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

Viktor Kálmán

On modeling and control of omnidirectional wheels

PhD. dissertation, excerpt

Advisor: Dr. László Vajta, associate professor

Budapest 2013.
1 Introduction

This dissertation is written with the aim of solving some of the problems related to 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. Modern engineering practice uses simulation whenever possible, to cut costs and speed up development. This is also true for mobile robotics. 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.

1. Omnidirectional wheel – the well known Mecanum wheel [24]

The need for eliminating human error becomes ever so obvious when human operators are managing expensive and/or powerful machinery, as the cost of failure can be very high. We meet systems that supervise and correct human control in dangerous situations every day while driving. Corrective systems such as ESP or ABS increasingly find their way into smaller markets such as industrial transport vehicles. Omnidirectional platforms are not immune to human error either, however due to their different working principle they require different methods.

2 Research goals

This dissertation is written about a class of robotic transport vehicles and aims to solve some of the related problems, in their simulation and control. In the following I outline the main directions of my research work.

- While omnidirectional wheels have undisputed mobility advantages, they require relatively complicated control systems to perform well. Simulation is a great development tool, but to be able to concentrate on a given task, it is useful to have a library of proven components. There is a great variety of wheel models available for automotive simulation, however omnidirectional wheel models are harder to come by. Realizing the many common aspects with regular tire models I decided to find a way to transform widely known wheel models, into omnidirectional wheel models.
• 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, and more importantly 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 nonlinear control problem and created a controller that makes braking of omnidirectional platforms more predictable and safe.

• A great proportion of ground vehicles, such as caterpillar drives, certain differential drives, hovercrafts and omnidirectional vehicles all suffer from the problem that they operate with high amounts of wheel slip – or no wheels at all – or their center of rotation is uncertain. This means that measuring wheel angular velocity to estimate vehicle movement speed is not practical. My goal was to design a system that can provide accurate velocity feedback, independent from platform kinematics. My idea was to use a method similar to the optical mouse, however with a couple of orders of magnitude faster operation.

• Usability and correctness of the above methods and algorithms have to be demonstrated in simulated and real world experiments.

2.1 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 Association1. “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 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.”[16] 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. Figure 2. shows a Mecanum wheeled forklift and a three wheeled omni-platform rendered in Dymola.

2. Mecanum platform and three-wheeled omni-platform, Dymola animations

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 (thesis 3.) 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.
3 Contributions

Scientific contributions and results of my research are summarized in the following sections.

3.1 Omnidirectional wheel model in simulation

Thesis 1: 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 modeled omnidirectional wheel – such as roller angle – are superposed.

[KJVS12], [KV12b], [VAL12]

Due to advances in control theory, mechatronics and manufacturing technology omnidirectional wheels – invented in the seventies [11] – are living their renaissance. They have a rich history in the literature, they have been used for various tasks and many different embodiments are known [5], [18]. Their use ranges from robotic soccer applications, through industrial heavy load transporters [20], and vehicle simulators [1], to educational and leisure projects [3], [14]. Figure 3. shows some of these applications. Modern engineering uses simulation for almost every task imaginable, to cut costs, speed up development and minimize changes late in the product life cycle. Mecanum platforms are no exception, since they require more complex mechanical design and control, than traditional vehicles.

3. Examples for the use of omnidirectional wheels

As I found no publicly available simulation libraries for Mecanum wheels, I decided to create one myself. The rollers themselves are often covered with rubber, especially in heavy duty applications, this means the rollers themselves behave much like a small solid tire. 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 [8]. 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.

### 3.1.1 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 contact with the ground and obstacles, and it has to generate the forces and torques that arise in the contact patch. Most of these calculations are nonlinear because of the characteristics of the tire material [4].

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, 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 friction force. 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 [17], [19]. This modeling approach implies that many of the model parameters have no explicit physical meaning, they only represent mathematical coefficients of the polynomials. This makes the use of these models complicated, one has to have real measurements of real tires to obtain useful parameters.

![Friction force generation vs. slip, static curve (after [10])]
- 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 5., 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:

\[ \mathbf{d}_i = R_0 \left\{ \sin \left( \frac{2\pi i}{n} \right), 0, \cos \left( \frac{2\pi i}{n} \right) \right\} \quad (1) \]

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

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, resulting in loss of accuracy.
- Crude contact model, most rollers are not a simple cylinder. In order to make them ride smoother, they are created with a varying cross section and rounded edges. A better geometry model would add even more complexity.

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.

