determined limit for velocity and acceleration, therefore it is possible to calculate these sensor parameters based on apriori information, such as maximal velocity (Figure 54). Outliers – false measurements – can be filtered out by a plausibility check, unrealistic acceleration values and jumps in measured velocity can be discarded.

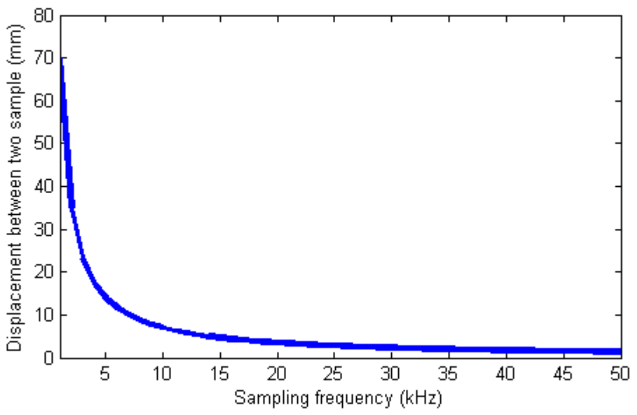


Figure 54. Displacement vs. sampling frequency (at 70m/s)

Platform kinematics: In case of using only one sensor - unless it is placed in the point of interest - the displacement measured needs to be transformed to platform coordinates. Additionally rotation information is lost, it has to be calculated from the kinematics of the platform. In the extreme case, when the origin of the rotation is in the centre of the sensor the angle of rotation cannot be estimated because the sensor does not measure any displacement.

In consequence it is necessary to apply multiple sensors and calculate the displacement from their geometry.

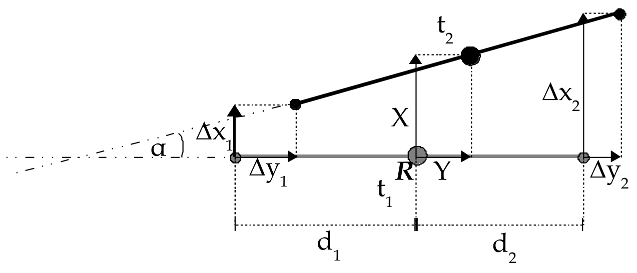


Figure 55. Multiple sensor displacement model

Figure 55. shows a possible case of sensor placement. A sensor is assumed to measure displacements in two dimensions. As mentioned above the orientation of the coordinate system is constant between two sampling instances because we approximate the movement of the sensors with a straight line. This introduces a small quantization error which can be modeled as noise. d_1 and d_1 are the distances of the sensors from the reference point R, Δx_1, Δy_1 and Δx_2, Δy_2 are the displacement values measured by sensors 1 and 2 respectively. From this model the displacement and orientation change X, Y and α of the reference point R between time instants t_1, t_2 can be derived easily:

$$X = \frac{d_1 \Delta x_2 + d_2 \Delta x_1}{d_1 + d_2} \tag{73}$$

$$Y = \frac{d_1 \Delta y_2 + d_2 \Delta \mathrm{y}_1}{d_1 + d_2} \tag{74}$$

$$\alpha = arcsin\left(\frac{\Delta x_2 - \Delta x_1}{d_1 + d_2}\right) \tag{75}$$

Displacement of any other point of the platform can be calculated with a simple geometrical transformation. If the reference point is in the origin of sensor 1 ($d_1 = 0$), then equations (73), (74) and (75) become simpler:

$$X = \Delta x_1 \tag{76}$$

$$Y = \Delta y_1 \tag{77}$$

$$\alpha = arcsin\left(\frac{\Delta x_2 - \Delta x_1}{d_2}\right) \tag{78}$$

This shows that the system is overdetermined, the y component of the second sensor is not needed. The equations show another very important property, in particular that the calculation of the motion information does not depend on the kinematical model of the platform. This is one of the greatest advantages of the method. This property has been noted by others too [40], [6], [48].
Another very important question is the connection between the distance of the sensors and the accuracy of the measurement. From equations (73), (74) and (75) it is clear that with greater sensor distance higher accuracy can be achieved. The sensor distance required for a given angular resolution can be reduced by increasing the sampling rate and/or resolution of the sensors as smaller displacements will be detectable.
In real applications parallel mounting of the sensors is not always guaranteed. This alignment error introduces systematic errors in odometry that can be eliminated by calibration as described in the literature, for example [7].

4.3.2 Advanced sensor system

Matrix cameras are very practical for the purpose of movement measurement as two dimensional displacement and even rotation can be calculated from a sequence of images. However they have certain disadvantages. With commercial matrix cameras, high (several kHz) sampling rates are currently unachievable, and the data rate at high speeds makes processing challenging. Lowering resolution may help, but at the price of losing accuracy. However there is a way to maintain resolution and low data rates, even at high speeds, by using line-scan cameras. These type of image sensors have relatively high resolution in one dimension, several thousand pixels are common. Frame rates at the order of 10 to 100 kHz and relatively low prices are also an attractive attribute.
The first thing that comes to mind as a disadvantage is, that achieving image overlap seems to be impossible, when only a narrow line is captured from the ground. To solve this problem appropriate optics, or wide pixel sensors need to be applied as these can realize an integrating effect. Figure 56 from [33] shows the projection of a cylindrical lens.

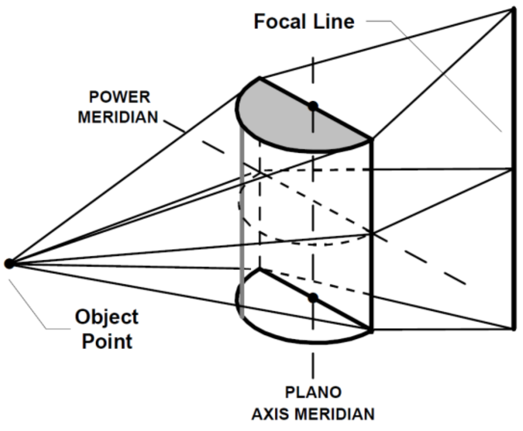


Figure 56. A cylinder lens focuses rays of light from a point object to a line image [33]

This kind of lens image a point source as a line, as opposed to a point, which is the case for common circular lens. They have magnification in only one plane. Light sources perpendicular to the focal line get projected to the same spot, thus achieving an integrating effect. This property is very important for speed measurement. Figure 57. shows a comparison between the imaging of a matrix camera and the line scan camera with a cylindrical lens, or wide pixels. The integrating effect is visualized by the different colors.

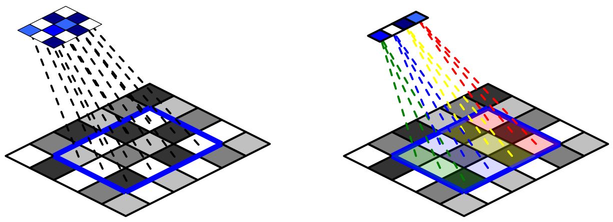


Figure 57. Illustration of the imaging of a matrix camera and a line-scan camera with cylindrical lens

Since a sensor equipped with a line-scan camera can only detect movements parallel to its axis, it only provides one dimensional information. Therefore the main problem to be solved, is to make measurements robust to motions perpendicular to the axis of the detector. These movements cause an error because they might change the pattern on the detector, even without any actual movement in the parallel direction. The problem is illustrated on Figure 58. The second snapshot might differ from the one that would have been obtained by pure parallel motion. This error cannot be totally eliminated but it is possible to decrease this effect with high frame rate and larger field of view of the camera. If the sampling frequency is high (which is no problem with line-scan cameras) then the perpendicular displacement between two consecutive images can be small enough that they will be taken of essentially the same texture element, making correlation in the parallel direction possible.

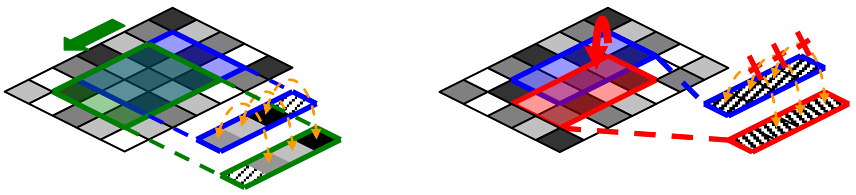


Figure 58. Illustration of the problem of sideways motion

This is of course a texture dependent effect, and has to be investigated with texture analysis. Also this effect can be enhanced by widening the field of view of the detector,

i.e. by integrating on a wider range in the perpendicular direction. By doing this the images can overlap, giving better correlation values.
A negative effect of this method, is that the integration on a wider field of view can cause contrast to reduce to the level of noise, or completely disappear, making estimation of displacement in the parallel direction impossible. For that reason the field of view of the sensor is an important design parameter.
A great advantage of using a one dimensional measurement is that a much simpler algorithm can be used than the optical flow algorithms described in section 4.2. In this case the algorithm reduces to a simple distance metric calculation and minimum search. The neighboring images in a sequence are two vectors of $1\ x\ n$ dimensions where n is the resolution of the line detector. As a distance metric, for example the following well known methods can be used:
Euclidean distance: it measures the straight line or "as the crow flies" distance. Between the points $X(x_1, x_2, \dots x_n)$ and $Y(y_1, y_2, \dots y_n)$ $(n \in \mathbb{Z}^+)$ the distance is calculated as follows:

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{79}$$

Sometimes the Euclidean squared distance is used, which is the same formula without the square root, for faster calculation.
Manhattan distance: it computes the distance that would need to be travelled between two points in a grid-like path. The Manhattan or – city block – distance between two entities is the sum of the differences of their corresponding components:

$$d = \sum_{i=1}^{n}|x_i - y_i| \tag{80}$$

Pearson's correlation: it measures the similarity between two profiles. The correlation coefficient is r:

$$d = 1 - r$$

$$r = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_i - \bar{X}}{s_X}\right)\left(\frac{\mathrm{y}_i - \bar{Y}}{s_Y}\right) \tag{81}$$

where $\frac{x_i - \bar{X}}{s_X}$, $\bar{X}$ and s_X are the standard score, mean, and standard deviation of X respectively. The result for r is less than or equal to 1.
Spearman correlation: this is essentially the same as the previous one, the most important difference is that instead of working on the vector coordinates x_i, y_i it works on the ranks of the coordinates that are obtained by assigning integers to the values giving the highest value to the minimum, essentially their position in a descending order. This method is useful for determining if there is a monotonous transformation that transforms one vector into the other. The use of ranks instead of raw numbers makes it more invariant to the presence of outliers and offsets. The formula for calculation and other characteristics are the same.
Cosine similarity: it measures similarity between two vectors by evaluating the cosine of the angle between them. For 0 it is 1 and for any other angles it is less than 1. The

cosine of the angle determines whether the two vectors are pointing roughly at the same direction. It can be derived from the dot product the following way:

$$X \cdot Y = \|X\|\|Y\| cos\vartheta \tag{82}$$

$$d = 1 - cos\vartheta = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2}} \tag{83}$$

For all these metrics the displacement can be calculated by shifting one of the vectors and calculating the distance metric until the smallest distance, is found. The distance – in pixels – is the value of the shift. No matter which of these metrics are used, they are robust to changes in lighting because adding an offset does not change the location of the minima. Another advantage of these methods is that all are conveniently implemented in Matlab by the pdist( ) function.

4.4 Simulation

To be able to decide the optimal parameters of the sensor and test the candidate algorithms, a simulator was created in Matlab with the help of my colleague, Tibor Takács in the lab.

My basic assumptions to start with were the following: low speed displacement measurement is most accurate if a relatively small area on the ground is observed with a high resolution image sensor to detect small movements accurately. However for high speed measurements the sensor needs to look at a bigger area to ensure that the consecutive images overlap. Alternatively, sampling rates may be raised to ensure overlapping images. Resolution on the other hand, can be lowered to achieve the same relative error rates. This contradiction can be resolved by using a variable image size by changing the magnification rate of the optics, making the image larger at higher speeds. Unfortunately this raises cost, causes calibration and accuracy problems, so we need to assume it to be constant. Therefore it is necessary to find a compromise, to be able to get reasonably accurate measurements in the whole speed range.

