Robin Ritz

Development of a Dynamical Model for Retina-based Control of an RC Monster Truck

Bachelor Project

Automatic Control Laboratory
Swiss Federal Institute of Technology (ETH) Zurich

Advisors

Thomas Besselmann
Tobi Delbrück
Prof. Manfred Morari

Abstract

At the Institute of Neurinformatics (INI), a shared institute of the University of Zurich (UZH) and the Swiss Federal Institute of Technology Zurich (ETHZ), a dynamic vision sensor for fast visual processing has been developed. The dynamic vision sensor uses a silicon retina to detect contrast changes in a scene. An ongoing project is to develop a controller based on a silicon retina for a small robotic monster truck. The goal of the project is to drive at high speed along a drawn route. In this semester project, a dynamical model of the RC monster truck is derived and a simulation environment to test retina-based control strategies is developed. The simulation is based on Blender, a powerful open-source modeling environment. Further, first control strategies for event-based lateral control are implemented and tested in the simulation, as well as on the real vehicle. At the end of this semester project, the truck is able to follow a chalk drawn line at a little more than walking pace.

Contents

1 Introduction

Cars with an automatic steering system are a current area of research. Modern cars already provide active assistance concerning steering and drive power control and prototypes have been developed, which are able to drive completely autonomous. Indeed, autonomous vehicles open a field of challenges which not all have been solved completely until today. One of these challenges, which are still a current area of research, is the visual processing. An autonomous vehicle which can be used on common roads must have visual sensors. Otherwise, the vehicle will not be able to track the road. Since common cameras capture pictures on a constant high frame rate, they cause high requirements on the memory resources. In addition, the pictures have to be processed in real time to detect the road and potentially to identify obstacles, what requires high processor resources. Hence, the cost for a system with good visual processing are high, since hardware resources are expensive. A further problem might be that the performance of the processors is limited, even if the cost have no importance.

A new approach to avoid these problems are event-based visual sensors. The key idea of an event-based sensor is that only changes in the measurement area are submitted. Thus, in the case of a visual sensor, if a pixel of the picture does not change, then no information is generated for this pixel. This approach of event based visual sensors allows fast handling of visual data with relatively small hardware resources. In this project, such an event-based visual sensor is used to detect the road. In our case, the road is defined by two parallel lines or by one single line.

In addition to a reliable identification of the road, an autonomous vehicle has to possess a robust control strategy, to track the road. Usually, a control strategy is tested virtually before it is applied to the real model, to avoid damages and to save time. But the control strategy for a vehicle to track a road is highly coupled with the output of the visual sensor, and therefore it is not sufficient to test the controller and the handling of the visual data separately. For that reason, a simulation environment, based on prior developments by Albert Cardona which allow coupled testing of event-based visual processing and controllers (see [4]), is built during this project to test the control performance including the visual part.

For the simulation, as well as for the controller design, a model of the vehicle is essential and therefore a model of the RC monster truck is established during this semester project.

In Section 2, the vehicle used during this project is described. Section 3 contains the modeling of the RC monster truck for the simulation, as well as for the controller design. In Section 4, the simulation of the truck in Blender is introduced and in Section 5, the first implemented control schemes for lateral control of the truck are described.

2 Description of the RC Monster Truck

The vehicle used in this project is a remote-controlled monster truck based on the standard model ‘Traxxas E-maxx’, which is about 0.5 m long and able to reach velocities of about 50 kph. Figure 1 shows a picture of the RC monster truck used in this project. The truck was upgraded with a USB servo motor controller for the actuation of the steering angle and the motor power (see A in Figure 1). The vehicle provides a three dimensional acceleration sensor with a range of 2 g (see B in Figure 1).

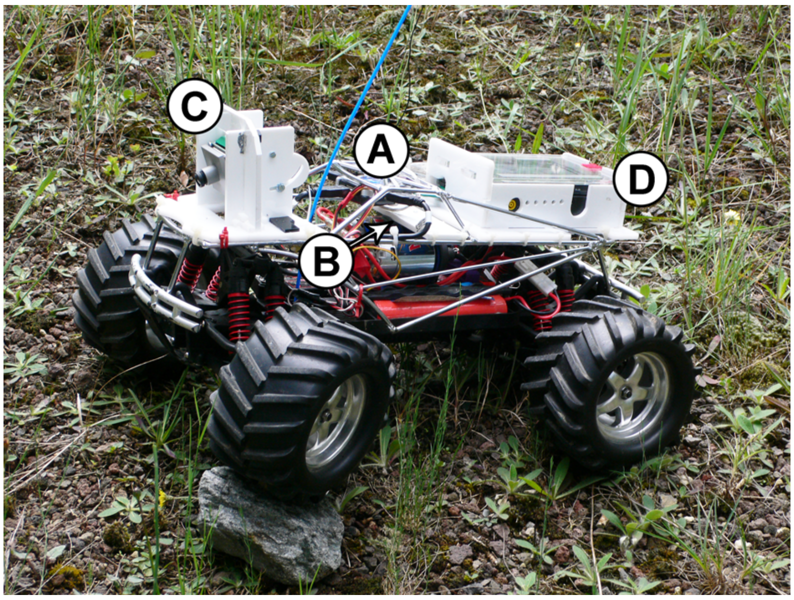


Figure 1: Picture of the RC monster truck used in this project. A: USB servo controller, B: Acceleration sensor, C: Silicon retina, D: Miniature Sony PC.

A further sensor is the silicon retina (see C in Figure 1), which is mounted in front of the truck. The silicon retina is a dynamic vision sensor, which allows fast visual processing. Conventional cameras are running at a constant frame rate and capture an entire picture on each frame. To handle each picture in real time, large processor and memory capabilities are required. The silicon retina does not capture pictures with a constant frame rate, but records only local contrast changes. Thus, if a pixel changes its contrast considerably, the silicon retina creates an event. The created event contains the position of the pixel in the visual field and the time, when the pixel has changed its contrast. Further, the event provides the information, if the contrast of the pixel has increased or decreased. These events are called spikes, so the output of the silicon retina are spike packages, containing information about the contrast change of the different pixels in the visual field. The spikes of the silicon retina are handled by jAER. jAER is a Java-based package for fast visual processing with spikes. In this project, jAER runs on a miniature Sony PC, which is mounted on the RC monster truck (see D in Figure 1). The control policy and the filter, which handles the spikes to detect the road, are implemented in jAER on the embedded PC. The embedded PC is connected with the sensors and the USB servo motor controller through USB cables.

3 Modeling the RC Monster Truck

In order to implement a realistic simulation of the vehicle dynamics and further to design a controller for the truck, it is necessary to identify models of different complexity. The models should include all relevant dynamics of the RC monster truck, but at the same time there are restrictions on the complexity, because otherwise the calculation effort raises too much.

3.1 Assumptions

In the following, the basic assumptions which are made for the identification of the complex model are explained.

For the modeling, the RC monster truck is divided into a rigid body and four wheels, which are coupled to the body by the suspensions. The suspensions are assumed to consist of a spring, a damper and an inflexible stick, which are fixed in a particular configuration with the body and the wheel. The detailed configuration of these three elements will be introduced when we derive the suspension forces. We make the assumption that the center of gravity of the body is the same point as the center of gravity of the whole vehicle. Further, we assume that we can directly set the driving torque and the steering angle without any delay.

We define a local coordinate system, which moves with the truck. The axes of the local coordinate system are denoted with lower case letters, while the axes of the inertial coordinate system are denoted with upper case letters. The x-axis of the local coordinate system targets always in forward direction of the vehicle and the z-axis, defined as the axis normal to the ground, keeps always the direction of the inertial Z-axis. Hence, the local coordinate system turns around the z-axis when the truck drives through a curve. The center of the local coordinate system is the center of gravity of the body. If the body turns around the x-axis or around the y-axis, the local coordinate system does not follow, otherwise the z-axis would no longer target in the direction of the inertial Z-axis. So, the centers of the four wheels do not move in the local coordinate system. This coordinate system is chosen because it provides the advantage that the longitudinal velocity of the vehicle always has the same direction as the x-axis and the lateral velocity of the vehicle always has the same direction as the y-axis. That simplifies the derivation of the differential equations for the longitudinal and the lateral acceleration of the truck.

For the modeling of the rotational dynamics of the body, a further coordinate system, which is rigidly coupled with the body of the truck, is defined. The axes of this coordinate system are denoted by x_b, y_b and z_b and the center point is the same as the center point of the local vehicle coordinate system. This coordinate system allows to describe the rotational dynamics of the body in a simple way.

Thus, relative to the inertial coordinate system, the vehicle coordinate system rotates around the Z-axis and the coordinate system rigidly coupled with the body rotates around all three axes. The rotation angle of the vehicle around the Z-axis is denoted by φ_z and the additional rotation angles of the body are denoted by φ_x and φ_y. We can assume that the angle φ_x is small and does influence the forces on the body.

Further, we assume that the silicon retina is rigidly coupled with the body at a constant height and at constant inclination.

3.2 Nonlinear Model

For the simulation (see Section 4), a model of the truck is required that represents the real dynamics of the vehicle as accurately as possible. For that reason, a complex model, including the suspensions and the wheel dynamics, is derived in this section.

Longitudinal Dynamics

The longitudinal dynamics of the truck are considered in an external view. We assume that the forces generated by the wheels directly affect the truck, i.e. the influence of the suspensions is neglected. This assumption can be made, because the suspensions do not allow a longitudinal departure of the wheels related to the body. The forces generated by the wheels are denoted $F_{x,i}$, with $i \in \{1, 2, 3, 4\}$ for each wheel. Another relevant force for the longitudinal dynamics of a vehicle is the aerodynamic drag, which is assumed to act on a constant height h_{air}. The longitudinal inert force $F_{x,t} = m_{tot}\ddot{x}$ acts on the center of gravity at the height h_g. Figure 2 shows a side view of the truck including all relevant forces for the longitudinal dynamics of the vehicle. The forces of the left wheels are not drawn, but of course they have to be considered as well. The longitudinal position of the center of gravity is described by its distance to the front wheels l_f, respectively its distance to the rear wheels l_r.

