

Inhaltsverzeichnis

Tabellen und Abbildungsverzeichnis.....	iii
Abkürzungsverzeichnis.....	iv
Verwendete Formelzeichen	v
Zusammenfassung (Abstract).....	vii
1. Einführung.....	1
1.1 Motivation und Zielsetzung	1
1.2 Aufbau der Arbeit.....	1
2. Physikalische Grundlagen	3
2.1 Gleichförmige Bewegung.....	3
2.2 Gleichförmig beschleunigte Bewegung	5
2.3 Kreisbewegung und Zentripetalkraft.....	6
2.4 Reibung.....	8
2.5 Die schiefe Ebene	10
2.6 Trägheitsmoment	11
2.7 Kontinuierliche Simulation.....	12
2.8 Numerische Integration nach Euler.....	13
3. Carrerabahn Aufbau	15
3.1 Segmente	15
3.2 Sensoren	17
4. Treiber.....	18
4.1 Anpassungen.....	18
5. Modell- und Simulationslösungen.....	20
5.1 Radumfangsschlupf	20
5.2 Lineares Reifenmodell	21
5.3 Reifenmodell nach Pacejka	22
5.4 Lineares Einspurmodell.....	22
5.5 Nichtlineares Einspurmodell.....	25
5.6 Nichtlineares Zweispurmodell.....	28
5.7 Einfaches Modell für den Schwimmwinkel.....	32
5.8 Winkelbestimmung durch neuronales Netz	36
5.9 Driftsensor als Wertetabelle	39

6.	Simulation	40
6.1	Betriebssystem	40
6.2	Programmiersprache und Entwicklungsumgebung	40
6.3	Verwendete Bibliotheken	42
6.4	Genereller Ablauf	43
6.5	Verarbeitung der Fahrzeugdaten	43
6.6	Verarbeitung der Streckendaten	45
6.7	Kommunikation Simulation – Treiber	47
6.8	Kommunikation Simulation - Steuerprogramm	48
7.	Dateien & Strukturen	52
7.1	HTWG_schienen.xml	52
7.2	61silber.xml	53
7.3	htwg.xml	54
7.4	struct carData	55
7.5	struct threadParam	56
7.6	struct vAccelerate	56
7.7	struct vDecelerate	56
7.8	struct speed	57
7.9	struct sSensorData	57
7.10	struct sTrackData	58
7.11	struct sSegData	59
7.12	struct sSlideData	59
	Ehrenwörtliche Erklärung	62
	Danksagung	63

Tabellen und Abbildungsverzeichnis

Abbildung 1: Gleichförmige Bewegung.....	4
Abbildung 2: Gleichförmig beschleunigte Bewegungen	5
Abbildung 3: Konstante Kreisbewegung	6
Abbildung 4: Reibung.....	8
Abbildung 5: Reibungskoeffizienten	9
Abbildung 6: Kräfte an der schiefen Ebene	10
Abbildung 7: Aufbauplan der Carrerabahn	16
Abbildung 8: Seitenführungskraft in Abhängigkeit des Schwimmwinkels.....	21
Abbildung 9: Lineares Einspurmodell	23
Abbildung 10: Nichtlineares Zweispurmodell (Horizontaldynamik).....	29
Abbildung 11: Nichtlineares Zweispurmodell (Nickdynamik)	29
Abbildung 12: Nichtlineares Zweispurmodell (Wankdynamik)	29
Abbildung 13: Kräfte bei der Kurvenfahrt mit Übersteuern	34
Abbildung 14: Neuronales Netz mit einer versteckten Zwischenschicht.....	37
Abbildung 15: Funktion logsig in MATLAB	38
Abbildung 16: Aufteilung der Daten	38
Abbildung 17: Aufbau der Simulation.....	43
Abbildung 18: Strukturzusammenhang für ein Auto der Carrerabahn.....	44
Abbildung 19: Kommunikation Steuerprogramm – Simulation	47
Abbildung 20: Kommunikation Simulation – Treiber.....	48

Abkürzungsverzeichnis

ESP	Elektronisches Stabilitätsprogramm
ASR	Anti Schlupf Regelung
ABS	Anti Blockier System
XML	Extensible Markup Language
ALU	Aluminium
Leg.	Legierung
DGL	Differentialgleichung
HTWG	Fachhochschule Konstanz
S1	Strecke 1
S2	Strecke 2
S3	Strecke 3
S4	Strecke 4
MKS	Mehr-Körper-System
FEM	Finite-Elemente-Model
KDE	K Desktop Environment
gcc	GNU Compiler Collection
gdb	GNU Debugger
SVN	Subversion
CVS	Concurrent Versions System
Qt	Quasar Toolkit
URL	Uniform Resource Locator
GNU	GNU's Not UNIX
LGPL	Library / Lesser General Public License
GPL	General Public License
usw	Und so weiter
z.B.	zum Beispiel
DIN	Deutsche Industrienorm

Verwendete Formelzeichen

\vec{v}	[m/s]	Geschwindigkeit
\vec{v}_0	[m/s]	Anfangsgeschwindigkeit
\vec{a}	[m/s ²]	Beschleunigung
s	[m]	Strecke/Weg
t	[s]	Zeit
F	[N]	Resultierende Kraft
\vec{F}_N	[N]	Normalkraft
\vec{F}_H	[N]	Haftreibungskraft
\vec{F}_K	[N]	Gleitreibungskraft
\vec{F}_{GH}	[N]	Hangabtriebskraft
\vec{F}	[N]	Kraft
μ_H	[-]	Haftreibungskoeffizient
μ_R	[-]	Rollreibungskoeffizient
μ	[-]	Gleitreibungskoeffizient
\vec{g}	[m/s ²]	Beschleunigung im Gravitationsfeld der Erde
F_{SV}	[N]	Seitenführungskraft am Vorderrad
F_{SH}	[N]	Seitenführungskraft am Hinterrad
a_y	[m/s ²]	Querbeschleunigung
a_x	[m/s ²]	Horizontalbeschleunigung
F_X	[N]	Resultierende Kraft in X-Richtung
F_Y	[N]	Resultierende Kraft in Y-Richtung
F_Z	[N]	Resultierende Kraft in Z-Richtung
J_X	[kg/m ²]	Trägheitsmoment der Wankbewegung
J_Y	[kg/m ²]	Trägheitsmoment der Nickbewegung
J_Z	[kg/m ²]	Trägheitsmoment der Gierbewegung
M_X	[Nm]	Wankmoment
M_Y	[Nm]	Nickmoment
M_Z	[Nm]	Giermoment
ψ	[Grad]	Gierwinkel
ϕ	[Grad]	Wankwinkel
θ	[Grad]	Nickwinkel
l_v	[m]	Abstand vom Schwerpunkt zum Vorderrad
l_h	[m]	Abstand vom Schwerpunkt zum Hinterrad
l	[m]	Spurabstand
l_a	[m]	Länge des Autos
b_a	[m]	Breite des Autos
c_a	[m]	Höhe des Autos
r	[m]	Kurvenradius
α_v	[Grad]	Schräglaufwinkel vorn
α_h	[Grad]	Schräglaufwinkel hinten
c_v	[N/rad]	Schräglaufsteifigkeit vorn
c_h	[N/rad]	Schräglaufsteifigkeit hinten
δ	[Grad]	Lenkwinkel
δ_v	[Grad]	Lenkwinkel vorn

δ_h	[Grad]	Lenkwinkel hinten
β	[Grad]	Schwimmwinkel
ω	[rad/s]	Winkelgeschwindigkeit
α	[rad/s ²]	Winkelbeschleunigung
φ	[Grad]	Drehwinkel
ρ	[kg/m ³]	Dichte
a_T	[m/s ²]	Tangentialbeschleunigung
a_R	[m/s ²]	Radialbeschleunigung
a_Z	[m/s ²]	Zentripetalbeschleunigung
F_Z	[N]	Zentripetalkraft
F_{XVL}	[N]	Kraft in X-Richtung am linken Vorderrad
F_{XVR}	[N]	Kraft in X-Richtung am rechten Vorderrad
F_{XHR}	[N]	Kraft in X-Richtung am rechten Hinterrad
F_{XHL}	[N]	Kraft in X-Richtung am linken Hinterrad
F_{ZVL}	[N]	Kraft in Z-Richtung am linken Vorderrad
F_{ZVR}	[N]	Kraft in Z-Richtung am rechten Vorderrad
F_{ZHL}	[N]	Kraft in Z-Richtung am linken Hinterrad
F_{ZHR}	[N]	Kraft in Z-Richtung am rechten Hinterrad
F_{YVL}	[N]	Kraft in Y-Richtung am linken Vorderrad
F_{YVR}	[N]	Kraft in Y-Richtung am rechten Vorderrad
F_{YHL}	[N]	Kraft in Y-Richtung am linken Hinterrad
F_{YHR}	[N]	Kraft in Y-Richtung am rechten Hinterrad
F_{AH}	[N]	Auftriebskraft hinten
F_{AV}	[M]	Auftriebskraft vorn
b_V	[m]	Abstand der beiden Vorderräder
b_H	[m]	Abstand der beiden Hinterräder
l_V	[m]	Abstand vom Schwerpunkt zur Vorderachse
l_H	[m]	Abstand vom Schwerpunkt zur Hinterachse
h_{SP}	[m]	Höhe des Schwerpunkts
h_{WA}	[m]	Höhe der Wankachse
h_{NA}	[m]	Höhe der Nickachse
S_X	[-]	Umfangsschlupf
μ_X	[-]	Umfangskraftbeiwert
μ_y	[-]	Seitenkraftbeiwert

Zusammenfassung (Abstract)

Thema:	Carrerabahn - Implementierung einer Simulation zur Verifizierung von Echtzeitprogrammen für die Steuerung von Hardware
Diplomand:	Marco Fahr
Firma:	HTWG Konstanz
Betreuer:	Prof. Dr. M. Mächtel
Abgabedatum:	07.05.2008
Schlagworte:	Carrerabahn, Echtzeit, Sensoren, Simulation, Signalverarbeitung,

Das Ziel dieser Diplomarbeit besteht darin mittels einer Simulation eine Testumgebung zu schaffen, in der Programme, welche die Aufgabe haben Autos auf einer Carrerabahn zu steuern, auf ihre Zuverlässigkeit und Fehlerfreiheit getestet werden können. Dazu ist es nötig, dass die zu testenden Programme keinen Unterschied während ihrer Ausführung feststellen, ob sie mit der Simulation oder mit der echten Hardware kommunizieren. Aus diesem Grund verwendet die Simulation den Treiber für die Carrerabahn genau in der gleichen Weise wie die reale Carrerabahn dies im Betrieb auch macht. Die Simulation versucht die reale Carrerabahn mit allen ihren Sensoren möglichst exakt nachzubilden.

Kapitel 1

1. Einführung

1.1 Motivation und Zielsetzung

Ziel dieser Arbeit ist die Erstellung einer Simulation in enger Zusammenarbeit mit dem Treiber der simulierten Hardware. Ziel dieser Arbeit ist unter es gängige Lösungen für Simulationsprobleme aus dem Bereich der Fahrdynamik dahingehend zu beurteilen, ob sie eine mögliche Lösung für auftretende Probleme in der Simulation bieten. Insbesondere die Realisierung des Driftsensors kann als eines der Hauptprobleme betrachtet werden.

1.2 Aufbau der Arbeit

Die Diplomarbeit ist in sechs Kapitel gegliedert. Das erste Kapitel besteht aus der Einführung. Im zweiten Kapitel wird auf die der Simulation zu Grunde liegende Physik und Mathematik eingegangen. Die verschiedenen physikalischen Gesetze werden in diesem Kapitel näher betrachtet und erklärt, insbesondere die Gesetze der Bewegungslehre.

Im dritten Kapitel werden der Aufbau und das Layout der realen Carrerabahn beschrieben und erklärt. Es werden die verschiedenen Sensoren in der Bahn besprochen und die Aufteilung der Bahn in verschiedene Abschnitte.

Kapitel vier beschäftigt sich mit dem Treiber und den nötigen Modifikationen um diesen mit der Carrerabahn der HTWG Konstanz zu benutzen. Dazu gehören die Erweiterung auf den Vierspurbetrieb, das Umstellen auf eine neue Strecke und die Verwendung von anderen Hardware und Benutzer Statuswörtern.

Der aktuelle Stand der Technik im Bezug auf die Simulation von nicht schienengebundenen Kraftfahrzeugen wird im fünften Kapitel beleuchtet. Insbesondere Ansätze zur Bildung von Reifen und Fahrzeugmodellen für die Simulation werden hier vorgestellt. Die Modelle werden insbesondere unter dem Aspekt, sie für die Simulation des Driftsensors zu gebrauchen vorgestellt. Des Weiteren werden eigene Ansätze zur Realisierung des Driftsensors vorgestellt.

Im sechsten und letzten Kapitel wird die Entwicklungsumgebung, die benutzten Werkzeuge und Programmiersprachen und Techniken und der Aufbau und Ablauf der Simulation

vorge stellt. Es wird auf die Realisierung und die Konzepte hinter der Simulation eingegangen und wie die einzelnen Aspekte der Simulation wie z.B. Aufbau der Steck e, Modellierung des Autos und der Sensoren verwirklicht wurden.

Kapitel 2

2. Physikalische Grundlagen

In den Folgenden Kapiteln wird an manchen Stellen von Kinematik, Kinetik und Dynamik gesprochen, weshalb ich an dieser Stelle Folgende Begriffsdefinition anführen möchte.

Kinetik

„Die Kinematik beschreibt die Bewegung der Körper, ohne dabei auf die Kräfte als Ursache für die Verschiedenen Bewegungsarten einzugehen.“ [DoKrVo07]

Kinematik

„Die Lehre von den Kräften und Bewegungen nennt man Kinetik. Sie arbeitet mit allen Grundgrößen der Mechanik, mit der Länge, der Kraft und der Zeit“ [Ass04]

Dynamik

„Wenn man nur vom Wortstamm ausgeht (griechisch = Kraft), dann ist die Dynamik die Lehre von den Kräften. So gesehen ist sie ein Oberbegriff für Statik und Kinetik.“ [Ass04]

2.1 Gleichförmige Bewegung

Eine der Aufgaben der Simulation besteht darin, die möglichen Bewegungen der Autos auf der Bahn zu simulieren. Eine dieser Bewegungen ist die gleichförmige Bewegung mit einer konstanten Geschwindigkeit v .

Man unterscheidet zwei Arten von gleichförmigen Bewegungen:

1. Die gleichförmige geradlinige Bewegung
2. Die gleichförmige krummlinige Bewegung

In diesem Abschnitt möchte ich nur auf die erste der gleichförmigen Bewegungen eingehen, da die zweite in einem der Folgenden Abschnitte über die Kreisbewegung behandelt wird.

Folgende drei Gesetze gelten für die gleichförmige geradlinige Bewegung:

- **Das Weg-Zeit-Gesetz**

$$s(t) = \vec{v}_0 + \vec{v} * t \quad (2.1)$$

- **Das Geschwindigkeits-Zeit-Gesetz**

$$\vec{v}(t) = \vec{v} = \textit{konstant} \quad (2.2)$$

- **Das Beschleunigungs-Zeit-Gesetz**

$$\vec{a}(t) = 0 \quad (2.3)$$

Es gelten weiterhin noch diese Beziehungen untereinander:

- $\dot{s}(t) = \frac{ds}{dt} = \vec{v} \quad (2.4)$

- $\ddot{s}(t) = \dot{v}(t) = \frac{dv}{dt} = \frac{d^2s}{dt^2} = \vec{a} \quad (2.5)$

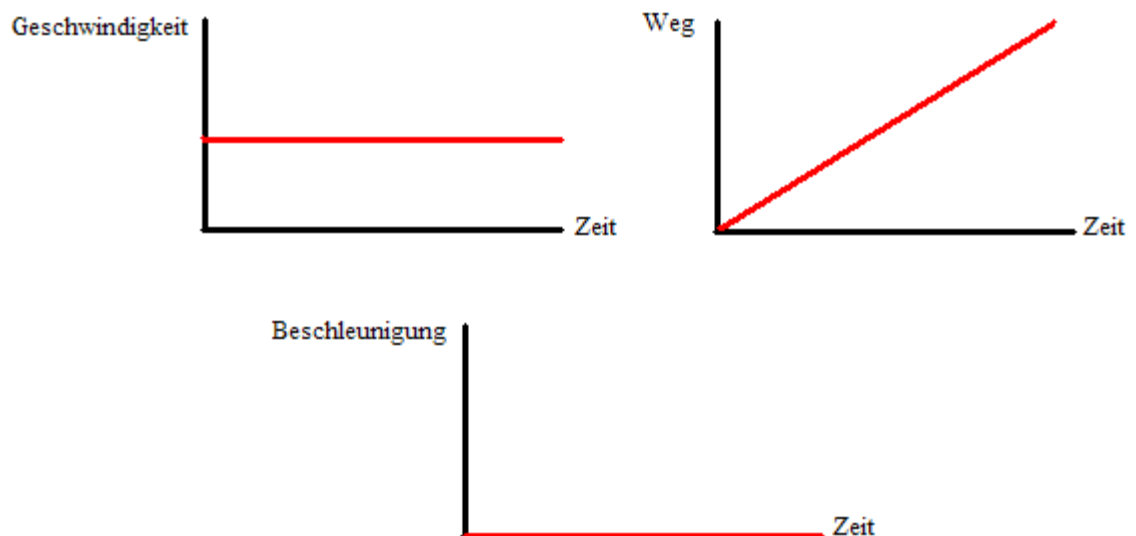


Abbildung 1: Gleichförmige Bewegung

2.2 Gleichförmig beschleunigte Bewegung

Eine weitere Bewegungsart welche durch die Simulation dargestellt werden muss ist die gleichförmig beschleunigte Bewegung. Diese Bewegung tritt immer dann auf wenn das Auto auf der Carrerabahn einen neuen Geschwindigkeitswert übermittelt bekommt. Es ist eine Bewegung mit einer konstanten Beschleunigung \vec{a} . Das Auto besitzt im Moment der Geschwindigkeitszuweisung eine Anfangsgeschwindigkeit v_0 und wird durch die Zuweisung auf die neue Geschwindigkeit \vec{v} beschleunigt. Diese Bewegung tritt auch beim Anfahren auf. In diesem Fall hat die Anfangsgeschwindigkeit \vec{v}_0 den Wert 0. Auch das abbremesen auf eine niedrigere Geschwindigkeit als die Momentane oder bis zum Stillstand ist eine beschleunigte Bewegung. In diesem Fall hat die Beschleunigung einen negativen Betrag.

