DHBW
Duale Hochschule
Baden-Württemberg

In der Regel haben wir einen zweizeiligen Bachelorthesistitel

Bachelorarbeit

für die Prüfung zum

Bachelor of Engineering

des Studiengangs Vorderasiatische Archäologie

an der Dualen Hochschule Baden-Württemberg Karlsruhe

von

Vorname Nachname

August 2011

Bearbeitungszeitraum	12 Wochen
Matrikelnummer, Kurs	1234510, ABC2008DE
Ausbildungsfirma	Firma GmbH, Firmenort
Betreuer	Dipl.-Ing. (FH) Peter Pan
Gutachter	Dr. Silvana Koch-Mehrin

Sperrvermerk

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anderslautende Genehmigung der Ausbildungsstätte vorliegt.

Abgabeort, August 2011

_____

Vorname Nachname

Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit mit dem Thema: *In der Regel haben wir einen zweizeiligen Bachelorthesistitel* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Abgabeort, August 2011

_____

Vorname Nachname

Abstract

Abstract normalerweise auf Englisch. Siehe: http://www.dhbw.de/fileadmin/user/public/Dokumente/Portal/Richtlinien_Praxismodule_Studien_und_Bachelorarbeiten_JG2011ff.pdf (8.3.1 Inhaltsverzeichnis)

Ein „Abstract“ ist eine prägnante Inhaltsangabe, ein Abriss ohne Interpretation und Wertung einer wissenschaftlichen Arbeit. In DIN 1426 wird das (oder auch der) Abstract als Kurzreferat zur Inhaltsangabe beschrieben.

Objektivität soll sich jeder persönlichen Wertung enthalten

Kürze soll so kurz wie möglich sein

Genauigkeit soll genau die Inhalte und die Meinung der Originalarbeit wiedergeben

Üblicherweise müssen wissenschaftliche Artikel einen Abstract enthalten, typischerweise von 100-150 Wörtern, ohne Bilder und Literaturzitate und in einem Absatz.

Quelle: http://de.wikipedia.org/wiki/Abstract Abgerufen 07.07.2011

Diese etwa einseitige Zusammenfassung soll es dem Leser ermöglichen, Inhalt der Arbeit und Vorgehensweise des Autors rasch zu überblicken. Gegenstand des Abstract sind insbesondere

- Problemstellung der Arbeit,
- im Rahmen der Arbeit geprüfte Hypothesen bzw. beantwortete Fragen,
- der Analyse zugrunde liegende Methode,
- wesentliche, im Rahmen der Arbeit gewonnene Erkenntnisse,
- Einschränkungen des Gültigkeitsbereichs (der Erkenntnisse) sowie nicht beantwortete Fragen.

Quelle: http://www.ib.dhbw-mannheim.de/fileadmin/ms/bwl-ib/Downloads_alt/Leitfaden_31.05.pdf, S. 49

Inhaltsverzeichnis

Abkürzungsverzeichnis

ROS	Robot Operating System
AGPL	Affero GNU General Public License

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Aufgabe

Erste Erwähnung eines Akronyms wird als Fußnote angezeigt. Jede weitere wird nur verlinkt: Affero GNU General Public License (AGPL). [3]

Verweise auf das Glossar: Glossareintrag, Glossareinträge

Nur erwähnte Literaturverweise werden auch im Literaturverzeichnis gedruckt: [1], [2]

Meine erste Fußnote[1]

1.1 Carolocup, Regeln

1.2 Simulation

1.3 Anforderungen

Immer wenn ich Fahrzeug sage meine ich das simulierte, wenn ich echtes Fahrzeug sage meine ich das echte.

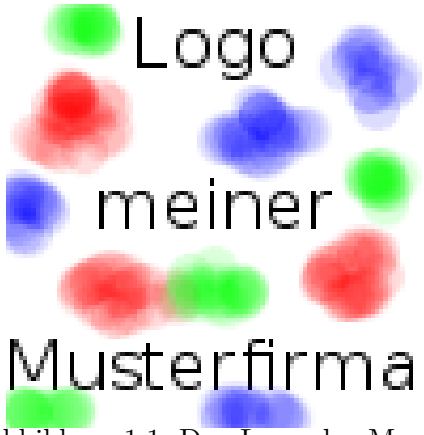

Abbildung 1.1: Das Logo der Musterfirma[2]

[1] Ich bin eine Fußnote

[2] aus [4]