A balance of forces in longitudinal direction yields

$$m_{tot}\ddot{x} = F_{x,1} + F_{x,2} + F_{x,3} + F_{x,4} - F_{air}, \tag{1}$$

where the mass m_{tot} includes the mass of the body and the mass of the four wheels. The aerodynamic resistance F_{air} depends on the velocity of

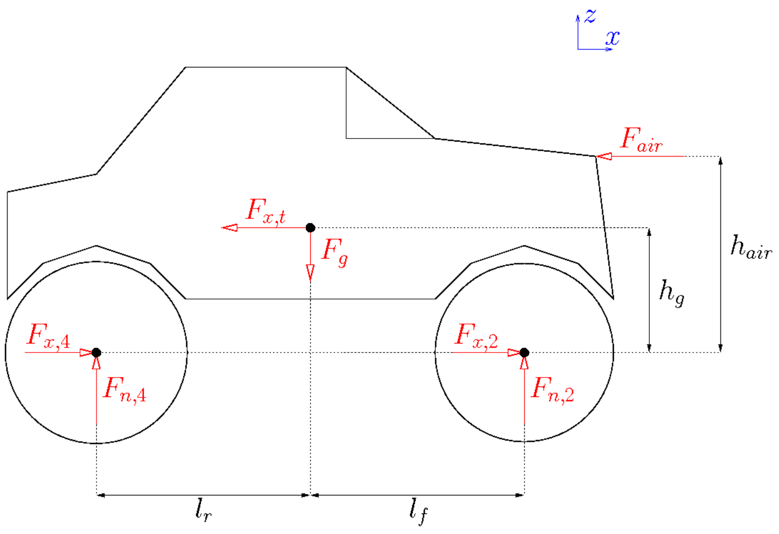


Figure 2: Side view of the RC monster truck with relevant forces.

the vehicle and according to [9], the force F_{air} can be written as

$$F_{air} = \frac{1}{2} c_a A \rho_{air} \dot{x}^2 \text{sign}(\dot{x}), \tag{2}$$

where c_a denotes the aerodynamic coefficient of the truck, A stands for the frontal area of the vehicle and ρ_{air} represents the specific weight of air. The local coordinate system is rotating, what causes the additional term $m_{tot}\dot{y}\dot{\varphi}_z$ in longitudinal direction. But this term is small and thus can be neglected.

Lateral Dynamics

For the development of the model for the lateral vehicle dynamics, we assume again that the suspensions have no influence. Thus, the lateral forces $F_{y,i}$, generated by the wheels, act directly on the lateral dynamics of the truck. For this assumption, we use the simplification that the suspensions do not let the wheels move in lateral direction. Aerodynamic drag does not have to be considered for the lateral dynamics, because the other forces dominate. Figure 3 shows a top view of the vehicle. The distance between the front respectively the rear wheels is denoted by b_w. The angle φ_z is shown in Figure 3, even if it does not affect the forces, to denote that the local coordinate system rotates. That causes an additional term in the equation of motion for the lateral dynamics.

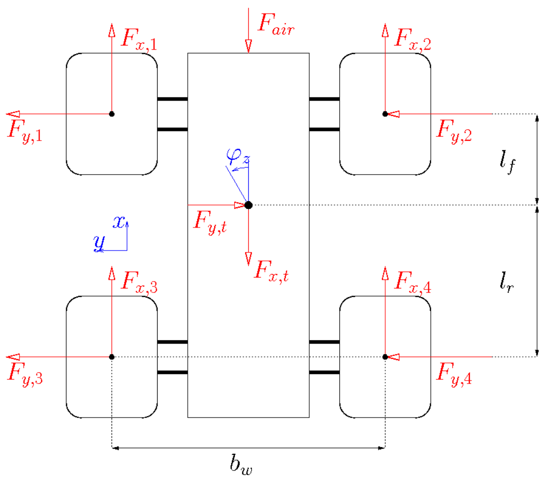


Figure 3: Top view of the RC monster truck with relevant forces.

The lateral equation of motion can be determined by a balance of forces in lateral direction. This results in

$$m_{tot}\ddot{y} = F_{y,1} + F_{y,2} + F_{y,3} + F_{y,4} - m_{tot}\dot{x}\dot{\varphi}_z. \tag{3}$$

For the lateral dynamics, the term $m_{tot}\dot{x}\dot{\varphi}_z$ caused by the rotating coordinate system can not be neglected, because the variables $\dot{x}$ and $\dot{\varphi}_z$ can reach considerable values.

Vertical Dynamics

The motion in the vertical direction influences the range of vision of the silicon retina and should therefore also be modeled. We assume that the wheels are always in contact with the ground, i.e. the wheels do not move in vertical direction. Thus, only the vertical motion of the body has to be modeled. Figure 4 shows a side view of the body. Again, the forces of the left wheels and suspensions are not drawn, to keep the figure well arranged. As mentioned before, the center of gravity of the body is assumed to be at the same point as the center of gravity of the whole vehicle. The vertical forces $F_{z,i}$ from the wheels are assumed to be independent of the suspensions and the point of application of these forces can be seen in Figure 4. The influence of the suspensions is modeled by the forces $F_{z,sus,i}$, which act on the height h_{sus} and on the same longitudinal position as the vertical forces $F_{z,i}$ from the wheels.

Neglecting the influence of φ_x and φ_y, we can determine the vertical

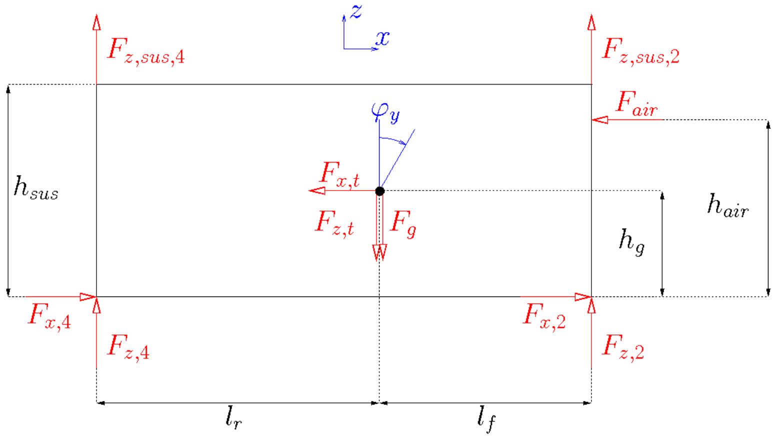


Figure 4: Side view of the body of the RC monster truck with relevant forces.

equation of motion using a balance of forces. We get

$$m_{body}\ddot{z} = \sum_{i=1}^{4} F_{z,i} + \sum_{i=1}^{4} F_{z,sus,i} - F_g \tag{4}$$

for the dynamics of the body in vertical direction with $F_g = m_{body}g$. The motion in vertical direction has no big influence on the whole vehicle model and that is why the influence of the angles φ_x and φ_y is neglected here to avoid a too complex model.

Rotational Dynamics related to the x_b-Axis

The rotation of the wheels around the x_b-axis has not to be considered, because we make the assumption, that the wheels are always in contact with the ground. So, we just have to find dynamic equations for the rotation of the body. In Figure 5, we can see a back view of the body, where the forces of the front wheels are not drawn. The width of the body is denoted by b_{top} on the top, and by b_{bot} on the bottom. The suspensions cause forces in the lateral direction ($F_{y,sus,i}$), as well as in the vertical direction ($F_{z,sus,i}$).

The rotation of the body around the x_b-axis is influenced by the forces $F_{x,i}$, $F_{y,i}$, $F_{z,i}$, $F_{y,sus,i}$ and $F_{z,sus,i}$. Thus, the angular acceleration around the x_b-axis can be written as

$$J_x\ddot{\varphi}_x = M_{F_x,\varphi_x} + M_{F_y,\varphi_x} + M_{F_{y,sus},\varphi_x} + M_{F_z,\varphi_x} + M_{F_{z,sus},\varphi_x}. \tag{5}$$

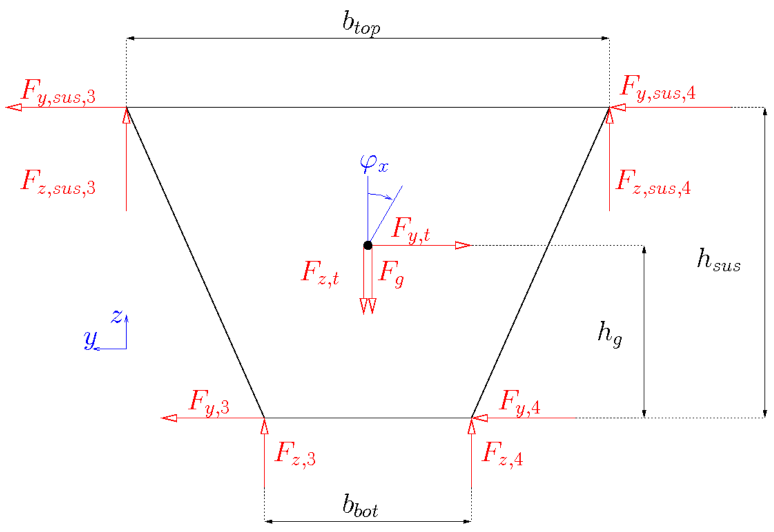


Figure 5: Back view of the body of the RC monster truck with the relevant forces.

The torques in Equation (5) can be evaluated by balances of moments according to Figure 5. We get

$$\begin{aligned}
M_{F_x,\varphi_x} &= \sin(\varphi_y)\Big(\tfrac{b_{bot}}{2}(F_{x,1} - F_{x,2} + F_{x,3} - F_{x,4})\Big),\\
M_{F_y,\varphi_x} &= h_g(F_{y,1} + F_{y,2} + F_{y,3} + F_{y,4}),\\
M_{F_{y,sus},\varphi_x} &= -(h_{sus} - h_{sp})(F_{y,sus,1} + F_{y,sus,2} + F_{y,sus,3} + F_{y,sus,4}),\\
M_{F_z,\varphi_x} &= \cos(\varphi_y)\Big(\tfrac{b_{bot}}{2}(F_{z,1} - F_{z,2} + F_{z,3} - F_{z,4})\Big),\\
M_{F_{z,sus},\varphi_x} &= \cos(\varphi_y)\Big(\tfrac{b_{top}}{2}(F_{z,sus,1} - F_{z,sus,2} + F_{z,sus,3} - F_{z,sus,4})\Big).
\end{aligned} \tag{6}$$

The influence of φ_x on the forces can be neglected, because the angle is assumed to be small.