**Single roller model:**

To overcome some of the disadvantages of the model presented in the previous paragraphs 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. The vectors describing a roller touching the ground are shown on Figure 6. The unit vectors $\mathbf{u}_x, \mathbf{n}_y, \mathbf{k}_z$ point in the direction of the W – wheel – system axes, set up according to the TYDEX notations [25]. To accommodate the omni-wheel, I defined a $\mathbf{n}_w$ unit vector in the direction of the rollers' axis that is the direction it can exert force – I call it active direction.

$$\mathbf{n}_w = \mathbf{u}_x \cdot \text{Rot}_\gamma$$

where $\text{Rot}_\gamma$ is a 3x3 rotation matrix of $\gamma$.

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

To the direction of $\mathbf{u}_w$ – 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. We can make a further step by defining slip for the omnidirectional wheel. In many 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 the idealized roller only generates force in the direction of its spin axis ($\mathbf{n}_w$) we shall only calculate slip in this direction. Rill [19] defines slip as "total slip":

$$s = \frac{v}{R|\omega| + v_{num}}$$

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

$R$ is the wheel radius, $\omega$ 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 $v$ to $v_{omni}$ in the slip equation:

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

where $v_{fw}$ is the projection of the velocity of the center of the wheel in the $\mathbf{n}_w$ 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 base model using the modified slip equation. The direction of this force has to be set to the direction of $\mathbf{n}_w$.

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 can be modified if needed. If more accuracy is required roller inertia and rolling resistance can be incorporated in the model. The roller discontinuities might be modeled by
modulating tire forces by an angle dependent function [7]. 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 the single roller model in my further work.

Functionality of the wheel model was verified by simulation, applying the inverse kinematic model of omnidirectional platforms [12]. Figure 7. shows spin by translate movements of a Mecanum forklift, and a three-wheeled omni-platform. They are moved with constant velocity in the x direction and with a sinusoidal velocity in the y direction, while spinning around their centers.

7. Snapshots from simulated spin – while - translate movements of omnidirectional vehicles
3.2 Brake assist for omnidirectional wheels

**Thesis 2:** 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 characteristics.

[Kal13], [KV12a], [VAL12]

Due to their design omnidirectional wheels in general lack the capability to generate substantial side forces, forces 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. Also 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 with regard to the center of gravity. 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.

![Diagram of omnidirectional wheels and forces](image)

8. Forces and velocities during braking

Friction force generated at the wheels of a braking vehicle always opposes the velocity of the wheel center, and its magnitude depends on the intensity of braking, wheel load and the contact properties of the wheel and the ground. This dependency can be described by appropriate wheel and ground models, as accurately as needed, however the result will be only as accurate as our knowledge of instantaneous load distribution and local ground characteristics, which are hard to measure. The load dependent effect is nonlinear it is often approximated with a negative quadratic function \[ f \propto v^2 \]. The direction of the force is a trigonometric function of wheel velocities.

Because of the uncertainties and nonlinearities present in the system I decided to apply a nonlinear control approach. I considered using sliding mode control as recommended by many, sources in the literature: [6], [13], [21], [26], [27]. The system to be controlled is a
parameter uncertain nonlinear MIMO system. Its state equation can be written in the general form:

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

(5)

where the usual notations are used – $x$ is the state vector, $u$ is the input and $f(\cdot)$ is a nonlinear matrix valued 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 – a braking omnidirectional platform – it is more practical to also include $v_i$, wheel center velocities: $x = \{v_i, v_c, \omega\}'$. These can be derived in a simple way from platform center linear and angular velocity.

$$v_i = v_c + \omega \times r_{ci}$$

(6)

where $r_{ci}$ is the vector pointing from the platform center to a given wheel center. The input vector consists of $f_i$, 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:

$$m \dot{v}_c = \sum f_i$$

(7)

$$J_c \dot{\omega} = \sum f_i \times r_i$$

(8)

These equations are written for the geometrical center of the vehicle. Uncertain parameters are $M$ mass of the vehicle, the magnitude of $f_i$ and the $J_c$ inertia matrix of the vehicle. State variables $v_c$ and $\omega$ are assumed to be measurable. (For example by my method described by Thesis 3.) Another important property of the system, that it is strongly cross coupled, all inputs have an effect on all state variables. This follows from the general kinematic equations of the system [12]:

$$M \begin{pmatrix} v_c \\ \dot{\omega} \end{pmatrix} = R \Omega \cos(\gamma)$$

(9)

$$M = \begin{pmatrix} n_{w1x} & n_{w1y} & r_1^T u_{w1} \\ n_{w2x} & n_{w2y} & r_2^T u_{w2} \\ \vdots & \vdots & \vdots \\ n_{wNx} & n_{wNy} & r_N^T u_{wN} \end{pmatrix}$$