The program simulates a configurable virtual camera, moving above one of some pre-recorded surfaces. These surfaces are represented by simple grayscale images taken of real textures (e. g. concrete, soil, stone, PVC etc.) with very high resolution. Figure 59. shows some of the surfaces I used. The reason for using real images was to test the real word applicability of the sensor, to avoid a false judgment by exploiting artificial characteristics of a texture, such as consistent texture and repetitivity. Common texture databases used in the image processing community – such as the Brodatz texture database[11] – are not fit for the simulator, for they have insufficient resolution and are not calibrated for size. Also these databases contain various textures from textile samples, to plants and aerial images, that are not a good test candidate for a sensor that is supposed to face the ground from a small distance. My pictures were taken with an upside down flatbed scanner (HP scanjet 3970) to ensure uniform conditions. By using this method a controllable environment was ensured, light, distance, image size, pixel/mm ratio and viewing angle were equivalent for all pictures taken. I selected these images due to their different properties with respect to texture-size, contrast and brightness. The images were taken at a 2400dpi resolution, and their size is 20cm x 20cm resulting in 18896 x 18896 pixels. This defines the relation between pixel sizes and Si distance dimensions.

[11] for example: www.ux.uis.no/~tranden/brodatz.html (Acc. 2012 Aug.)

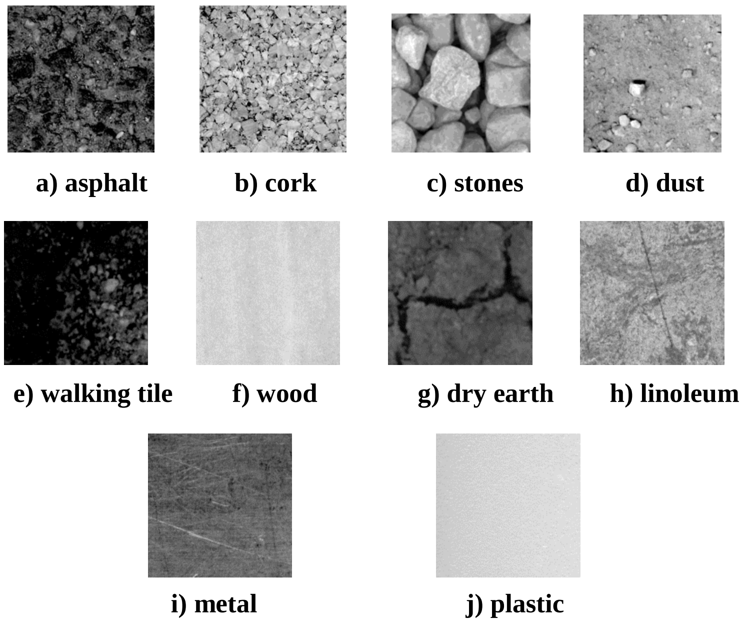


Figure 59. Sample images from the textures used in the simulator (2x2cm)

The virtual camera and sensor implemented in the simulator have several adjustable parameters:

- movement speed
- angle of movement
- frame rate
- field of view in two dimensions
- signal to noise ratio
- resolution
- distance metric.

Using the virtual surfaces and line-scan cameras it is possible to simulate a lot of different movement scenarios.

The simulator works the following way: The ground is represented by one of the high resolution images. An image detector is chosen by defining its $n\,x\,m$ resolution and a pixel size. Naturally n has to be set to 1 for a line detector. The field of view is determined as a $k\,x\,l$ mm rectangle. The image on the detector is created by resampling a $k\,x\,l$ mm portion of the high resolution image onto the $n\,x\,m$ detector image, optionally with additional white noise, with an expected value of 0 and a standard deviation of choice. The typical value of $n\,x\,m$ is around $1\,x\,1024$, the value of $k\,x\,l$ is dependent on the optics, and a typical value is for example $50\,x\,50$mm. These values have to be chosen carefully. Moving the camera at a very high speed limits the number of snapshots taken, providing individual measurements rather than an average for a ground type. The simulation is based on the assumption that the ground is continuous i.e. its resolution is several magnitudes higher than the camera. This poses a lower limit on field of view and an upper limit on camera resolution. Also it is not practical to define a field of view of 15cm as it will hardly fit in the 20x20cm virtual ground, also optics of that size would make the sensor extremely expensive.

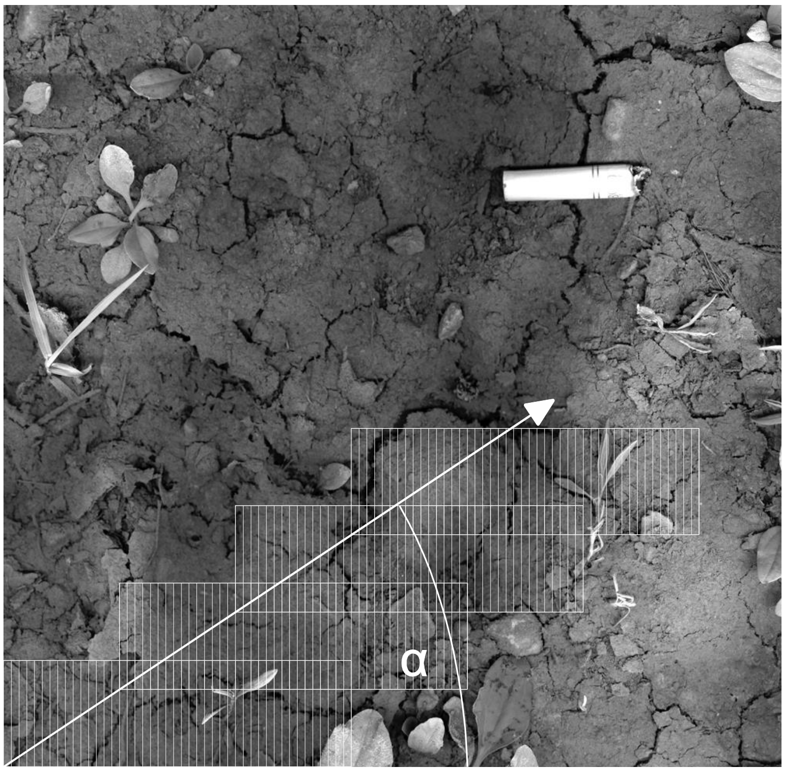


Figure 60. Illustration of the concept of the simulator (dry earth)

This concludes the setup. When running the simulation the consecutive image is chosen automatically by translating the $k\ x\ l$ mm window on the ground image, with a certain amount of pixels, according to the pre-defined movement speed, frame rate and direction. The direction of movement can be chosen by specifying the angle α. It is important to verify if the sensor gives reliable measurements at high sideways movements. This concept is illustrated on Figure 60.

The two neighboring images are then shifted and compared according to a distance measure of choice such as correlation, Manhattan or cosine distances as described in section 4.3.2. The shift, corresponding to the smallest distance measure is the estimated displacement. The exact traveled distance in pixels, is known from the simulated speed, so the error of the measurement can be obtained easily, by subtracting the estimate from the set value and normalizing. The algorithm is illustrated on Figure 61.

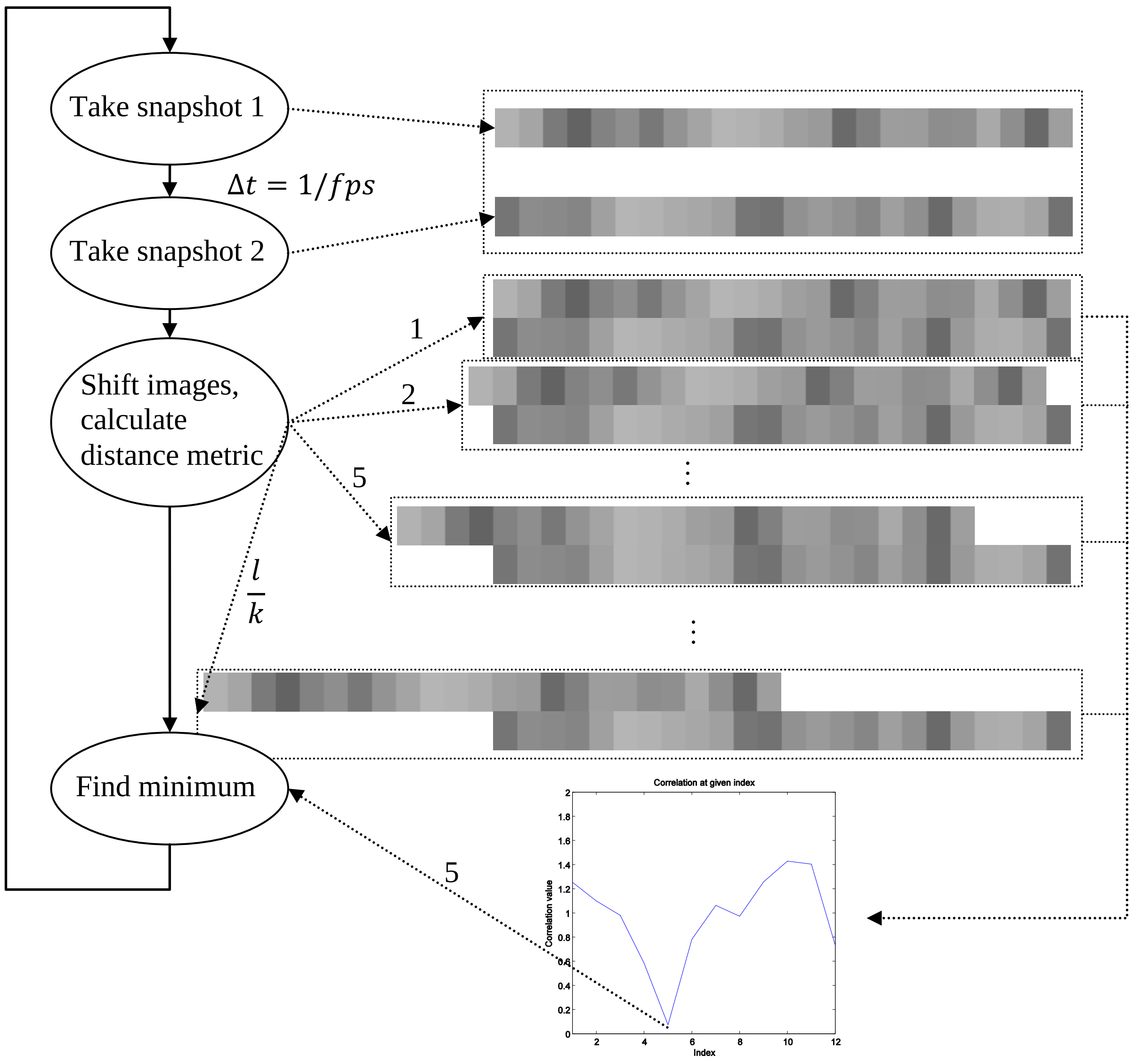


Figure 61. Illustration of the algorithm

The shift values are in pixel index, $\frac{l}{k}$ is length in pixels, divided by a limit value. A practical value for k is for example 2, to limit the shift to half the image length, as there is a high chance for false readings, when only a small section of the images are compared. The purpose of the simulator was to determine the feasibility, and the best parameter values for line-scan cameras in optical velocity measurement.

4.5 Simulation results

The most important parameter of the sensor is the field of view and the shape factor of the optics. As I modeled the imaging system with rectangular frames, a practical shape factor choice is width/length of the field of view in %. A sensor with a small field of view is more compact and therefore cheaper. If it is possible to avoid the use of cylindrical lens the optics can be simpler and easier to develop. Therefore an important purpose of the tests is to find a connection between accuracy and field of view.