Rotational Dynamics related to the y_b-Axis

The angular acceleration of the body around the y_b-axis is influenced by the forces $F_{x,i}$, $F_{z,i}$, $F_{z,sus,i}$ and F_{air} (see Figure 4). As above, we can write the angular acceleration as a sum of moments. This results in

$$J_y\ddot{\varphi}_y = M_{F_x,\varphi_y} + M_{F_z,\varphi_y} + M_{F_{z,sus},\varphi_y} + M_{F_{air},\varphi_y}. \tag{7}$$

The calculation of the torques is not trivial, because the lever arms of the forces depend on the angle φ_y. For the calculation of the torque M_{F_x,φ_y}, we define $h_{x,12}$ as the lever arm of the forces $F_{x,1}$ and $F_{x,2}$, respectively $h_{x,34}$ as the lever arm of the forces $F_{x,3}$ and $F_{x,4}$. Using these definitions, the moment M_{F_x,φ_y} can be written as

$$M_{F_x,\varphi_y} = h_{x,12}(F_{x,1} + F_{x,2}) + h_{x,34}(F_{x,3} + F_{x,4}), \tag{8}$$

where $h_{x,12} = f_{h_{x,12}}(\varphi_y)$ and $h_{x,34} = f_{h_{x,34}}(\varphi_y)$ are functions of the angle φ_y. Through geometrical considerations, the functions $f_{h_{x,12}}(\varphi_y)$ and $f_{h_{x,34}}(\varphi_y)$ can be established. The square of the maximum lever arm of the forces $F_{x,1}$ and $F_{x,2}$ is $l_f^2 + h_g^2$ (see Figure 4) and for $\varphi_y = 0$ the angular displacement of the lever arm is equal to $\arctan\left(\frac{h_g}{l_f}\right)$. Thus, the lever arm for the forces $F_{x,1}$ and $F_{x,2}$ can be written as

$$h_{x,12} = \cos\left(\arctan\left(\frac{h_g}{l_f}\right) + \varphi_y\right)\sqrt{l_f^2 + h_g^2}. \tag{9}$$

Using the same considerations, it can be shown that the lever arm of the forces $F_{x,3}$ and $F_{x,4}$ is given by

$$h_{x,34} = \cos\left(\arctan\left(\frac{h_g}{l_r}\right) - \varphi_y\right)\sqrt{l_r^2 + h_g^2}. \tag{10}$$

For the calculation of the lever arms for the forces $F_{z,i}$ and $F_{s,sus,i}$, we assume that the angular displacement of the maximum lever arm is zero and that the maximum lever arm is l_r respectively l_f. This assumption can be made, because the height of the center of gravity h_g is small, relative to the length of the body $l_r + l_f$. Hence, the torque M_{F_z,φ_y} is given by

$$M_{F_z,\varphi_y} = \cos(\varphi_y)\Big(l_r(F_{z,3} + F_{z,4}) - l_f(F_{z,1} + F_{z,2})\Big) \tag{11}$$

and the torque $M_{F_{z,sus},\varphi_y}$ can be written as

$$M_{F_{z,sus},\varphi_y} = \cos(\varphi_y)\Big(l_r(F_{z,sus,3} + F_{z,sus,4}) - l_f(F_{z,sus,1} + F_{z,sus,2})\Big). \tag{12}$$

The last moment to derive for the angular acceleration around the y_b-axis is M_{F_{air},φ_y}, which can be written as

$$M_{F_{air},\varphi_y} = h_{F_{air}} F_{air}, \tag{13}$$

where the lever arm $h_{F_{air}} = f_{h_{F_{air}}}(\varphi_y)$ also depends on the angle φ_y and can be calculated in the same way as the lever arms $h_{x,12}$ and $h_{x,34}$ before. This yields in

$$h_{F_{air}} = \cos\left(\arctan\left(\frac{h_{air} - h_g}{l_f}\right) - \varphi_y\right)\sqrt{l_f^2 + (h_{air} - h_g)^2}. \tag{14}$$

Rotational Dynamics related to the z-Axis

The part of the model for the rotational dynamics around the z-axis has to include the entire vehicle, because not only the body rotates in this direction, but also the wheels. Therefore, we consider an external view on the vehicle while we establish the following equations. According to Figure 3, the balance of moments yields in

$$J_z \ddot{\varphi}_z = M_{F_x,\varphi_z} + M_{F_y,\varphi_z}. \tag{15}$$

In this case, the lever arms are constant and the torque caused by the longitudinal forces $F_{x,i}$ is given by

$$M_{F_x,\varphi_z} = \frac{b_w}{2}(-F_{x,1} + F_{x,2} - F_{x,3} + F_{x,4}). \tag{16}$$

Further, the torque due to the lateral forces $F_{y,i}$ can be written as

$$M_{F_y,\varphi_z} = l_f(F_{y,1} + F_{y,2}) - l_r(F_{y,3} + F_{y,4}). \tag{17}$$

Wheel Dynamics

The rotational speed of the wheels highly influences the tire forces and that is why the wheel dynamics need to be modeled for a good simulation. Figure 6 shows a wheel from the side. $F_{r,i}$ denotes the rolling friction and has no influence on the rotational speed $\dot{\omega}_i$, because its point of application is the center of the wheel. The radius R of the wheel is not constant because of elastic deformation. The relevant radius for the lever arm of the longitudinal tire force $F_{l,i}$ is denoted by R_{eff}.

The only forces, which have influence on the rotational speed of the wheels, are the tire force $F_{l,i}$ in direction of the related wheel and the driving power force. The driving power is modeled as a torque $M_{w,i}$, which acts directly on the wheel. Hence, the differential equation for the angular wheel acceleration is

$$J_w \dot{\omega}_i = M_{w,i} - R_{eff} F_{l,i}. \tag{18}$$

Suspension Forces

The dynamic equations contain the suspension forces $F_{y,sus,i}$ and $F_{z,sus,i}$, so they have to be derived. The suspensions are modeled as an inelastic stick and a spring-damper element. Figure 8 shows the configuration of the suspension, the body and a wheel. The stick couples the related

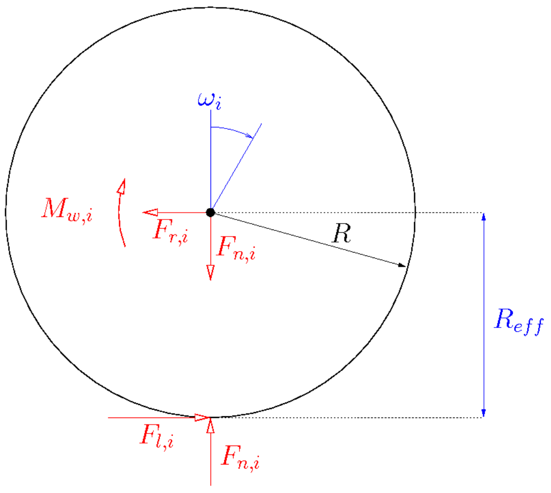


Figure 6: Side view of a wheel.

wheel with the body and the spring-damper element is mounted between the stick and the body. The elements are connected using swivel joints. The variables a and b in Figure 8 are auxiliary constants, which stand for the lateral difference between the top edge and the bottom edge of the body (a) and for the tilted height of the body (b).

The suspensions do not allow longitudinal departure between the wheels and the body, hence each suspension has one degree of freedom. Thus, the configuration of suspension i can be described by the angular displacement β_i of the concerning stick (see Figure 7).

The force of the spring-damper element $F_{sus,i}$, called suspension force, is a function of its length $l_{SD,i}$ and the derivative of this length. The force of the spring-damper element is given by

$$F_{sus,i} = c_s(l_{SD,i} - l_{SD,0}) + d_s \frac{d}{dt}\{l_{SD,i}\}, \tag{19}$$

where c_s denotes the spring constant and d_s stands for the damper constant. Thus, we have to derive an expression for the length of the spring-damper element $l_{SD,i}$. If the configuration β_i of the suspension is known, then the length $l_{SD,i}$ is given, too. So, we first derive an expression for the angle β_i. Using trigonometric functions, the angle β_i can be written as

$$\beta_i = \arcsin\left(\frac{\Delta h_i}{l_{sus,1} + l_{sus,2}}\right), \tag{20}$$

where Δh_i represents the vertical displacement of the point where the stick is joined with the body. In Figure 8, we can see the the forces that

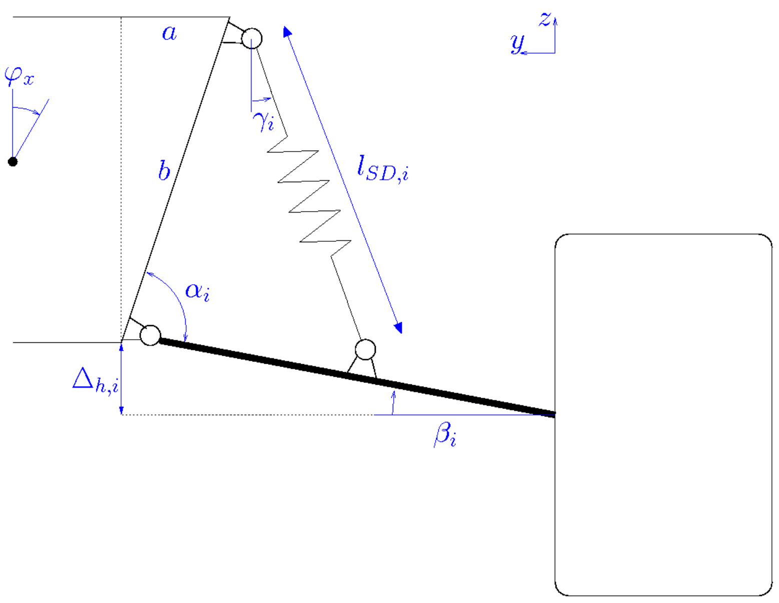


Figure 7: Suspension connecting the body and a wheel.

act on the suspension stick. The variables $l_{sus,1}$ and $l_{sus,2}$ define where the spring-damper element is connected to the stick.

The vertical displacement Δh_i of the joining point is a function of the variables z, φ_x and φ_y. The influence of the vertical displacement z of the center of gravity is linear, but the influence of the angles φ_x and φ_y is more complex. Since the angle φ_x is assumed to be small, we neglect its influence on Δh_i for simplification reasons. The influence of the angle φ_y is given by $-l_f \sin\varphi_y$ for the front wheels and by $l_r \sin\varphi_y$ for the rear wheels. Hence, the vertical displacement Δh_i of the joining point can be written as

$$\begin{aligned} \Delta h_i &= z - l_f \sin\varphi_y \qquad \text{for} \quad i = 1, 2, \\ \Delta h_i &= z + l_r \sin\varphi_y \qquad \text{for} \quad i = 3, 4. \end{aligned} \tag{21}$$

Using Equation (20), we are now able to determine the angle β_i and this allows us to calculate the angle between the side of the body and the stick of the suspension. We denote this angle by α_i (see Figure 7) and

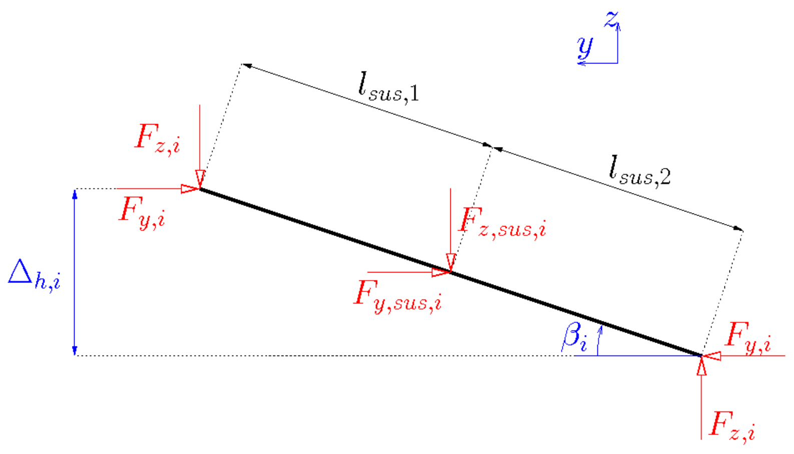


Figure 8: Forces on the stick of the suspension.