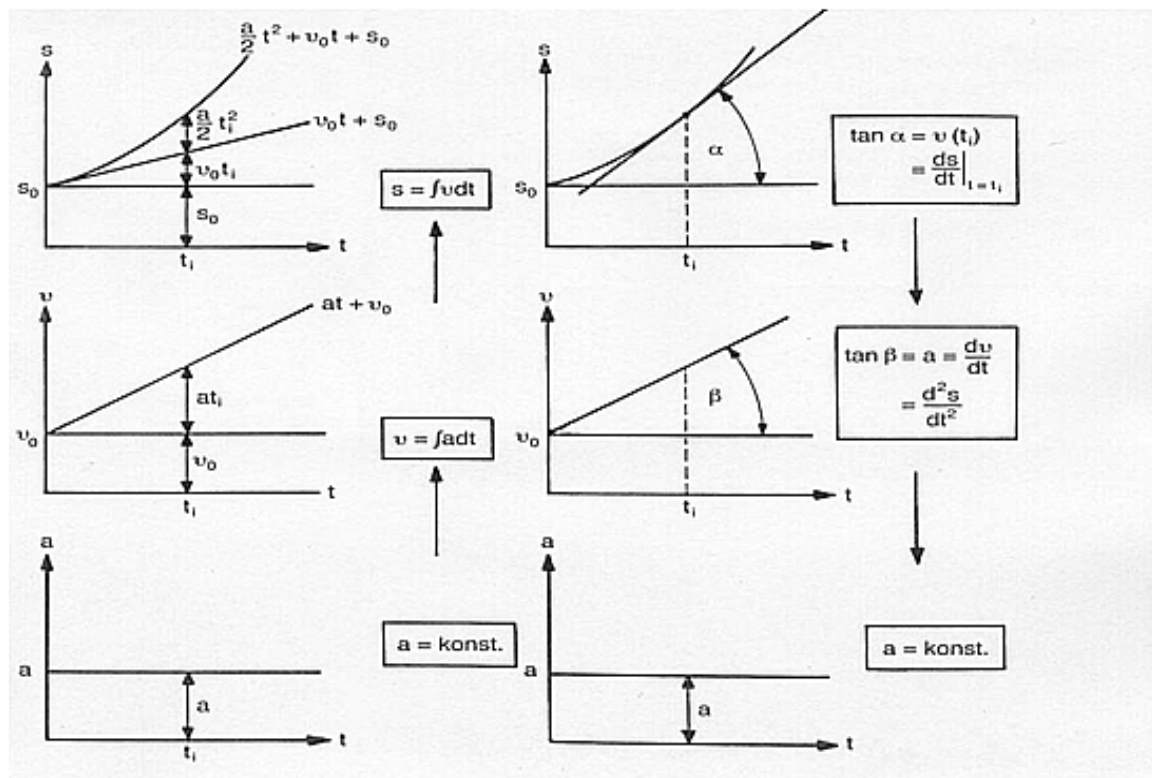


Abbildung 2: Gleichförmig beschleunigte Bewegungen

Quelle [www_eri]

Folgende drei Gesetzte gelten für die gleichförmige beschleunigte Bewegung:

- **Das Weg-Zeit-Gesetzt**

$$s(t) = s_0 + \vec{v}_0 * t \pm \frac{1}{2} * \vec{a} * t^2 \quad (2.6)$$

- **Das Geschwindigkeits-Zeit-Gesetz**

$$\vec{v}(t) = v_0 \pm \vec{a} * t \quad (2.7)$$

- **Das Beschleunigungs-Zeit-Gesetz**

$$\vec{a}(t) = \textit{konstant} \quad (2.8)$$

Weiterhin gelten noch folgende Beziehungen untereinander:

- $\dot{s}(t) = \frac{ds}{dt} = \vec{v}$
- $\ddot{s}(t) = \dot{v}(t) = \frac{dv}{dt} = \frac{d^2s}{dt^2} = \vec{a}$

2.3 Kreisbewegung und Zentripetalkraft

Bei der Kreisbewegung handelt es sich bei der Anfangs schon erwähnten gleichförmig krummlinigen Bewegung. Jede Kurvenfahrt die von einem Auto auf der Carrerabahn unternommen wird ist eine Kreisbewegung. Bei der konstanten Kreisbewegung werden in gleichen Zeitabständen Δt gleiche Kreisbogenabschnitte Δs durchlaufen. Die Geschwindigkeit \vec{v} ist dabei konstant.

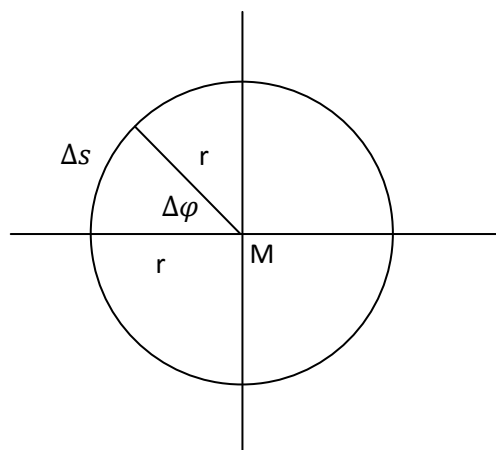


Abbildung 3: Konstante Kreisbewegung

Es gelten die folgenden Beziehungen:

$$\vec{v} = \frac{\Delta s}{\Delta t} \quad (2.9)$$

$$\vec{v} = \frac{\Delta \varphi}{\Delta t} * r \quad (2.10)$$

$$\omega = \frac{d\varphi}{dt} = \dot{\varphi} \quad (2.11)$$

(2.11) in (2.10) eingesetzt führt dann zu:

$$\vec{v} = \omega * r = \dot{\varphi} * r \quad (2.12)$$

$$\varphi = \varphi_0 + \omega * t \quad (2.13)$$

Die radiale Beschleunigung ist bei dieser Bewegung immer zum Kreismittelpunkt gerichtet und vom Betrag konstant. Sie lautet:

$$a_R = a_Z = \omega^2 * r = \frac{v^2}{r} \quad (2.14)$$

Unter Berücksichtigung des zweiten newtonschen Axioms $F = m * a$ ergibt sich die Zentripetalkraft der gleichförmigen Kreisbewegung zu:

$$F_Z = \frac{m * v^2}{r} \quad (2.15)$$

Bei der gleichförmig beschleunigten Kreisbewegung gilt für die Winkelbeschleunigung:

$$\vec{\alpha} = \dot{\vec{\omega}} = \ddot{\vec{\varphi}} = \frac{a_T}{r} \quad (2.16)$$

Für den Drehwinkel gilt, analog zu gleichförmig beschleunigter Bewegung, die folgende Beziehung:

$$\varphi(t) = \varphi_0 + \omega_0 * t + \frac{1}{2} \vec{\alpha} * t^2 \quad (2.17)$$

2.4 Reibung

Wenn sich zwei Körper an einer Stelle berühren und es wird eine horizontale Kraft auf diese Kontaktstelle ausgeübt, so tritt an dieser Stelle Reibung zwischen diesen Objekten auf. Wenn ein z.B. ein Objekt auf einer Straße aufliegt so wird dieses mit der Kraft F_N auf den Untergrund gedrückt. Die Kraft F_N ist der Anteil der Gewichtskraft des Objektes der im rechten Winkel (senkrecht) an der Auflagefläche angreift. Wird nun versucht das Objekt über den Untergrund zu ziehen oder zu drücken, so entsteht die Reibkraft welcher der ziehenden oder drückenden Kraft entgegen wirkt. Diese Kraft verhindert ein Rutschen des Körpers über den Untergrund. Dies nennt man Haftreibung. Wird die Kraft um den Körper in Bewegung zu versetzen erhöht so wird irgendwann eine Schwelle erreicht bei der die Gegenkraft der Haftreibung kleiner ist als die anliegende Kraft. Das Objekt beginnt über den Untergrund zu gleiten und die Haftreibung geht in Gleitreibung über. Der Widerstand der dem Körper durch das Gleiten auf dem Untergrund entgegengebracht wird ist die Reibkraft.

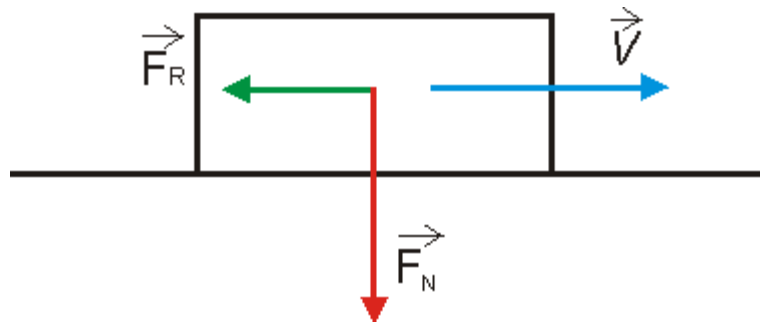


Abbildung 4: Reibung

Quelle [www_wü]

$$\vec{F}_H \leq \mu_H * \vec{F}_N \quad (2.18)$$

$$\vec{F}_{Hmax} = \mu_H * \vec{F}_N \quad (2.19)$$

$$\vec{F}_K = \mu * \vec{F}_N \quad (2.20)$$

μ_H ist der Haftreibungskoeffizient und er beschreibt das Reibungsverhalten zwischen zwei Objekten die nicht in Bewegung sind. μ ist der Gleitreibungskoeffizient welcher das Verhalten der Reibung zwischen zwei Objekten beschreibt die an einer Fläche aneinander gleiten. Damit sich ein Objekt bewegt muss eine Kraft aufgebracht werden die größer als die Haftreibungskraft ist. Ist der Körper in Bewegung so muss nur noch die Gleitreibungskraft aufgebracht werden um den Körper in Bewegung zu halten. Aus diesem Grund gilt immer $\mu \leq \mu_H$.

Die Folgende Tabelle wurde aus [Rob04] entnommen:

Material	Haftreibungskoeffizient μ_H	Gleitreibungskoeffizient μ
Eisen – Eisen	0,45 – 0,80	1,0
Kupfer – Kupfer	0,45 – 0,80	0,60 – 1,0
Stahl – Stahl	0,45 – 0,80	0,40 – 0,70
Chrom – Chrom	0,45 – 0,80	0,41
Nickel – Nickel	0,45 – 0,80	0,39 – 0,70
ALU Leg. – ALU Leg.	0,45 – 0,80	0,15 – 0,60
Stahl – Kupfer	0,45 – 0,80	0,23 – 0,29
Stahl – Weißmetall	0,45 – 0,80	0,21
Stahl – Grauguss	0,18 – 0,24	0,17 – 0,24
Bremsbelag – Stahl		0,50 – 0,60
Leder - Metall	0,60	0,20 – 0,25
Polyamid - Stahl		0,32 – 0,45
PTFE – Stahl		0,04 – 0,22
Eis - Stahl	0,027	0,014

Abbildung 5: Reibungskoeffizienten

2.5 Die schiefe Ebene

Auch die schiefe Ebene spielt in der Simulation eine Rolle. Da es auf der Carrerabahn je einen Abschnitt mit einer Bergfahrt und einen Abschnitt mit einer Talfahrt gibt, ist es notwendig sich mit den Gesetzen an der schiefen Ebene vertraut zu machen.

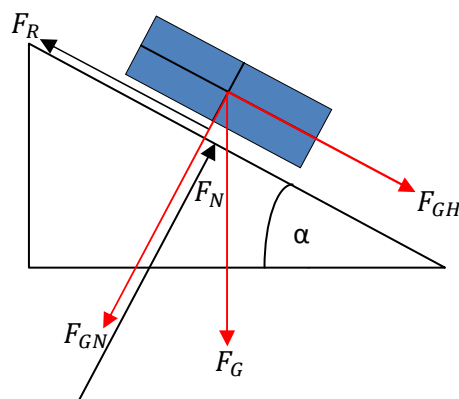


Abbildung 6: Kräfte an der schiefen Ebene

Ein Körper, der die schiefe Ebene mit einer konstanten Geschwindigkeit hinauffährt, beschreibt eine gleichförmig verzögerte Bewegung. Ein Körper, der die schiefe Ebene hinunterfährt, beschreibt eine gleichförmig beschleunigte Bewegung. Die Bewegung kommt dadurch zu Stande, weil eine Komponente, die Hangabtriebskomponente \vec{F}_{GH} , welche parallel zur schiegen Ebene verläuft, den Körper beschleunigt oder abbremst. Die Komponente, welche senkrecht zur schiegen Ebene verläuft, die Normalkomponente \vec{F}_{GN} bzw. \vec{F}_N , ist verantwortlich für die Reibungskraft \vec{F}_R , die zwischen den Auflageflächen entsteht.

$$\vec{F}_{GN} = \vec{F}_G * \cos \alpha \quad (2.21)$$

$$\vec{F}_{GH} = \vec{F}_G * \sin \alpha \quad (2.22)$$

$$\vec{F}_R = \mu * \vec{F}_{GN} \quad (2.23)$$

Bei der Abwärtsbewegung ist die Gleitreibung der Hangabtriebskraft entgegengesetzt. Bei der Aufwärtsbewegung addieren sich Hangabtriebskraft und Gleitreibungskraft.

Die jeweilige Beschleunigung ergibt sich zu:

- **Aufwärts**

$$\vec{a} = \vec{g} (\sin \alpha + \mu * \cos \alpha) \quad (2.24)$$

- **Abwärts**

$$\vec{a} = \vec{g} (\sin \alpha - \mu * \cos \alpha) \quad (2.25)$$

Diese Beschleunigung verzögert oder beschleunigt eine gleichförmige Bewegung an der schiefen Ebene. Bei gleichförmig beschleunigten oder verzögerten Bewegungen unterstützt diese Beschleunigungskomponente die Bewegung.

2.6 Trägheitsmoment

Der Trägheitsmoment, auch Massenträgheitsmoment genannt, ist die physikalische Größe die angibt wie sich ein starrer Körper bei einer Veränderung der eigenen Rotation um eine Achse verhält.

Das Trägheitsmoment wird durch die Folgende Summe berechnet, r_i ist der senkrechte Abstand des i - ten Teilchens von der Drehachse:

$$J = \sum_i m_i * r_i^2 \quad (2.26)$$

Da in der Simulation der Körper des Autos zu einem Quader vereinfacht wird, ist in diesem Fall nur das Trägheitsmoment der z-Achse von Bedeutung ergibt sich die Formel zu:

$$J = \sum_i m_i * (l_a^2 + b_a^2) \quad (2.27)$$

Hieraus folgt mit $dm = \rho \, dl_a \, db_a \, dh_a$:

$$J_z = \int_m (l_a^2 + b_a^2) \, dm \quad (2.28)$$

$$J_z = \rho \int_{-\frac{h_a}{2}}^{\frac{h_a}{2}} \int_{-\frac{b_a}{2}}^{\frac{b_a}{2}} \int_{-\frac{l_a}{2}}^{\frac{l_a}{2}} (l_a^2 + b_a^2) dl_a db_a dh_a \quad (2.29)$$

$$J_z = \rho \int_{-\frac{h_a}{2}}^{\frac{h_a}{2}} \int_{-\frac{b_a}{2}}^{\frac{b_a}{2}} (l_a^3 + b_a^2 * l_a) db_a dh_a \quad (2.30)$$

$$J_z = \rho \int_{-\frac{h_a}{2}}^{\frac{h_a}{2}} \left(\frac{1}{12} l_a^3 * b_a + \frac{1}{12} b_a^3 * l_a \right) dh_a \quad (2.31)$$

$$J_z = \rho \left(\frac{1}{12} l_a^3 * b_a * h_a + \frac{1}{12} b_a^3 * l_a * h_a \right) \quad (2.32)$$

$$J_z = \frac{1}{12} \rho * l_a * b_a * h_a (l_a^2 + b_a^2) \quad (2.33)$$

Mit $m = \rho * l_a * b_a * h_a$ ergibt sich das Trägheitsmoment um die z-Achse des Autos zu:

$$J_z = \frac{1}{12} * m * (l_a^2 + b_a^2) \quad (2.34)$$

Dieses Moment wird bei der späteren Betrachtung der dynamischen Fahrzeugmodelle in Kapitel 5 benötigt.

2.7 Kontinuierliche Simulation

Das Ziel einer kontinuierlichen Simulation ist es den zeitlichen Verlauf der Zustandsgrößen eines Modells möglichst exakt zu beschreiben. Als Zustandsgrößen bezeichnet man diejenigen Größen, die durch Aufsummierung eines kleinen Teilbetrags zum aktuellen Wert, innerhalb eines Zeitintervalls, berechnet werden können. Neben diesen Größen existieren noch die Funktionalen Größen des Modells. Diese lassen sich durch Gleichungen mit vorhandenen Werten sofort bestimmen.

Für die Bestimmung des zeitlichen Verlaufs werden in der Regel Differentialgleichungen verwendet. Diese Gleichungen versuchen das dynamische Verhalten mathematisch auszudrücken. Mit diesen Gleichungen wird das physikalische Verhalten der Simulation modelliert und im Rechner abgebildet. Der Zustand der Simulation wird durch Integration der abgeleiteten Zustandsgrößen ermittelt.

Der Funktionsverlauf der Simulation ist nur zu bestimmten Zeitpunkten genau gegeben und muss deshalb zwischen diesen Zeitpunkten durch Funktionen ermittelt werden. Mit Hilfe dieser Funktionen ist es möglich den Simulationsablauf kontinuierlich zu berechnen. Dies geschieht durch Integration dieser Funktionen. Ausgehend von einem Startzustand kann so das Modell der Simulation das Verhalten zu jedem beliebigen Zeitpunkt berechnen. Modellgrößen, wie z.B. Geschwindigkeiten, Beschleunigungen, Drehmomente oder Kräfte die die Veränderung des Modells bewirken, werden während der Simulation zwischen zwei Zeitpunkten als konstant angenommen, da eine genauere Bestimmung in der Regel nicht möglich ist.

2.8 Numerische Integration nach Euler¹

Integrale müssen auf dem Rechner numerisch gelöst werden. Das einfachste und auch bekannteste Verfahren ist die Integration nach Euler. Dieses Verfahren ist auch als Polygonzug- oder Euler-Cauchy-Verfahren bekannt. Von einem Startwert ausgehend werden in einem bekannten Zeitintervall, für eine definierte Schrittweite, Stützstellen berechnet und die Funktion zwischen diesen Stützstellen linear interpoliert. Die so berechneten Werte sind nur Näherungen, je kleiner jedoch die Schrittweite gewählt wird, desto genauer liegen diese Werte bei dem simulierten realen System.

Wird einem Körper mit der Masse m , der Position s und der Geschwindigkeit v eine Kraft F über einen Zeitraum t zugeführt, so erfährt dieser Körper eine Beschleunigung die durch die folgende Gleichung gegeben ist:

$$F_{(t)} = m * a_t = a_t = \frac{F_{(t)}}{m} \quad (2.35)$$

Mit den Gleichungen

$$v_{(t)} = v_0 + a * t \equiv v_{(t+\Delta t)} = v_{(t)} + \int_t^{t+\Delta t} a_{(t)} dt \quad (2.36)$$

und

$$s_{(t)} = s_0 + v * t \equiv s_{(t+\Delta t)} = s_{(t)} + \int_t^{t+\Delta t} v_{(t)} dt \quad (2.37)$$

¹ Vgl. [Her04]

sind die Geschwindigkeit und die Position des Körpers, in Abhängigkeit von der Beschleunigung, zu jedem Zeitpunkt exakt bestimmt. Auf Grund der Tatsache, dass der Verlauf der Beschleunigung über der Zeit nicht bekannt ist, sind die Integrale nicht exakt bestimmbar. Die Beschleunigung kann nur zum Zeitpunkt t und den vorherigen Zeitpunkten bestimmt werden. Durch das Integrationsverfahren wird nun versucht sich dem wirklichen Verlauf der Beschleunigung möglichst exakt anzunähern. Hier kommen die numerischen Integrationsverfahren ins Spiel. Eines der einfachsten ist das Euler-Verfahren. Mit dessen Hilfe wird nun der Verlauf der Beschleunigung zwischen den berechneten Stützpunkten linear interpoliert.