Error measurement: In this simulation it is not trivial how the error should be interpreted and visualized. In the background the Matlab code measures all movements in image pixels, as the smallest quantum. The difference between two images is calculated from the simulated speed, the frame rate and the scaling factor $r_i \left[\frac{ipx}{mm}\right]$ that determines the number of pixels in a mm. The error can be measured in image pixels, or mm as the difference of estimated and simulated movement, however this does not relate well to the camera parameters and the speed of the movement, making comparison of experiments difficult. It is possible to relate the distance error to the velocity – dividing by the frame rate – and give it as a percent of simulated velocity, but still this would make different setups complicated to compare. The error measure which worked for me best, is the displacement error measured in camera pixels because this shows clearly whether correlation was successful for a certain image pair or not. The error is calculated the following way:

$$E_{cpx} = d_{m_{cpx}} - Round\left(d_{r_{ipx}} \cdot \frac{r_i}{r_c}\right) \tag{84}$$

This reads in English: the error in camera pixels equals the difference of the distance measured in camera pixels, and the real distance traveled in ground pixels – scaled down to camera pixels. The subscript cpx refers to camera pixel and ipx to image pixel, r_i and r_c stand for the scaling factors for mm per image pixel and mm per camera pixel respectively. It has to be noted though that this way the error of a high resolution camera can be found to be larger than a low one, so this error is only useful when absolute speed measurement errors are not the main question. Speed percentage errors for example can be calculated this way:

$$E_{s\%} = \frac{v_m}{v_{sim}} \cdot 100 = \frac{E_{cpx} \cdot r_c}{f_r v_{sim}} \cdot 100 \tag{85}$$

where f_r stands for frame rate.

Image overlap: the effect of overlap between consecutive frames needs to be investigated. For this I set the frame rate to a constant 2500 fps, I changed the field of view length from 10 mm to 100 mm while setting the field of view width to 40% of length. I changed the simulated velocity from 5 m/s to 200 m/s to be able to see faulty measurements too. Figure 62. shows the experiments for the “stone” and “cork” textures, and also the percentage of overlap as a function of field of view length and simulated velocity. It is visible from the figures that the error mainly depends on the overlap, according to the last figure it is around 50%. What this really means is that the algorithm gets correct results until the limit set by maximal image shift i.e. 50% (see Figure 61., $k = 2$).

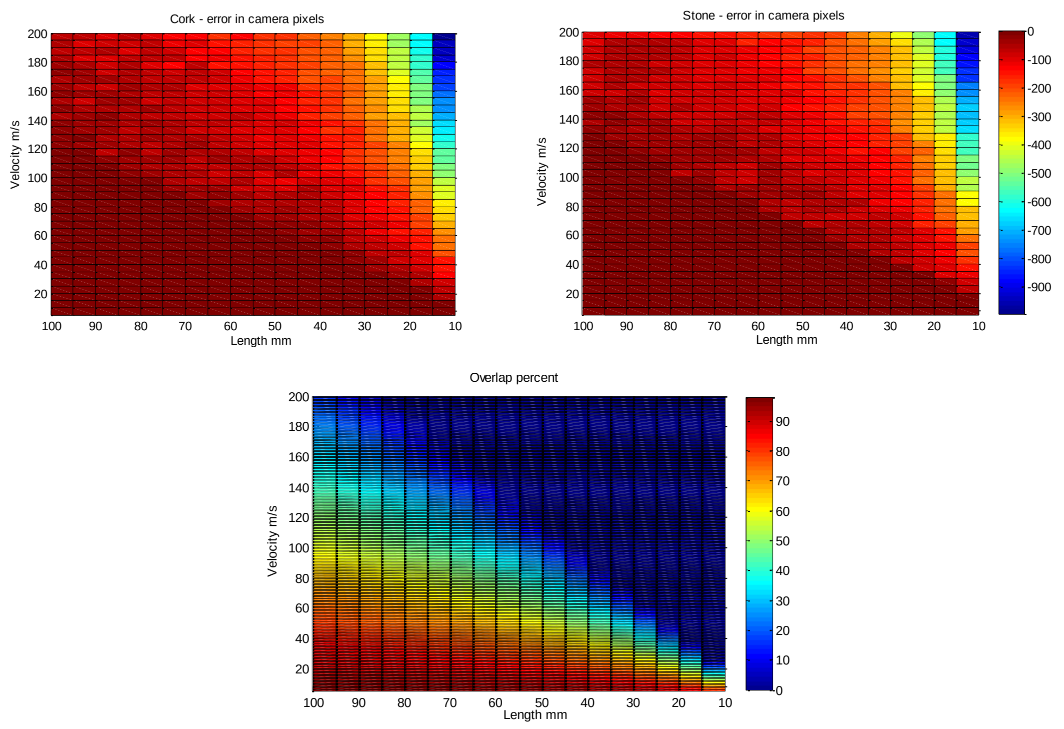


Figure 62. The effect of image overlap on the error – moving straight

The effect of shape factor of field of view : Since there are a lot of parameters to tune, if we want to investigate the relations of a certain two, we have to fix the rest to sensible values. To see the connection between field of view shape factor and accuracy I set the field of view length to be between 10 and 70 mm in increments of ten and the width from 10 to 100% of the length in increments of ten. To obtain reasonable measurements the simulated speed and frame rate have to be set together to allow high values of overlap between images. When the sensor moves at a small angle, the measurements are free from texture induced noise, therefore to find out about the boundaries of its capabilities, I fixed the movement angle to 90° [12]. I fixed the sensor speed to 12 m/s and the frame rate to 2500fps. Many other speed and fps values can be used, the important thing is to be able to measure the overlap, between frames. This way the sensor travels approximately 4.8mm between frames, perpendicular to the sensors axis, and hardly moves in the parallel direction.

[12] in fact it was set to 89° to avoid some numerical errors, this has no significant effect on the outcome

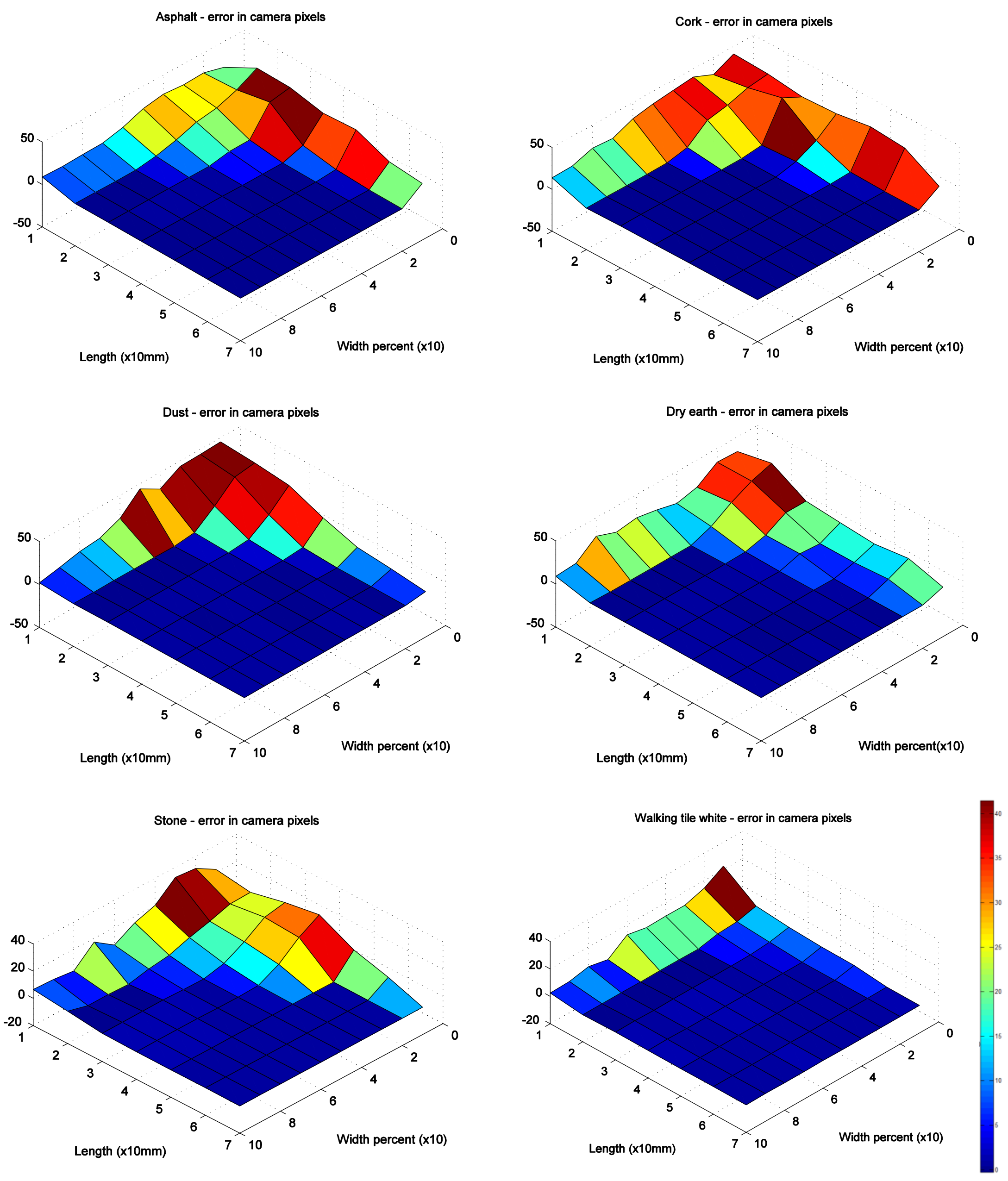
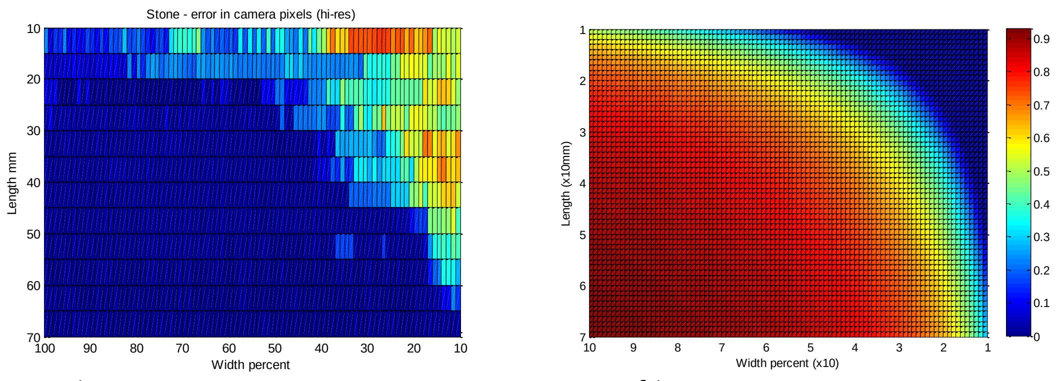


Figure 63. Error surfaces as a function of field of view ratio

Figure 63. shows the error surface as a function of the two dimensions of the field of view. The main axis of the line detector is called length, width of the field of view is scaled in percentage of the length, 100% meaning a square field of vision. The values along the z axis are the averaged errors of twenty consecutive measurements, with the same settings, in camera pixels. It is clear from the images that apart from texture dependence, the most important factor is overlap between snapshots. This is verified by Figure 64. where the overlap percentage is displayed for the same 12 m/s at 2500fps settings.



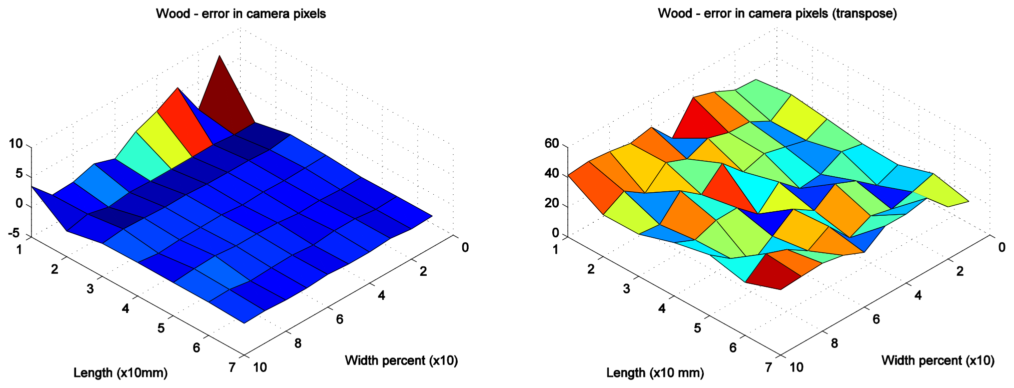
a) b)

Figure 64. (a) Percent of overlap vs. shape factor at 12 m/s, 2500fps, stone. (Dark blue means no error.) (b) Overlap percentage at the same movement parameters . Similarity of the two patterns is obvious.