its value is given by

$$\begin{aligned} \alpha_i &= \beta_i + \varphi_x + \arctan\left(\frac{h_{sus}}{a}\right) \qquad \text{for} \quad i = 1,3, \\ \alpha_i &= \beta_i - \varphi_x + \arctan\left(\frac{h_{sus}}{a}\right) \qquad \text{for} \quad i = 2,4, \end{aligned} \tag{22}$$

where a is the auxiliary constant described above. The distance a can be written as

$$a = \frac{b_{top} - b_{bot}}{2}. \tag{23}$$

Using the law of cosine, we can finally calculate the length of the spring-damper element $l_{SD,i}$, what yields

$$l_{SD,i} = \sqrt{l_{sus,1}^2 + b^2 - 2l_{sus,1} b \cos\alpha_i}, \tag{24}$$

where the tilted height b is given by

$$b = \sqrt{h_{sus}^2 + a^2}. \tag{25}$$

The program Mathematica was used, to calculate the derivative of the length $l_{SD,i}$. According to Mathematica, the derivative of the length $l_{SD,i}$ of the spring-damper element is given by

$$\begin{aligned} \frac{d}{dt}\{l_{SD,i}\} &= \frac{l_{sus,1} d \sin\alpha_i}{\sqrt{l_{sus,1}^2 + d^2 - 2l_{sus,1} d \cos\alpha_i}} (\dot{\beta}_i + \dot{\varphi}_x) \qquad \text{for} \quad i = 1,3, \\ \frac{d}{dt}\{l_{SD,i}\} &= \frac{l_{sus,1} d \sin\alpha_i}{\sqrt{l_{sus,1}^2 + d^2 - 2l_{sus,1} d \cos\alpha_i}} (\dot{\beta}_i - \dot{\varphi}_x) \qquad \text{for} \quad i = 2,4. \end{aligned} \tag{26}$$

So, now we are able to determine the absolute force of the spring-damper element $F_{sus,i}$, but we need also the components $F_{y,sus,i}$ and $F_{z,sus,i}$. Therefore, the direction of the suspension force $F_{sus,i}$ has to be established. The angle between the normal direction of the ground and the spring-damper element defines the direction of the force $F_{sus,i}$ and we call this angle γ_i (see Figure 7). To determine the angle γ_i, the law of sines can be used. We get

$$\gamma_i = \arctan\left(\frac{h_{sus}}{c}\right) + \arcsin\left(\frac{l_{sus,1}}{l_{SD,i}} \sin\alpha_i\right) - \frac{\pi}{2}. \tag{27}$$

Based on the expressions which were derived above, we are now able to determine the components $F_{y,sus,i}$ and $F_{z,sus,i}$ of the suspension force $F_{sus,i}$. The lateral component $F_{y,sus,i}$ is given by

$$\begin{aligned} F_{y,sus,i} &= F_{sus,i} \sin\gamma_i \qquad \text{for} \quad i = 1, 3, \\ F_{y,sus,i} &= -F_{sus,i} \sin\gamma_i \qquad \text{for} \quad i = 2, 4, \end{aligned} \tag{28}$$

and the vertical component $F_{z,sus,i}$ can be written as

$$F_{z,sus,i} = -F_{sus,i} \cos\gamma_i. \tag{29}$$

Tire Forces

The local coordinate system of a wheel is distorted by the related steering angle δ_i from the vehicle coordinate system. Figure 9 shows the top view of a wheel. The forces $F_{x,i}$ and $F_{y,i}$ relative to the vehicle coordinate system are given by the forces $F_{l,i}$, $F_{c,i}$ and $F_{r,i}$ in the local coordinate system of the wheel and the steering angle δ_i.

To determine the tire forces, the velocities of the wheels relative to the ground in their local coordinate system are required. For the calculation of the relative velocities of the wheels, it is necessary to know the distances between the center the body and the center of the wheels, because the rotation of the vehicle around its center of gravity influences the relative velocities of the wheels. These distances are denoted by e_f for the front wheels and by e_r for the rear wheels. According to the theorem of Pythagoras, the distances are given by

$$e_f = \sqrt{\frac{b_w^2}{4} + l_f^2} \tag{30}$$

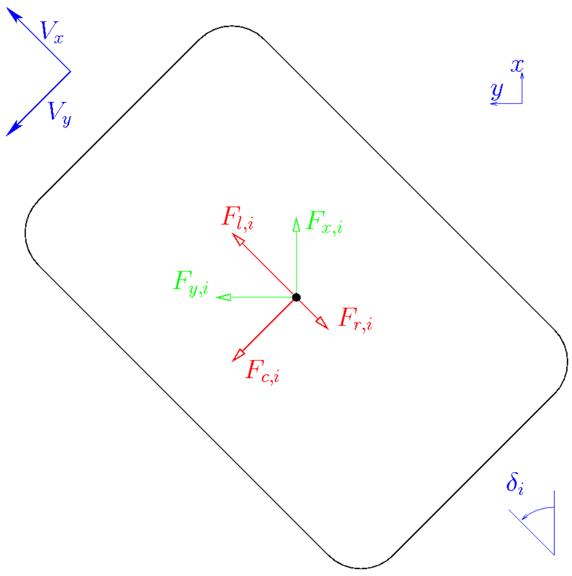


Figure 9: Top view of a wheel.

for the front wheels and

$$e_r = \sqrt{\frac{b_w^2}{4} + l_r^2} \tag{31}$$

for the rear wheels. Furthermore, the angle between the x-Axis and the line, connecting the center of gravity of the body and the center of the wheel, is required. This angle is denoted by $\gamma_{w,f}$ respectively $\gamma_{w,r}$ and can be determined using trigonometric functions. This results in

$$\gamma_{w,f} = \arctan\left(\frac{b_w}{2l_f}\right) \tag{32}$$

for the front wheels and

$$\gamma_{w,r} = \arctan\left(\frac{b_w}{2l_r}\right) \tag{33}$$

for the rear wheels. Thus, the velocities of the wheels relative to the ground in the vehicle coordinate system are given by

$$\begin{aligned}
v_{x,1} &= \dot{x} - e_f\dot{\varphi}_z \sin\gamma_{w,f}, \\
v_{x,2} &= \dot{x} + e_f\dot{\varphi}_z \sin\gamma_{w,f}, \\
v_{x,3} &= \dot{x} - e_r\dot{\varphi}_z \sin\gamma_{w,r}, \\
v_{x,4} &= \dot{x} + e_r\dot{\varphi}_z \sin\gamma_{w,r},
\end{aligned} \tag{34}$$

in the forward direction, i.e. in the direction of the x-axis. Further, we get

$$\begin{aligned}
v_{y,1} &= \dot{y} + e_f\dot{\varphi}_z \cos\gamma_{w,f}, \\
v_{y,2} &= \dot{y} + e_f\dot{\varphi}_z \cos\gamma_{w,f}, \\
v_{y,3} &= \dot{y} - e_r\dot{\varphi}_z \cos\gamma_{w,r}, \\
v_{y,4} &= \dot{y} - e_r\dot{\varphi}_z \cos\gamma_{w,r},
\end{aligned} \tag{35}$$

for the lateral direction. Considering the steering angles δ_i and using the Equations (34) and (35), the velocities of the wheels relative to the ground in their local coordinate system become

$$\begin{aligned}
V_{x,1} &= \cos\delta_1(\dot{x} - e_f\dot{\varphi}_z \sin\gamma_{w,f}) + \sin\delta_1(\dot{y} + e_f\dot{\varphi}_z \cos\gamma_{w,f}), \\
V_{x,2} &= \cos\delta_2(\dot{x} + e_f\dot{\varphi}_z \sin\gamma_{w,f}) + \sin\delta_2(\dot{y} + e_f\dot{\varphi}_z \cos\gamma_{w,f}), \\
V_{x,3} &= \cos\delta_3(\dot{x} - e_r\dot{\varphi}_z \sin\gamma_{w,r}) + \sin\delta_3(\dot{y} - e_r\dot{\varphi}_z \cos\gamma_{w,r}), \\
V_{x,4} &= \cos\delta_4(\dot{x} + e_r\dot{\varphi}_z \sin\gamma_{w,r}) + \sin\delta_4(\dot{y} - e_r\dot{\varphi}_z \cos\gamma_{w,r}),
\end{aligned} \tag{36}$$

in the longitudinal direction and

$$\begin{aligned}
V_{y,1} &= -\sin\delta_1(\dot{x} - e_f\dot{\varphi}_z \sin\gamma_{w,f}) + \cos\delta_1(\dot{y} + e_f\dot{\varphi}_z \cos\gamma_{w,f}), \\
V_{y,2} &= -\sin\delta_2(\dot{x} + e_f\dot{\varphi}_z \sin\gamma_{w,f}) + \cos\delta_2(\dot{y} + e_f\dot{\varphi}_z \cos\gamma_{w,f}), \\
V_{y,3} &= -\sin\delta_3(\dot{x} - e_r\dot{\varphi}_z \sin\gamma_{w,r}) + \cos\delta_3(\dot{y} - e_r\dot{\varphi}_z \cos\gamma_{w,r}), \\
V_{y,4} &= -\sin\delta_4(\dot{x} + e_r\dot{\varphi}_z \sin\gamma_{w,r}) + \cos\delta_4(\dot{y} - e_r\dot{\varphi}_z \cos\gamma_{w,r}),
\end{aligned} \tag{37}$$

in the lateral direction. The cornering forces $F_{c,i}$ are the forces generated by the wheels in the local lateral direction, i.e. in the direction of the velocity $V_{y,i}$. The slip angles of the tires are assumed to be small, because then the cornering force $F_{c,i}$ of each wheel is proportional to the slip angle. In this case, according to [9], the cornering force $F_{c,i}$ of a tire can be written as

$$F_{c,i} = C_\alpha \Big(\delta_i - \frac{V_{y,i}}{V_{x,i}} \Big), \tag{38}$$

where C_α represents the constant cornering stiffness of the tire. The longitudinal slip ratio is also assumed to be small, hence the longitudinal tire force $F_{l,i}$ is proportional to the slip ratio and can, according to [9], be written as

$$F_{l,i} = C_\sigma \Big(\frac{R_{eff}\omega_i - V_{x,i}}{R_{eff}\omega_i} \Big) \tag{39}$$

during acceleration and

$$F_{l,i} = C_\sigma \Big(\frac{R_{eff}\omega_i - V_{x,i}}{V_{x,i}} \Big) \tag{40}$$