(10)

where

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

The idea behind the controller is that the swerving effect is caused by the wheels starting to move in the “wrong direction”. The sliding manifold can be constructed by describing a trajectory with no movement in unwanted directions. This results in an error function for each wheel, serving as a sliding manifold

$$s_{tot} = s_{th} + s_{\omega} |r_i| + s_{ij}$$

(11)

where $s_{th}$ corresponds to a wheels movement in the direction perpendicular to the platform velocity at the start of braking.
\[ s_{th} = \| F_i \| \nu_{th} \]  
(12)

see Figure 8 for notations. \( \| F_i \| \) is a unit vector in the direction of the friction force \( \nu_{th} \) is the projection of \( \nu_i \) on the vector perpendicular to \( \nu_c \) initial platform velocity.

\[ s_{lo} = \| \beta \| \Omega \]  
(13)

where \( \| \beta \| \) is a unit vector in the direction of the angular acceleration caused by the wheel i.e. the torque generated by friction force of wheel \( i \).

\[ s_{ll} = \psi(\delta_i) ||f_i||\nu_i || \]  
(14)

where \( \delta_i \) is the angle between the active direction of the wheel and \( \nu_i \), \( \psi \) is a nonlinear weighing function.

The control strategy is to switch off the brake to the wheel when \( s_{tot} \geq 0 \) and apply the brake otherwise.

\[ u_i = f_i(\zeta(s_{tot})) \]  
(15)

where

\[ \zeta(x) = \begin{cases} 0 \text{ for } \forall x \geq 0 \\ 1 \text{ for } \forall x < 0 \text{ } \times \in \mathbb{R} \end{cases} \]  
(16)

The control loop is shown on Figure 9.

9. The closed control loop

To find out if the closed loop control system is stable I decided to use the so called passivity approach [13] (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. Generally a system is called passive if there exists a storage function \( V(t) \geq 0 \), such that for all \( t_0 < t_1 \),

\[ V(t_1) \leq V(t_0) + \int_{t_0}^{t_1} y(t)u(t)dt \]  
(17)

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 systems, 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:

$$V = \frac{Mv^2_c}{2} + \frac{J\Omega^2}{2} \quad (18)$$

considering the linear and rotational kinetic energies. The “supply rate” – in this case would be more appropriately called the “dissipation 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 (17) holds true for $\forall t_0, t_1$ time instants. However for a braking maneuver a strict inequality is a necessary condition 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 $90^\circ$ to the direction of movement

The second criterion is guaranteed by design, if the vehicle is created so that the rank of $M$ in equation (9) 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 $\forall s_{itot} \geq 0$.

### 3.2.1 Demonstration in simulation

I created a Mecanum wheeled forklift model and a three wheeled platform with omni-wheels (a.k.a. kiwi drive), and executed experiments to demonstrate the capabilities of the controller. I investigated several kinds of disturbances. The first thing I included in the simulation was to quantize the feedback signal in time, thus creating a time dependent error in the feedback, also getting closer to the behavior of a real digital sensor.

![Image of simulation](image.png)

10. Braking in a straight line, Mecanum platform slowing from 10m/s

The sampling time of the simulated velocity sensor was 0.01s and a zero order hold supplied the feedback between sampling instants. Figure 10 shows snapshots from two experiments with the Mecanum platform, slowing down from a straight 10m/s movement, with quantized feedback.

To introduce another kind of disturbance I created a patch on the ground with a smaller friction coefficient and made the vehicle brake with some of its wheels on the “icy patch” and the others on regular ground. I also added some noise to the feedback. The noise is simulated...
by a uniform distribution centered around the real value $\eta$. It can be set independently for the linear and the angular velocities. Also an offset can be incorporated in the signal.

$$\tilde{v}_c = v_c + \eta(v_c) + v_0$$  \hspace{1cm} (19)

Noise is added to the angular velocity feedback similarly.

Figure 11 shows overlaid snapshots of as assisted and non-assisted ice patch maneuver of the three wheeled platform, with a noisy feedback signal. The brighter colored platform had an active brake assist with a noisy feedback, the dark platform had no brake assist.

11. Demonstration of disturbance rejection, three wheeled platform on ice

This brake assist controller is generally applicable for omnidirectional vehicles. The controller eliminates the swerving tendency that manifests during braking. I created the control law under the assumption that there are high uncertainties in the platform model, exact wheel characteristics and load distribution are deemed unknown. Also I assumed noise to be present in the velocity feedback and the friction coefficient of the ground. Due to these uncertainties and the nonlinear nature of the control problem I applied sliding mode control. The sliding manifold is constructed so that the velocity components of each wheel due to platform angular velocity and sideways linear velocity are driven to zero. The cross coupled nature of the platform kinematic model ensures that this control approach stops the platform in finite time. The stability of the closed control loop is guaranteed by the passivity approach.