Die lineare Interpolation ist durch die Folgenden zwei Gleichungen gegeben:

$$v_{(t+\Delta t)} = v_{(t)} + \frac{a_{(t)} - \frac{a_{(t+\Delta t)} - a_{(t)}}{2}}{\Delta t} \quad (2.38)$$

$$s_{(t+\Delta t)} = s_{(t)} + \frac{v_{(t)} - \frac{v_{(t+\Delta t)} - v_{(t)}}{2}}{\Delta t} \quad (2.39)$$

Die Qualität des Ergebnisses, sprich die Größe des gemachten Fehlers durch die lineare Interpolation, hängt von der Größe der gewählten Schrittweite Δt ab. Je kleiner die Schrittweite gewählt wird, desto kleiner der Fehler.

Kapitel 3

3. Carrerabahn Aufbau

3.1 Segmente

Bei der aufgebauten Rennbahn im Labor für Echtzeitsysteme der HTWG Konstanz handelt es sich um das Modell Exclusiv der Firma Carrera. Auf der Bahn können Fahrzeuge im Maßstab 1:24 und 1:32 betrieben werden. Ein Element der Bahn ist 198 mm breit und der Spurabstand beträgt 100 mm. Die Bahn ist mit vier Spuren aufgebaut. Die Gesamtbreite der Carrerabahn summiert sich dadurch auf 396 mm plus eventuell angebrachte Seitenstreifen für die Leitplanken. Die Carrerabahn ist in insgesamt neun Segmente aufgeteilt. Zwei Segmente beinhalten je eine Kreuzung. In einem Segment befindet sich eine Steigung, im anschließenden Segment das Gefälle. Die Bahn verfügt über insgesamt zehn Kurven und eine Brücke.

Das erste Segment besteht nur aus einer langen Geraden. Es hat seinen Anfang nach dem Start/Ziel Element und endet an der ersten Kurve. Es besteht aus zwei 1/3 Geraden mit je 115mm Länge und sechs Standardgeraden mit je 345 mm Länge. Diese Bauteile werden für die Spuren eins, zwei, drei und vier verwendet.

Das zweite Segment besteht aus einer 210° Rechtskurve. Die Spuren eins und zwei sind aus sieben 2/30° Kurvenelementen, die Spuren drei und vier aus drei 1/60° und einem 1/30° Kurvenelement aufgebaut.

Segment drei besteht aus einer weiten 120° Linkskurve gefolgt von einer kurzen 30° Rechtskurve. Spur eins und zwei bestehen aus zwei 1/3 Geraden mit je 115 mm Länge, vier 2/30° und einem 3/30° Kurvenelement, Spur drei und vier auch aus zwei 1/3 Geraden, vier 3/30° und einem 2/30° Kurvenelement.

Das vierte Segment besteht aus einem weiten 360° Linkskurve mit zusätzlicher Steigung. Spur eins und zwei sind aus einer 1/3 Geraden und zwölf 2/30° Kurvenelementen aufgebaut. Die Spuren drei und vier aus einer 1/3 Geraden und zwölf 3/30° Kurvenelementen.

Segment fünf beinhaltet eine Kreuzung und eine kurze 30° Linkskurve. Die Spuren eins und zwei sind aus zwei 1/3 Geraden, einer Kreuzung mit 345 mm Länge, einem 3/30° Kurvenelement und einer 345 mm langen Standardgeraden aufgebaut. Spur drei und vier

bestehen auch aus zwei $1/3$ Geraden, einer Kreuzung, einem $4/15^\circ$ Kurvenelement und einer Standardgeraden.

Segment sechs besteht aus einer 240° Rechtskurve mit Gefälle und einer 30° Linkskurve. Die erste und die zweite Spur sind aus acht $4/15^\circ$ Kurvenelementen, zwei Standardgeraden, einem $3/30^\circ$ Kurvenelement und einem $1/4$ Geradenelement mit 00 mm Länge zusammengebaut. Spur drei und vier aus acht $3/30^\circ$ Kurvenelementen, zwei Standardgeraden, einem $4/15^\circ$ Kurvenelement und einem $1/4$ Geradenelement.

Das siebte Segment ist eine 180° Linkskurve. Spur eins und zwei bestehen aus einer Standardgeraden und drei $1/60^\circ$ Kurvenelementen. Die Spuren drei und vier aus einer Standardgeraden und sechs $2/30^\circ$ Kurvenelementen.

Im achten Segment befindet sich wieder eine Kreuzung und eine 120° Rechtskurve. Die erste und zweite Spur bestehen aus einer Standardgeraden, einer Kreuzung, vier $2/30^\circ$ Kurvenelementen und einer $1/3$ Geraden. Die dritte und vierte Spur aus einer Standardgeraden, einer Kreuzung, zwei $1/60^\circ$ Kurvenelementen und einem $1/3$ Geradenelement.

Im neunten und letzten Segment befindet sich eine 120° Rechtskurve. Die Spuren eins und zwei bestehen aus einer Standardgeraden, vier $3/30^\circ$ Kurvenelementen, zwei Standardgeraden und einem Start/Ziel-Element mit 345 mm Länge. Die Spuren drei und vier aus einer Standardgeraden, vier $2/30^\circ$ Kurvenelementen, zwei Standardgeraden und einem Start/Ziel-Element.

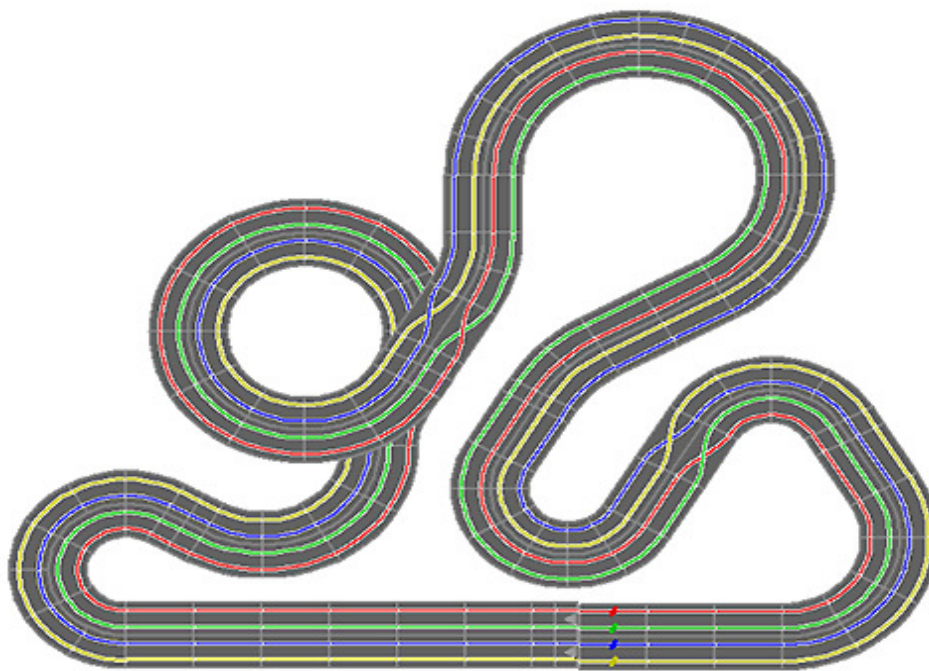


Abbildung 7: Aufbauplan der Carrerabahn

3.2 Sensoren

Die Carrerabahn ist mit zwei Arten von Sensoren ausgestattet. Beim ersten Sensortyp handelt es sich um eine Lichtschranke. Die Lichtschranken sind immer am Anfang vor dem jeweiligen Segment angebracht. Mit ihnen ist es möglich die Position und die Geschwindigkeit eines Autos auf der Bahn zu bestimmen. Die Segmente der Carrerabahn werden durch die Position der Sensoren vorgegeben. Immer zwischen zwei Sensoren ist ein Segment. Es gibt insgesamt neun Sensoren und die Bahn ist auch in neun Segmente aufgeteilt. Der erste Sensor wurde direkt nach dem Start in der Mitte der ersten 1/3 Geraden angebracht. Der zweite Sensor wurde in der Standartgeraden vor der ersten Kurve angebracht. Sein Abstand beträgt 86 mm vom Anfang der Standartgeraden. Der dritte Sensor wurde in der Mitte der ersten 1/3 Geraden nach der ersten Kurve angebracht. Die vierte Lichtschranke wurde in der Mitte der 1/3 Geraden vor der dritten Kurve, dem Vollkreis, angebracht. Lichtschranke Nummer fünf wurde in der Mitte der 1/3 Geraden nach dem Vollkreis angebracht. Die sechste Lichtschranke wurde in der Standartgeraden vor der vierten Kurve angebracht. Der Abstand beträt 86 mm vom Kurvenanfang in die Standartgerade hinein. Der siebte Sensor wurde in der Standartgeraden vor der fünften Kurve angebracht. Sein Abstand beträgt 86 mm vom Kurvenanfang in die Standartgerade hinein. Sensor Nummer acht wurde in die Standartgerade vor der Kreuzung des achten Segments angebracht. Hier beträgt der Abstand auch 86 mm vom Anfang der Kreuzung in die Standartgerade hinein. Die letzte Lichtschranke wurde 86 mm vor der letzten Kurve in der Standartgeraden angebracht.

Der zweite Sensortyp ist ein sogenannter Driftsensor. Auf Grund der Tatsache, dass während der Erstellung der Diplomarbeit noch keine Driftsensoren installiert waren, wurde in Absprache mit Prof. Mächtel die Annahme getroffen, dass die Sensoren immer bei 40 Grad in einer Kurve angebracht sind. Ein solcher Driftsensor ermittelt die Auslenkung des Autos beim driften in der Kurve. Sein Wertebereich liegt zwischen eins und vierzehn. Gemessen wird mit diesem Sensor nur die Auslenkung nach außen. Auf der fertigen Bahn sind insgesamt sieben Driftsensoren verbaut. Wird ein Driftsensor ausgelöst, so werden zwei Statuswörter von der Carrerabahn an den Treiber übermittelt. Das erste Statuswort enthält Informationen über das aktuelle Segment, ob eine Lichtschranke ausgelöst wurde, ob ein Driftsensor ausgelöst wurde und vieles mehr. Das zweite Hardware Statuswort wird nur übertragen wenn ein Driftsensor ausgelöst wurde. In diesem Statuswort ist die Information über welchen Sensor es sich handelt und welchen Wert dieser Sensor angenommen hat kodiert.

Kapitel 4

4. Treiber

4.1 Anpassungen

Eine der ersten Tätigkeiten während der Diplomarbeit bestand im studieren der Funktionsweise des Treiber, welcher mir von Prof. Mächtel zur Verfügung gestellt wurde. Die Version musste um einige Funktionalität erweitert werden, da sie für eine andere Rennbahn gedacht war. In der Ursprünglichen Version war der Treiber für eine Rennbahn mit nur zwei Spuren geschrieben worden. Es mussten also Gerätedateien für zwei zusätzliche Spuren angelegt werden. Diese Änderung betraf die Gerätedateien für die Steuerprogramme und die Gerätedateien für die Simulation. Es wurden also zusätzliche vier neue Gerätedateien eingebaut. Im Treiber wie auch in der Simulation wird eine Spur durch eine Struktur dargestellt. Diese Struktur wurde so erweitert, dass eine Spur nicht nur Informationen über den direkten Gegner in Form seines Hardware Statuswortes hat, sondern von allen an einem Rennen beteiligten Spuren. Diese Struktur musste auch um das zweite Hardware Statuswort für den Driftsensor erweitert werden. In diesem Zuge wurde auch die Routine die für den Austausch der Statuswörter zuständig ist so überarbeitet, dass der Austausch der Hardware Statuswörter zwischen den Spuren auch mit der neue Carrerabahn mit vier Spuren funktioniert.

Im nächsten Schritt musste dem Treiber der Aufbau der Carrerabahn an der HTWG bekannt gemacht werden. Die Bahn wurde als ein Feld mit so vielen Elementen wie es Segmente auf der Bahn gibt realisiert. Jedem Segment wurde ein hexadezimaler Wert gegeben der mit den Werten der Hardware Statuswörtern der Segmente in der Simulation übereinstimmt. So ist gesichert wenn ein Statuswort von der Simulation an den Treiber geschickt wird, dass der Treiber herausfinden kann in welchem Segment das Auto sich gerade befindet. Im nächsten Schritt mussten die Routinen angepasst werden welche die Statuswörter die von der Simulation an den Treiber geschrieben werden auswerten. Da das 32 Bit Hardware Statuswort eine andere Belegung hat wie die für den der Treiber Ursprünglich geschrieben wurde.

Da bei der Übertragung der Driftsensorwerte der HTWG-Carrerabahn zwei Statuswörter benötigt werden, wurde der Treiber so umgeschrieben, dass er mit einem Feld von Statuswörtern umgehen kann. Der Treiber kann dann auf das zweite Element des Feldes zugreifen wenn er das zweite Statuswort benötigt. Auf diese Weise ist Sichergestellt, dass beide Statuswörter mit einem write()- Aufruf der Simulation gleichzeitig beide Statuswörter

übertragen werden und die Simulation nicht zweimal den write()-Aufruf hintereinander ausführen muss. Es wurde noch ein Fehler behoben, der es nicht erlaubte den Treiber ein zweites Mal zu laden nachdem er schon einmal vom Betriebssystem geladen wurde. Durch den Fehler konnten beim erneuten Laden keine Gerätedateien für die Simulation und die Steuerprogramme angelegt werden, da beim löschen zuvor die Nummern der Geräte nicht richtig freigegeben wurden.

Der Treiber kommuniziert mit dem Steuerprogramm eines Autos über ein sogenanntes Benutzer Statuswort. Immer wenn von der Bahn ein Hardware Statuswort an den Treiber geschickt wird, weil sich z.B. das aktuelle Segment geändert hat oder ein Sensor ausgelöst wurde, wird daraufhin vom Treiber ein Benutzer Statuswort generiert und auf die zuständige Gerätedatei geschrieben. Die Routine die dieses Benutzer Statuswort generiert musste auch an die neue Syntax, welche die HTWG-Bahn und ihre Steuerprogramme verwendet angepasst werden.

Kapitel 5

5. Modell- und Simulationslösungen

In diesem Kapitel sollen verschiedene Modelle die als Lösung für die Implementierung des Driftsensors geeignet sind diskutiert werden. Die Modelle werden heute in allen Bereichen der Fahrdynamik und der Simulation von Fahrzeugen angewendet und die meisten sind stellen den heutigen Stand der Technik dar. Fahrassistenzsysteme wie ESP, ASR und DSC basieren im Wesentlichen auf diesen Modellen. Mit diesen Modellen ist eine Grundlage geschaffen worden, Kraftfahrzeuge physikalisch korrekt im Computer abzubilden. Es wird in diesem Kapitel jedoch nur auf Modelle mit maximal zwei Freiheitsgraden eingegangen. Modelle die mehr Freiheitsgrade bieten, wie z.B. die MKS-Modelle oder die FEM-Modelle werden nicht vorgestellt, da diese mehr bieten als für die Lösung des Simulationsproblems benötigt wird.

Für die Simulation ist besonders die Querdynamik dieser Modelle entscheidend, da diese die Kräfte und Bewegungen beschreiben welche an Autos beim Driftvorgang in der Carrerabahn vorherrschen.

Die meisten der diskutierten Modelle beschreiben leider nur das Verhalten des Autos bis in den kritischen Bereich jedoch nicht über den Grenzbereich hinaus.

5.1 Radumfangsschlupf²

Im Allgemeinen bedeutet Schlupf das Verhalten eines mit festgelegter Drehzahl rotierenden Körpers zu einem oder mehreren anderen Bauteilen nicht ganz synchron zu rotieren. Eines der besten Beispiele hierfür wäre ein Riemenantrieb. Der Riemen rotiert mit einer geringfügig anderen Drehzahl wie die Welle die ihn antreibt. Die Kraft die von einem Reifen übertragen werden kann hängt direkt vom sogenannten Reifenschlupf ab. Dieser Schlupf ist nach DIN 70000 folgendermaßen definiert:

$$S_x = \frac{\omega - \omega_0}{\omega_0} \quad (5.1)$$

² Vgl. [HeEr07] S. 36 - 37

In der Formel bezeichnet ω_0 die Drehgeschwindigkeit eines geradeaus frei rollenden Rades und ω bezeichnet den Schlupf eines Rades.

Nach DIN 7000 sind der Umfangs- und Seitenkraftbeiwert folgendermaßen definiert:

$$\mu_X = \frac{F_X}{F_N} \quad (5.2)$$

$$\mu_X = \frac{F_Y}{F_F} \quad (5.3)$$

5.2 Lineares Reifenmodell

Das einfachste der bekannten Reifenmodelle zur Berechnung der Reifenseitenkraft, ist das lineare Reifenmodell. Es behält seine Gültigkeit bei kleinen Schräglaufwinkeln. Dem Modell liegt die Vereinfachung zu Grunde, dass die Seitenführungskraft des Reifens durch die folgende Beziehung gegeben ist:

$$F_y = c_\alpha * \alpha \quad (5.4)$$

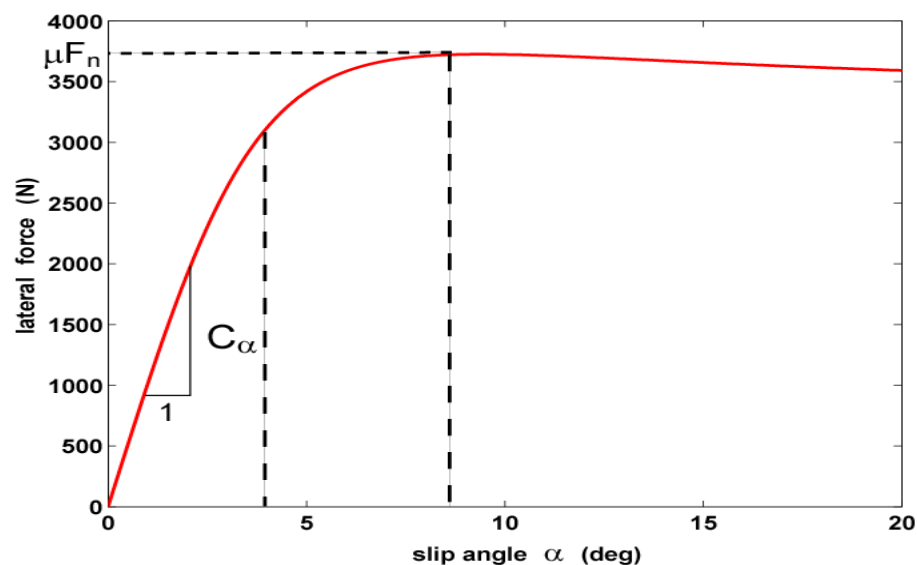


Abbildung 8: Seitenführungskraft in Abhängigkeit des Schwimmwinkels Quelle [JudGer05]

Bei diesem Modell steht c_α für den Seitenkraftbeiwert und α für den Schwimmwinkel. Der Seitenkraftbeiwert hängt vom Radumfangsschlupf S_x des jeweiligen Rades ab. Das bedeutet der Seitenumfangsschlupf hat dieselbe Steigung wie der lineare Anteil des Radumfangsschlupf. Da dieser lineare Bereich nur begrenzt eingesetzt werden kann sind die Möglichkeiten mit diesem Modell sehr begrenzt. Das Verhalten eines Autos das den Grenzbereich überschritten hat lässt sich diesem Modell nicht simulieren. Sein Haupteinsatzgebiet liegt beim linearen Einspurmodell.