during braking. The constant C_σ stands for the longitudinal tire stiffness. The elasticity of the tires causes the rolling friction force $F_{r,i}$, which can be calculated by

$$F_{r,i} = C_r F_{n,i} \text{sign}(V_{x,i}), \tag{41}$$

as shown in [9]. The rolling friction force $F_{r,i}$ has to be subtracted from the longitudinal tire force $F_{l,i}$. The forces generated by the tire in the vehicle coordinate system, denoted by $F_{x,i}$ and $F_{y,i}$, have now been derived and are given by

$$F_{x,i} = (F_{l,i} - F_{r,i}) \cos\delta_i - F_{c,i} \sin\delta_i \tag{42}$$

and

$$F_{y,i} = (F_{l,i} - F_{r,i}) \sin\delta_i + F_{c,i} \cos\delta_i. \tag{43}$$

Normal Forces

For the calculation of the normal forces on the tires, we neglect the influence of the angles φ_x and φ_y. Using this assumption, the normal forces can be written as

$$F_{n,i} = \frac{l_r}{2(l_f + l_r)} m_{tot} g + F_{dyn,i} \tag{44}$$

for the front wheels and

$$F_{n,i} = \frac{l_f}{2(l_f + l_r)} m_{tot} g + F_{dyn,i} \tag{45}$$

for the rear wheels. The dynamic terms $F_{dyn,i}$ are caused by the longitudinal acceleration $\ddot{x}$, the lateral acceleration $\ddot{y}$ and the aerodynamic drag F_{air}. The forces $F_{dyn,i}$ can be established with balances of moments on the vehicle x-axis and on the vehicle y-axis. This yields in

$$\begin{aligned} F_{dyn,1} &= -\tfrac{F_{air}h_{air}+m_{tot}\ddot{x}}{2(l_f+l_r)} - \tfrac{m_{tot}\ddot{y}}{2b_w}, \\ F_{dyn,2} &= -\tfrac{F_{air}h_{air}+m_{tot}\ddot{x}}{2(l_f+l_r)} + \tfrac{m_{tot}\ddot{y}}{2b_w}, \\ F_{dyn,3} &= \tfrac{F_{air}h_{air}+m_{tot}\ddot{x}}{2(l_f+l_r)} - \tfrac{m_{tot}\ddot{y}}{2b_w}, \\ F_{dyn,4} &= \tfrac{F_{air}h_{air}+m_{tot}\ddot{x}}{2(l_f+l_r)} + \tfrac{m_{tot}\ddot{y}}{2b_w}. \end{aligned} \tag{46}$$

The constant b_w stands for the width of the vehicle, i.e. for the distance between the center of two opposite wheels (see figure 3). To get the normal forces $F_{z,i}$ on the body, the weight of the wheels has to be subtracted from the normal forces $F_{n,i}$ on the tires. Thus, we get

$$F_{z,i} = F_{n,i} - m_w g. \tag{47}$$

Silicon Retina

Even if the silicon retina does not influence the dynamics of the truck, it is important to know its position and orientation to place it correctly in the virtual model. The silicon retina is rigidly coupled with the body of the truck at a constant height h_{sr}. The orientation has an angular displacement of φ_{sr} to the x_b-axis, while using y_b as the rotating axis.

3.3 Reduced Model for Controller Design

To get a realistic simulation in Blender, it was necessary to model all states which influence the view of the silicon retina considerably. But for controller design, a model consisting of less dynamic states is advantageous. Therefore, the complex model of the truck has to be simplified.

Reduced Longitudinal Dynamics

According to Equation (1), the longitudinal dynamics of the vehicle are

$$m_{tot}\ddot{x} = F_{x,1} + F_{x,2} + F_{x,3} + F_{x,4} - F_{air}. \tag{48}$$

F_{air} is the aerodynamic drag and can be calculated using Equation (2). The variables $F_{x,i}$ are the forces generated by the wheels in x-direction of the vehicle coordinate system. As derived before, they can be written as

$$F_{x,i} = (F_{l,i} - F_{r,i})\cos\delta_i - F_{c,i}\sin\delta_i, \tag{49}$$

where $F_{l,i}$ and $F_{c,i}$ represent the tire forces and $F_{r,i}$ is the rolling friction of the wheel i (see Equation (42)). δ_i stands for the steering angle of the relating wheel. Now, we make the assumption that we can set $F_{l,i}$ for each wheel, thus the torque on the wheel as input variable is replaced by the longitudinal tire force $F_{l,i}$. Using this assumption, we no longer have to model the rotational speed ω_i of the wheels and that reduces the order of the dynamic model. The cornering force of the wheels $F_{c,i}$ is given by

$$F_{c,i} = C_\alpha\left(\delta_i - \frac{V_{y,i}}{V_{x,i}}\right), \tag{50}$$

what has been established before. $V_{x,i}$ and $V_{y,i}$ are the absolute velocities of the wheels in their local coordinate system. Using Equations (36) and (37) and neglecting that the wheels are not on the longitudinal axis of the vehicle, we can simplify the expressions for the wheel velocities $V_{x,i}$ and $V_{y,i}$. We get the terms

$$\begin{aligned} V_{x,i} &= \dot{x}\cos\delta_i + (\dot{y} + l_f\dot{\varphi}_z)\sin\delta_i \qquad \text{for} \quad i = 1,2, \\ V_{x,i} &= \dot{x}\cos\delta_i + (\dot{y} - l_r\dot{\varphi}_z)\sin\delta_i \qquad \text{for} \quad i = 3,4, \end{aligned} \tag{51}$$

and

$$\begin{aligned} V_{y,i} &= -\dot{x}\sin\delta_i + (\dot{y} + l_f\dot{\varphi}_z)\cos\delta_i \qquad \text{for} \quad i = 1,2, \\ V_{y,i} &= -\dot{x}\sin\delta_i + (\dot{y} - l_r\dot{\varphi}_z)\cos\delta_i \qquad \text{for} \quad i = 3,4, \end{aligned} \tag{52}$$

for the wheel velocities in the local coordinate system. Thus, we can write the cornering forces $F_{c,i}$ as

$$\begin{aligned} F_{c,i} &= C_\alpha\left(\delta_i - \tfrac{-\dot{x}\sin\delta_i + (\dot{y}+l_f\dot{\varphi}_z)\cos\delta_i}{\dot{x}\cos\delta_i + (\dot{y}+l_f\dot{\varphi}_z)\sin\delta_i}\right) \qquad \text{for} \quad i = 1,2, \\ F_{c,i} &= C_\alpha\left(\delta_i - \tfrac{-\dot{x}\sin\delta_i + (\dot{y}-l_r\dot{\varphi}_z)\cos\delta_i}{\dot{x}\cos\delta_i + (\dot{y}-l_r\dot{\varphi}_z)\sin\delta_i}\right) \qquad \text{for} \quad i = 3,4. \end{aligned} \tag{53}$$

Now, we have established all equations of the reduced longitudinal vehicle model. We can rewrite the differential Equation (48) using Equation (49) and get the reduced longitudinal system dynamics:

$$\frac{d}{dt}\begin{Bmatrix} x \\ \dot{x} \end{Bmatrix} = \begin{Bmatrix} \dot{x} \\ \frac{1}{m_{tot}}\left[\sum_{i=1}^{4}\Big((F_{l,i} - F_{r,i})\cos\delta_i - F_{c,i}\sin\delta_i\Big) - F_{air}\right] \end{Bmatrix}. \tag{54}$$

Reduced Lateral Dynamics

According to Equation (3), the lateral acceleration can be described by

$$m_{tot}\ddot{y} = F_{y,1} + F_{y,2} + F_{y,3} + F_{y,4} - m_{tot}\dot{x}\dot{\varphi}_z. \tag{55}$$

The forces in y-direction of the vehicle coordinate system $F_{y,i}$ are given by Equation (43):

$$F_{y,i} = (F_{l,i} - F_{r,i})\sin\delta_i + F_{c,i}\cos\delta_i. \tag{56}$$

The equations for the forces $F_{l,i}$ and $F_{c,i}$ relative to the wheels have already been established for the reduced model in this section. Thus, we can directly write the differential equations for the reduced lateral dynamics as:

$$\frac{d}{dt}\begin{Bmatrix} y \\ \dot{y} \end{Bmatrix} = \begin{Bmatrix} \dot{y} \\ \frac{1}{m_{tot}}\left[\sum_{i=1}^{4}\Big((F_{l,i} - F_{r,i})\sin\delta_i + F_{c,i}\cos\delta_i\Big)\right] - \dot{x}\dot{\varphi}_z \end{Bmatrix}. \tag{57}$$

Reduced Rotational Dynamics

The only rotation we consider in the reduced model, is the rotation around the z-axis. As derived before, the angular acceleration is

$$J_z\ddot{\varphi}_z = M_{F_x,\varphi_z} + M_{F_y,\varphi_z}, \tag{58}$$

where the torque M_{F_x,φ_z} is given by

$$M_{F_x,\varphi_z} = \frac{b_w}{2}(-F_{x,1} + F_{x,2} - F_{x,3} + F_{x,4}) \tag{59}$$

and the torque M_{F_y,φ_z} can be written as

$$M_{F_y,\varphi_z} = l_f(F_{y,1} + F_{y,2}) - l_r(F_{y,3} + F_{y,4}). \tag{60}$$

To reduce the complexity of the model, we assume that the forces M_{F_x,φ_z} cause no torque on the vehicle, i.e. M_{F_x,φ_z} is not considered for the reduced model. This assumption can be made, because the truck used for this project provides no possibility to control each wheel particular. We assume that the truck is front driven and that the drive force of the left front wheel is always equal to the drive force of the right front wheel. The truck we use in this project is four wheel driven, but for the reduced model we can make the assumption that it is front driven. Thus, we avoid two additional forces and keep the model as simple as possible. Further, we consider that the steering angle of the rear wheels always equals zero in our model, i.e. the we can write $F_{y,i} = F_{c,i}$ for $i = 3, 4$. Using these assumptions and Equation (56) for the lateral forces $F_{y,i}$, the angular differential equations for the reduced model can be written as:

$$\frac{d}{dt}\begin{Bmatrix}\varphi_z \\ \dot{\varphi}_z\end{Bmatrix} = \begin{Bmatrix}\dot{\varphi}_z \\ \frac{1}{J_z}\Big[l_f \sum_{i=1}^{2}\Big((F_{l,i} - F_{r,i})\sin\delta_i + F_{c,i}\cos\delta_i\Big) - l_r\Big(F_{c,3} + F_{c,4}\Big)\Big]\end{Bmatrix}. \tag{61}$$

Combined Reduced Dynamics

To combine the reduced system equations, we define the state vector

$$\mathbf{x} = [x, \dot{x}, y, \dot{y}, \varphi_z, \dot{\varphi}_z]^T. \tag{62}$$