2 Umfeld

2.1 ROS

Das Robot Operating System (ROS) ist ein Betriebssystem das speziell auf Roboter angepasst ist. Es läuft auf einem anderen Betriebssystem, in diesem Fall Ubuntu. Es stellt Funktionalitäten wie Hardware Abstraktion, Paket Management, Kommunikation zwischen Prozessen... zu verfügung. Das Ziel von Robot Operating System (ROS) ist die Wiederverwendbarkeit von Code. Dies gelingt dadurch das Robot Operating System (ROS) ein verteiltes System an Prozessen (Nodes) ist. Die Nodes kommunizieren untereinander über Topics. Topics sind die Busse auf denen Nodes Nachrichten senden. Nachrichten zu senden oder zu empfangen ist anonym, dadurch sind Sender und Empfänger komplett entkoppelt voneinander. [6] Nodes lassen sich so austauschen und wiederverwenden. [5] Die Regelung des Fahrzeugs läuft auf einer ROS Architektur.

2.2 Architektur

Als Ausgangssituation für die Simulation wird die open source Simulationsumgebung des KITs genutzt. Diese ist eine Gazebo Simulation mit verschiedenen ROS Nodes. HIER ARCHITEKTUR BILD EINFÜGEN Die eigentliche Fahrsimulation des Fahrzeugs ist nicht Teil der Simulationsumgebung. Um das Fahrzeug zu bewegen erwarted die Simulation ein Topic auf dem neue Position gepublished werden. Da die Simulation aber den Lenkwinkel als Input nutzen soll, muss die Simulation dahingehend erweitert werden.

2.3 Hardware

3 Stand der Technik

3.1 Wie kann man Physik simulieren

3.2 Was gibt Ros für Möglichkeiten

4 Implementierung

4.1 Ausgangslage

4.2 Simulation als Datenproduktion

Strecke modelieren Man kann in der Simulation Strecken aufbauen. Dazu wird auch zu jeder Strecke eine Ideallinie für das Fahrzeug generiert. Diese Strecke besteht aus einer Liste von Punkten. Das Modul AutomaticDrive kann genutzt werden damit das Fahrzeug entlang dieser Strecke fährt. Das Fahrzeug fährt dann aber in gerader Linie von Punkt zu Punkt. Dadurch kommt es in Kurven zu einem ruckeligen Fahrverhalten. WIE LÖST MAN DAS PROBLEM. INTERPOLATION IN KURVEN BEI AUTODRIVE

Die Positionierung und Orientierung funktioniert mit dem Hinterachsenmittelpunkt H als Basis. Das Fahrzeug fährt indem der Punkt H mit einer konstanten Geschwindigkeit entlang der Fahrspur fährt. Die Orientierung des Fahrzeugs ist so definiert, dass das Fahrzeug am Punkt H eine Tangente zur Fahrspur bildet. Die Tangente kann mit einer Funktioner der Datenstruktur der Fahrspur erechnet werden. In einer Kurve kann das Fahrzeug dann so aussehen:

Positionierung und Rotation um Punkt H hat sich als realitätsnah erwiesen, da die Kamera, die vorne am Fahrzeug montiert ist, in den Kurven ein ähnliches Bild geliefert hat wie das echte Fahrzeug.

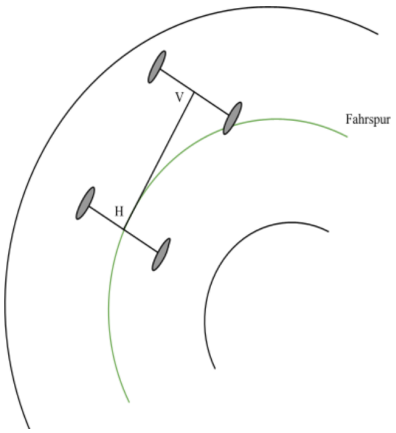

Abbildung 4.1: BlaBla[2]

Kamera entlang der Strecke fahren lassen

Um Algorithmen und Model in der Spurerkennung und Situationsklassifizierung zu verifizieren und verbessern, benötigt der Auto drive noch eine Möglichkeit den Lenkwinkel auszugeben. Für den AutomaticDrive nehmen wir ein paralleles Lenksystem an. Punkt V nach H. $\vec{VH}$

Die Vorderräder sind ausgerichtet entlang der Tangente VT am Punkt V zur Fahrspur. Der Lenkwinkel β ist der Winkel zwischen $\vec{VH}$ und $\vec{Lenkrichtung}$.