5.3 Reifenmodell nach Pacejka

Ein Ansatz der auf trigonometrischen Funktionen basiert ist die sogenannte „Magic Formula“ von Pacejka.

$$y = D * \sin[C * \tan^{-1}(B * x - \tan^{-1} B * x)] \quad (5.5)$$

Die Formel wird hier nur der Vollständigkeit halber vorgestellt, da sie ein häufig genutztes Modell in der Simulation von Kraftfahrzeugen ist. Sie scheidet aber als Möglichkeit für die Simulation aus. Je nach dem was mit der Formel berechnet wird, entweder die lateralen Reifenkräfte oder die longitudinalen Reifenkräfte, werden die einzelnen Faktoren D, C und B entweder aus fünfzehn oder elf Reifenkennwerten berechnet. Die Kennwerte werden durch die Reifenhersteller empirisch in Versuchen ermittelt. Da man die Vollgummireifen eines Carreraauto schlecht mit den luftgefüllten Reifen eines Kraftfahrzeugs vergleichen kann, wird dieses Modell nicht weiter betrachtet. Der interessierte Leser sei auf [Bec01] verwiesen. In den Kapiteln 21, 22 und 29 wird besonders auf die Verwendung der Magic Formula eingegangen

5.4 Lineares Einspurmodell³

Das am häufigsten bei der Realisierung von Simulationen verwendete Modell für ein Fahrzeug ist das lineare Einspurmodell. Dieses Modell besitzt einen Freiheitsgrad welcher durch die Längsdynamik gegeben ist. Es entstand durch die Arbeit von Rieker und Schunk 1939 [RieSchu39].

Dieses Modell ist Stand der Technik im Bereich der linearen Theorie der Fahrzeugtechnik. Es ist ein stark vereinfachtes Modell bei dem der Schwerpunkt des Autos in die Fahrbahnebene gelegt wird. Die Räder einer jeden Achse werden in der Mitte zu einem Rad zusammengefasst, daher auch die Bezeichnung Einspurmodell. Dieses Modell behält

³ Vgl. [HeEr07] S. 90 - 95

grundsätzlich seine Gültigkeit solange Querbeschleunigungen von ca. 4 m/s^2 nicht überschritten werden. Desweiteren behält es nur seine Gültigkeit wenn die Bewegungsgleichungen linearisiert werden, so dass angenommen werden kann es gilt $\sin(\alpha) \approx \alpha$ und $\cos(\alpha) \approx 1$. Darüber hinaus behält es seine Gültigkeit nur bei trockener Fahrbahn. Es treten keine Beschleunigungen in Längsrichtung. Hub-, Nick- und Wankbewegungen werden nicht berücksichtigt. Die Radlasten an den Vorder- und Hinterachsen bleiben konstant, es tritt also kein Lastwechsel auf. Für eine eventuelle Verwendung in der Simulation ist das dynamische Verhalten des linearen Einspurmodells von Bedeutung. Hier ist die Betrachtung des dynamischen Gleichgewichts in der Querrichtung und um die Hochachse maßgebend.

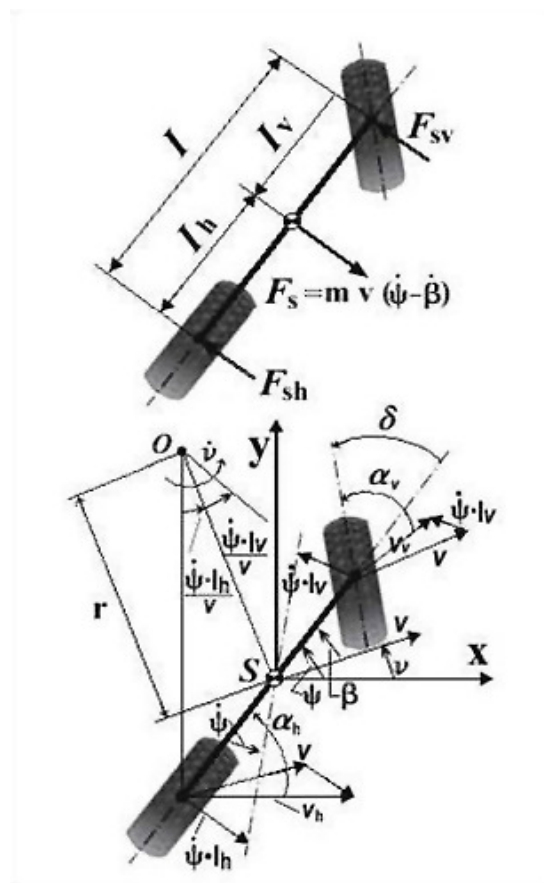


Abbildung 9: Lineares Einspurmodell

Quelle [HeEr07]

Im ersten Schritt werden die Gleichgewichtsbedingungen des linearen Einspurmodells aufgestellt. Kräftegleichgewicht der Längsachse:

$$m * a_y = F_{SV} + F_{SH} \quad (5.6)$$

Momentengleichgewicht:

$$J_z * \ddot{\psi} = F_{SV} * l_v + F_{SH} * l_h \quad (5.7)$$

Die Näherung der Schräglaufwinkel für kleine Lenkwinkel und Giergeschwindigkeiten:

$$\alpha_v = \beta + \delta - l_v * \frac{\dot{\psi}}{v} \quad (5.8)$$

$$\alpha_h = \beta + l_h * \frac{\dot{\psi}}{v} \quad (5.9)$$

Das lineare Reifenmodell:

$$F_{SV} = c_v * \alpha_v \quad (5.10)$$

$$F_{SH} = c_h * \alpha_h \quad (5.11)$$

Die Beschleunigung ergibt sich zu:

$$a_y = \frac{v^2}{r} = \frac{v}{r} * v = \frac{v}{r} * r * (\dot{\psi} - \dot{\beta}) = v(\dot{\psi} - \dot{\beta}) \quad (5.12)$$

Durch einsetzen von (5.5), (5.6), (5.7), (5.8) in (5.3) und (5.4) ergibt sich das folgende lineare Differentialgleichungssystem für den Schwimmwinkel und die Gierwinkelgeschwindigkeit:

$$\dot{\beta} = -\frac{c_h + c_v}{m * v} * \beta + \left(1 - \frac{c_h * l_h - c_v * l_v}{m * v^2}\right) * \dot{\psi} - \frac{c_v}{m * v} * \delta \quad (5.13)$$

$$\ddot{\psi} = \frac{c_v * l_v - c_h * l_h}{J_z} * \beta - \frac{c_h * l_h^2 + c_v * l_v^2}{J_z * v} * \dot{\psi} + \frac{c_v * l_v}{J_z} * \delta \quad (5.14)$$

Da der Lenkwinkel δ bei den Autos der Carrerabahn immer null ist und der Drehpunkt nicht im Schwerpunkt des Autos, sondern an der Finne an der Vorderachse liegt, ergeben sich die Differentialgleichungen zu:

$$\dot{\beta} = -\frac{c_h}{m * v} * \beta + \left(1 - \frac{c_h * l}{m * v^2}\right) * \dot{\psi} \quad (5.15)$$

$$\ddot{\psi} = \frac{c_h * l}{J_z} * \beta - \frac{c_h * l^2}{J_z * v} * \dot{\psi} \quad (5.16)$$

Mit diesem Gleichungssystem lassen sich nun Schwimmwinkel und die Gierwinkelgeschwindigkeit eines Autos, mittels numerischer Integration nach Euler oder Range-Kutta, zu jedem Simulationszeitpunkt bestimmen. Als Eingangsgröße fungiert hier der Schwimmwinkel β und die Giergeschwindigkeit $\dot{\psi}$.

Da der Lenkwinkel δ bei den Autos der Carrerabahn immer null ist ergeben sich die Differentialgleichungen zu

Aufgrund der Tatsache, dass dieses Modell jedoch nur für Querschleunigungen konzipiert ist welche $\leq 0,4 \text{ g}$ betragen, ist es für die Simulation des Driftsensors eher ungeeignet. Desweiteren hat ein Auto der Carrerabahn eine starre Vorderachse und somit keine Lenkwinkel. Auch basiert dieses Modell auf der stationären Kreisfahrt, die nie über den Grenzbereich einer Kurvenfahrt hinausgeht und ist somit meines Erachtens für den konkreten Fall der Carrerabahn ungeeignet. Bei der Bestimmung des Driftwinkels wird ja gerade das Verhalten eines Autos bei extremer Überschreitung des Grenzbereichs untersucht und in diesem Fall scheint mir das vorgestellte Modell mehr als ungeeignet und es wurde von einer Implementierung abgesehen.

5.5 Nichtlineares Einspurmodell⁴

Werden größere Schwimmwinkel und Schräglaufwinkel in der Bewegung erreicht, so wird das lineare Einspurmodell ungenau, da die auftretenden Reifenseitenkräfte nicht mehr als linear angesehen werden können. In diesem Fall wird das Modell des linearen Einspurmodells zum nichtlinearen Einspurmodell erweitert.

⁴ Vgl. [HeEr07] S.90 - 96

Mit diesem Modell ist es möglich die Querdynamik eines Fahrzeugs auch über den linearen Bereich hinaus zu simulieren. Bei der bisherigen Betrachtung des linearen Einspurmodells wurden die Reifenseitenkräfte senkrecht zur Fahrzeugmittellachse angenommen. Diese greifen jedoch senkrecht zu den Vorder- und Hinterradachse an.

Bei Betrachtung dieses Sachverhaltes kommt man unter Verwendung von Gleichung (5.6) zu den Folgenden beiden Gleichungen für die Längs- und Querrichtung:

$$m * a_x = -F_{SV} * \sin \delta_v - F_{SH} * \sin \delta_h \quad (5.17)$$

$$m * a_y = F_{SV} * \cos \delta_v + F_{SH} * \cos \delta_h \quad (5.18)$$

Durch Aufspaltung der Kraft in eine laterale und eine longitudinale Komponente, ergeben sich die Beschleunigungen mit Hilfe von:

$$\vec{a} = \dot{v} + \frac{v^2}{r} \quad (5.19)$$

Zu:

$$a_x = \dot{v} * \cos \beta + v(\dot{\psi} - \dot{\beta}) * \sin \beta \quad (5.20)$$

$$a_y = -\dot{v} * \sin \beta + v(\dot{\psi} - \dot{\beta}) * \cos \beta \quad (5.21)$$

Eingesetzt in (5.17), (5.18) und jeweils nach \dot{v} umgestellt ergibt sich draus:

$$\dot{v} = \frac{F_{SV} * \sin \delta_v + F_{SH} * \sin \delta_h}{m * \cos \beta} + \frac{v(\dot{\psi} - \dot{\beta}) * \sin \beta}{\cos \beta} \quad (5.22)$$

$$\dot{v} = \frac{F_{SV} \cdot \cos \delta_v + F_{SH} \cdot \cos \delta_h}{m \cdot \sin \beta} - \frac{v(\dot{\psi} - \dot{\beta}) \cdot \cos \beta}{\sin \beta} \quad (5.23)$$

Gleichstellen liefert:

$$\frac{F_{SV} \cdot \cos \delta_v + F_{SH} \cdot \cos \delta_h}{m \cdot \sin \beta} - \frac{F_{SV} \cdot \sin \delta_v + F_{SH} \cdot \sin \delta_h}{m \cdot \cos \beta} = v(\dot{\psi} - \dot{\beta}) \cdot \left[\frac{\cos \beta}{\sin \beta} + \frac{\sin \beta}{\cos \beta} \right] \quad (5.24)$$

Unter zu Verwendung des Satz des Pythagoras mit $\sin^2 \alpha + \cos^2 \alpha = 1$ und erweitern der rechten Seite ergibt sich:

$$\frac{(F_{SV} \cdot \cos \delta_v + F_{SH} \cdot \cos \delta_h) \cdot \cos \beta}{m \cdot \cos \beta \cdot \sin \beta} - \frac{(F_{SV} \cdot \sin \delta_v + F_{SH} \cdot \sin \delta_h) \cdot \sin \beta}{m \cdot \sin \beta \cdot \cos \beta} = \frac{v(\dot{\psi} - \dot{\beta})}{\sin \beta \cdot \cos \beta} \quad (5.25)$$

$$\frac{F_{SV} [\cos \delta_v \cdot \cos \beta - \sin \delta_v \cdot \sin \beta]}{m \cdot \cos \beta \cdot \sin \beta} + \frac{F_{SV} [\cos \delta_h \cdot \cos \beta - \sin \delta_h \cdot \sin \beta]}{m \cdot \sin \beta \cdot \cos \beta} = \frac{v(\dot{\psi} - \dot{\beta})}{\sin \beta \cdot \cos \beta} \quad (5.26)$$

Das Additionstheorem $\cos(\alpha + \beta) = \cos \alpha \cdot \cos \beta - \sin \alpha \cdot \sin \beta$ auf die Gleichung (5.21) angewandt, die beiden Gleichungen (5.10) und (5.11) eingesetzt, führen zu der Differentialgleichung für den Schwimmwinkel:

$$\dot{\beta} = \dot{\psi} - \frac{1}{m \cdot v} \left[c_v \cdot \left(\delta_v + \beta - \frac{l_v \cdot \dot{\psi}}{v} \right) \cos(\delta_v + \beta) + c_h \cdot \left(\delta_h + \beta + \frac{l_h \cdot \dot{\psi}}{v} \right) \cos(\delta_h + \beta) \right] \quad (5.27)$$

Da der vordere und hintere Lenkwinkel vom Betrag her null sind und der Drehpunkt beim Auto der Carrerabahn nicht im Schwerpunkt liegt ergibt sich die DGL zu:

$$\dot{\beta} = \dot{\psi} - \frac{1}{m \cdot v} \left[c_h \cdot \left(\beta + \frac{l \cdot \dot{\psi}}{v} \right) \cos(\beta) \right] \quad (5.28)$$

Der Drallsatz aus (5.4) wird mit Betrachtung der Lenkwinkel zu:

$$J_z * \ddot{\psi} = F_{SV} * l_v * \cos \delta_v - F_{SH} * l_h * \cos \delta_h \quad (5.29)$$

Auch hier werden wieder die Beziehungen aus (5.8) und (5.9) eingesetzt und es ergibt sich:

$$\ddot{\psi} = \frac{1}{J_z} \left[c_v * \left(\delta_v + \beta - \frac{l_v * \dot{\psi}}{v} \right) \cos \delta_v - c_h * \left(\delta_h + \beta + \frac{l_h * \dot{\psi}}{v} \right) \cos \delta_h \right] \quad (5.30)$$

Auch hier werden die beiden Lenkwinkel δ_v und δ_h gleich null gesetzt, der Drehpunkt in die Vorderachse gelegt und es ergibt sich nun die DGL:

$$\ddot{\psi} = \frac{1}{J_z} \left[c_h * \left(\beta + \frac{l * \dot{\psi}}{v} \right) \right] \quad (5.31)$$

Auch beim nichtlinearen Einspurmodell kam ich zu dem Schluss dass es für die Simulation ungeeignet ist. Im Prinzip stellt es nur eine Erweiterung des linearen Einspurmodells dar, bei dem die hinteren Räder lenkbar sind. Ansonsten gelten dieselben Einschränkungen für dieses Modell wie für das lineare Einspurmodell.

5.6 Nichtlineares Zweispurmodell⁵

Ein genaueres Modell um das Fahrverhalten eines Autos zu simulieren ist das nichtlineare Zweispurmodell. Dieses Modell schließt die Kipp- und Wankdynamik eines Fahrzeugs mit in die Betrachtung der kinematischen und dynamischen Vorgänge ein. Dies ist für die Simulation zum jetzigen Zeitpunkt nicht von Bedeutung, könnte aber in späteren Szenarien, z.B. mit LKW – Modellen, durchaus eine Rolle spielen und soll deshalb trotzdem hier kurz vorgestellt werden. Dieses Modell besitzt gegenüber den Einspurmodellen zwei Freiheitsgrade. Es basiert auf den Newton - Euler'schen Gleichungen.

⁵ Vgl. [Ise06]

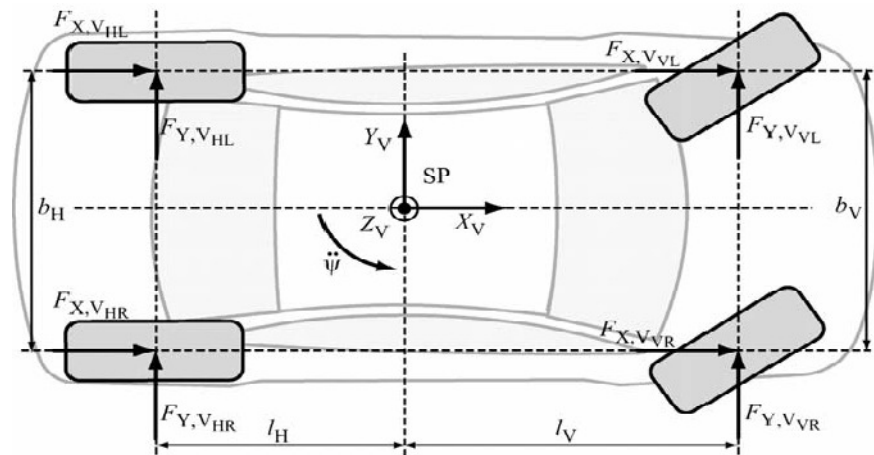


Abbildung 10: Nichtlineares Zweispurmodell (Horizontaldynamik)

Quelle [Ise06]

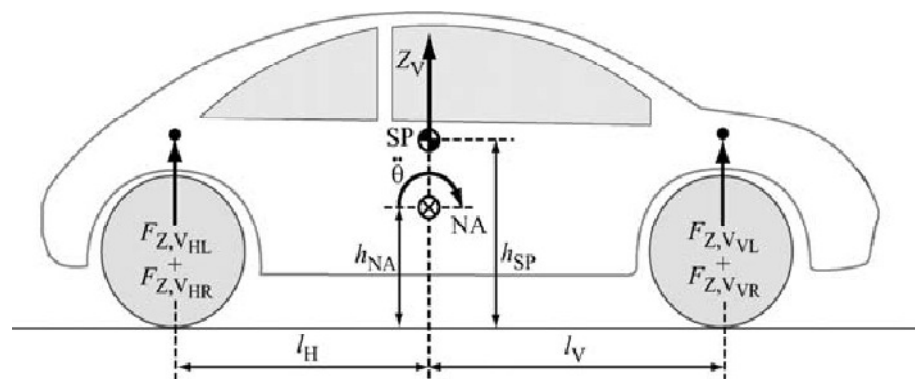


Abbildung 11: Nichtlineares Zweispurmodell (Nickdynamik)

Quelle [Ise06]

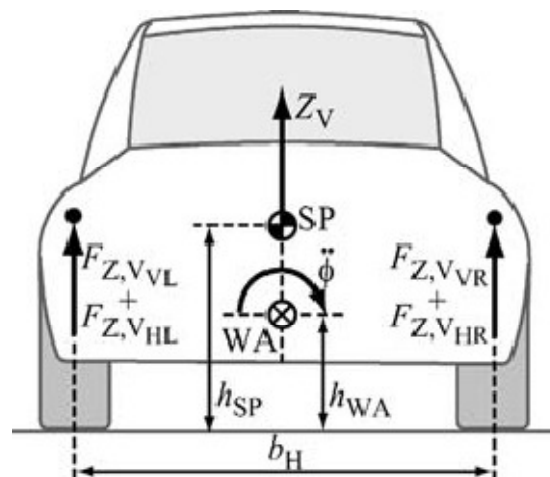


Abbildung 12: Nichtlineares Zweispurmodell (Wankdynamik) Quelle [Ise06]

Mit Hilfe des Impulssatzes nach Newton können die Kräftegleichungen entlang der drei Bewegungsrichtungen x, y und z aufgestellt werden. Die drei Gleichungen lauten wie folgt:

$$F_X = m * \dot{v}_X + (\dot{\Theta} * v_Z + \dot{\psi} * v_Y) \quad (5.32)$$

$$F_Y = m * \dot{v}_Y + (\dot{\psi} * v_X + \dot{\phi} * v_Z) \quad (5.33)$$

$$F_Z = m * \dot{v}_Z + (\dot{\phi} * v_Y + \dot{\Theta} * v_X) \quad (5.34)$$

Die Formeln zur Berechnung der resultierenden Kräfte in Längs- und Querrichtung und der Fahrzeughochachse lauten folgendermaßen:

$$F_X = F_{XVL} + F_{XVR} + F_{XHL} + F_{XHR} - m * g * \sin \alpha_X - F_{XLW} \quad (5.35)$$

$$F_Y = F_{YVL} + F_{YVR} + F_{YHL} + F_{YHR} - m * g * \sin \alpha_Y - F_{YLW} \quad (5.36)$$

$$F_Z = F_{ZVL} + F_{ZVR} + F_{ZHL} + F_{ZHR} - m * g + F_{ZAUF} \quad (5.37)$$

Hierbei sind die α_X und α_Y die Neigung der Fahrbahn in Quer- und Längsrichtung. Die Kräfte F_{XLW} und F_{YLW} stellen die Luftkräfte in Quer- und Längsrichtung dar, F_{ZAUF} die Kraft die durch den Auftrieb des Autos erzeugt wird. Zur Berechnung der Kräfte sei an dieser Stelle auf [Ise06] verwiesen.