By comparing images of Figure 63. and Figure 64., the conclusion can be drawn that approximately 60% overlap is needed for pixel correct displacement calculation, if we want good results on any texture. Longer field of view yields better results, however it also means larger absolute errors, because a single pixel displacement corresponds to a larger distance on the ground. We can also see, that by ensuring sufficient overlap the effect of off-axis movement is negligible.

4.5.1 Effects of texture

Optical motion measurement in general relies heavily on texture of the moving objects. This is of course also true for the sensor system presented herein. If there is no texture, no motion can be detected. This problem is addressed in optical mice by using sideways illumination, thus creating shadows of surface roughness, and calculating motion from the movement of shadows. Another method uses laser speckle patterns (Figure 49).



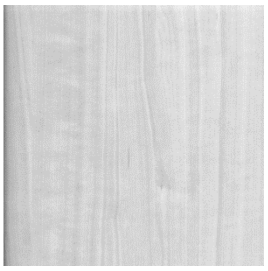


Figure 65. Moving sideways on wood – in perpendicular directions

In this section I summarize my texture related findings, using the simulation environment – this means no special lighting effects.

Texture reliance also means that the distance measure can give false readings on “difficult” texture, A good example of this is the well known aperture problem of optical flow. A perfect repetitive texture, or a direction dependent texture can easily cause false correlation. A good example of this is the “wood” texture that I captured on a tabletop in the lab. The texture resembles a barcode, in a sense that it has repetitive texture in one direction and no, or very small changes in the other. Figure 65. was created using the same settings as Figure 63. The first image shows sideways movement, parallel to the grain, zero movement is detected perfectly. The second image shows movement perpendicular to this direction, that was executed on the transpose of this image, this is movement at a right angle to the grain, where it is not possible to detect movement with this method.
The textures “plastic” and “metal” are not detailed enough, measurements are only useful, when no noise is added to the simulation, otherwise the texture gets lost in the noise. They are shown on Figure 66.

Figure 66. Difficult textures - metal and plastic

In the small extreme resolution image database I created, there are three textures that can be considered regular, that is they have repetitive texture elements of a similar size. This is useful for making some texture related conclusions. The other ground images have features that stand out of the image thus providing good “landmarks” for correlation. The three textured images are “stone”, “asphalt” and “cork” (Figure 67.).

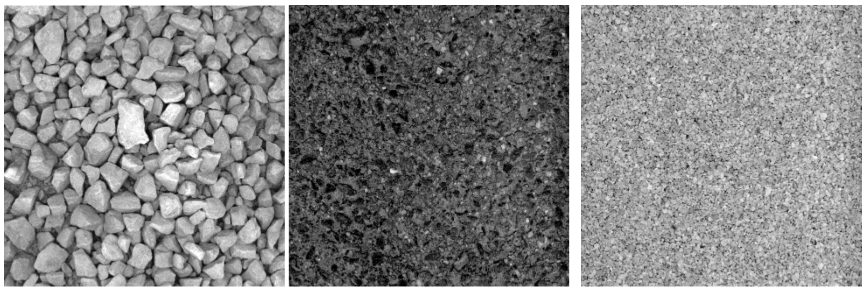


Figure 67. The three ground patterns with the most regular texture – stone, asphalt and cork (20 x 20 cm)

Figure 68. shows correlation values for these three textures with the exact same settings. The graphs display correlation value versus the index, at which the two images are shifted and matched. The simulation parameters are the following:

- Velocity: 20 m/s
- Angle of movement: 45°
- Field of view length: 50mm
- Field of view shape factor: 0.4
- Resolution: 1024
- Frame rate: 2500fps
- Noise: 5 grayscale value

From the graphs it is clear that the repetitivity of the texture is reflected in the local minima of the correlation functions, stone with the bigger texture size gives two distinct minima, while asphalt and cork gives more minima as their texture elements are smaller. It is also worth noting, that the minimum of stone is much less pronounced, this means that with smaller resolution, higher errors can be expected due to the combined effect of noise and flat minimum location.

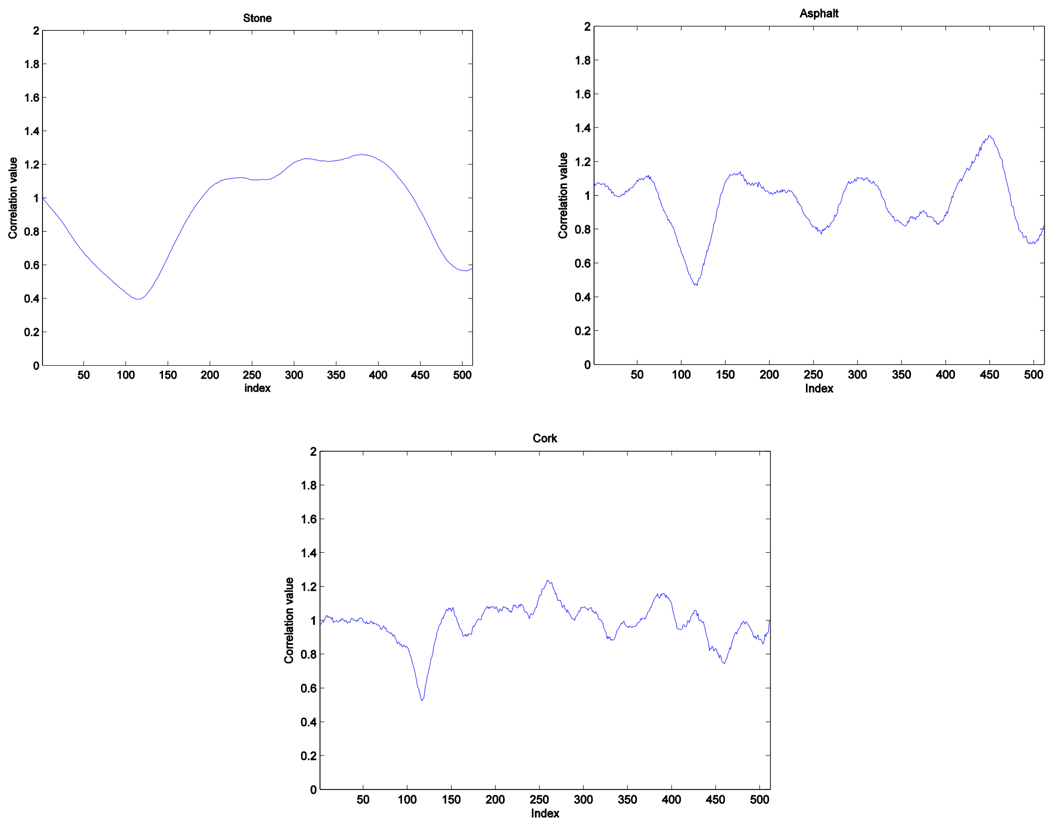


Figure 68. The effect of texture size on correlation functions – smaller texture gives more frequent and more pronounced local minima

By widening the field of view higher overlap values can be ensured, however widening the field of view has a negative effect on contrast. This can be seen on Figure 69.

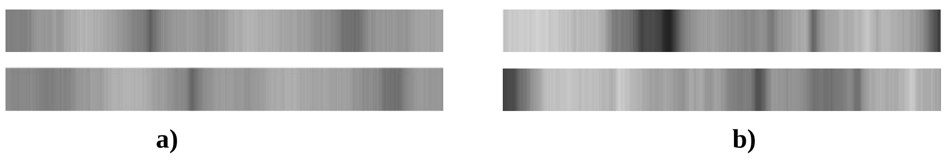


Figure 69. The effect of field of shape factor

On sub-figure a) a wider field of view was used than on b). Both image pairs are one sampling period apart, taken on the same surface (Stone) at the same speed, and frame rate. It is clearly visible that a) has less contrast, due to the integration effect, but the samples correlate well, b) on the other hand has more contrast but the correlation is less evident. It is important to note here that increasing image width much further leads to total loss of contrast making measurements impossible. However on the surfaces that give good results that limit is higher than 100% width/length, which is impractical anyway. Below that the simulations show that the errors are not dependent on image width, provided that sufficient overlap is ensured.

4.6 Evaluation, enhancements, practicalities

Sensor placement: according to the calculations in section 4.3.2 at least three one dimensional sensors placed at separate locations are necessary to obtain movement information independent from platform kinematics. This is no surprise as two dimensional ground movement has three degrees of freedom. In some cases it is possible to use only two sensors and calculate the missing degree of freedom from platform kinematics. This is relevant for non-holonomic platforms such as differential or Ackermann drive. For these platforms the best placement is right next to the driven, or the steered wheels. The proposed sensor placement is illustrated on Figure 70. For holonomic platforms three sensors are necessary and they have to be placed, so that there is no singular configuration.

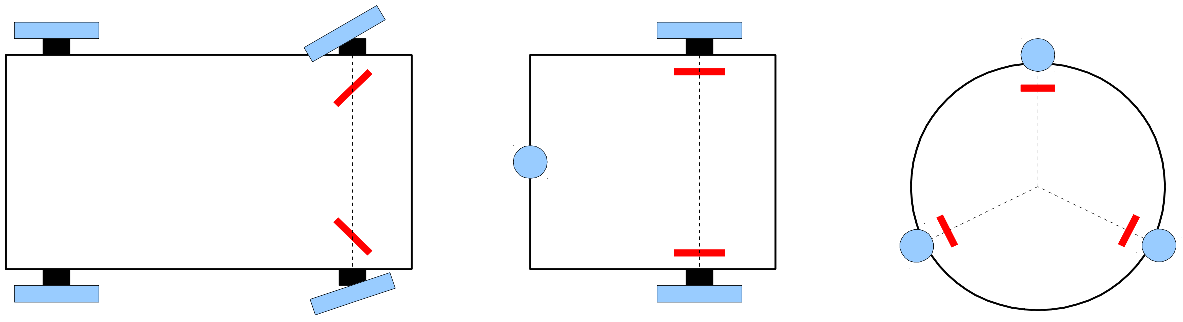


Figure 70. Suggested sensor placements for different drive configurations – red line represents sensor

When the third degree of freedom is calculated from kinematics a great advantage of the sensor is given up, when the platform skids – moves perpendicular to wheel rotation the movement is measured inaccurately or not at all.

For Ackermann drive vehicles the use of the sensor is not limited to velocity measurement, a useful application could be the measurement of sideslip angle [BKS06]. In this case a single sensor can be mounted at a wheel of interest, parallel to the wheel's axis of rotation, the sideslip angle can be determined as a function of longitudinal and sideways velocity of the wheel: $\alpha = -arctan\left(\frac{v_{\parallel}}{|v_{\perp}|}\right)$ where $v_{\parallel}$ is the forward, $v_{\perp}$ is the sideways velocity of the wheel.

Environmental effects: as all optical sensors, this sensor is sensitive to dirt. A good way to get rid of it is to use a special coating for the optics, or use some mechanical way of removal, such as high frequency vibration, or blowing air away from the sensor as it is common with industrial sensors. None of these solutions are cheap and/or perfect, this remains a disadvantage, and has to be considered for the application. In a controlled environment such as a warehouse, or a racetrack, an occasional cleaning should be sufficient.

Water poses another problem, when it does not splash on the optics, it still acts as a mirror and introduces errors with its movements. Some of these problems may be helped by special lighting, however this is also a disadvantage to consider.

Lighting: since the sensor is an optical one, lighting plays an important role. If the sensor is mounted under a vehicle chassis it can be fairly independent from external light. Since the practical image sensor to use is a tall pixel line scan-camera, the amount of light needed does not need to be much compared to a matrix camera as a tall pixel's relatively large area ensures high sensitivity, the integration of the cylindrical lens has a similar effect. The issue of lighting however is something that the simulator is not designed to test.

From the simulations I concluded that for pixel accurate operation in the perpendicular direction, an overlap of 60% is needed. However this value can be lowered by the help of filters and intelligent processing.