4.3 Kamera

Auflösung, Position, FoV eingestellt.

Die Auflösung der Kamera ist 2x2 Pixel. Die Position und Orientierung der Kamera wird genau ermittelt, um die simulierte Kamera an die selbe Position zu bringen.

Problem: Für die IPM umrechnung muss der Kamerafeed sehr genau den echten wiederspiegeln. Neue matrix wäre eine Scheiß Lösung.

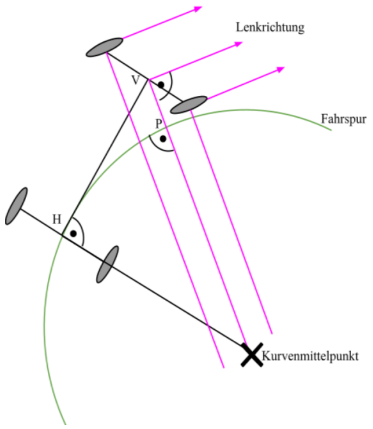

Abbildung 4.2: BlaBla[3]

4.4 statische Fortbewegung

Problem: Lenkwinkel muss irgendwie in eine Transformation des Fahrzeugs umgewandelt werden. Der genaue Lösung wäre eine dynamische Simulation der Räder und Lenkung. Dies war nicht so einfach abzustimmen, da sich die Simulation anders Verhalten hat als das echte Fahrzeug. Vorläufige Lösung: Eine Rechnung die die Transformation abschätzt, aber sehr genau ist bei kleinen Lenkwinkel und hoher Fps.

Gegeben ist der Lenkwinkel alpha Die Geschwindigkeit des Fahrzeugs v Der daraus resultierende Vektor Lenkrichtung steering Position des Fahrzeugs p Orientierung des Fahrzeugs und der daraus resultierende Vektor direction

Als erstes wird aus dem Lenkwinkel alpha der Vektor steering bestimmt. Danach wird die neue Position pNeu bestimmt mit pNeu = p + steering * (deltaTime * v) Die neue Orientierung des Fahrzeugs directionNeu wird erechnet mit directionNeu = direction rotieren um k*alpha dabei ist k Abhängig von der gefahrenen Strecke, sowie der Fahrzeuglänge.

4.5 dynamische Fortbewegung

Vier drehende Räder, Antrieb, Lenkung vorne. Problem mit, Kräften, massen, Kollision... on...

Die Räder des Fahrzeugs werden definiert innerhalb einem Link. Jeder der vier Links besitzt eine visuelle Komponente und ein Kollisions-Komponente. Das visuelle Objekt und das Kollisions-Objekt stimmen überein, beide definieren den gleichen Zylinder. Jeder

physikalisch simulierte Link benötigt einen inertial-Tag. Dieser definiert die Masse "massünd die Massenverteilung als inertia Tensor ïnertia".

Damit sich ein Rad drehen kann, wird es mit einem Joint an dem Fahrzeug befestigt. Der Joint definiert die beiden Links die verbunden werden sollen: child und parent. Der Typ des Joints ist für die Räder "continuos". Das bedeuted, dass sich die Räder ohne Einschränkung um eine Achse drehen können. Der Joint definiert außerdem die Position und Orientierung der Achse.

DiskJoints.

Um die Räder und die Länkung zu steuern, benötigt es einen Motor. Für urdf gibt es hierfür Actuator und Transmissions. Eine Transmission verbindet einen Actuator mit einem Joint. ACTUATOR TRANSMISSIONS CODE HIER

Controller Um die Actuator zu steuern kann man das Package ros_control nutzten. Dies bietet eine Reihe verschiedener Controller. Controller können einen Actuator nach PID-Steuerung steuern. Um die Controller zu benutzen wird ein neues Package erstellt. Das Package $/simulation/src/gazebo_controller$ beinhaltet eine Launchdatei und eine Parameterdatei. Die Parameterdatei definiert die vier verschiedenen Controller. Zwei für den Hinterradantrieb und zwei für die Vorderlenkung. Die Controller für den Antrieb sind JointVelocityController, die Controller für die Lenkung sind JointPositionController. Von dem Antrieb soll die Geschwindigkeit gesteuert werden, von der Lenkung soll die Position bzw. die Rotation gesteuert werden. In der Parameterdatei werden auch die zugehörigen Joints und die PID Parameter definiert. Die Launchdatei nutzt die Parameterdatei und lädt damit die Controller über einen controller_spawner. Die Launchdatei wird in der master.launch-Datei wie folgt aufgerufen:

```
1 <include if="\$(eval include_automatic_drive == false)" file="\$(find
      gazebo_controller)/launch/robot_control.launch"></include>
```

Die Controller werden nur initialisiert, wenn AutomaticDrive nicht genutzt wird.