The goal is to get a differential equation for the system dynamics of the form

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \tag{63}$$

where $\mathbf{u}$ represents the input vector. The input vector contains the longitudinal wheel forces $F_{l,i}$ and the steering angles δ_i. As mentioned before, we assume that the vehicle is front driven and steered by the front wheels. This means that the longitudinal wheel forces and the steering angles for the rear wheels are always zero. Thus, the input vector can be written as

$$\mathbf{u} = [\delta_f, F_{l,f}], \tag{64}$$

where $\delta_1 = \delta_2 = \delta_f$ and $F_{l,1} = F_{l,2} = F_{l,f}$. Combining Equation (54), Equation (57) and Equation (61), the entire reduced system can be derived. Thus, the reduced system dynamics are:

$$\frac{d}{dt}\left\{\begin{array}{c} x \\ \dot{x} \\ y \\ \dot{y} \\ \varphi_z \\ \dot{\varphi}_z \end{array}\right\} = \left\{\begin{array}{c} \dot{x} \\ \frac{1}{m_{tot}}\left[\sum_{i=1}^{4}\Big((F_{l,i}-F_{r,i})\cos\delta_i - F_{c,i}\sin\delta_i\Big) - F_{air}\right] \\ \dot{y} \\ \frac{1}{m_{tot}}\left[\sum_{i=1}^{4}\Big((F_{l,i}-F_{r,i})\sin\delta_i + F_{c,i}\cos\delta_i\Big)\right] - \dot{x}\dot{\varphi}_z \\ \dot{\varphi}_z \\ \frac{1}{J_z}\left[l_f\sum_{i=1}^{2}\Big((F_{l,i}-F_{r,i})\sin\delta_i + F_{c,i}\cos\delta_i\Big) - l_r\Big(F_{c,3}+F_{c,4}\Big)\right] \end{array}\right\}. \tag{65}$$

Linearization of the Vehicle Dynamics

To create a simple controller to track a line with the truck, a linear state space model of the truck is required. This can be achieved by linearizing the system equations of the form $\dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u})$, to get equations of the form

$$\dot{\mathbf{x}} = \mathrm{A}\mathbf{x} + \mathrm{B}\mathbf{u}. \tag{66}$$

Hence, we have to linearize each column of the function $f(\mathbf{x},\mathbf{u})$ for each state. This can be done by using Mathematica to solve the complex derivatives. As mentioned before, the state vector is given by

$$\mathbf{x} = [y, \dot{x}, y, \dot{y}, \varphi_z, \dot{\varphi}_z]^T \tag{67}$$

and the linearization point is chosen as

$$\begin{aligned} \mathbf{x}_0 &= [0, \dot{x}_0, 0, 0, 0, 0]^T, \\ \mathbf{u}_0 &= [0, F_{l,f,0}]^T, \end{aligned} \tag{68}$$

where $\dot{x}_0$ is the steady state velocity for $\mathbf{u} = \mathbf{u}_0$. For the linearization, we assume static normal force distribution. Thus, the normal forces on the wheels are

$$\begin{aligned} F_{n,f} &= \tfrac{l_r}{2(l_f+l_r)} m_{tot} g, \\ F_{n,r} &= \tfrac{l_f}{2(l_f+l_r)} m_{tot} g, \end{aligned} \tag{69}$$

depending if the wheel is a front or a rear wheel. Calculating the linearization yields the system matrices A and B:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{c_a A \rho_{air} \dot{x}_0}{m_{tot}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{4C_\alpha}{m_{tot}\dot{x}_0} & 0 & \frac{2C_\alpha}{m_{tot}\dot{x}_0}(l_r - l_f) - \dot{x}_0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{2C_\alpha}{J_z \dot{x}_0}(l_r - l_f) & 0 & -\frac{2C_\alpha}{J_z \dot{x}_0}(l_r^2 + l_f^2) \end{bmatrix}, \tag{70}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & \frac{2}{m_{tot}} \\ 0 & 0 \\ \frac{2F_{l,f,0} - 2C_r F_{n,f} + 4C_\alpha}{m_{tot}} & 0 \\ 0 & 0 \\ \frac{2l_f(F_{l,f,0} - C_r F_{n,f} + 2C_\alpha)}{J_z} & 0 \end{bmatrix}.$$

Linearized Lateral Dynamics

Regarding the system matrix A in Equation (70), we can see that in the linear system the lateral dynamics are not coupled with the longitudinal dynamics. Thus, the longitudinal dynamics do not have to be considered for lateral controller design. The state vector can be reduced to

$$\mathbf{x} = [y, \dot{y}, \varphi_z, \dot{\varphi}_z]^T \tag{71}$$

and the system matrix A becomes

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{4C_\alpha}{m_{tot}\dot{x}_0} & 0 & \frac{2C_\alpha}{m_{tot}\dot{x}_0}(l_r - l_f) - \dot{x}_0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2C_\alpha}{J_z\dot{x}_0}(l_r - l_f) & 0 & -\frac{2C_\alpha}{J_z\dot{x}_0}(l_r^2 + l_f^2) \end{bmatrix}. \tag{72}$$

Further, we can see that the longitudinal wheel forces of the front wheels $F_{l,f}$ have no influence on the lateral system dynamics, thus the input vector can be reduced to

$$\mathbf{u} = \delta_f. \tag{73}$$

Under that condition, the input matrix B becomes

$$B = \begin{bmatrix} 0 \\ \frac{2F_{l,f,0} - 2C_r F_{n,f} + 4C_\alpha}{m_{tot}} \\ 0 \\ \frac{2l_f(F_{l,f,0} - C_r F_{n,f} + 2C_\alpha)}{J_z} \end{bmatrix}. \tag{74}$$

3.4 Parameter Identification

To identify the parameters of the model, the acceleration sensor was used. Therefore, a Matlab-function which simulates the complex model of the truck in Simulink was written. The function returns the difference between the simulated acceleration values and the real acceleration measurements, while we feed the same input values to the simulation as we used during the acceleration measurements on the real truck. To find the optimal set of parameters, the Matlab-function is minimized numerically by adjusting the parameters of the model in the simulation. In Figure 10, we can see a comparison between measured and simulated acceleration values, after the parameters of the model have been optimized.

Figure 10 shows that the acceleration measurements have a substantial variation of about 2 g when the car is driving, what reduces the accuracy of the parameter identification. A further drawback is that we have to identify a lot of parameters only with the acceleration measurements.

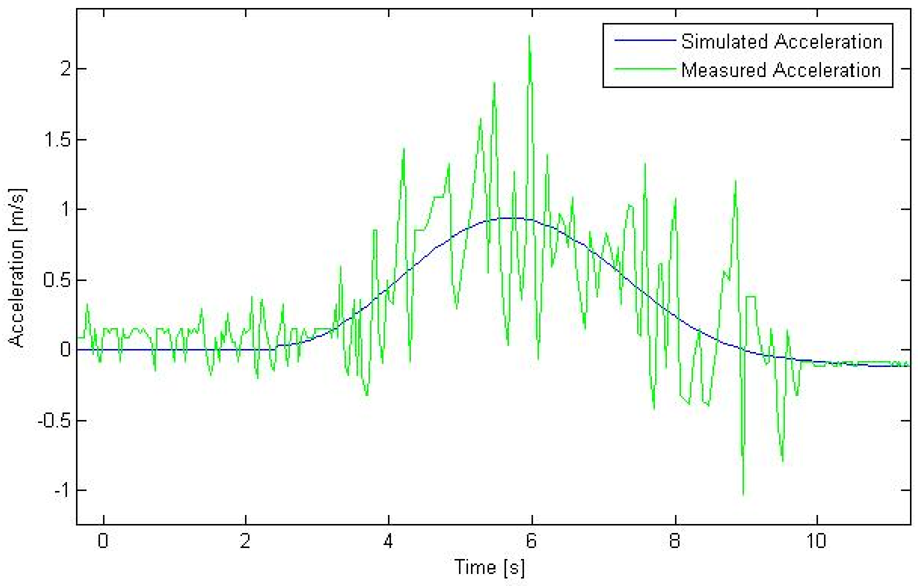


Figure 10: Measured acceleration (green line) and simulated acceleration (blue line) for a particular set of input values.

For that reason, some parameters have to be estimated and can hardly be verified using the acceleration sensor.

Furthermore, it has to be considered that some parameters of the truck are not constant, e.g. the drive torque depends on the battery power and the tire forces depend on the ground, on which the truck is driving.

On the attached CD, a list of the parameters and their measured, estimated or identified values can e found.

4 Simulation in Blender

In order to test a particular controller for the RC monster truck, a virtual three dimensional environment is employed, which allows to simulate the silicon retina. Therefore, a virtual model of the vehicle with nearly the same dynamics as the original has to be created. Blender, an open-source program for three dimensional modeling, was chosen as our test environment. Blender provides the possibility to integrate Python scripts, which allow to implement the complex dynamics of the truck.

4.1 Simulation Concept

The dynamics of the vehicle in the simulation should approximate the real behavior of the truck as good as possible, to get a realistic test environment. In Section 3, a complex model of the vehicle was derived. This model contains the relevant dynamics which influence the field of view of the silicon retina. Thus, if we are able to run the simulation based on this model, we have the possibility to test a particular controller for the truck including the visual part.

For a useful test of control strategies, it is not adequate to simulate only realistic dynamics, but it is also necessary to react accurately on the inputs and to create practical outputs. In our case, the inputs are the steering angle and the drive torques of the wheels. The influence of these inputs was also modeled in Section 3. The output of the simulation are the spikes recorded by the silicon retina in front of the vehicle. The spikes are sent to the jAER interface, which includes the visual filter and the controller (see Section 2). In Figure 11, the interaction between the simulation and the jAER interface is visualized.

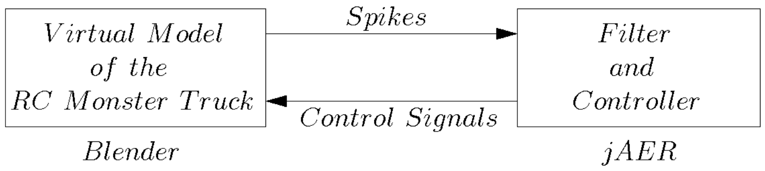


Figure 11: Schematic visualization of the interaction between the simulation in Blender and the Java-based jAER package.