---

$\eta$ Uniform distribution might not be the best simulation of sensor noise, however my purpose here is to demonstrate disturbance rejection of the controller.
3.3 Velocity feedback

**Thesis 3:** 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.

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

The wheels of omnidirectional platforms in general operate with large amounts of slip, which is perfectly normal and characteristic of their operation. This fact however makes traditional – wheel rotation based – velocity measurement methods highly inaccurate. To deal with this problem, I proposed a new optical speed measurement system that produces accurate three degree of freedom velocity measurements up to a very high speed, independent from wheel rotations. The method in itself is not new, it is similar to the working principle of a common optical mouse [9]: 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. 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 not common 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 – several thousand pixels are common – resolution in one dimension, frame rates at the order of 10 to 100 kHz and relatively low prices.

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 12. shows an illustration of the difference between a matrix camera and a line detector with optical integration.

![Illustration of optical integration and the problem of sideways movement](image-url)

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, without any actual movement in the parallel direction. The problem is illustrated on Figure 12. 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 easy to reach 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.

A great advantage of using a one dimensional measurement is that a very simple algorithm can be used, a 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 any of the well known methods can be used, I found correlation to work very well.

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

\[
d = 1 - r
\]

\[
r = \frac{1}{n-1} \sum_{i=1}^{n} \left( \frac{x_i - \bar{X}}{s_X} \right) \left( \frac{y_i - \bar{Y}}{s_Y} \right)
\]

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.

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

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. The simulator moves a virtual camera above the ground. The ground 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.

The virtual camera and sensor implemented in the simulator has 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.

When running the simulation consecutive images are chosen automatically by translating a $k \times l$ mm window on the ground image with a certain amount of pixels according to the pre-defined movement speed, frame rate and direction. The working principle is illustrated by Figure 14. The direction of movement can be chosen by specifying the angle $\alpha$. The algorithm is illustrated on Figure 15. The two neighboring images are shifted and compared according to the distance measure. The shift resulting the smallest distance, is the estimated displacement. The exact traveled distance in pixels is known from the simulated speed, the error of the measurement can be obtained easily by subtracting the estimate from the set value and normalizing.

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.

### 3.3.1 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 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. The error is calculated the following way:

\[ E_{cpx} = d_{m_{cpx}} - \text{Round} \left( d_{r_{ipx}} \cdot \frac{r_i}{r_c} \right) \] (21)

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.

It became clear from the experiments with different settings that apart from texture dependence the most important factor is overlap between snapshots. This is verified by Figure 0 where the overlap percentage is displayed for sideways movement at 12 m/s and 2500fps settings. Width and Length is the size of the patch on the ground sensed by the detector. By comparing the images of Figure 0 and measurements on different textures (not displayed here) 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 of quantization noise, a single pixel displacement corresponds to a larger distance on the ground.

![Stone - error in camera pixels (hi-res)](image)

### 3.3.2 Enhancements practicalities

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 kind of filtering cannot help.

Other than filtering methods using apriori information can be applied. For instance we know that the value of acceleration is limited, so the velocity does not change abruptly 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 15000 fps image sensor this means that there is 60 ms between images. The algorithm used in the simulator lends itself naturally for FPGA based processing, since the distance metric values between consecutive images can be computed for each shifted image in a parallel manner. According to examples found in the literature [15], [2] algorithms of similar purpose and complexity can easily be executed in this timeframe. It is safe to say that an algorithm similar in function that 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 [23]. 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 field of view on the ground, therefore price depends heavily on the field of view.

**Texture:** plays an important role, since without surface variation the sensor does not work. This can be helped by special lighting, but at this point this remains untested. For example the ground textures “plastic” and “metal” from Figure 13 were useless in the presence of noise.

**Water, dirt:** the system is sensitive to dirt, blocking the optics and shiny surfaces, therefore it is best used in a controlled environment such as a racetrack, or indoors.
4 Outlook

The findings of this dissertation have been published in international conferences and journals. The main tool of my research work, Modelica was introduced to me as one of the best multidomain physical simulation languages that is becoming more and more widely used in the industry. The Virtual Engineering Laboratory of the IFF Fraunhofer Institute Magdeburg – a well recognized research and development company in Germany – uses it as their preferred tool. The simulation in Modelica can easily be used for research and development of omnidirectional platforms, both for mechanical and control system design, thanks to the configurable models. The results of the second thesis can be easily adapted to any omnidirectional vehicle since only the kinematics of the platform has to be known, granted that there is adequate velocity feedback.

Development of the system presented in my third thesis started in 2005 when the basis of the idea received a government grant János Irinyi. A prototype of the sensor was developed with the help of this grant, and several students at the University in the context of MSc thesis work. Figure 17 shows an early prototype in the lab, and a mouse chip based system mounted on a hovercraft that participated in an engineering contest.

5 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 at 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.
6 Bibliography

6.4 Publications referenced in the theses


6.5 Other publications


6.6 References