Mit Hilfe der Euler'schen Kreisgleichung werden nun noch die drei möglichen Drehbewegungen um die Längs-, Quer- und Hochachse bestimmt.

Dies führt zu den Folgenden drei Gleichungen:

$$J_X * \ddot{\phi} = M_X + \dot{\Theta} * \dot{\psi} * (J_Y - J_Z) \quad (5.38)$$

$$J_Y * \ddot{\Theta} = M_Y + \dot{\psi} * \dot{\phi} * (J_Z - J_X) \quad (5.39)$$

$$J_Z * \ddot{\psi} = M_Z + \dot{\phi} * \dot{\theta} * (J_X - J_Y) \quad (5.40)$$

Durch bremsen und beschleunigen wird eine Nickbewegung an der Karosserie eingeleitet. Diese wird durch die im Schwerpunkt des Fahrzeugs angreifende Trägheitskraft ausgelöst. Das Durchfahren einer Kurve bewirkt eine Wankbewegung des Autos durch die im Schwerpunkt angreifende Zentrifugalkraft. Die Wank- und Nickkräfte sind durch die folgenden zwei Gleichungen gegeben:

$$F_{TX} = -F_X = -m * a_X \quad (5.41)$$

$$F_{TY} = -F_Y = -m * a_Y \quad (5.42)$$

Mit der Annahme das der Nick- und Wankpol senkrecht unter dem Schwerpunkt des Autos liegen, lassen sich die Gleichungen für die resultierenden Momente folgendermaßen angeben:

$$M_X = (F_{ZVL} - F_{ZVR}) * \frac{b_V}{2} + (F_{ZHL} - F_{ZHR}) * \frac{b_H}{2} - (h_{SP} - h_{WA}) * F_{TY} \quad (5.43)$$

$$\begin{aligned} M_Y = & (F_{ZHL} + F_{ZHR}) * l_H - (F_{ZVL} + F_{ZVR}) * l_V - (h_{SP} - h_{NA}) * F_{TX} + \dots \\ & \dots + (l_H * F_{AH} - l_V * F_{AV}) \end{aligned} \quad (5.44)$$

$$\begin{aligned} M_Z = & (F_{XVR} - F_{XVL}) * \frac{b_V}{2} + (F_{XHR} - F_{XHL}) * \frac{b_H}{2} + (F_{YVL} + F_{YVR}) * l_V - \dots \\ & \dots - (F_{YHL} + F_{YHR}) * l_H \end{aligned} \quad (5.45)$$

Durch einsetzen von (5.43) in (5.38), (5.44) in (5.39) und (5.45) in (5.40) können die Winkelbeschleunigungen um die X -, Y - und Z - Achse des Fahrzeugs berechnet werden. Durch Integration der Gleichungen (5.38), (5.39) und (5.40) können die jeweiligen Winkelgeschwindigkeiten bestimmt werden.

Setzt man (5.35) in (5.32), (5.36) in (5.33) und (5.37) in (5.34) ein so können unter Berücksichtigung der Gleichungen (5.38), (5.39) und (5.40) die Beschleunigungen in X -, Y -

und Z - Richtung des Fahrzeugs ermittelt werden. Auch hier kann durch Integration der Gleichungen (5.32), (5.33) und (5.34) die Geschwindigkeit in der jeweiligen Richtung ermittelt werden.

Von einer Implementierung dieses Modells als Lösung für die Bestimmung des Driftwinkels hab ich abgesehen, da es meiner Meinung nach zu viel Overhead mit sich bringt. Für die Simulation insbesondere für die Realisierung des Driftwinkels denke ich reicht eine einfache schnelle Lösung aus, die auch auf Kosten der Genauigkeit gehen kann. Da bei diesem Modell alle möglichen Rotationsbewegungen des Kraftfahrzeugs berücksichtigt werden halte ich diese Modell für zu kompliziert um nur den Driftwinkel eines Autos auf der Carrerabahn zu bestimmen.

5.7 Einfaches Modell für den Schwimmwinkel

Das Modell wird auf der Basis von einfachen Bewegungs- und Kräftegleichungen entwickelt. Das Ziel bei der Simulation ist es im Moment des Kurveneintritts die Zeit zu bestimmen, die das Auto bis zur Auslösung des Sensors benötigt, unter der Berücksichtigung das das Auto zu schnell in die Kurve einfährt und auszubrechen beginnt. Für die Simulation ist nur die Dynamik in Querrichtung und Längsrichtung von Bedeutung. Kipp-, Nick- oder Lastwechsellvorgänge wie sie in der Realität bei diesem Vorgang auftreten können vernachlässigt werden. Da die Autos der Carrerabahn durch eine Schiene geführt werden, wird beim überschreiten des Grenzbereichs nur das Übersteuern betrachtet, da sich ein Drehmoment um die Schienenführung aufbaut. Dieses Drehmoment veranlasst das Auto über das Heck zum Kurvenäußeren auszubrechen. Die Beschleunigung die für diesen Vorgang notwendig ist wird durch die Zentrifugalkraft erzeugt.

Die Seitenführungskräfte der Reifen werden in diesem vereinfachten Modell durch die Gleitreibungskraft zwischen Reifen und Fahrbahn ersetzt. Diese Vereinfachung der Seitenführungskräfte wird damit begründet, dass die Gleichungen für die Reifenführungskräfte für echte Fahrzeugreifen gedacht sind, die bekanntlich einen komplizierten Aufbau aus unterschiedlichsten Materialien und Profilen haben. Ein echter Reifen ist mit Luft gefüllt wo durch es zu komplexen physikalischen Vorgängen zwischen Reifen und Fahrbahnoberfläche kommt. Die Reifen an einem Auto der Carrerabahn sind Vollgummireifen, weshalb für dieses Modell nur die geeigneten Reibungskoeffizienten für Haft- und Gleitreibung gefunden werden müssen. Die Koeffizienten können mittels Versuche an einer Steigung der Bahn empirisch Ermittelt werden. Dazu wird die Steigung der Bahn solange erhöht bis das Auto, das quer zur Fahrbahn steht, von alleine aus dem Stillstand zu rutschen beginnt. Man bestimmt den Winkel der Bahnsteigung und ist somit in der Lage den Haftreibungskoeffizienten für die Reifen des entsprechenden Fahrzeugs zu bestimmen. Der Haftreibungskoeffizient ist gerade $\tan \alpha$.

Um den Rollreibungskoeffizienten zu ermitteln wird der Vorgang der Bergabfahrt betrachtet. Man kann die Zeit und die Geschwindigkeit die ein Auto zum herunterfahren mit konstanter Geschwindigkeit und ohne Reibungseinflüsse benötigen würde berechnen. Diese Rechnung vergleicht man mit dem Vorgang in der Realität. Durch Anbringung von zwei Lichtschranken

am Berg und im Tal kann man die Zeit und die Geschwindigkeit messen. Durch Anwendung des Geschwindigkeits-Zeit-Gesetzts für die beschleunigte Bewegung an der schiefen Ebene kann unter der Verwendung der gemessenen Werte für Endgeschwindigkeit und Zeit der Wert für den Gleitreibungskoeffizienten ungefähr berechnet werden

$$F = F_H - F_K \quad (5.46)$$

$$a = g(\sin \alpha - \mu * \cos \alpha) \quad (5.47)$$

$$v_{(t)} = v_0 + a * t \quad (5.48)$$

(5.47) eingesetzt in (5.48) und nach μ_R umgestellt ergibt dann:

$$\mu_R = \frac{v_0 - v + g * \sin \alpha * t}{g * \cos \alpha * t} \quad (5.49)$$

Um den Gleitreibungskoeffizienten zu bestimmen muss auch hier das Auto in der Bewegung betrachtet werden. Die beste Methode um den Koeffizienten zu bestimmen ist eine lange gerade, wie bei einer Steilkurve, schräg aufzubauen. Nun lässt man ein Auto langsam auf dieser geraden fahren. Man erhöht nun den Winkel der Fahrbahn solange bis das Heck des Autos bei der langsamen Fahrt wegrutscht. Auch hier ergibt sich der Reibungskoeffizient wieder zu $\tan \alpha$.

Der gesamte Vorgang beim Driften besteht aus zwei Kreisbewegungen die sich gegenseitig beeinflussen. Die Zentrifugalkraft, die durch die erste Kreisbewegung erzeugt wird, hängt von der Geschwindigkeit des Autos bei der Kurvenfahrt ab. Durch das Ausbrechen des Hecks wird die zweite Kreisbewegung um den Drehpunkt in der Vorderachse eingeleitet und es entsteht nach Überwindung der Haftreibung des Reifens eine Gleitreibungskraft. Diese Gleitreibungskraft sorgt dafür, dass das Auto bei der Kurvenfahrt langsamer wird und die Zentrifugalkraft abnimmt. Diese Abnahme der Zentrifugalkraft beeinflusst die Geschwindigkeit mit dem der Schwimmwinkel beim Driftvorgang zu nimmt.

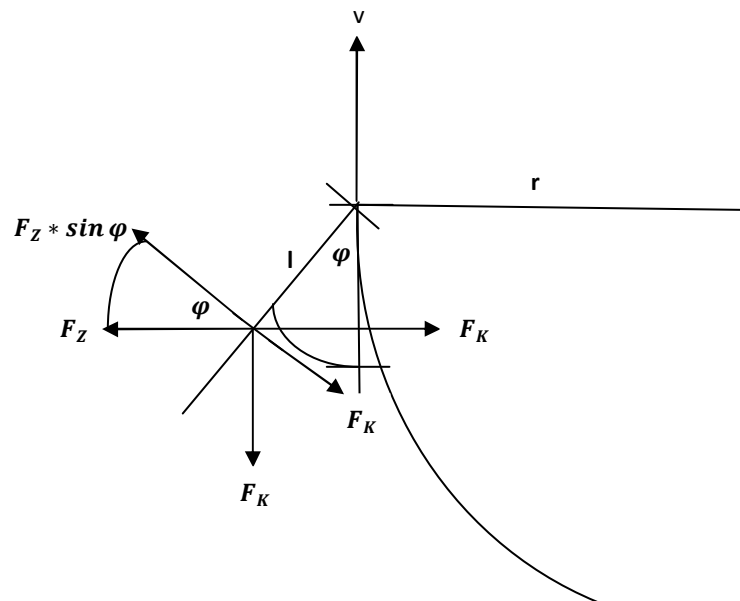


Abbildung 13: Kräfte bei der Kurvenfahrt mit Übersteuern

Bei diesem Modell wird angenommen, dass das Auto beim überschreiten des Grenzbereich in der Kurve eine gleichförmig verzögerte Bewegung beschreibt. Diese Bewegung ist durch die folgende Gleichung gegeben.

Ansatz mit Gleichgewichtsbedingung für stationäre Kreisfahrt:

$$F_Z \leq F_R \quad (5.50)$$

Resultierende Beschleunigung in lateraler Richtung:

$$F = F_Z - F_K \quad (5.51)$$

$$m * a_y = m * \frac{v^2}{r} - m * g * \mu \quad (5.52)$$

$$a_y(t) = \frac{v^2(t)}{r} - g * \mu \quad (5.53)$$

Mit der Verzögerung die durch die Reibung der zweiten Kreisbewegung auftritt, ergibt sich die Geschwindigkeit zu:

$$v^2(t) = (v_0 - g * \mu * t)^2 \quad (5.54)$$

(5.54) eingesetzt in (5.53) ergibt:

$$a_y(t) = \frac{(v_0 - g * \mu * t)^2}{r} - g * \mu \quad (5.55)$$

Die Zeit die das Auto bis zum Erreichen des Drift-Sensors benötigt lässt sich mit dem Weg-Zeit-Gesetz der gleichförmig verzögerten Bewegung ausrechnen:

$$s_{(t)} = v_0 * t - \frac{1}{2} a * t^2 \quad (5.56)$$

(5.56) nach t umgestellt ergibt sich folgende quadratische Gleichung:

$$t_{1/2} = \frac{v_0}{\mu * g} \pm \sqrt{\left(\left(\frac{v_0}{\mu * g}\right)^2 - \frac{2s}{\mu * g}\right)} \quad (5.57)$$

Die Geschwindigkeit am Sensor ergibt sich aus dem Geschwindigkeits-Zeit-Gesetz der gleichförmig verzögerten Bewegung:

$$v_{(t)} = v_0 - \mu * g * t \quad (5.58)$$

Für die gleichförmig beschleunigte Kreisbewegung gelten Folgende Beziehungen:

$$\omega_{(t)} = \alpha * t = \frac{d\varphi}{dt} \quad (5.59)$$

$$\varphi_{(t)} = \int_0^t \omega dt = \int_0^t \alpha * t = \frac{1}{2} \alpha * t^2 \quad (5.60)$$

Dies führt nun zu Folgender Beziehung:

$$\varphi_{(t)} = \frac{1}{2} \left((a_y(t) * \sin \varphi) - \mu * g \right) * t^2 \quad (5.61)$$

Gleichung (5.55) in (5.61) eingesetzt ergibt:

$$\varphi_{(t)} = \frac{1}{2} \left(\left(\left(\frac{(v_0 - g * \mu * t)^2}{r} - g * \mu \right) * \sin \varphi \right) - \mu * g \right) * t^2 \quad (5.62)$$

Dieser Ansatz führte leider nicht zum Ziel, da der zu berechnende Winkel auch in der rechten Seite der Gleichung auftaucht und nicht zu eliminieren war. Aus diesem Grund wurde dieser Ansatz nach längerer Überlegung nicht weiter verfolgt. Lediglich die Ansätze zur Bestimmung der Reibungskoeffizienten erweisen sich als nützlich. Mit ihnen kann man eine Vereinfachung für das lineare Einspurmodell treffen in dem man die Schräglaufsteifigkeit durch die Gleitreibungskraft ersetzt.

5.8 Winkelbestimmung durch neuronales Netz

Eine weitere Möglichkeit zur Implementierung des Driftsensors wäre das entwerfen und trainieren eines neuronalen Netzes. Da die Carrerabahn während der Erstellung der Diplomarbeit noch nicht komplett fertig aufgebaut war konnte dieser Ansatz nicht durchgeführt werden. Die Driftsensoren sind noch nicht in der Bahn eingebaut und deshalb war es nicht möglich Trainingsdaten für ein neuronales Netz zu gewinnen. Ich möchte hier aber trotzdem ein paar Überlegungen zu diesem Ansatz und zu einer möglichen Architektur des neuronalen Netzes anstellen. Ich würde für dieses Problem ein vorwärtsgerichtetes Backpropagation-Netz wählen. Diese Art von neuronalen Netzen bestehen aus einer Eingangsschicht, einer Ausgangsschicht und ein oder mehreren verdeckten Zwischenschichten. Es würde das überwachte Lernen bei diesem neuronalen Netz eingesetzt.

Die Theorie besagt das jede kontinuierliche Abbildung der Form „ $y = f(x)$ “ durch ein Netz mit nur einer verdeckten Schicht gelöst werden kann [Bit07], deshalb würde ich zur Bestimmung des Driftwinkels ein Netz mit genau einer verdeckten Schicht verwenden. Bei den Neuronen in den Zwischenschichten gibt es kein Rezept mit dem man die Anzahl ermitteln könnte. Man kann nur soviel sagen, dass zu wenig Neuronen ein langsames lernen zur Folge hat und zu viel Neuronen das Netz in einen Speicher verwandeln können da es schlechter generalisiert. Die Anzahl sollte durch ausgiebiges testen und experimentieren herausgefunden werden.

Die Anzahl der Neuronen der Eingangsschicht und der Ausgangsschicht sind klar durch das zu lösende Problem vorgegeben. Bei der Ausgangsschicht würde ich vierzehn Neuronen verwenden, da der Driftsensor vierzehn Werte annehmen kann. Das Ergebnis wäre dann in der 1:n Darstellung der Ausgangsschicht ablesbar. Die Anzahl der Neuronen in der Eingangsschicht wird durch die Anzahl der betrachteten Merkmale, also der Eingangsparameter bestimmt. Als mögliche Eingangsparameter könnte man die Geschwindigkeit, den Radius, das Bogenmaß von Kurveneintritt bis zur Sensorposition, das Fahrzeuggewicht, den Haft- und Gleitreibungskoeffizient, die Distanz von der Finne zur Hinterachse, die Bahnsteigung und die Reifenbreite verwenden. Dies sind dann neun Merkmale für das zu lösende Problem. Daraus ergeben sich neun Neuronen in der Eingangsschicht. Es ist beim Anwenden des fertigen neuronalen Netzes noch zu untersuchen ob eine Normierung der Werte z.B. in einem Intervall zwischen null und eins eine positive Auswirkung auf die Genauigkeit der Ergebnisse hat.