4.2 Implementation

The dynamics of the vehicle are not linear and their calculation is numerically expensive. But the frequency of the calculation of the dynamic variables of the model should be as high as possible, so that the quality of the visual output is not affected by studdering. For that reason, we create a state mesh and linearize the complex model at each point of the mesh. Therefore, a Matlab script has been created, which allows to linearize a Simulink model at each point of the user defined state mesh. This provides also the advantage that it is not needed to change the Python code in the Blender file, when we want to change the model of the truck. It is only necessary to modify the Simulink model and, after executing the Matlab script, the simulation in Blender is adjusted to the changes. The Matlab script builds a text file for each linearization point and a folder containing the information about the used state mesh. Each text file comprehends the relating matrices A and B, which describe the concerning linear system. The folder for the state mesh contains a text file for each state with information about the resolution and the range of the related state. All text files are labeled in a defined format, so that the Python scripts can access the files without manual modifications. Figure 12 visualizes the handling of the linearized models.

The dynamics, which are simulated and visualized using the strategy described above, are the motions in all three directions, the rotations around all axes and the rotations of the wheels, everything relative to the local coordinate system of the vehicle. The simulation in Blender is based on eight separate Python scripts, which are explained in the following.

Python Script ‘Initialize’

This Python script sets up the simulation environment and initializes the global variables. Global variables are the current values of the dynamic states and their current derivatives, the resolution and the range of the state mesh, and some other factors and pointers which are necessary for the simulation. The resolution and the range of the state mesh are calculated using the text files created by the Matlab code. This Python script is executed once when the simulation starts.

Python Script ‘Dynamics’

The Python script for the dynamics handles the motions of the virtual vehicle. First, the new values of the dynamic states are calculated us-

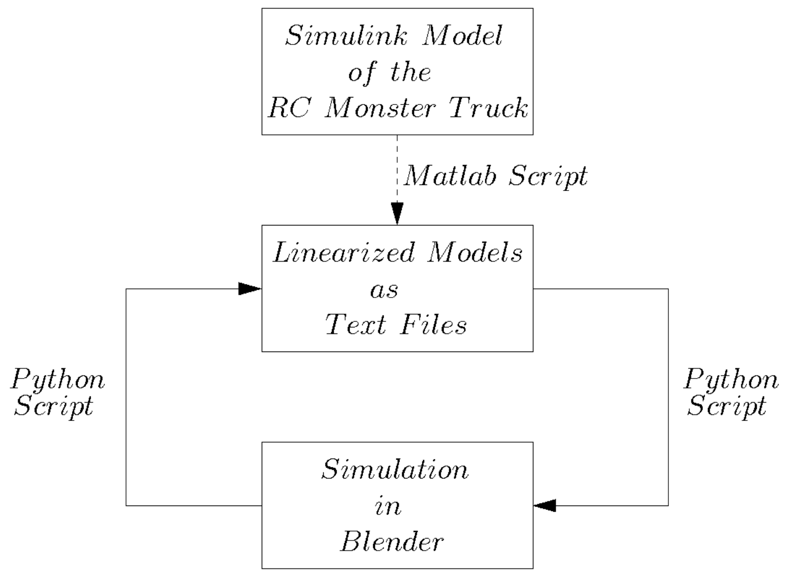

Simulink Model of the RC Monster Truck
Matlab Script
Linearized Models as Text Files
Python Script
Python Script
Simulation in Blender

Figure 12: Schematic visualization of the simulation. The Matlab script builds the text files containing the linearized models and the Python scripts access the text files to get the linearized model relating to the current values oft the dynamic states.

ing an Euler discretization. The derivative of the dynamic states are calculated at the end of this script and saved as global variables. After that, the new values of the states are applied to the three dimensional model. Then the inputs, i.e. the steering angle and the drive torque of the front wheels, are updated, either according to the manual input of the keyboard or according to the control signal coming from the external controller. The influence of manual steering will be introduced later. Based on the new inputs and on the current values of the dynamic states, the script reads in the new system matrix A and the new input matrix B. Finally, the current derivatives of the states are calculated using the loaded matrices and saved as global variables for the next call of this script. The script for the dynamics is executed every logic tick, and the time between two logic ticks is set to 15 ms in our simulation.

Python Script 'Corrections'

Because the dynamics are simulated numerically and visualized in parallel, some states drift away, i.e. the calculated values do not correspond with the visualization. For example the vertical position z drifts away and some time after the start of the simulation, the truck seems to fly or to drop into the ground. For that reason, the python script for the corrections is called once in ten game logic ticks. The script reads in the drifting states from the three dimensional model and uses these values to adjust the numerical simulation. Another problem is that the wheels tend to drift away from the body of the truck during the simulation, also because of numerical errors. The Python script for the corrections handles this problem as well.

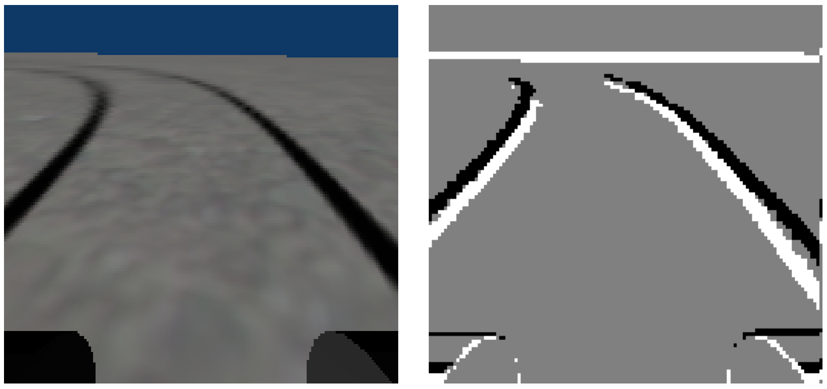
Python Script ‘Settings’

During the simulation, it is possible to change between manual control and automatic control. Manual control means that the virtual vehicle is controlled by the keyboard arrows, while automatic control denotes that the vehicle is controlled by the external controller implemented in Java. Thus, we need two settings, one for steering and one for speed control. Both of these two options can be changed independently from manual to automatic and vice versa. In the integrated command line of Blender, the inputs from the controller are always displayed during the simulation, even if the manual control mode is activated. So, it is possible to control the virtual truck manually and to observe at the same time, what the controller would do in the current situation. The Python script for the settings is executed, whenever the user pushes one of the keys to change the steering or the speed control mode, and handles the changing.

Python Script ‘Camera Feeder’

This script simulates the output of the silicon retina, i.e. it creates the spikes. For that, a virtual camera is mounted in the front of the truck, at the same place where the silicon retina is located on the original vehicle. The view of the camera is captured and used to extract spikes. Therefore, the current view of the virtual camera is rasterized and each element of the raster is analyzed regarding to its brightness. If the changing of the brightness, compared to the last execution of the script, raises above a particular value, then a spike is created for the related element of the raster. Figure 13 shows a screen shot of the virtual camera in the simulation, and the associated spikes, which have been created by the Python script. This script is based on a script published by Albert Cardona in [4] and has been adapted for our simulation.

To send the spikes to jAER, a socket is set up, which provides the possibility to send a continuous stream to the jAER client. Through this stream, the spikes are transmitted to the filter and the external



(a) Screen shot of the three dimensional simulation in Blender.

(b) Spikes which are generated at the same time of the screen shot.

Figure 13: Comparison between a screen shot of the simulation in Blender (a) and the related spikes, created by a Python script (b).

controller, which are implemented in Java. This script handles also the inputs, which are transmitted through the socket stream as well, coming from the controller. The script decodes the input stream into the desired steering angle and the desired drive torque. Because the execution of this script is expensive in terms of calculation steps, it is only called every few game logic ticks, but the flow of the spikes is still sufficient.

Python Script 'Capture Route'

To analyze the performance of a particular controller, it is advantageous to capture the driven path. Therefore, this Python script captures a trajectory which is driven during the simulation. The captured path, which consists of the position and the orientation of the truck for each game logic tick, is saved in a text file. This script is called every game logic tick, if the route capturing is activated.

Python Script 'Rerun Captured Route'

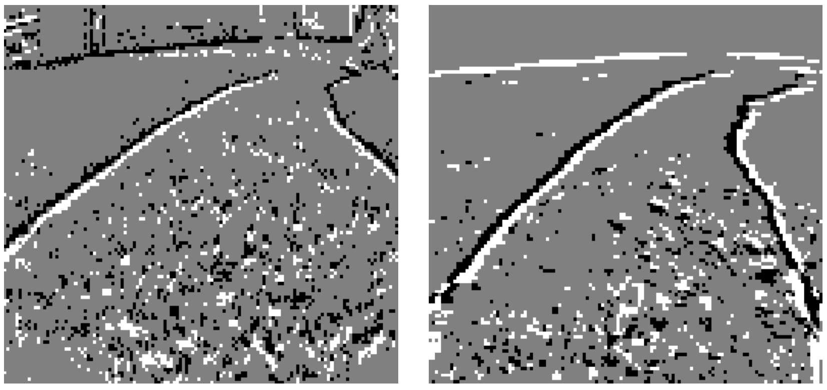
This script allows to rerun a trajectory, which has been recorded before. If a captured path is driven again, then it is possible to switch between different views, to analyze the performance of the controller. This script is called every game logic tick, if the simulation is in the rerun mode.

Python Script ‘Get Captured Route’

To test a controller without the influence of the filter which handles the spikes, this Python script determines the lateral and the angular errors relative to a reference route and feeds them to the controller. Thus, we assume that the filter of the silicon retina always recognizes the route perfectly. The reference route can either be captured by the Python script ‘Capture Route’, or be generated by a program, for example Matlab.

4.3 Simulation Results

Based on the implementation introduced above, a realistic simulation of the RC monster truck can be performed, provided that we have a good model of the truck. The simulated output of the silicon retina looks as expected. The flow of the spikes created in Blender during the simulation is very similar to the flow of the spikes resulting from the real silicon retina, except that the timestamps of the events are quantized to the simulation frame rate. Depending on the texture of the ground in the simulation, we are able to create more or less noise in the simulated output of the silicon retina. Hence, the simulation provides the possibility to test a controller under idealized conditions without noise, as well as the possibility to test a controller under conditions which are similar to reality. Figure 14 shows a comparison between spikes which were simulated and spikes which were recorded using the silicon retina on the truck. In Blender, a texture with high contrast differences was chosen, what causes noise in the virtual output of the silicon retina.



(a) Screen shot of a recorded output of the silicon retina.

(b) Screen shot of the simulated output of the silicon retina.

Figure 14: Comparison between recorded spikes (a) and simulated spikes (b).

It has to be mentioned that some manual adjustments in the system matrices are necessary, after the Simulink model of the truck has been linearized at each point of the state mesh. The reason for that is the fact that the used Simulink model fails if the car does not move, because of zero divisions.

On the attached CD, some videos of the simulation can be found, including different perspectives and the output of the visual filter using simulated spikes.

5 Controller Design

In this section, a controller for the lateral position of the truck is designed, based on the reduced lateral model of the system dynamics. We assume that the silicon retina is the only available sensor and its output is filtered to get a measurement signal. The measurement signal contains the lateral departure of the truck relative to the middle of the route and the angular displacement of the vehicle relative to the direction of the route.