PID Wie funktioniert PID???

Ziel der Simulation ist es bei gegebenem Lenkwinkel und Geschwindigkeit das Fahrzeug entsprechend zu bewegen. Die Geschwindigkeit wird an die zwei Geschwindigkeitscontroller gegeben. Der Lenkwinkel wird an die zwei Lenkcontroller gegeben. Hierbei treten jedoch zwei Probleme auf:

Wenn sich beide Hinterräder in der genau gleichen Geschwindigkeit drehen, kann es zu Problemen in Kurven kommen. Das äußere Rad muss eine längere Strecke in der gleichen Zeit wie das innere Rad fahren. Dadurch kann es zu ungewünschten Kräften und Rutschen

kommen. Im echten Fahrzeug wird dieses Problem durch ein Differentialgetriebe gelöst. In der Simulation hat sich das als kein großes Problem erwiesen und es bleibt erstmal bei zwei Controllern mit der genau gleichen Geschwindigkeit, auch in den Kurven.

Das zweite Problem ist die Lenkung. Wenn man den Lenkwinkel direkt an beide Controller weitergibt hat man eine Parallellenkung. Das Fahrzeug mit Parallellenkung sieht in einer Kurve so aus:
Das Fahrzeug fährt um den Kurvenmittelpunkt herum. Wenn ein Rad eine Kurve fährt, ist es so ausgerichtet, dass es entlang einer Tangenten auf dem Kreis um den Kurvenmittelpunkt fährt. Die Hinterräder des Fahrzeugs fahren um den Kurvenmittelpunkt. Die Vorderräder sind allerdings nicht entlang eines Kreises um den Kurvenmittelpunkt orientiert sondern parallel zur Lenkrichtung. Die Lenkrichtung geht aus vom Vorderachsenmittelpunkt V. Dies kann ebenfalls zu Rutschen und ungewollten Kräften auf Räder und Lenkung führen. Um das zu verhindern wird im echten Fahrzeug eine Ackermann-Lenkung verwendet. Diese sieht in einer Kurve so aus: Im Gegensatz zu Parallellenkung sind nun auch die Vorderräder entlang einem Kreis um den Kurvenmittelpunkt ausgerichtet. Das linke Rad hat nun einen anderen Winkel wie das rechte Rad.

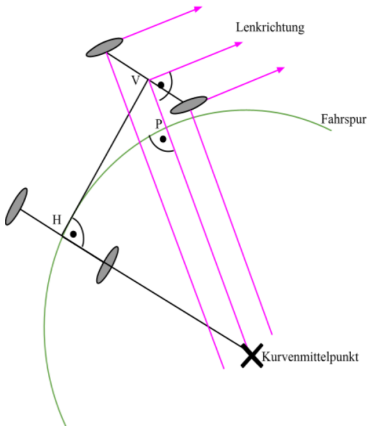

Abbildung 4.3: BlaBla[4]

Wie die Ackermann-Lenkung, per Hardware, im echten Fahrzeug umgesetzt ist hier unwichtig. Um die Ackermann-Lenkung zu simulieren, können aus einem gegebenen Lenkwinkel, jeweils der Winkel für das rechte und linke Rad berechnet werden.

Ziel ist es die Winkel β und γ zu bestimmen. Gegeben ist der Lenkwinkel am Vorderachsenmittelpunkt α. Mit der Fahrtrichtung und α lässt sich die pinke Linie konstruieren. Außerdem ist gegeben dass die Hinterachse auf einer Geraden mit dem Kurvenmittelpunkt liegt. Der Schnittpunkt der pinken Linie und dieser Geraden ist der Kurvenmittelpunkt. Die Strecke von H zum Kurvenmittelpunkt wird definiert als r. Die Strecke w ist der Radabstand zwischen den rechten und linken Rädern. Die Strecke l ist der Radabstand zwischen