Enhancements: I noticed that when approaching the limits of reliable operation (less and less overlap) point errors start to appear in the course of consecutive measurements, i.e. a large error appears for a single image pair. As we increase speed, these errors become more frequent and start to grow in area – several consecutive measurements are bad. This can be helped by the use of median filtering, by taking n consecutive measurements and taking the average of their median filtered values the correct value can be obtained. That is of course when there are not too many errors, in which case this method cannot help. By substituting each n measurement with the average of their median filtered values the data rate of the sensor is divided by n. Since the frame rate can be quite high – the order of 10^4 Hz – using for example $n = 10$ would yield measurement data rates at the order of kHz, which is acceptable. Figure 71. demonstrates the effect of median filtering on the walking tile pattern, taken from the sidewalk in front of building I of the University. The figure on the left shows averages of twenty consecutive measurements and the right shows the same with median filtering.

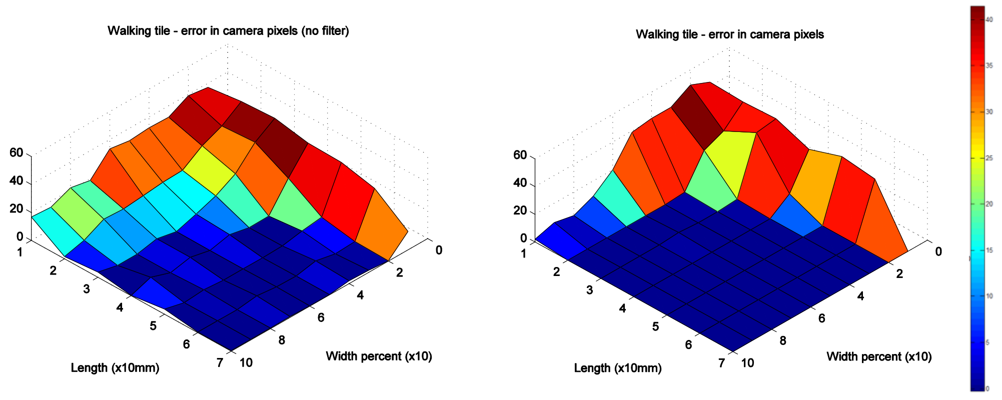


Figure 71. Raw and filtered data for the “walking tile” ground, median filtering effectively removes point errors

This method emphasizes the sensors sudden degradation behavior, it can be noted that especially after applying the filter, the sensor is either pixel correct or several pixels off.

Other than filtering methods, **apriori information** can be used. For instance we know that the value of acceleration is limited, so the velocity “does not change much” between two measurements. This information can be used to create a weighing function for the distance measure, that gives higher values for distances that are “far” from the previous measurement value, thus favoring small changes in velocity between samples. This method can both speed up the measurements and decrease errors.

Real time algorithm implementation: for the sensor to work as intended real time processing is needed, taking for example a 2500 fps image sensor this means that there is 400 μs between images. The algorithm used in the simulator (section 4.4) lends itself naturally for FPGA based processing, because the distance metric values between consecutive images can be computed for each shifted image, in a parallel manner. Lindoso [32] presents an application where similar calculations for fingerprint matching on 50x50 images are executed in a comparable amount of time on a Xilinx Virtex 4-SX. Bilal et al [4] proposes enhancements of an FPGA implemented correlation algorithm to speed up processing and use hardware resources more efficiently. I think it is safe to say that an algorithm, similar in function to the one I used in the simulation, can be implemented in real time, with contemporary FPGA hardware, without making a compromise in speed.

Optics: a telecentric lens has proven to be a good solution for ground distance related problems [TK07]. If the use of cylindrical lens can be avoided and a suitable tall pixel image sensor is used, off the shelf lens can be applied. The largest diameter of a telecentric lens has to be the same size as the largest dimension of the rectangular field of view on the ground, therefore price depends heavily on the field of view.

4.6.1 Applications, future prospects

The concept of the sensor won a Hungarian state grant – Irinyi János – in 2005. This made it possible to create the first proof of concept prototype, made with a mouse chip. This sensor proved the usability of telecentric optics and the experience gained catalyzed the creation of a compact odometry sensor subsystem for mobile robots that was used in MSc theses and robotic competitions in the 3DMR lab. Figure 72. shows the first prototype (left) the wooden housing holds the telecentric lens and the electronics. It is mounted on an oscilloscope trolley, to make it mobile.

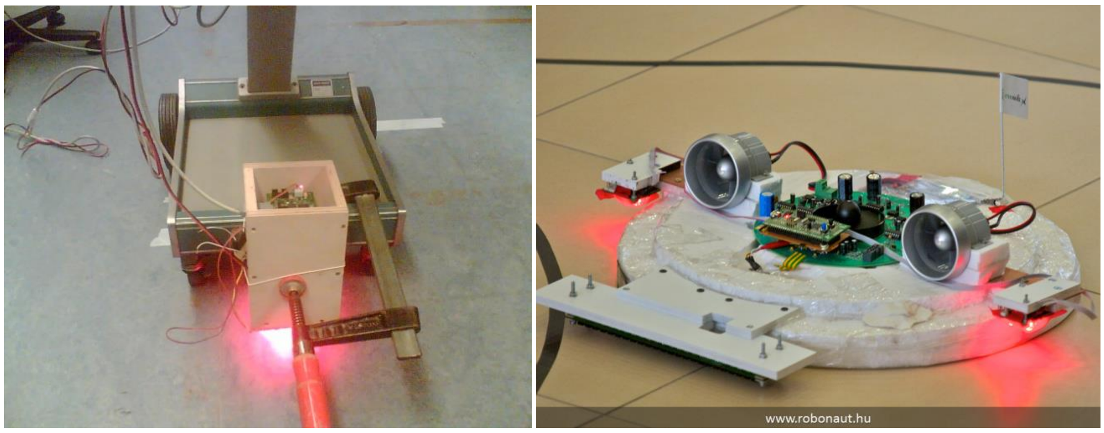


Figure 72. First prototype and application on a hovercraft

The hovercraft (Figure 72. right) was made by an MSc student (Gábor Baracsi) under my guidance as a master's thesis project. For this project he used the sensor as a building block, simply integrating it with his own work, leveraging my and another MSc student's previous work. I started experiments with the line scan camera sensor, connecting an imaging sensor with a PC. I also created plans according to the simulation results described in section 4.5. Unfortunately I did not win the second part of the Irinyi János grant and the realization of a line-scan prototype was put on hold due to insufficient funds.

At the writing of this thesis several wide pixel sensors were available at reasonable prices for example LIS-770i from Panavision Imaging (available from 2011) and S3901 from Hamamatsu[13] (available from 2010) both feature wide pixels and high frequency readout. Table 3. summarizes their most important features:

[13] http://www.panavisionimaging.com, http://jp.hamamatsu.com/products

Sensor	**LIS-770i**	**S3901**
Image size (max)	312.5 x 6006μm (selectable in 5 steps)	2.5 x 6.4 mm
shape factor	5%	39%
resolution	335 or 770 px	128
frame rate	2985 or 1300 fps	15600 fps
pixel clock	1 MHz	2 MHz

Table 3. Comparison of the main features of two linear image sensors as an example

Both these sensors are considered wide pixel, the S3901 is extremely wide. I cited these image sensors to demonstrate the capabilities of my speed sensor with off the shelf components i.e. no special integrating optics with cylindrical lens, the only integrating effect is achieved by the image sensor itself. Table 4. summarizes the achievable speed sensor parameters with the two detectors. Field of view length is assumed to be 50mm. For the LIS-770i resolution is assumed to be the smaller value, 335px to have a better line rate value. It has to be noted that the S3901 is a lot more expensive than the LIS-770i.

Parameters	**LIS-770i**	**S3901**
field of view length	50 mm	50 mm
field of view width	2.5 mm	19.5 mm
line rate	2985 fps	15600 fps
resolution	335 px	128 px
area projected on a pixel	0.15 x 2.5 mm	0.39 x 19.5 mm
max speed - assuming 60% overlap, sideways movement	3 m/s	120 m/s

Table 4. Projected speed sensor parameters with the example image sensors

From the data it is clear that a sensor with excellent qualities can be assembled from off the shelf components.

4.7 Conclusion

Thesis III: Accurate, high velocity contactless displacement measurement can be realized using a sensor system based on a line scan camera; by choosing appropriate system parameters, accurate one dimensional measurements can be made during arbitrary two dimensional motions.

The experiments conducted with the simulator show that using a line-scan camera for optical speed measurements is a viable idea. Practical parameter choices have lead to exact displacement calculations for most of the investigated textures, in the presence of simulated noise. I gave examples of commercially available off the shelf components, that enable high speed measurements, exceeding the range required for most land vehicles.

Related publications:

[BKS06], [KT07], [TK07], [KT08], [TK08], [TKV08]

5 Conclusion

My research work summarized in this dissertation addresses a special segment of mobile robotics and industrial logistics. Due to their exceptional maneuvering capabilities, – independent movement in three degrees of freedom – omnidirectional vehicles are used for a great variety of tasks. They are very popular among hobbyists, specialized competitions are organized for them each year, they even have their own soccer league. Also there are several industrial applications that leverage the advantages stemming from the fact that they do not need to run time consuming y or k turns in the corridors of warehouses; they can save time and headaches by efficiently maneuvering into the most awkward spaces, changing their orientation on the go. The scientific community has also been interested in these platforms for a long time, due their complex mechanical design and unique control needs.

In this section I give a short summary of my contributions to this field.

During my work with omnidirectional platforms I found that even though these machines have been invented a long time ago, due to their somewhat limited niche applications, their design and control methodology is not as well worked out, as for conventional vehicles. However there is a great wealth of models, tools and methodologies that can be borrowed from conventional vehicle technology. Modern engineering increasingly uses simulation, to gain insight into the inner workings of systems and save precious time and resources, by not having to build absolutely untested prototypes of a new product. Arguably the most important component of a vehicle simulation is the wheel model, since that is the part that defines the contact with the environment and generates the forces and torques that make the vehicle move. Tire simulation for regular wheels is a mature, well researched domain with useful results. An omnidirectional wheel is by no means the same as a regular tire, however many aspects are very similar, such as the calculation of the contact point, and many aspects of the force generation characteristics. Tire materials, such as rubber are similar and exhibit the same friction characteristics. Considering all this, it is a straightforward idea to adapt already existing tire models from the automotive vehicle simulation domain and create an universally applicable omnidirectional wheel model, to facilitate research and simulation of omnidirectional platforms.

I created a method for modifying regular tire model force generation characteristics to behave like an omnidirectional wheel with a given δ roller angle and material characteristics. The modification transforms wheel dynamical variables such as slip and tire deformation in the active – parallel to roller axis – direction of the wheel. It calculates with zero force in the perpendicular direction. Alternatively forces due to bearing friction and inertia can be added. The procedure is demonstrated in chapter 2. by modification of the widely used Rill model, which was chosen for its easy parameter modification characteristics. For the verification of the model, I chose to work with a simulator based on the Modelica language. Modelica is an open source, object oriented, declarative programming language, created especially for the modeling of complex physical systems. Together with the Dymola environment of DAssault Systemes the simulation system includes a powerful GUI and animation capabilities for easy, qualitative model verification.

Modern physical simulation software packages usually include libraries of components related to a certain engineering discipline; this is also true for Dymola. The Academic Bundle of version 7.4 includes a vehicle simulation library, with several different tire models, for example: linear, Rill, Bakker, Pacejka. For empirical validation of my tire model I created two different platform models, a simple industrial forklift with $\delta = 45°$ roller angles and four wheels – commonly known as a Mecanum platform – and a three wheeled robot with $\delta = 0°$, a popular choice for robot soccer competitions. I derived the inverse kinematic equations governing their movement, using related literature, and created a drive unit in the simulator that moved these platforms according to the equations. The results were satisfactory, by supplying platform linear and angular velocity vectors, as input to the simulation, I was able to move both platforms according to expectations, in a qualitatively correct manner. As a simplification I neglected the effect of contact point variation caused by the change of rollers touching the ground, and I also neglected the effect of bearing friction, rolling resistance and roller inertia in the passive direction of the wheel.