5.1 PID Control Scheme

The first approach we consider is to use a standard PID controller for the lateral control of the truck.

Approach

A PID controller consists of three parts. The proportional part multiplies the error by a constant factor, where the error is the current departure to the setpoint value. The integral part integrates and the differential part differentiates the error and the sum of all these parts is fed to the plant. Figure 15 shows a schematic PID controller consisting of the proportional, the integral and the differential part.

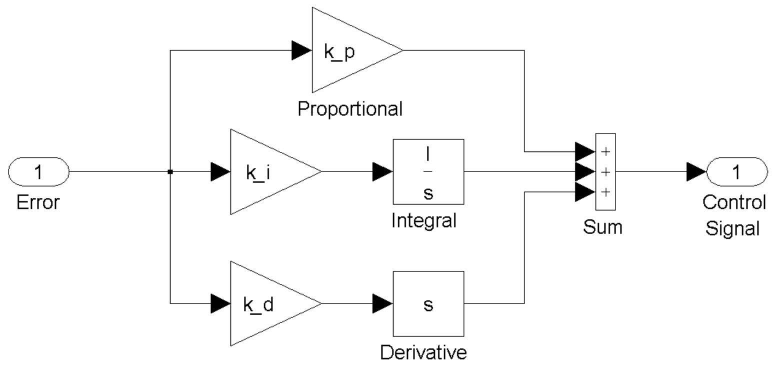


Figure 15: Schematic visualization of a PID controller.

To determine the range of the factors k_p, k_i and k_d, which weight the three parts of the PID controller, the method of Ziegler and Nichols, as described in [7], was applied in the simulation.

Implementation

A PID controller has one input and one output. The truck has one input, which is the steering angle, and two outputs, which are the lateral and the angular displacement of the vehicle relative to the route. Since the PID controller can only handle one input, we combine the two outputs of the truck to feed them to the PID controller. Therefore, the two outputs of the truck are added after they have been weighted by constant factors. This approach restricts the control performance because the controller reacts in the same way on both outputs of the truck, aside from the different weighting factors. But at the current state of the project, this restriction does not influence the performance of the truck considerably. Since the outputs of the vehicle do not represent absolute values of system states, but lateral and angular errors, we can feed them directly to the PID controller. In Figure 16, we can see a feedback system using a PID controller. The constants K_1 and K_2 are the weighting factors of the lateral and the angular error and have to be determined manually.

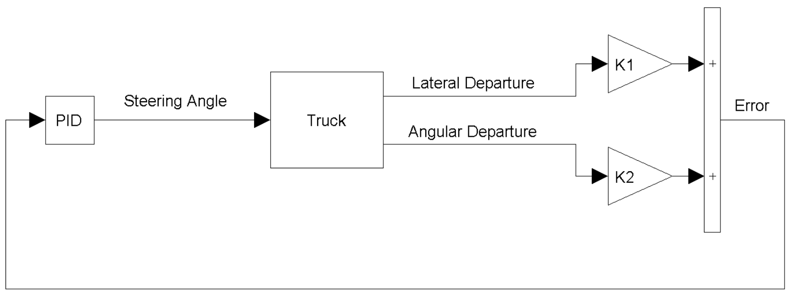


Figure 16: Feedback system with a PID controller.

Because we add the lateral and the angular error, we have to avoid that they cancel themselves. But in our case, there is no steady state, in which the errors cancel, because the lateral error changes, if the angular error is not zero. The only situation, in which the sum of the errors becomes zero, is if the vehicle moves towards the route from the side on a particular angle, which depends on the weighting factors K_1 and K_2. But this causes no problems, because it does not make the system unstable and the only steady state occurs if the lateral as well as the angular error equals zero.

5.2 LQR Control Scheme

The second approach we consider is to track the route using an LQR controller.

Approach

An LQR controller minimizes the cost function

$$J(\mathbf{u}) = \int_0^\infty \mathbf{x}(t)^T Q \mathbf{x}(t) + \mathbf{u}(t)^T R \mathbf{u}(t) dt \tag{75}$$

by choosing an adequate control signal $\mathbf{u}(t)$. Q is a quadratic weighting matrix of the system states $\mathbf{x}$ and R penalizes the control effort. In [5], it is shown that the cost function defined in Equation (75) is minimized if the control signal has the form

$$\mathbf{u}(t) = -K\mathbf{x}(t), \tag{76}$$

where K is a constant matrix. According to [5], the matrix K is given by

$$K = R^{-1} B^T \Phi, \tag{77}$$

where Φ can be determined by solving the Algebraic Ricatti Equation

$$\Phi B R^{-1} B^T \Phi - \Phi A - A^T \Phi - Q = 0. \tag{78}$$

The system matrices A and B of the lateral model of the truck were derived in Section 3.

Implementation

Since we are not able to measure all four states of the lateral model of the truck, we have to add an observer to the control loop, which determines the system states based on the outputs of the truck. Figure 17 shows the control loop using an LQR controller including the state observer.

A common approach to determine the system states is to simulate the dynamics of the plant in the observer. But since the silicon retina is event based, the calculation of the integrators in the simulated plant is problematic, because the sampling rate depends on the number of spikes, which are currently generated. Therefore, we use another approach to determine the system states.

Since the states y and φ_z do not influence the system dynamics, we can assume that the position and the orientation of the route always equal

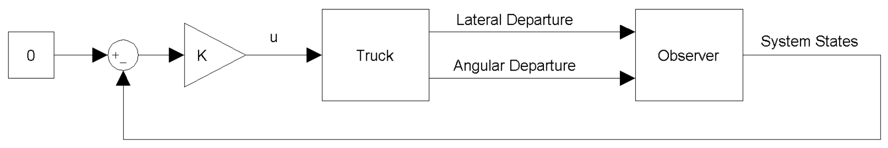


Figure 17: Feedback system with an LQR controller. The control signal u contains only the steering angle, if just the lateral model of the truck is used.

zero in the vehicle coordinate system, i.e. the setpoint values are always zero. Since the output of the truck gives us the lateral and the angular error, the output equals y respectively φ_z, if the setpoint values are zero. Thus, we are able to determine the system states y and φ_z. To calculate the system states $\dot{y}$ and $\dot{\varphi}_z$, we differentiate the known system states y and φ_z. Usually, differentiate system outputs is problematic, because the noise of the sensors is amplified. But in our case, the output of the system comes from the filter of the silicon retina and has been processed before, so we are not considered with sensor noise. Further, we assume that the route is smooth, so there are no steps in the filter output, provided that the filter does always recognize the route correctly.

5.3 Experimental Results

On the attached CD are some videos of the experimental results in the simulation and in reality. In all videos, the truck was steered by the proportional part of the PID controller. In the simulation, the speed was set to a constant value of about 7 kph during line tracking and about 4 kph during lane tracking. On the real truck, the acceleration was controlled manually and the car drove at a velocity of a little more than walking pace.

6 Conclusion

The result of this semester project is a nonlinear model of the RC monster truck and a reduced linear model for the controller design. Further, a simulation environment in Blender has been developed, to simulate the behavior of the truck including the silicon retina. Using this simulation environment, control strategies to track a route with the RC monster truck can be tested.

Two first control strategies, a PID and an LQR controller, were implemented in jAER and tested in Blender as well as in reality. On the real vehicle, some problems occurred, e.g. that the vehicle dynamics do not only depend on the velocity, but also on the battery power, and that there is too much noise under some terrain and light conditions, so that the visual filter is not able to handle all the spikes. Regarding to the controller design, it has to be considered that the sampling rate is not constant, what complicates the handling of virtual integrators.

In the simulation in Blender under idealized conditions without noise, both controllers provided good results, i.e. the controllers were able to track a line at a velocity of about 7 kph and a lane at a velocity of about 4 kph.

On the real vehicle, the observer for the LQR controller failed because of the inconstant sampling rate. The best results were achieved by using just the proportional part of the PID controller. In this case, the truck was able to follow a line at a little more than walking pace or to follow a lane if it is pushed manually.

Future issues related to this project are to develop a method to handle the varying sampling rate of the silicon retina robustly and to establish good filter and control parameters, which may depend on the current conditions. Further, considering the acceleration sensor for the controller design and using cascaded control loops could improve the performance of the control strategies.

Another future work is to install additional sensors for a more precisely identification of the model parameters. At the current state of the project, the imprecise model parameters do not yet influence the performance of the controller considerably, because there are other more fundamental problems to solve. But in the future when more complex controllers will be applied and a simulation very near to the reality is needed, then the identification of the parameters has to be preciser.

Acknowledgement

I would like to thank Christian Brändli for the nice collaboration during this project. My thanks go also to Tobi Delbrück and Thomas Besselmann for the supervision and the support and to Patrick Lichtsteiner for the construction of a new camera mount.

List of Figures

Attachments

The attached CD contains several digital files of this project. In addition to the documentation of the project, there are five folders on the CD. The content of the folders is introduced in the following.

Folder ‘jAER’

This folder contains the controller, which is called FancyDriver and implemented in Java.

Folder ‘Matlab’

In this folder the Matlab-files are stored, i.e. the Simulink model of the truck and the Matlab script to linearize the model for the simulation.

The Readme-file in this folder gives more information about the handling of the Matlab scripts.

Folder ‘Presentation’

This folder contains the PowerPoint-presentation of our work and the videos which were shown during the presentation.

Folder ‘Simulation’

This folder includes all files which are needed for the simulation. The sub-folder ‘Blender’ contains the Blender-file for the simulation (Truck.blend) and Blender version 2.45 for Windows, since the simulation works only in this version. The sub-folder ‘PythonScripts’ contains the scripts, which were written for the simulation.

For further instructions relating to the simulation, take a look at the Readme-file, which is placed in this folder.

Folder ‘Videos’

In this folder some videos of test runs in the simulation as well as in reality are stored. In all videos, the steering angle of the truck is controlled by the PID controller using just the proportional part. In Blender, the velocity of the truck is set to a constant value and in the videos of the real truck, the acceleration is controlled manually.

References

[1] http://jaer.wiki.sourceforge.net.

[2] http://siliconretina.ini.uzh.ch.

[3] http://www.blender.org.

[4] Albert Cardona. Blender's game engine as a 3d environment simulator for external programs. *blender conference*, 2007.

[5] H. P. Geering. *Regelungstechnik*. Springer Berlin, 2001.

[6] L. Guzzella. Modeling and analysis of dynamic systems, 2005/2006. Lecture Notes at ETH Zurich.

[7] L. Guzzella. *Analysis and Synthesis of Single-Input Single-Output Control Systems*. vdf, September 2007.

[8] The MathWorks. Matlab. http://www.mathworks.com.

[9] R. Rajamani. *Vehicle Dynamics and Control*. Springer, 2005.

[10] Wolfram Research. Mathematica. http://www.wolfram.com.