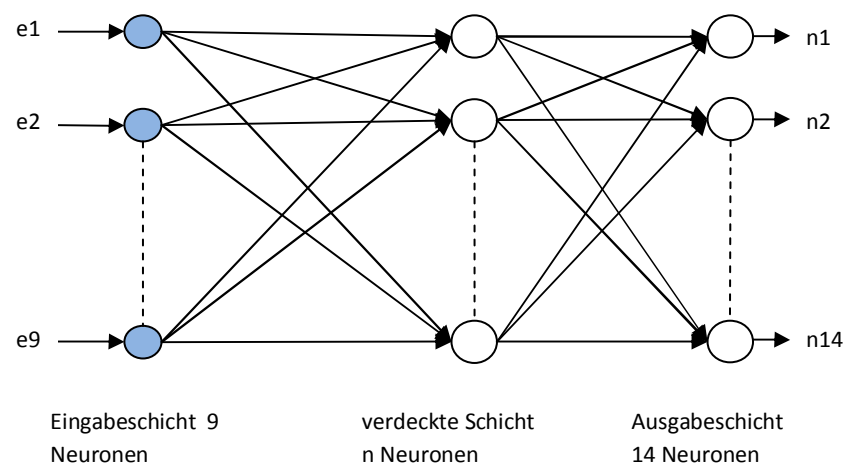


Abbildung 14: Neuronales Netz mit einer versteckten Zwischenschicht

Als Aktivierungsfunktion in der verdeckten Schicht wird eine binär, sigmoide Funktion benötigt. Eine solche Funktion hat eine s-förmige Darstellung. Sie stellt eine reelle und differenzierbare Funktion innerhalb des Intervalls $[0, 1]$ dar. Diese Darstellung wird benötigt, da in der Ausgabeschicht eine 1:n Darstellung verwendet wird und die Differenzierbarkeit von einem Backpropagation Netz vorausgesetzt wird. In MATLAB wird einem hierfür nur die logistische Funktion zur Verfügung gestellt. Diese würde dann im neuronalen Netz zum Einsatz kommen. Diese Funktion wird durch folgende Gleichung beschrieben:

$$P_{(t)} = \frac{1}{1+e^{-t}} \quad (5.63)$$

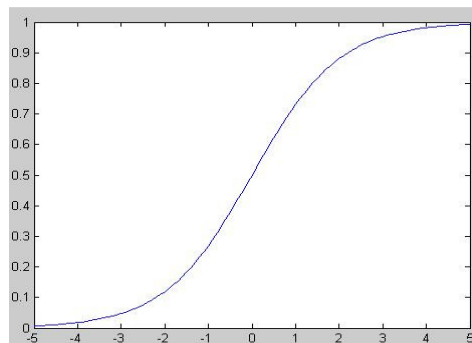


Abbildung 15: Funktion logsig in MATLAB

Als Trainingsalgorithmus könnte man den Levenberg-Marquard-Algorithmus oder den Resilient-Backpropagation Algorithmus verwenden. Der Levenberg-Marquard-Algorithmus ist schneller als die meisten anderen Algorithmen die zum trainieren von neuronalen Netzen verwendet werden. Er kann aber zu Problemen führen, da er einen hohen Speicherbedarf hat wenn mit großen Lernmengen gearbeitet wird. Als Netzperformancefunktion würde ich den „mean-square-error“, also den mittleren quadratischen Fehler verwenden. Die Einstellungen für maximale Epochenzahl und für das Fehlerziel muss wieder durch testen und experimentieren herausgefunden werden. Der Wert für das Fehlerziel darf nicht zu groß und nicht zu klein gewählt werden, da das Netz sonst untrainiert bleibt oder nicht innerhalb der eingestellten Epochenzahl zum Ergebnis führt. Des Weiteren wäre noch zu Untersuchen in wie fern sich eine Vorinitialisierung des neuronalen Netzes auf die Performance und die Qualität des Ergebnisses auswirkt [Bit07].

Es ist auch zu Untersuchen wie sich die Aufbereitung der Daten in Trainings- und Testdaten auf die Lernfähigkeit und die Ergebnisse des neuronalen Netzes auswirken. Die Daten die man durch Versuche auf der Carrerabahn ermitteln muss sind in 14 Cluster aufteilbar. Jeder Driftsensorwert zwischen eins und vierzehn bildet ein Cluster. Man könnte hier zwei Strategien verfolgen. Entweder teilt man die Daten prozentual oder in immer gleichen Teilen auf.

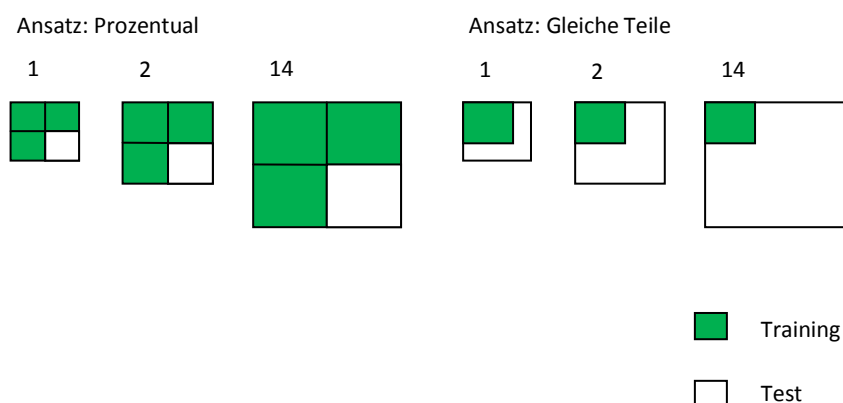


Abbildung 16: Aufteilung der Daten

Ich denke dass ein neuronales Netz eine gute Möglichkeit wäre um den Driftwinkel eines Autos zu bestimmen. Da es mir aber Aufgrund der Tatsache, dass die Driftsensoren noch nicht auf der Carrerabahn angebracht waren, nicht möglich war Test-und Trainingsdaten zu ermitteln, wurde dieser Ansatz verworfen.

5.9 Driftsensor als Wertetabelle

Für die Simulation habe ich mich für ein einfaches Modell entschieden um das Fahrzeug und damit auch den Driftsensor zu realisieren. Jedes Fahrzeug, das auf der Carrerabahn benutzt wird ist als XML-Datei hinterlegt. In dieser Datei sind die möglichen Geschwindigkeitswerte die das Fahrzeug annehmen kann hinterlegt. In der Datei gibt es eine „Speedmap“, welche alle von dem Fahrzeug realisierbaren Geschwindigkeiten enthält. Die Werte sind durch Messungen mit den Fahrzeugen auf der realen Carrerabahn zu ermitteln. Ein Geschwindigkeitseintrag beinhaltet zwei Werte und eine weitere Struktur für den Schleudersensor. Der erste Wert ist ein hexadezimaler Zahlenwert der zweite Wert entspricht der Geschwindigkeit in mm/s. Die Schleudersensorstruktur enthält wiederum je eine Struktur pro Fahrzeugspur.

In den Strukturen für die Fahrzeugspuren sind die Werte des Schleudersensors, für die aktuelle Geschwindigkeit, für die sieben möglichen Kurven der Carrerabahn hinterlegt. Der Sensor kann Werte zwischen 1 und 14 annehmen. Die jeweiligen Werte sind wie die Geschwindigkeitswerte auf der realen Bahn für das Fahrzeug durch Messungen und Versuche zu ermitteln. Wenn der Schleudersensor einen Wert von -1 eingetragen hat, so bedeutet dies, dass das Auto bei der betreffenden Geschwindigkeit, auf seiner Spur in der jeweiligen Kurve heraus geflogen ist, da seine Geschwindigkeit für diese Kurve zu schnell war.

Ich denke dass dieser Ansatz effizient ist, da auf viele Rechnungen verzichtet, was sich wiederum positiv auf das Laufzeitverhaltender Simulation auswirkt. Das nachschauen in einer Tabelle ist um ein vielfaches schneller als die Integration von zwei oder mehreren Differentialgleichungs- systemen. Ich denke auch die damit erreichbare Genauigkeit der Simulation ist ausreichend, da ja nicht die exakte physikalische Simulation der Carrerabahn als Ziel im Vordergrund steht, sondern eine möglichst genaue. Ich denke das dieser Ansatz ausreichend ist um die Simulation in ihrem Einsatzgebiet, dem Testen von Programmen die Autos auf der Carrerabahn steuern, erfolgreich einzusetzen.

Kapitel 6

6. Simulation

6.1 Betriebssystem

Die Simulation wurde unter dem Betriebssystem Linux entwickelt. Die Wahl dieses Betriebssystems begründet sich durch den Vorschlag von Prof. Dr. Michael Mächtel, die Simulation auf einer schon vorhandenen, ähnlichen Simulation aufzubauen. Gerade die Tatsache, dass der Treiber den mir Prof. Mächtel zur Verfügung gestellt hat, für das Betriebssystem Linux geschrieben wurde, schränkte die Auswahl eines anderen Betriebssystems enorm ein. Da Linux eine offene API zur Treiberentwicklung bietet und der Treiber von Prof. Mächtel für das Betriebssystem Linux geschrieben wurde, wurde Linux verwendet.

Zum Einsatz kam die Distribution Ubuntu 7.10 als 32 Bit Installation. Als Betriebssystem-kernel wurde Version 2.6.22-14 verwendet. Ubuntu ist eine freie Linux-Distribution und basiert auf der Distribution Debian. Das Projekt Ubuntu wird von der Firma Canonical Ltd. unterstützt und finanziert. Ziel dieser Distribution ist es ein einfach zu installierendes und zu wartendes Betriebssystem auf Linux-Basis bereitzustellen, zu garantieren das die benutzten Pakete und Bibliotheken auf einander abgestimmt sind und Benutzern die das arbeiten mit Windows gewöhnt sind einen schnellen Einstieg zu gewähren. Es gibt verschiedenen Versionen der Distribution mit die auf unterschiedliche Desktops aufsetzen. Die von mir gewählte Version benutzt als Desktop KDE und wird deshalb Kubuntu genannt. Diese Wahl hat aber keinen Einfluss auf die Qualität der Arbeit sondern ist eine persönliche Vorliebe meinerseits. Genauso gut hätte man Ubuntu, welches auf Gnome aufsetzt, oder Xubuntu welches auf Xfce aufsetzt zur Entwicklung benutzen können.

6.2 Programmiersprache und Entwicklungsumgebung

Als Programmiersprache wurde Ansi C eingesetzt. Die Wahl hierauf viel auch auf Grund der Tatsache, dass die Simulation und der Treiber, welcher mir von Prof. Mächtel zur Verfügung gestellt wurde in dieser Sprache geschrieben wurde. Eine andere Möglichkeit wäre eine Neuentwicklung bzw. Teilportierung in der Sprache C++ gewesen. Diese Alternative wurde aber wegen unnötigem Mehraufwand von mir verworfen. Bei der Versionsverwaltung wurde auf die Open-Source Software Subversion SVN gesetzt, da es über einige Vorteile gegenüber der Alternative CVS verfügt. So ist SVN wesentlich schneller als CVS und

Subversion versioniert auch Verzeichnisse, Umbenennungen und Kopien von Dateien. Mit SVN ist es möglich mit binären Dateien umzugehen und die einzelnen Commits sind atomar, d.h. ein Schreibzugriff wird erst dann wirksam wenn der gesamte Vorgang abgeschlossen ist. Bei der Wahl einer geeigneten Entwicklungsumgebung standen mehrere Alternativen zur Diskussion. Allen sollte gemein sein, dass es frei verfügbare, sogenannte Open-Source-Software sein sollte. Die erste Möglichkeit wäre die Verwendung von einem normalen Texteditor mit Hilfe der Kommandozeile, in Verbindung mit den Werkzeugen gcc und gdb, unter Kubuntu. Diese Art der Entwicklung wurde aber schon im Vorfeld von mir abgelehnt, da ich wenn möglich eine Entwicklungsumgebung zum erstellen des Codes benutzen wollte. Der Vorteil einer Entwicklungsumgebung gegenüber der ersten Möglichkeit liegt klar in der besseren Bedienbarkeit und Effizienz beim erstellen von Code. Dies fängt mit der Hilfe bei der Codeformatierung an und geht bis hin zur Codevervollständigung.

Eine Entwicklungsumgebung die es für Kubuntu gibt und die auch einfach über den Paketmanager Adept unter Kubuntu installiert werden kann heißt KDevelop. Mit dieser IDE ist es möglich Projekte mit C/C++ und vielen anderen Programmiersprachen anzulegen. Die Entwicklungsumgebung verfügt über eine grafische Debugger-Schnittstelle die den gdb verwendet. Über Plugin-Werkzeuge ist es jederzeit möglich den Funktionsumfang der IDE zu erweitern. Um grafische Benutzeroberflächen zu erstellen verfügt KDevelop über einen Designer der das Trolltech Framework Qt benutzt. Desweiteren verfügt die IDE über einen Automake Manager, der die Verwendung der GNU Entwicklungstools zum erstellen von Software, vereinfacht. Mit ihm wird die Erstellung von Makefiles und configure – Skripts einfacher.

Meine Wahl fiel dann aber auf die Plattform Eclipse. Da Eclipse heute schon in vielen kommerziellen Softwareentwicklungen eingesetzt wird und von einer Reihe großer namhafter Firmen unterstützt wird, fiel die Wahl auf diese IDE recht schnell. Eclipse ist eine in Java geschriebene freie Entwicklungsumgebung. Daraus ergibt sich ein weiterer nicht zu unterschätzender Vorteil von Eclipse gegenüber anderen Entwicklungsumgebungen, die Plattformunabhängigkeit. Alles was zum arbeiten mit dieser IDE benötigt wird ist ein Betriebssystem auf dem eine Java Runtime läuft. Der Funktionsumfang kann auch bei dieser Entwicklungsumgebung mit Hilfe von Plugin-Werkzeugen erweitert werden. Unter Kubuntu Eclipse wie die meiste Software auch über den Adept Paketmanager heruntergeladen und installiert werden. In dieser Diplomarbeit kam die Version 3.3.1.1 zum Einsatz. In der Grundinstallation verfügt Eclipse nur über die notwendigen Plugins um ein Java Projekt zu erstellen. Für die Verwendung als Entwicklungsumgebung um die Simulation zu erstellen, mussten erst noch die notwendigen Plugins installiert werden. Hier kam schon ein großer Vorteil von Eclipse gegenüber anderen IDE zum tragen. Eclipse kann aus der IDE heraus auf dem Neuesten Stand gehalten und um neue Funktionalität erweitert werden. Als erste Erweiterungen wurden das CDT Plugin in der Version 4.0.3.2 installiert. Mit diesem Plugin ist es möglich C/C++ Projekte unter Eclipse zu erstellen. Es benutzt die Standard GNU Entwicklungstools gcc, gdb und make, zum linken, kompilieren und debuggen von C/C++ Programmen und Bibliotheken. Auch in diese IDE bietet eine grafische Schnittstelle zum komfortablen debuggen mit dem gdb. Um aus der Entwicklungsumgebung direkt mit der Versionsverwaltung arbeiten zu können, wurde das Plugin Subclipse in der Version 1.2.4 installiert. Subclipse ist ein Plugin das auf der Versionsverwaltungssoftware Subversion SVN

aufsetzt und dadurch die gesamte Funktionalität von SVN zur Verfügung stellt. In einem extra Ansichtsfenster, unter Eclipse heißen diese Fenster Perspektive, können bekannte Repositories durch die Eingabe von URL und Benutzerdaten ein gepflegt werden. In der SVN-Perspektive kann das Repository dann in einer Baumansicht betrachtet werden. In der Perspektive für die C/C++ Entwicklung können die Änderungen im Projektbaum an das Repository übertragen werden. Mit Eclipse hat man die Wahl ob man sich die Makefiles zum erstellen der ausführbaren Datei selbst erstellen will oder ob das Eclipse automatisch übernimmt. Es bedarf am Anfang einer gewissen Einarbeitungszeit bis man sich mit allen Funktionen und Einstellungen der IDE vertraut gemacht hat aber wenn diese erste Hürde geschafft ist, so ist das Arbeiten und der Funktionsumfang mit bzw. von Eclipse eine Erleichterung. Lediglich die Geschwindigkeit beim Starten der IDE könnte man als einen Kritikpunkt anbringen, da alle Plugins beim Start geladen werden. Man sollte also darauf achten den Funktionsumfang der IDE dem entsprechenden Projekt anzupassen und lieber für unterschiedliche Projekte mehrere angepasste Installationen zu benutzen, als eine Eclipse Installation mit allen verfügbaren Erweiterungen.

6.3 Verwendete Bibliotheken

Um mit dem Betriebssystem Kubuntu Software zu entwickeln, sollte sichergestellt sein, dass die Standard GNU Entwicklungswerkzeuge installiert sind. Die Standard C-Bibliothek wird automatisch bei der Installation des Betriebssystems mit installiert, die Entwicklungswerkzeuge jedoch nicht. Man kann die Werkzeuge jedoch wie bei den Entwicklungsumgebungen einfach über den Adept-Paketmanager nachinstallieren. Dazu muss nur das Paket „build-essentials“ installiert werden. Mit diesem Paket werden alle wichtigen Werkzeuge zum erstellen von C/C++ Programmen installiert. Zum debuggen sollte man dann noch das Paket gdb über den Adept Paketmanager installieren.

Da alle Daten für die Autos und die Strecke in XML-Dateien gespeichert sind, benötigt man noch eine Bibliothek, die den Umgang mit diesem Standard erleichtert. Hierfür benötigt man einen XML-Parser. Ein frei erhältlicher XML-Parser ist der Expat-Parser. Er ist in C geschrieben und ist unter der GPL Lizenz veröffentlicht. Um den XML-Parser zu benutzen müssen die Pakete „libexpat1“ und „libexpat1-dev“ auf dem Entwicklungsrechner installiert werden. Dies kann auch wieder einfach über den Adept Paketmanager vorgenommen werden. Das Paket „libexpat1“ muss auch auf dem Rechner installiert sein auf dem die Simulation letztendlich läuft. Handelt es sich dabei um eine 64 Bit Installation des Betriebssystems Linux, so muss das Paket „lib64expat1“ installiert werden. Um den Umgang mit den XML-Dateien noch weiter zu erleichtern, wurde eine weitere Bibliothek verwendet. Die Bibliothek heißt „scew“ und stellt nichts anderes als eine einfache in C geschriebene Wrapper-Bibliothek dar. Über simple Schnittstellen kann hiermit ein Zugriff auf die Funktionalitäten des Expat-Parsers erfolgen. Auch diese Software ist frei erhältlich und fällt unter die LGPL Lizenz. Diese Bibliothek ist jedoch nicht mit dem Adept Paketmanager installierbar. Man kann die Quellen auf der Seite <http://www.nongnu.org/scew/> herunterladen. Die Quellen wurden dann einfach kompiliert und auf dem Rechner installiert.

6.4 Genereller Ablauf

In diesem Abschnitt soll ein kurzer Überblick über den generellen Ablauf der Simulation gegeben werden, insbesondere welche Schritte beim Start des Programms ausgeführt werden. Nach dem Start der Simulation wird geprüft ob das Programm mit Parametern gestartet wurde. Ist dies der Fall so werden diese ausgewertet und das Programm mit der entsprechenden Konfiguration gestartet. Werden keine Parameter gestartet, so startet das Programm in der Default-Konfiguration. Im nächsten Schritt werden die Geräte, welche die Spuren eins bis vier darstellen, geöffnet und der entsprechende Filedeskriptor wird der jeweiligen Threadstruktur zugewiesen. Als nächstes werden die Fahrzeugdaten aus den entsprechenden XML-Dateien gelesen, die zugehörigen Strukturen damit initialisiert und benötigte Daten berechnet.