Building on my results with omnidirectional wheel simulation, I started to work on another problem, related to the behavior of omnidirectional platforms during heavy braking. These wheels allow true omnidirectional movement, by only generating substantial force in the active direction of the wheel, as opposed to regular wheels that are usually quite hard to move in the lateral direction. This property results in unwanted swerving behavior when braking. Wheel forces depend strongly on wheel vertical load and local ground characteristics, causing substantial differences in the friction force generated at the wheel contacts. The lack of sideways force lets the vehicle lose its original orientation and turn around the strongest – most loaded wheel – possibly causing accidents and at the least, serious operator concern. The problem of keeping vehicle orientation while braking is a difficult one, due to 1ack of exact knowledge of load distribution, and wheel parameters, consequently the magnitude of friction force. In chapter 3. I present the dynamic equations governing platform movement. For the control of parameter, or model uncertain nonlinear systems the literature suggests the use of sliding mode control. I created a general switching surface for omnidirectional wheels, that is based on the simple approach of turning the brakes off for a given wheel when brake force of that wheel increases unwanted velocity components. Otherwise the brake is applied with the maximum force set by the driver. Stability of the feedback loop is guaranteed by the passivity of the system. Correct operation under various wheel and platform parameters, is verified by simulation in the Modelica – Dymola environment. The most important property of the control law is that it is highly independent from platform dynamic characteristics, as it only builds on the kinematics of a given platform.

The proposed control algorithm relies on accurate measurement of the linear and angular velocity of the vehicle. Omnidirectional wheels generally operate with higher amounts of slip, heavy braking in general also causes wheel slippage, therefore it is important to have a feedback method independent from wheel rotation. For this task I propose an optical feedback method, similar in principle to the well known optical mouse, that uses correlation to compare ground texture intensities to measure displacement. My method however uses a line scan camera to achieve greater speed and resolution, using less computational resource than a matrix camera of similar parameters would require. A line scan camera however can only measure displacement in one dimension, therefore movements at an angle to its main axis may cause serious measurement errors. To solve this problem I proposed the use of cylindrical lens, or tall pixel cameras to achieve an optical integrating effect. I created a simulation in Matlab and showed that error free operation can be achieved on a wide range of textures by insuring sufficient overlap between sampled images. My results show that accurate one dimensional measurements can be made with a line sensor, while moving freely along the surface. The proposed sensor system is capable of measuring two dimensional movements of land vehicles up to the range of 100m/s with high accuracy.

6 Publications

[Kal13] Viktor Kalman. Controlled braking for omnidirectional wheels. *International Journal of Control Science and Engineering*, 3(2):1–10, 2013.

[KV12a] Viktor Kalman and Laszlo Vajta. Designing and tuning a brake assistant for omnidirectional wheels. *PERIODICA POLYTECHNICA-ELECTRICAL ENGINEERING*, 4:1–11, 2012.

[KV12b] Viktor Kálmán and László Vajta. Slip based center of gravity estimation for transport robots. In *Factory Automation*, pages 50–55, Veszprém, Hungary, May 21-22. 2012. University of Pannonia.

[ATV$^+$12] Majdik András, Takács Tibor, Kálmán Viktor, Tamás Levente, and Vajta László. Vizuális hasonlóságon alapuló pozíció felismerés városi környezetekben. In *Számítástechnika és Oktatás Konferencia*, pages 303.–307., Alba Iulia, Romania, 2012.10.11-2012.10.14. 2012. (in Hungarian)

[VAL12] Kálmán Viktor, Dr. Majdik András, and Dr. Vajta László. Omnidirekcionális járművek szimulációja és irányítási kérdései. In *Számítástechnika és Oktatás Konferencia*, pages 269–275, Alba Iulia, Romania, 2012.10.11-2012.10.14. 2012. (in Hungarian)

[KJVS12] Viktor Kálmán, Tamás Juhász, László Vajta, and Ulrich Schmucker. Mecanum wheel library in modelica. In *15. IFF-Wissenschaftstage: Digitales Engineering zum planen, testen und betreiben technischer systeme.*, page 6, Magdeburg, Germany, June 26-28. 2012. IFF Fraunhofer Institute.

[KTT10] Viktor Kálmán, Tibor Takács, and Attila Tekes. Intelligens fogyasztásmérés a víziközmű-társaságoknál. *VÍZMŰ PANORÁMA*, 5:26–29., 2010. (in Hungarian)

[Kál08] Viktor Kálmán. Internal modeling for mobile robots. In *Robotica 2008 - 4th International Conference on Robotics.*, pages 455–463., Brasov, Romania, November 13-14. 2008.

[DVK$^+$08] András Dán, László Vajta, Viktor Kálmán, Tibor Takács, and Zoltán Molnár. Intelligens kommunális villamos fogyasztásmérés lehetőségének vizsgálata kutatás fejlesztés keretében az e.on tiszántúli Áramszolgáltató zrt-nél. In *MEE 55. Vándorgyűlés.*, Eger, Hungary, September 10-12. 2008. (in Hungarian)

[KT08] Viktor Kálmán and Tibor Takács. True ground speed measurement: A novel optical approach. In *CLAWAR 2008: 11th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines.*, page 8., Coimbra, Portugal, August 8-10. 2008.

[TK08] Tibor Takács and Viktor Kálmán. Velocity measurement using line-scan cameras. In *Robotica 2008 - 4th International Conference on Robotics.*, pages 345–352., Brasov, Romania, November 11-13. 2008.

[TKV08] Tibor Takács, Viktor Kálmán, and László Vajta. *Frontiers in Robotics, Automation and Control, Optical Speed Measurement and applications.*, chapter 10, pages 165–188. InTech Open Access Publisher, 2008.

[KVV08] Viktor Kálmán, Miklós Vogel, and László Vajta. Demining in shallow inland water areas. In *HUDEM'2008 The 7th IARP Workshop Robotics and Mechanical assistance in Humanitarian De-mining and Similar risky interventions.*, pages 144–147., Cairo, Egypt, March 28-30. 2008.

[KT07] Viktor Kálmán and Tibor Takács. Továbbfejlesztett optikai sebességmérő. *A JÖVŐ JÁRMŰVE*, 3-4:46–49., 2007. (in Hungarian)

[TK07] Tibor Takacs and Viktor Kalman. Optical navigation sensor - Incorporating vehicle dynamics information in mapmaking. *ICINCO 2007*, 4th International Conference on Informatics in Control, Automation and Robotics, Angers, FRANCE, MAY 09-12, 2007. pages 271–274

[VVJ+07] László Vajta, Ferenc Vajda, Tamás Juhász, Tamás Urbancsek, Viktor Kálmán, Miklós Vogel, and Ádám Helybély. Emser. Technical report, BME, VIK, IIT, 3DMR, 2007. GVOP-3.3.1.-2004-04-0059/3.0.

[BKS06] Gergely Bári, Viktor Kálmán, and Bálint Szabó. Jármûdinamikai állapotbecslõ algoritmus kifejlesztése. *A JÖVŐ JÁRMŰVE*, 1-2:37–40., 2006. (in Hungarian)

[DHJ+06] Á. Dálnoki, Á. Helybéli, T. Juhász, V. Kálmán, T. Urbancsek, F. Vajda, and L. Vajta. Rendszer és eljárás valós objektumok mozgatására virtuális térben végrehajtott műveletekkel., 2006.

[KBHK04] Viktor Kálmán, Károly Béres, Gábor Hesz, and Szabolcs Kautny. Coconut rugby: Bute's robot in the eurobot 2004 cup. In *1st CLAWAR/EURON Workshop on Robots for Entertainment, Leisure and Hobby.*, number 009, page 4, Vienna, Austria, December 2-4. 2004.

[Kál04] Viktor Kálmán. Building a teleoperated mobile robot. In L. Lehoczky and L. Kalmár, editors, *MicroCad 2004 International Scientific Conference.*, page 4, Miskolc, Hungary, March 18-19. 2004.

7 Bibliography

[1] R. Ahmad, P. Toonders, M.J.D. Hayes, and R.G. Langlois. Atlas mecanum wheel jacobian empirical validation. In *CSME International Congress*, Winnipeg, MA, Canada, 2012.

[2] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *Int. J. Comput. Vision*, 12(1):43–77, February 1994.

[3] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Comput. Surv.*, 27(3):433–466, September 1995.

[4] M. Bilal and S. Masud. Efficient computation of correlation coefficient using negative reference in template matching applications. *Image Processing, IET*, 6(2):197 –204, march 2012.

[5] Magnus Jonason Bjärenstam and Michael Lennartsson. Development of a ball balancing robot with omni wheels. Master’s thesis, Lund University, Department of Automatic Control, 2012.

[6] A. Bonarini, M. Matteucci, and M. Restelli. A kinematic-independent dead-reckoning sensor for indoor mobile robotics. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3750 – 3755 vol.4, sept.-2 oct. 2004.

[7] J. Borenstein and L. Feng. Umbmark - a method for measuring comparing and eliminating dead-reckoning errors in mobile robots. In *Proceedings of SPIE Conference on Mobile Robots*, Philadelphia, USA, October 1994.

[8] Raymond M. Brach and R. Matthew Brach. Tire models for vehicle dynamic simulation and accident reconstruction. Technical report, Brach Engineering, 2009. SAE Technical Paper.

[9] Jochen Brunhorn, Oliver Tenchio, and Raúl Rojas. Robocup 2006: Robot soccer world cup x. chapter A Novel Omnidirectional Wheel Based on Reuleaux-Triangles, pages 516–522. Springer-Verlag, Berlin, Heidelberg, 2007.

[10] Kyung-Seok Byun and Jae-Bok Song. Design and construction of continuous alternate wheels for an omnidirectional mobile robot. *Journal of Robotic Systems*, 20(9):569–579, 2003.

[11] S. Chhaniyara, P. Bunnun, and K. Seneviratne, L. D. & Althoefer. Optical flow algorithm for velocity estimation of ground vehicles: A feasibility study. *International Journal on Smart Sensing and Intelligent Systems*, 1(1):246–268, March 2008.

[12] Correvit(R)-SL. Non-contact optical sensor for slip free measurement of longitudinal and transversal dynamics. Technical report, Corrsys-Datron Sensorsysteme GmbH, 2001. Available: www.corrsys-datron.com (Accessed 2008 May).

[13] John Dainty, Anthony Ennos, Maurice Françon, Joseph Goodman, Thomas McKechnie, Gareth Parry, and J. Dainty. Introduction. In *Laser Speckle and Related Phenomena*, volume 9 of *Topics in Applied Physics*, pages 1–7. Springer Berlin / Heidelberg, 1975. 10.1007/BFb0111435.

[14] E. R. Davies. *Machine Vision (Theory, Algorithms, Practicalities)*. Morgan Elsevier, UK-London, 3 edition, 2005. ISBN 0-12-206093-8.

[15] R.A. DeCarlo, S.H. Zak, and G.P. Matthews. Variable structure control of nonlinear multivariable systems: a tutorial. *Proceedings of the IEEE*, 76(3):212 –232, mar 1988.

[16] Douwe Dresscher, Yury Brodskiy, Peter Breedveld, Jan Broenink, and Stefano Stramigioli. Modeling of the youbot in a serial link structure using twists and wrenches in a bond graph. In *Proceedings of SIMPAR 2010 Workshops*, pages 385–400, Germany, November 2010. SIMPAR.

[17] David J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *Int. J. Comput. Vision*, 5(1):77–104, September 1990.

[18] A. Gfrerrer. Geometry and kinematics of the mecanum wheel. *Computer Aided Geometric Design*, 25(9):784–791, December 2008.

[19] Thomas D. Gillespie. *Fundamentals of Vehicle Dynamics*. SAE International, 1 edition, 1992. ISBN: 978-1-56091-199-9.

[20] David J. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, 1:279–302, 1988. 10.1007/BF00133568.

[21] W. Hirschberg, G. Rill, and H. Weinfurter. Tire model TMeasy. *VEHICLE SYSTEM DYNAMICS*, 45(S):101–119, 2007.

[22] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *ARTIFICAL INTELLIGENCE*, 17:185–203, 1981.

[23] Jan Horn, Alexander Bachmann, and Thao Dang. A fusion approach for image-based measurement of speed over ground. In *Multisensor Fusion and Integration for Intelligent Systems, 2006 IEEE International Conference on*, pages 261 –266, sept. 2006.

[24] Bengt Erland Ilon. Wheels for a course stable selfpropelling vehicle movable any desired direction on the ground or some other base, 1975. US Patent No. 3876255.