Als nächstes werden die Streckeninformationen aus XML-Dateien eingelesen und die Strukturen, welche die Strecke im Rechner darstellt, werden für jede Spur aufgebaut. Danach werden zwei Threads generiert. Einer zum Lesen der Gerätedatei und einer der auf die Gerätedatei schreibt. Die Vorgänge des Streckenaufbaus und der Initialisierung der beiden Threads werden für jede Spur wiederholt, wenn darauf ein Auto fahren soll.

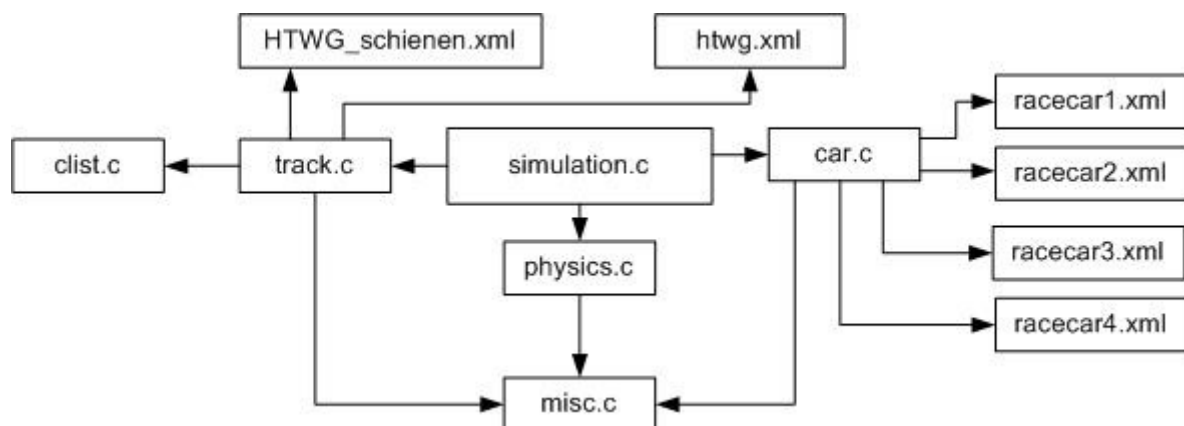


Abbildung 17: Aufbau der Simulation

6.5 Verarbeitung der Fahrzeugdaten

Nach dem die Simulation gestartet ist werden die benötigten Fahrzeugdaten zum Teil aus einer Datei gelesen und zum Teil berechnet. Dies wird in der Datei car.c realisiert. Jedes Auto hat in der Simulation eine eigene Datenstruktur. In dieser Struktur sind alle wichtigen Daten für die Simulation hinterlegt. Die Struktur repräsentiert gewissermaßen den Zustand und die Möglichkeiten des jeweiligen Autos. In der Struktur sind die Breite, die Länge und das Gewicht eines Autos gespeichert, die Haft- und Gleitreibungskonstanten der Autoreifen, der Abstand vom Drehpunkt bei der Vorderachse bis zur Hinterachse, die Länge der Finne am Drehpunkt, der Schräglauf- steifigkeitswert der Reifen, die Bremsverzögerung und eindimensionales Feld mit den von dem jeweiligen Auto erreichbaren

Beschleunigungswerten. Die von einem Auto möglichen Geschwindigkeiten werden in einem eindimensionalen Feld abgespeichert, das vom Typ einer zweiten Struktur ist. Diese Struktur beinhaltet einmal den Geschwindigkeitswert und ein Feld der Größe vier vom Typ einer dritten Struktur.

Die Idee dahinter ist Folgende, um den Driftsensor als Wertetabelle zu realisieren, muss zu jeder möglichen Geschwindigkeit, der Sensorwert einer Kurve abgespeichert und abrufbar sein. Deshalb wird zu jedem Geschwindigkeitswert der Sensorwert der Spuren eins bis vier in jeder der möglichen sieben Kurven gespeichert. Die Abbildung 18 verdeutlicht den strukturellen Aufbau noch einmal.

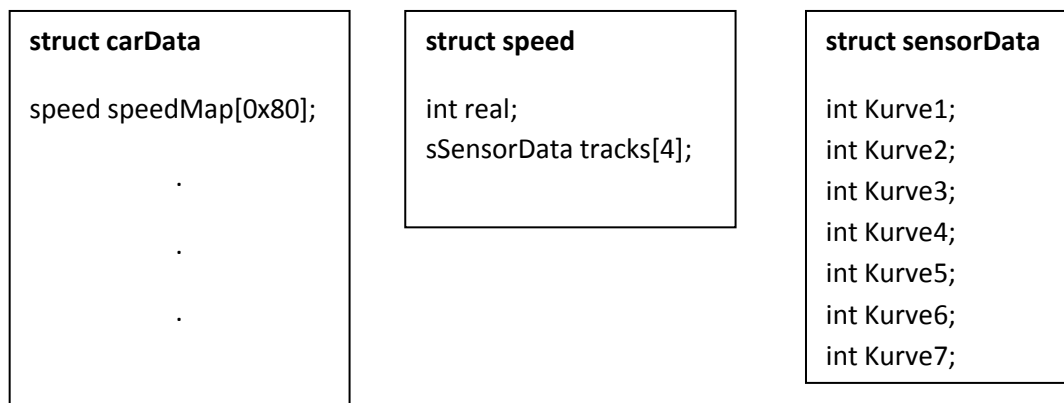


Abbildung 18: Strukturzusammenhang für ein Auto der Carrerabahn

Die Struktur des Autos beinhaltet noch ein dreidimensionales Feld vom Typ einer Struktur in der die Geschwindigkeit beim beschleunigen mit Anfangsgeschwindigkeit und beim verzögern mit Anfangsgeschwindigkeit enthalten ist. In dieser Struktur werden die Geschwindigkeitswerte nach den Geschwindigkeitsänderungen abgespeichert. Die erste Dimension des Feldes gibt die möglichen Beschleunigungen oder Verzögerungen an. Die Größe richtet sich nach den mit dem betreffenden Auto möglichen Beschleunigungen. In der zweiten Dimension des Feldes werden die Wege in 50mm Schrittweite abgelegt. Die maximale Größe ergibt sich aus der Länge des Segmentes mit der längsten Beschleunigungsstrecke geteilt durch die Schrittweite. Das Segment mit der längsten Beschleunigungsstrecke der HTWG-Bahn ist die Zielgerade, welche aus fünf Schienen mit 345 mm und zwei Schienen mit 115 mm besteht. Davon werden noch die Sensorpositionen abgezogen. Bei der HTWG-Bahn kommt man dann auf einen Wert von 50. Die dritte Dimension des Feldes gibt die Anfangsgeschwindigkeit an. Die Größe wird durch die Anzahl der möglichen Geschwindigkeitswerte des Autos vorgegeben. Für die Simulation werden immer Anfangsgeschwindigkeiten betrachtet die acht Einträge auseinander liegen. Also der erste dann der achte dann der sechzehnte usw. Die Größe ist also die Anzahl an möglichen Geschwindigkeiten geteilt durch acht.

Ein Beispiel:

Ein Aufruf von **vAccelerate[2][4][1]** würde die Geschwindigkeit liefern wenn das Auto eine Anfangsgeschwindigkeit hätte, welchem dem achten Eintrag in der speedMap entspricht, und auf einer Distanz von $4 * 50$ mm mit einer Beschleunigung, die dem 3 Eintrag im Feld der möglichen Beschleunigungen entspricht, beschleunigt oder verzögert würde.

Für das Bremsen mit der maximalen Bremsverzögerung existiert ein zweidimensionales Feld in der Struktur des Autos. Dieses Feld liefert die Geschwindigkeit, wenn mit maximaler Bremsverzögerung eine Distanz die ein Vielfaches von 50 ist, gebremst wird. Die erste Dimension gibt hier den Weg an. Sie hat genau die gleiche Größe wie im Feld für die Geschwindigkeitsänderung. Es werden 50 Elemente gespeichert. Die zweite Dimension gibt die Geschwindigkeit an bei der gebremst wird. Die maximale Anzahl an Elementen ergibt sich analog zu der Struktur für die Geschwindigkeitsänderungen. Auch hier wird wieder jede achte Geschwindigkeit gespeichert.

Ein Beispiel:

Ein Aufruf von **vDecelerate[2][1]** würde die Geschwindigkeit liefern wenn das Auto auf einer Strecke von $2 * 50$ mm mit einer Anfangsgeschwindigkeit, die dem achten Eintrag in der Struktur speedMap entspricht, mit maximaler Verzögerung abbremst.

Das Auto hat also insgesamt fünf Strukturen. Die Struktur carData enthält ein Feld vom Typ der Struktur speed welches ein Feld vom Typ der Struktur sensorData enthält. Des Weiteren enthält die Struktur carData noch ein dreidimensionales Feld vom Typ der Struktur vAccelerate und ein zweidimensionales Feld vom Typ der Struktur vDecelerate.

Im ersten Schritt werden die Werte des Autos wie z.B. Gewicht, Reibungskoeffizienten usw. aus der XML-Datei ausgelesen und den entsprechenden Variablen in der Struktur carData zugewiesen. Als nächstes wird das Feld für die Geschwindigkeits- und Sensorwerte mit den Werten aus der Datei gefüllt. Zum Schluss wird noch das Feld mit den Beschleunigungswerten gefüllt und die Variable mit der maximalen Bremsverzögerung initialisiert. Damit ist das Einlesen der Daten aus der XML-Datei beendet. Im nun Folgenden Schritt werden die Werte für das dreidimensionale und das zweidimensionale Feld berechnet. Die berechneten Geschwindigkeitswerte werden mit den von den Autos realisierbaren Geschwindigkeitswerten verglichen und es wird die Geschwindigkeit eingetragen welche zu dem berechneten Wert den kleinsten Unterschied hat.

6.6 Verarbeitung der Streckendaten

Nach dem die Strukturen für die Autos mit ihren Werten initialisiert wurden, wird im nächsten Schritt die Rennstrecke aufgebaut. Die Funktionalität hierfür ist in der Datei track.c implementiert. Auch die Daten für die Rennbahn sind in XML-Dateien gespeichert. Für die Strecke werden allerdings zwei Dateien benötigt. In der Datei htwg.xml sind alle verwendeten Schienen der Carrerabahn gespeichert. In der Datei HTWG_schienen.xml sind die Daten für die Aufteilung in die einzelnen

Streckensegmente und ihren Kennwerten enthalten. Für den Aufbau der Strecke werden drei Strukturen benötigt.

In der ersten Struktur werden alle Daten eines Streckenabschnitts der Rennbahn gespeichert. In der Struktur sind der Name des Segments, eine ID, die Länge der jeweiligen Spuren eins bis vier, die Länge bis zum Driftsensor auf der jeweiligen Spur, die Grenzggeschwindigkeiten für die Spuren für den Fall das eine Kurve im Segment enthalten ist, die Information ob es eine Rechts- oder Linkskurve ist, der Steigungswinkel für den Fall das im Streckenabschnitt eine Steigung oder ein Gefälle vorhanden ist, die Information ob man auf der Innen- oder Außenspur ist und die Information ob es in diesem Segment einen Driftsensor gibt gespeichert.

In der zweiten Struktur werden die Länge einer Spur, die Spurnummer, der Abstand vom Eintritt des Segments bis zum Anfang einer eventuellen Kreuzung, der Abstand vom Eintritt in das Segment bis zum Ende einer eventuellen Kreuzung, der Name des Segments und ein Zeiger auf die erste Struktur gespeichert.

Im ersten Schritt wird die Datei HTWG_schienen.xml verarbeitet. Für jeden Eintrag <schiene> wird eine Instanz der ersten Struktur angelegt. Die Strukturen werden mit den zugehörigen Werten aus der Datei gefüllt und dann in einer doppelt verketteten Ringliste immer als letztes Element eingefügt. Auf diese Weise wird die Carrerabahn durch die verkettete Liste im Rechner dargestellt die aus genau so vielen Elementen besteht wie die Bahn Schienen hat.

Anhand dieser verketteten Liste wird im nächsten Schritt eine zweite doppelt verkettete Ringliste aufgebaut, welche die die Bahn in ihre unterschiedlichen Segmente aufteilt. Dies wird für jede Spur der Carrerabahn gemacht. Die Bahn wird in insgesamt neun Segmente aufgeteilt. Die Namen dieser Segmente lauten:

- START/ZIEL
- KURVE_210_ENG
- KURVE_120
- KURVE_360_MITTEL
- KREUZUNG
- KURVE_240_MITTEL
- KURVE_120_ENG
- KURVE_180_MITTEL
- KURVE_120

Bei diesem Vorgang werden die Längen der Schienen die in einem Segment zusammengefasst werden addiert. Danach wird das ein Hardware Statuswort zusammengebaut welches für die Kommunikation mit dem Treiber benötigt wird. Dieses Statuswort ist 32 Bit breit und enthält Informationen über die Spur, um welches der neun Segmente es sich handelt, ob man sich auf der Innen- oder Außenbahn befindet, um welche Strecke es sich handelt und welche Lichtschranke sich in diesem Segment befindet. Für jedes so gebildete Segment wird eine Instanz vom Typ der zweiten Struktur angelegt. In dieser Struktur werden die berechnete Strukturlänge, das Hardware Statuswort, die Liniennummer, Positionsinformationen über eventuelle Kreuzungen und die Driftsensorposition

abgespeichert. Diese Strukturen werden analog zur ersten verketteten Liste immer an der letzten Stelle der zweiten verketteten Liste eingefügt. Nach Beendigung der Schleife hat man die Liste mit den neun Segmenten der Carrerabahn. Im letzten Schritt werden die Längen der Segmente noch um die Abstände der Lichtschrankensensoren korrigiert, da die Segmente durch die Positionen der Lichtschranken von einander getrennt werden und nicht am Anfang oder Ende einer Schiene beginnen.

6.7 Kommunikation Simulation – Treiber

Der Treiber legt im Geräteverzeichnis von Linux für jede Spur ein eigenes Gerät an. Für die Simulation wird ebenfalls zusätzlich für jede Spur ein eigenes Gerät angelegt. Im laufenden Betrieb werden von den Anwendungsprogrammen, welche die Autos steuern, neue Geschwindigkeitswerte an den Treiber übermittelt. Im Simulationsmodus wird dann der neue Geschwindigkeitswert, der auf das Gerät einer Spur in den kernel-space geschrieben wurde, auf das entsprechende Gerät der Simulation in den user-space zurückgeschrieben. Dadurch wird der Simulation mitgeteilt, dass das Auto auf der entsprechenden Spur, auf eine neue Geschwindigkeit beschleunigt. Durch diesen Ansatz merkt ein Steuerprogramm keinen Unterschied ob es gerade mit der Simulation oder mit der realen Carrerabahn zusammenarbeitet.

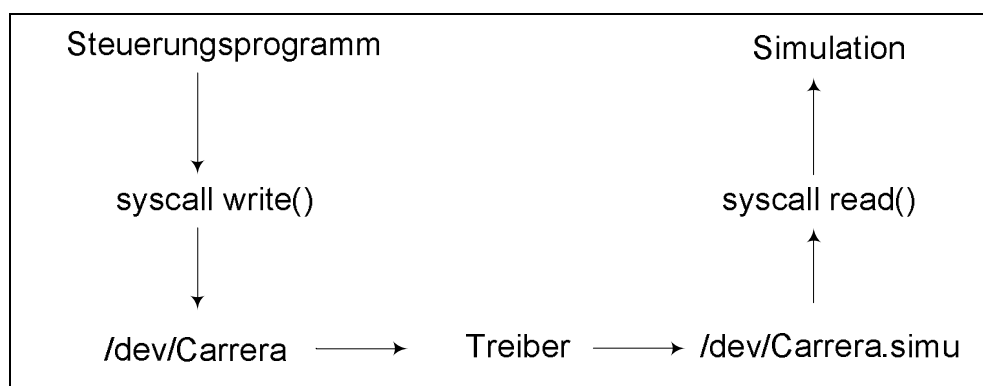


Abbildung 19: Kommunikation Steuerprogramm – Simulation

Fährt ein Auto in der Simulation durch eine virtuelle Lichtschranke oder an einem virtuellen Drift-Sensor vorbei, so dreht sich dieser Vorgang um. Ein Statuswort welches Informationen über den Sensor und die Spur enthält wird nun auf umgekehrtem Weg von der Simulationssoftware auf die entsprechende Gerätedatei der Simulation geschrieben und liegt somit im Treiber bereit.

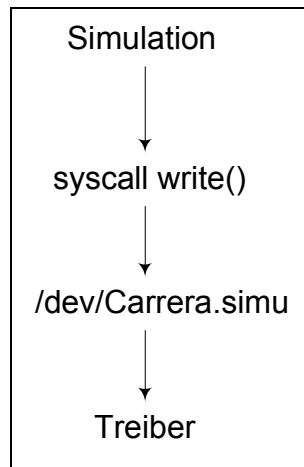


Abbildung 20: Kommunikation Simulation – Treiber

6.8 Kommunikation Simulation - Steuerprogramm

Nach der Initialisierung der Autos und der Strecke in der Simulation werden die beiden Threads pro Spur generiert. Der Thread der die Hardware Statuswörter an den Treiber übermittelt, wird mit der Methode WriteThread gestartet. Diese Methode bildet das Kernstück der Simulation. Für jede Spur wird eine Instanz der Struktur threadParam angelegt. In dieser Struktur alle wichtigen Daten die der Thread benötigt gespeichert. Dies wären:

- Dateideskriptor
- Spurnummer
- Variable um zu identifizieren ob die Spur die eigene ist
- Der letzte Geschwindigkeitswert
- Nummern der eigenen Threads (Read- und Writethread der Spur)
- Mutex und Condition-Variable für das Schlafenlegen und Wecken des Threads
- Verkettete Liste mit den Segmenten der Spur
- Die Struktur die das Auto repräsentiert
- Variable die Bremsen signalisiert

Nach dem starten des Threads für das schreiben auf die Gerätedatei wartet dieser bis ein Wert vom Steuerprogramm in die entsprechende Gerätedatei geschrieben wird. Das Warten wird mit Hilfe des Mutex und der Condition-Variablen realisiert. Wird ein neuer Wert in die

Gerätedatei geschrieben, so wird der zugehörige Lese-Thread geweckt. Dieser Thread weckt über einen Broadcast alle Threads die auf die Condition-Variable warten. Damit wird ein schlafender Schreibe-Thread geweckt und dieser kann die neue Geschwindigkeit verarbeiten.