[25] Giovanni Indiveri. Swedish wheeled omnidirectional mobile robots: Kinematics analysis and control. *IEEE Transactions on Robotics*, 25:164 – 171, 2009.

[26] T. A. Johansen, I. Petersen, J. Kalkkuhl, and J. Lüdemann. Gain-scheduled wheel slip control in automotive brake systems. *IEEE Transactions on Control Systems Technology*, 11(6):799–811, 2003.

[27] Hassan K. Khalil. *Nonlinear systems*. Prentice-Hall, 2 edition, 2002. ISBN 0-13-228024-8.

[28] M. Killpack, T. Deyle, C. Anderson, and C.C. Kemp. Visual odometry and control for an omnidirectional mobile robot with a downward-facing camera. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 139 –146, oct. 2010.

[29] J. Kim, S. Woo, J. Kim, J. Do, S. Kim, and S. Bae. Inertial navigation system for an automatic guided vehicle with mecanum wheels. *International Journal of Precision Engineering and Manufacturing*, 13(3):379–386, 2012.

[30] Masaaki Kumaga and Takaya Ochiai. Development of a robot balanced on a ball - application of passive motion to transport. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 4106 –4111, may 2009.

[31] T.B. Lauwers, G.A. Kantor, and R.L. Hollis. A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2884 –2889, may 2006.

[32] Almudena Lindoso and Luis Entrena. High performance fpga-based image correlation. *Journal of Real-Time Image Processing*, 2:223–233, 2007. 10.1007/s11554-007-0066-5.

[33] Darryl J. Meister. *Introduction to Ophthalmic Optics*. Carl Zeiss Vision, 2000.

[34] P. Muir and C. Neuman. Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 1772 – 1778, mar 1987.

[35] T.W. Ng. The optical mouse as a two-dimensional displacement sensor. *The optical mouse as a two-dimensional displacement sensor*, A 107:21–25, 2003.

[36] T.W. Ng and K.T. Ang. The optical mouse for vibratory motion sensing. *The optical mouse for vibratory motion sensing*, A 116:205–208, 2004.

[37] Navid Nourani-Vatani and Paulo Vinicius Koerich Borges. Correlation-based visual odometry for ground vehicles. *Journal of Field Robotics*, 28(5):742–768, 2011.

[38] Martin Otter and Hilding Elmqvist. *Modelica - Language, Libraries, Tools, Workshop and EU-Project RealSim*. German Aerospace Center, Dynasim AB, June 2001.

[39] Hans B. Pacejka. *Tyre and vehicle dynamics*. Butterworth-Heinemann, 2002.

[40] J. Palacin, I. Valganon, and R. Pernia. The optical mouse for indoor mobile robot odometry measurement. *Sensors and Actuators A: Physical*, 126(1):141 – 147, 2006.

[41] F.G. Pin and S.M. Killough. A new family of omnidirectional and holonomic wheeled platforms for mobile robots. *Robotics and Automation, IEEE Transactions on*, 10(4):480 –489, aug 1994.

[42] Prof. Dr.-Ing. Georg Rill. *Simulation von Kraftfahrzeugen*. Vieweg+Teubner Verlag, 1994.

[43] Daniel Ruf and Jakub Tobolár. Omnidirektionale fahrzeuge für schwerlasttransport in produktion und logistik. *Logistik und Verkehr in Bayern*, 12:34–35, 2011.

[44] A. Sandabanovic. Variable structure systems with sliding modes in motion control - a survey. *Industrial Informatics, IEEE Transactions on*, 7(2):212 –223, may 2011.

[45] S. M. Savaresi, M. Tanelli, and C. Cantoni. Mixed slip-deceleration control in automotive braking systems. *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 129(1):20–31, 2007.

[46] Siemens. Osmes the optical speed measurement system. Technical report, Siemens Transportation Systems, 2004. Available: www.transportation.siemens.com (accessed 2008 April).

[47] Jean-Jacques E. Slotine and Weiping Li. *Applied nonlinear control*. Prentice Hall, 1991. ISBN 0-13-040890-5.

[48] D. K. Sorensen. On-line optical flow feedback for mobile robot localization/navigation. Master's thesis, A&M University, Texas, USA, 2003. MSc Thesis.

[49] D. Stonier, Se-Hyoung Cho, Sung-Lok Choi, N.S. Kuppuswamy, and Jong-Hwan Kim. Nonlinear slip dynamics for an omniwheel mobile robot platform. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2367 –2372, april 2007.

[50] M.A. Sutton, J.H. Yan, V. Tiwari, H.W. Schreier, and J.J. Orteu. The effect of out-of-plane motion on 2d and 3d digital image correlation measurements. *Optics and Lasers in Engineering*, 46(10):746 – 757, 2008.

[51] J. Tobolár, F. Herrmann, and T. Bünte. Object-oriented modelling and control of vehicles with omni-directional wheels. In *Computational mechanics 25th conference with international participation*, Hrad Nectiny, Czech Republic, November 9-11 2009.

[52] H.-J. Unrau and J. Zamow. *TYDEX-Format, Description and Ref. ManualTYDEX-Format, Description and Ref. Manual*. Initiated by the TYDEX Workshop, release 1.3 edition, Sept. 1997.

[53] Claudio Vecchio. *Sliding Mode Control: theoretical developments and applications to uncertain mechanical systems*. PhD thesis, Universitá Degli Studi di Pavia, Dipartimento di Informatica e Sistemistica, 2008.

[54] Masahiro Watanabe and Shree K. Nayar. Telecentric optics for computational vision. In *Proc. of European Conference on Computer Vision*, pages 439–451, 1995.

[55] II Williams, R.L., B.E. Carter, P. Gallina, and G. Rosati. Dynamic model with slip for wheeled omnidirectional robots. *Robotics and Automation, IEEE Transactions on*, 18(3):285 –293, jun 2002.

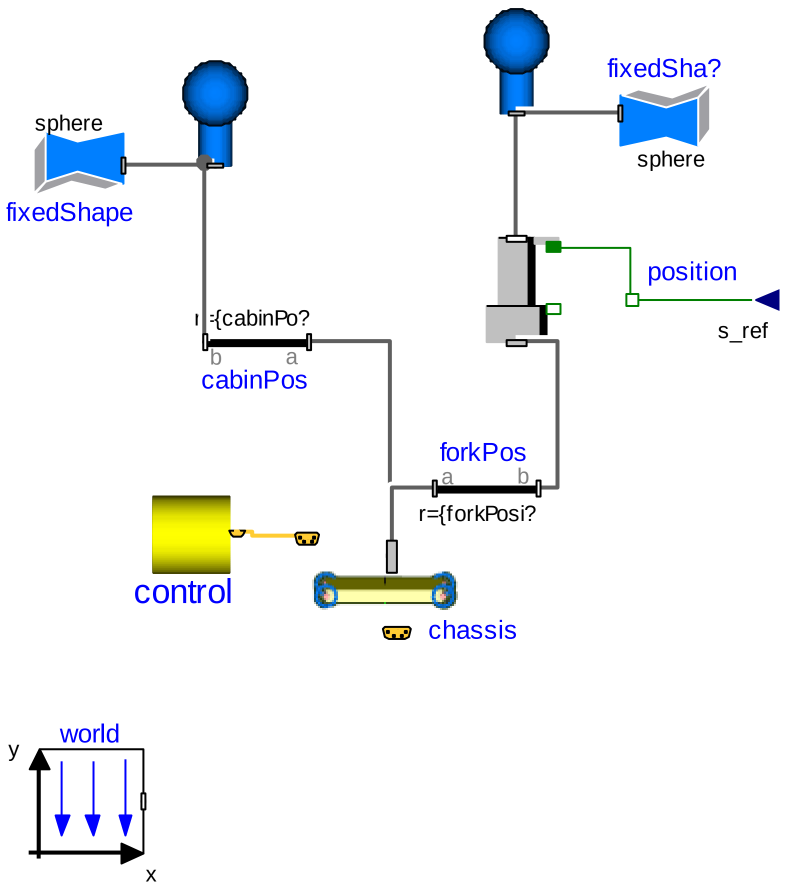
[56] K.D. Young, V.I. Utkin, and U. Ozguner. A control engineer's guide to sliding mode control. *Control Systems Technology, IEEE Transactions on*, 7(3):328 – 342, may 1999.

[57] Dirk Zimmer and Martin Otter. Real-time models for wheels and tyres in an object-oriented modelling framework. *Vehicle System Dynamics*, 48(2):189–216, 2010.

Appendix

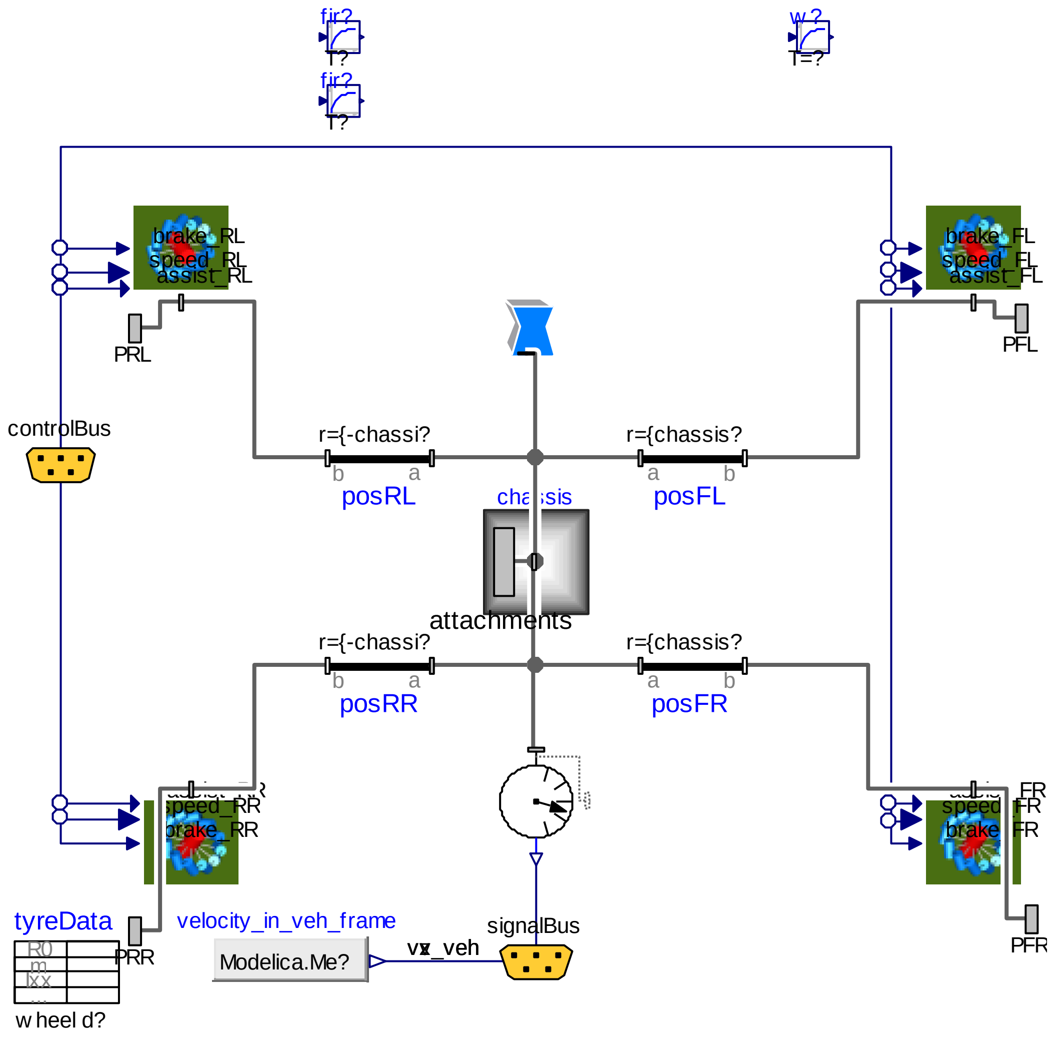
A. Details of the omnidirectional platform simulation

In the following I give some more details on how I created the simulation. The following images were exported from the Modelica-Dymola simulation environment. This simulation environment works on two levels, many models can be built up just by using the first level, using elements from the library, dragging building blocks on the drawing board. The second level is the textual representation, which is automatically generated from the graphical one, but it can be edited by the user adding functionality and modifications, by writing equations. That is of course not visible on these screenshots, I simply include them to give an overview to the interested reader.
Figure A shows the Mecanum forklift high level model. The object **`body`** represents the platform mass and inertia, it can be moved in relation to the chassis with **`cabinPos`** a fixed 3D translation. **`load`** is the cargo, its relative position is set by **`forkPos`** it can be lifted up, or lowered by driving the **`position`** signal through the **`s_ref`** input. The **`world`** component in the lower left corner represents the global coordinate system and defines the direction of gravity.