Wenn nun vom Steuerprogramm ein erster Geschwindigkeitswert auf die entsprechende Gerätedatei geschrieben wird, so wacht der betreffende Thread auf und fängt mit der Simulation des fahrenden Autos an. Im ersten Schritt wird berechnet wie lange das Auto für das Erreichen der neuen Geschwindigkeit benötigt. Dazu wird als erstes über den Geschwindigkeitsunterschied zwischen dem neuen Wert der in die Gerätedatei geschrieben wurde (Endgeschwindigkeit) und dem Wert der kleinsten möglichen Geschwindigkeit (Anfangsgeschwindigkeit) des Autos die aktuelle Beschleunigung ermittelt. Mit den Werten für Beschleunigung, Anfangsgeschwindigkeit und Endgeschwindigkeit kann mit den Gesetzen der gleichförmig beschleunigten Bewegung die benötigte Zeit und der benötigte Wert berechnet werden. Im Anschluss daran tritt der Thread in eine Schleife ein, die die gesamten Ereignisse für die Lichtschranken, Driftsensoren und das Durchfahren von Gefahrenbereichen (Kreuzungen) behandelt. Es gibt insgesamt fünf mögliche Ereignisse. Das erste Ereignis wird ausgelöst wenn die Finne des Autos an der Lichtschranke vorbeifährt. Das zweite Ereignis wird ausgelöst wenn das Auto die Lichtschranke passiert hat. Passiert das Auto den Driftsensor, so wird das dritte Ereignis ausgelöst. Wenn das Auto den Gefahrenbereich einer Kreuzung betritt, so wird das vierte Ereignis ausgelöst und beim Verlassen des Gefahrenbereichs der Kreuzung wird das fünfte Ereignis ausgelöst. Die fünf Ereignisse heißen folgendermaßen:

- LIGHT_IN
- LIGHT_OUT
- SLIDE
- DANGER_START
- DANGER_END

In der Simulation wird davon ausgegangen, dass die Autos unmittelbar an der Lichtschranke vor dem ersten Segment starten. In Folge dieser Annahme ist das erste Event das ausgelöst wird, das Betreten der Lichtschranke (LIGHT_IN) des ersten Segments. Aus der verketteten Liste mit den Segmentdaten wird das Hardware Statuswort des aktuellen Segments ermittelt und das erste Bit auf eins gesetzt um zu vermitteln, dass die Lichtschranke ausgelöst wurde. Da bei der Übermittlung der Driftsensorauslösung an dem Treiber zwei Hardware Statuswörter übermittelt werden müssen, habe ich mich dazu entschieden immer ein Feld mit zwei Statuswörtern in die Gerätedatei zu schreiben und diese so dem Treiber zu übermitteln. Wird nur ein Statuswort benötigt, so kann das zweite Feldelement einfach leer gelassen werden. Das neue Statuswort wird nach dem setzen des Bits in die Gerätedatei geschrieben und das nächste Event wird auf das Verlassen der Lichtschranke (LIGHT_OUT) gesetzt. Bevor dieser Event behandelt werden kann muss erst noch die Zeit berechnet werden die zwischen den zwei Events vergeht. Das ist grad die Zeit die das Auto benötigt um die Länge der Finne mit der gegebenen Beschleunigung zurückzulegen. Es wird geprüft ob die Endgeschwindigkeit schon erreicht wurde, wenn das nicht der Fall ist wird geprüft ob die

Beschleunigung innerhalb der Distanz zwischen den zwei Events abgeschlossen wird. Es sind also mehrere Fälle zu überprüfen.

Der erste Fall wäre der wo der Beschleunigungsvorgang innerhalb der Distanz abgeschlossen werden kann. Dann muss nur die Zeit berechnet werden die bis zum auslösen des nächsten Events benötigt wird und der Weg innerhalb der Distanz, der noch mit konstanter Geschwindigkeit zurückgelegt wird. Die Zeit setzt sich dann aus der Beschleunigungszeit und der Zeit für das konstante Fahren zusammen. Sind die Werte berechnet, so wird der Thread für die berechnete Zeit schlafen gelegt. Wird in dieser Zeit ein neuer Geschwindigkeitswert auf die Gerätedatei geschrieben, so wird der Thread sofort aufgeweckt und dieselben Überlegungen müssen noch einmal durchgespielt werden.

Beim zweiten Fall, also wenn angenommen wird, dass der Beschleunigungsvorgang nicht innerhalb der zur Verfügung stehenden Zeit abgeschlossen werden kann, wird in der Struktur des Autos nach der zu erwartenden Geschwindigkeit nach der Distanz nachgeschaut. Dann wird mit diesen Werten wieder die benötigte Zeit berechnet und auch hier der Thread schlafen gelegt. Auch in diesem Fall ist es möglich, dass ein neuer Geschwindigkeitswert auf die Gerätedatei geschrieben wird. Dann wird auch hier der Thread sofort aufgeweckt und die Berechnungen von neuem gestartet.

In beiden Fällen besteht die Möglichkeit, dass der Vorgang des Beschleunigens oder des Fahrens mit einer konstanten Geschwindigkeit an einer schiefen Eben, also einer Bergauf- oder Bergabfahrt erfolgt. Handelt es sich bei dem betrachteten Segment um eines mit einer Steigung oder einem Gefälle, so werden die Zeiten und zurückgelegten Wege mit den Gesetzen der schiefen Eben berechnet.

Wird kein neuer Geschwindigkeitswert in die Gerätedatei geschrieben, so wacht der Thread zu dem Zeitpunkt wieder auf wenn er einen neuen Event auslöst, was nach dem LIGHT_IN ein LIGHT_OUT ist. Wenn der Beschleunigungsvorgang noch nicht abgeschlossen war, werden jetzt der noch benötigte Weg und die dafür benötigte Zeit berechnet.

Bei der Behandlung dieses Events wird analog zum vorherigen als erstes das Hardware Statuswort gebildet und auf die Gerätedatei geschrieben. Der schon zurückgelegte Weg wird auf die Länge der Finne aktualisiert. Bei diesem Event ist nun zu prüfen welches Event als nächstes eintreten kann. Es besteht die Möglichkeit, dass eine Kreuzung oder ein Driftsensor als nächstes folgen. Und wenn dies der Fall ist, so muss noch in Erfahrung gebracht werden, ob jeweils nur das eine oder das andere, beide und wenn ja in welcher Reihenfolge diese Events auftreten. Es kann natürlich auch sein, dass weder ein Driftsensor noch eine Kreuzung in diesem Segment vorkommen und dann ist der nächste Event logischerweise wieder ein Lichtschrankeneintritt.

Tritt als nächster Event das potenzielle Auslösen eines Driftsensors auf, so wird die Zeit berechnet die das Auto bei seiner aktuellen Bewegungsart benötigt um bis zu dem Sensor zu kommen. Der Thread wird wieder für diese Zeit schlafen gelegt und nach dem die Zeit vergangen ist wieder aufgeweckt. Bei der Verarbeitung dieses Events wird zunächst verglichen ob die aktuelle Geschwindigkeit bei Kurveneintritt größer als die Grenzgeschwindigkeit der Kurve ist. Ist dies nicht der Fall, so wird der Sensor nicht ausgelöst

und es werden keine Statuswörter auf die Gerätedatei geschrieben. Ist die Geschwindigkeit jedoch höher, so wird in der Struktur des Autos in der Speedmap bei der entsprechenden Geschwindigkeit nach dem Sensorwert des Driftsensors geschaut. Es wird jetzt das erste Hardware Statuswort mit der Information über das Auslösen des Driftsensors versehen und im zweiten Hardware Statuswort wird der Sensorwert kodiert. Danach wird das Feld mit den Statuswörtern in die Gerätedatei geschrieben. Wenn das Auto zu schnell gefahren ist wird eine Meldung ausgegeben.

Fährt ein Auto als nächstes in eine Kreuzung ein, so wird das entsprechende Event gesetzt und nach dem Ablauf der berechneten Zeit die das Auto bis zum betreten der Kreuzung benötigt wird dieses behandelt. Hierbei ist zu beachten, dass ein gleichzeitiges befahren der Kreuzung von dem eigenen Auto und dem Auto auf der gegnerischen Spur zu verhindern ist. Dies kann effizient mit zwei Integer Variablen gelöst werden. Je eine Variable ist für ein Spurpärchen verantwortlich. Wenn nun ein Auto in den Gefahrenbereich einfährt, so wird diese Variable abgefragt. Ist die Variable gesetzt so bedeutet dies, dass das gegnerische Fahrzeug vor dem eigenen Fahrzeug in den Gefahrenbereich einfahren wird und das eigene Fahrzeug hat abzubremesen. Nach dem Durchfahren der Kreuzung wird im darauffolgenden Event, die Variable wieder freigegeben. Nun muss noch bestimmt werden ob eventuell noch ein Driftsensor innerhalb des Segmentes kommt oder ob als nächster Event eine Lichtschranke ausgelöst wird. Dies lässt sich durch vergleichen der Abstände einfach bewerkstelligen. Wenn der Abstand vom Segmentanfang zum Gefahrenbereichsanfang kleiner ist als vom Segmentanfang bis zur eventuellen Driftsensorposition, so ist der Driftsensor vor dem Gefahrenbereich, ansonsten danach.

Anhang

7. Dateien & Strukturen

7.1 HTWG_schienen.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
- <schienen>
- <schiene>
  <id>1</id>
  <name>startziel</name>
  <Spur_1>345</Spur_1>
  <Spur_2>345</Spur_2>
  <Spur_3>345</Spur_3>
  <Spur_4>345</Spur_4>
  <SENSOR_POS_4>0</SENSOR_POS_4>
  <SENSOR_POS_3>0</SENSOR_POS_3>
  <SENSOR_POS_2>0</SENSOR_POS_2>
  <SENSOR_POS_1>0</SENSOR_POS_1>
  <V4>0</V4>
  <V3>0</V3>
  <V2>0</V2>
  <V1>0</V1>
  <R4>0</R4>
  <R3>0</R3>
  <R2>0</R2>
  <R1>0</R1>
  <alpha>0</alpha>
</schiene>
+ <schiene>
+ <schiene>
+ <schiene>
+ <schiene>
- <schiene>
  <id>6</id>
  <name>kurve_210_eng</name>
  <Spur_4>917</Spur_4>
  <Spur_3>1274</Spur_3>
  <Spur_2>1638</Spur_2>
  <Spur_1>2002</Spur_1>
  <SENSOR_POS_4>451</SENSOR_POS_4>
  <SENSOR_POS_3>531</SENSOR_POS_3>
  <SENSOR_POS_2>607</SENSOR_POS_2>
  <SENSOR_POS_1>687</SENSOR_POS_1>
  <V4>1400</V4>
  <V3>1700</V3>
  <V2>1900</V2>
  <V1>2100</V1>
  <R4>246</R4>
  <R3>348</R3>
  <R2>444</R2>
  <R1>546</R1>
  <alpha>0</alpha>
</schiene>
+ <schiene>
```

7.2 61silber.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <fahrzeug name="61silber">
- <data>
  <width>70.0</width>
  <distance>100.0</distance>
  <rail>27.0</rail>
  <length>250.0</length>
  <friction_G>0.85</friction_G>
  <friction_R>0.014</friction_R>
  <distance2>100.0</distance2>
  <mass>500.0</mass>
  <ch>0.1</ch>
  <cv>0.1</cv>
</data>
+ <tiremodell>
- <speedmap>
- <geschwindigkeit>
  <hex>0x20</hex>
  <real>509</real>
- <schleudersensor>
- <track1>
  <K_210_E>0</K_210_E>
  <K_120>0</K_120>
  <K_360>0</K_360>
  <K_240>0</K_240>
  <K_180>0</K_180>
  <K_120_E>0</K_120_E>
  <K_210>0</K_210>
</track1>
- <track2>
  <K_210_E>0</K_210_E>
  <K_120>0</K_120>
  <K_360>0</K_360>
  <K_240>0</K_240>
  <K_180>0</K_180>
  <K_120_E>0</K_120_E>
  <K_210>0</K_210>
</track2>
- <track3>
  <K_210_E>0</K_210_E>
  <K_120>0</K_120>
  <K_360>0</K_360>
  <K_240>0</K_240>
  <K_180>0</K_180>
  <K_120_E>0</K_120_E>
  <K_210>0</K_210>
</track3>
- <track4>
  <K_210_E>0</K_210_E>
  <K_120>0</K_120>
  <K_360>0</K_360>
  <K_240>0</K_240>
```


7.4 struct carData

```
typedef struct _carData
{
    speed speedMap[0x80];
    int aPositive[6];
    vAccelerate sVAccelerate[6][50][12];
    int aNegative;
    vDecelerate sVDecelerate[50][12];
    // Breite des Autos
    double width;
    // Abstand Hinterachse bis Schienenführung
    double distance;
    // Länge der Fuehrungsfinne
    double rail;
    // Länge des Autos
    double length;
    // Konstante für Gleitreibung (Reifen/Fahrbahn)
    double friction_G;
    // Konstante für Rollreibung (Reifen/Fahrbahn)
    double friction_R;
    // Abstand Drehpunkt-Finne bis Magnet
    double distance2;
    // Masse des Autos in g
    double mass;
    // Schräglaufsteifigkeit Hinterrad
    double ch;
}carData;
```

7.5 struct threadParam

```
typedef struct _ThreadParam
{
    int fp;
    int line;
    int own;
    __u32 value;
    __u32 breaking;
    pthread_t readthread;
    pthread_t writethread;
    pthread_mutex_t speedMut;
    pthread_cond_t speedCond;
    C_List segments;
    carData sCarData;
} threadParam;
```

7.6 struct vAccelerate

```
typedef struct _vAccelerate
{
    int realPositive;
    int hexPositive;
    int realNegative;
    int hexNegative;
}vAccelerate;
```

7.7 struct vDecelerate

```
typedef struct _vDecelerate
{
    int real;
    int hex;
}vDecelerate;
```

7.8 struct speed

```
typedef struct _speed
{
    int real;
    int sinner;
    int souter;
    // Pro Geschwindigkeit 4 Spuren und 7 moegl. Kurven
    sSensorData tracks[4];
}speed;
```

7.9 struct sSensorData

```
typedef struct _sensorData
{
    int seg_210_E;
    int seg_120;
    int seg_360;
    int seg_240;
    int seg_180;
    int seg_120_E;
    int seg_210;
}sSensorData;
```


7.10 struct sTrackData

```
typedef struct _Trackdata
{
    char *name;
    int id;
    int inner;
    int outer;

    // die einzelnen Spuren
    int track1;
    int track2;
    int track3;
    int track4;
    // die einzelnen SensorPos(im Kreissektor)
    int s1;
    int s2;
    int s3;
    int s4;
    // die einzelnen Grenzgeschwindigkeiten für Konstante = 0.8
    int v1;
    int v2;
    int v3;
    int v4;
    // die Radien der jeweiligen Spuren
    int r1;
    int r2;
    int r3;
    int r4;

    double alpha;
    int direction;
    int slide;
} sTrackData;
```

7.11 struct sSegData

```
typedef struct __SegData
{
    int    length;
    int    line;
    int    dangerAreaStart;
    int    dangerAreaEnd;
    char   name[20];
    sTrackData *trackData;
    sSlideData *slideData;
    __u32  hw_status;
}sSegData;
```

7.12 struct sSlideData

```
typedef struct __slideData
{
    int    slidePos;
    int    slideSensor;
}sSlideData;
```

Literaturverzeichnis

- [Rob04]. **Robert Bosch GmbH**. *BOSCH Automotive Handbook*. s.l. : John Wiley and Sons, 2004.
- [Ass04]. **Assmann, Bruno**. *Technische Mechanik. 3. Kinematik und Kinetik*. s.l. : Wissenschaftsverlag, 2004.
- [Bec01]. **Beckmann, Brian**. *Physics of Racecars*. 2001.
- [Bit07]. **Bittel, Oliver**. *NeuroNetze_4.pdf*. 2007.
- [DoKrVo07]. **Dobrinski, Paul, Krakau, Gunter und Vogel, Anselm**. *Physik für Ingenieure*. s.l. : Vieweg+Teubner Verlag, 2007.
- [Gen97]. **Genta, Giancarlo**. *Motor Vehicle Dynamics: Modeling and Simulation*. s.l. : World Scientific, 1997.
- [HaHo03]. **Halfmann, Christoph und Holzmann, Henning**. *Adaptive Modelle für die Kraftfahrzeugdynamik*. s.l. : Springer, 2003.
- [HeEr07]. **Heißing, Bernd und Ersoy, Metin**. *Fahrwerkhandbuch: Grundlagen, Fahrdynamik, Komponenten, Systeme, Mechatronik*. s.l. : Vieweg Verlag, 2007.
- [Her04]. **Hermann und Martin**. *Numerik gewöhnlicher Differentialgleichungen: Anfangs- und Randwertprobleme*. s.l. : Oldenbourg Wissenschaftsverlag, 2004.
- [www_ika]. **ika-achen**. http://www.ika.rwth-aachen.de/lehre/kfz-labor/6_fahrdynamik_de.pdf.
- [Ise06]. **Isermann, Rolf**. *Fahrdynamikregelung*. s.l. : Vieweg Verlag, 2006.
- [JudGer05]. **Judy Hsu, Yung-Hsiang und Gerdes, J. Christian**. *Stability at the Limits PowerPoint Präsentation*. s.l. : Stanford University, 2005.
- [MiWa04]. **Mitschke, Manfred und Wallentowitz, Henning**. *Dynamik der Kraftfahrzeuge*. s.l. : Springer, 2004.
- [Mar]. **Monster, Marco**. *Physics Tutorial*.
- [RieSchu39]. **Riekert, P und E., Schunk T**. *Zur Fahrmechanik des gummibereiften Kraftfahrzeugs*. s.l. : Springer Berlin / Heidelberg, 1939.
- [RiFiSchFu05]. **Rittenschober, Thomas, et al**. *Fahrdynamikregelung mit differentialgeometrischen Methoden der Regelungstechnik*. s.l. : Institut für Regelungstechnik und Prozessautomatisierung, Johannes Kepler Universität Linz, 2005.

[www_erl]. **uni-erlangen**. http://www.tp2.uni-erlangen.de/Lehrveranstaltungen/medi/Medi_Materialien/W-G-B_Zeit_Diagramme.pdf.

[www_wü]. **uni-würzburg**. <http://www.physik.uni-wuerzburg.de/physikonline/repetmech/zusammenfassungen/zodynamik1p1.html>.

[SSV98]. **Vary, Peter, Heute, Ulrich und Hess, Wolfgang**. *Digitale Sprachsignalverarbeitung*. Stuttgart : Teubner, 1998.

[Wal96]. **Wallentowitz, H.** *Vertikal-/Querdynamik von Kraftfahrzeugen*. s.l. : Umdruck zur Vorlesung, Aachen, 1996.

Ehrenwörtliche Erklärung

Hiermit erkläre Ich, Marco Fahr, geboren am 07.03.1976 in Stockach (DE),

(1) dass Ich meine Diplomarbeit mit dem Titel:

Carrerabahn - Implementierung einer Simulation zur Verifizierung von Echtzeitprogrammen für die Steuerung von Hardware

An der HTWG Konstanz unter Anleitung von Professor Dr. M. Mächtel selbständig und ohne fremde Hilfe angefertigt habe und keine anderen als die angeführten Hilfen benutzt habe;

(2) dass Ich die Übernahme wörtlicher Zitate, von Tabellen, Zeichnungen, Bildern und Programmen aus der Literatur oder anderen Quellen (Internet) sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe.

Diese Arbeit hat in dieser oder ähnlicher Form noch nicht im Rahmen einer anderen Prüfung oder an einem anderen Ort vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Konstanz, 07.05.2008

Danksagung

Schließlich möchte Ich mich herzlich bedanken, bei allen, die mir für die Erstellung dieser Arbeit den Rücken freigehalten haben.

Besonders Danken möchte ich hierbei Herrn Prof. Dr. M. Mächtel und der gesamten Fakultät IN, welche durch die technische Ausrüstung die Erstellung dieser Arbeit ermöglicht haben.

Meinen Eltern und Jasmina danke Ich für die Unterstützung während des gesamten Studiums.

Außerdem danke Ich allen, die mich während der Entstehungszeit dieser Arbeit Motiviert, Unterstützt oder Versorgt haben.

Für Jerome