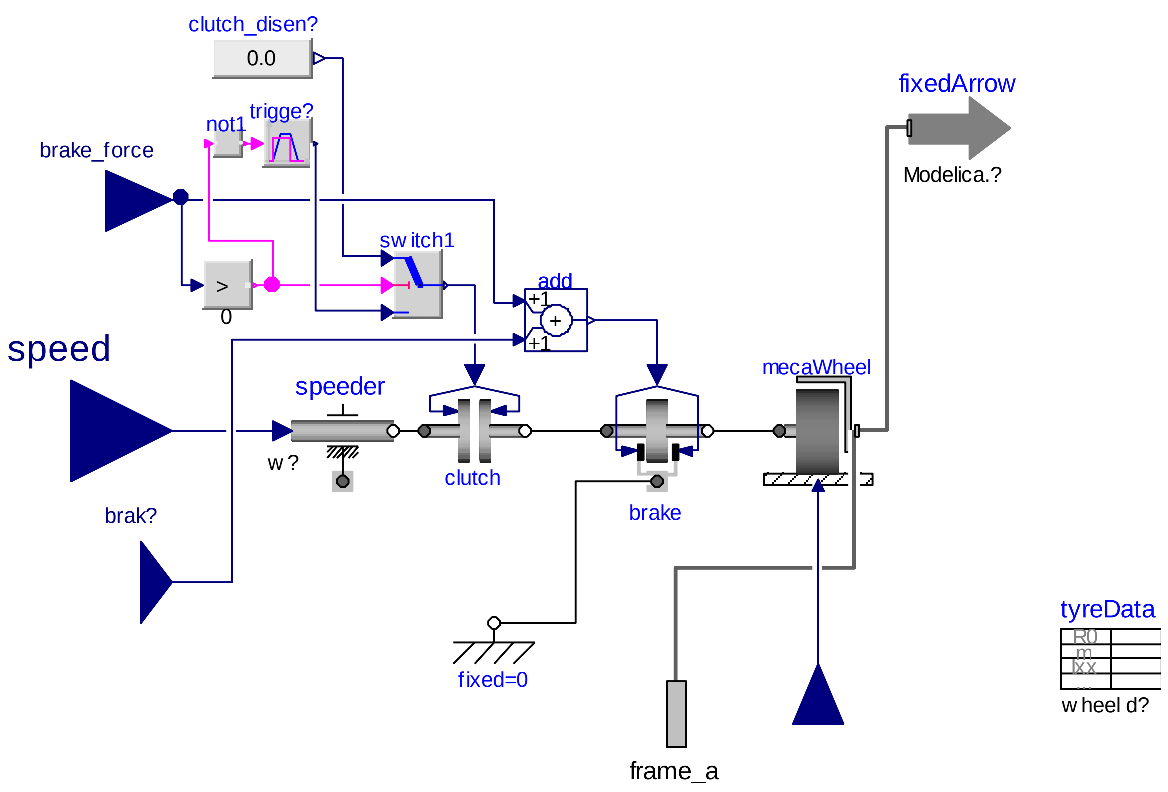
A. The Mecanum forklift model from the simulator

Figure B. shows the internals of the **`chassis`** module from Figure A., the position of the four wheels is determined relative to the central frame, the wheels are driven from the **`controlBus`** from where they receive angular velocity, brake and brake assist signals, also the friction coefficient can be changed for any wheel independently. This is useful for the “ice patch” experiments. Platform state variables are measured by the **absoluteSensor** module and fed to the **signalBus** of the vehicle. The **tyreData** table holds wheel parameters, so that they are the same for all wheels.



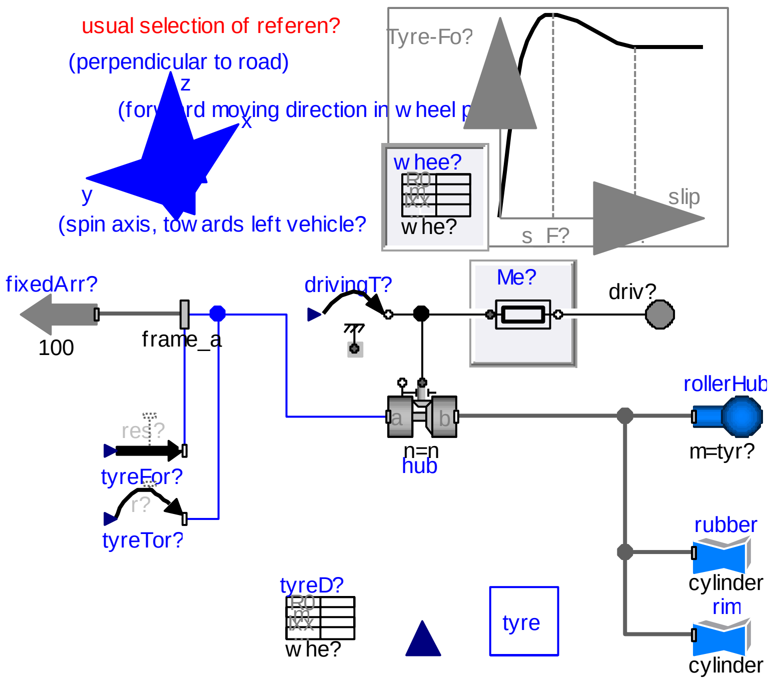
B. Mecanum chassis internals

Figure C. shows the omnidirectional wheel and its actuators. The attachment point is **frame_a**, which is connected to the wheel hub. The wheel body is driven by the **speed** angular velocity signal, through a **clutch** and a **brake**. These modules are simulations of a hydraulic clutch and disc brake, used in the vehicle library of the simulation package. I designed the system so that above a predefined signal magnitude on the **brake_force** input, the clutch automatically disengages, and the wheel can be slowed by the brake. This was necessary because **speeder** is an exact module, it drives its output flange according to the value on its input, therefore it cannot be slowed. In a real system the drive train would be realized with an appropriate electric motor and brake system, in the simulation I decided to make it simpler as modeling this part was out of the scope of my research, at the writing of this thesis. The **brake_assist** (lowest input on the left) can modulate the brake by adding or subtracting from the force pressing the brake pads to the disc, trough the **add** module. The input mue_ext can be driven by any real valued function for the friction coefficient, I use a function of the global position of the wheel frame to define areas of different friction characteristics. **mecaWheel** is the omnidirectional wheel. **fixedArrow** is an arrow that shows in the animation for debugging purposes.



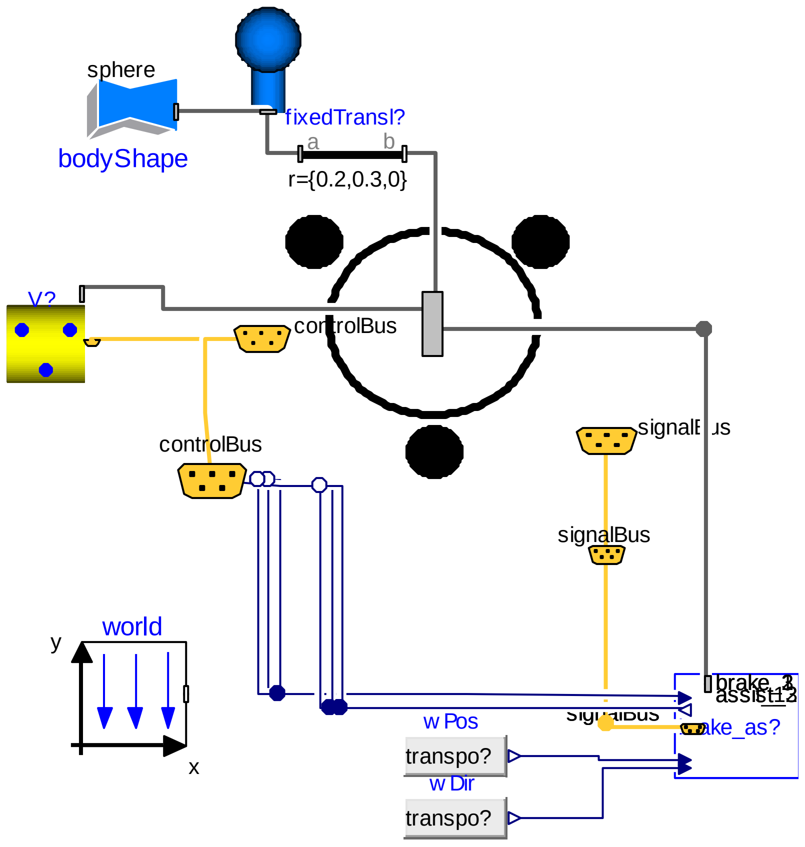
C. The drivetrain of a wheel

Figure D. is the representation of the tire model, that is included in the vehicle library most of the functionality is in the text level of the model. This is the part that I modified to behave like an omnidirectional wheel.

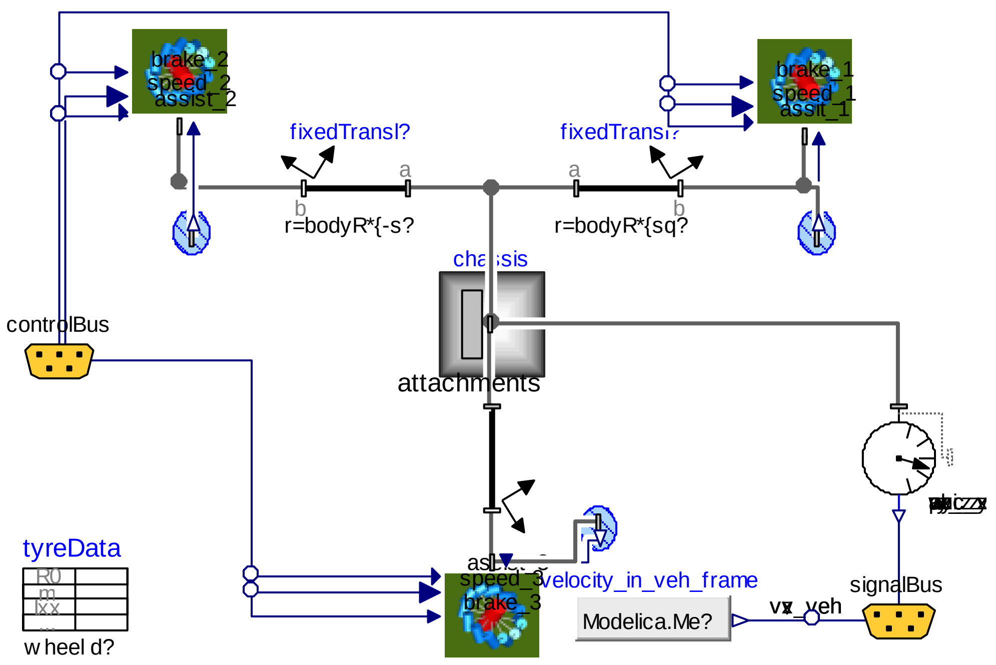


D. Wheel model

For the sake of completeness I present the model of the three-wheeled robot on Figure E. The concept is very similar to that of the Mecanum forklift, the base of the robot is the omnidirectional chassis, to which a body is attached, represented by a mass with inertia. The platform is driven by a control box, and a brake assistant is attached to it. Figure F. shows the three-wheeled chassis, again the concept is very similar to the previous model. On this figure the friction coefficient generators are visible as small blue circles connected to each wheel. The wheels themselves are similar to the Mecanum wheels, the important difference is that their force generation is different, because their "rollers" are set at 0°.



E. **The three-wheeled platform model**



F. Internals of the three-wheeled chassis