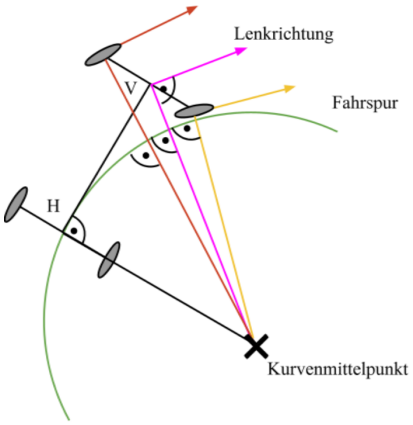

Abbildung 4.4: BlaBla[5]

den vorderen und hinteren Rädern. Aus der Pinken Strecke und den Strecken r und l bildet sich ein Dreieck mit rechtem Winkel bei H. Daraus folgt

$$\tan(\alpha) = \frac{l}{r} \tag{4.1}$$

Wenn man den Punkt H um $\frac{w}{2}$ nach rechts und links verschiebt kann man zwei weitere rechtwinklige Dreiecke konstruieren. Diese beinhalten jeweils die braune oder gelbe Strecke. Aus den zwei Dreiecken gehen folgende Gleichungen hervor:

$$\tan(\beta) = \frac{l}{r - w/2} \tag{4.2}$$

$$\tan(\gamma) = \frac{l}{r + w/2} \tag{4.3}$$

Umformung Gleichung 4.1

$$r = \frac{l}{\tan(\alpha)} \tag{4.4}$$

Setzte man Gleichung 4.1 in Gleichung 4.2 ein so erhält man

$$\tan(\beta) = \frac{l}{\frac{l}{\tan(\alpha)} - w/2}$$

$$\iff \beta = \arctan(\frac{l}{\frac{l}{\tan(\alpha)} - w/2})$$

Analog dazu erhält man die Gleichung für γ

$$\gamma = \arctan(\frac{l}{\frac{l}{\tan(\alpha)} + w/2})$$

Mit diesen zwei Gleichungen lassen sich die Winkel β und γ bestimmen. Die Radabstände l und w sind fest in der Simulation definiert. Beide individuelle Lenkwinkel β und γ sind nur von einer Variablen abhängig: Dem Lenkwinkel am Vorderachsenmittelpunkt α. [7]

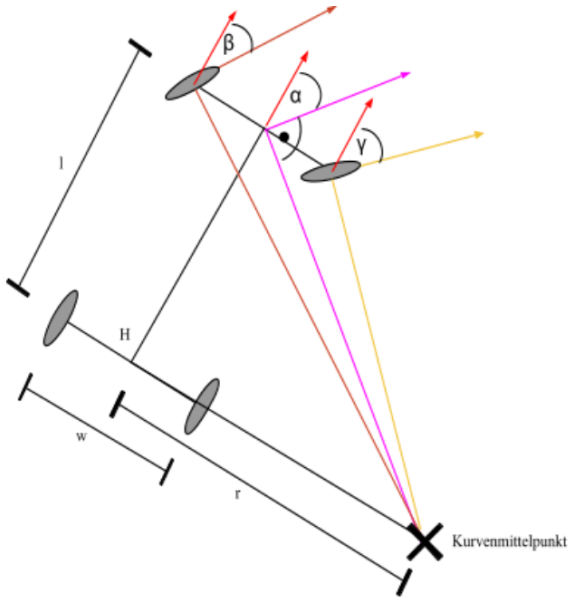

Abbildung 4.5: BlaBla[6]

5 Ergebnis

5.1 Anforderungen erreicht?

Literatur

[1] Peter Baumgartner, Hartmut Häfele und Kornelia Maier-Häfele. *E-Learning Praxishandbuch : Auswahl von Lernplattformen; Marktübersicht, Funktionen, Fachbegriffe*. Innsbruck: StudienVerl., 2002. ISBN: 3-7065-1771-X.

[2] Stuard E. Dreyfus und Hubert L. Dreyfus. *A Five-Stage Model Of The Mental Activities Involved In Directed Skill Acquisition*. Techn. Ber. University of California, Berkley, 1980. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA084551&Location=U2&doc=GetTRDoc.pdf.

[3] Free Software Foundation. *GNU AFFERO GENERAL PUBLIC LICENSE*. 2007. URL: http://www.gnu.org/licenses/agpl-3.0.de.html.

[4] Max Mustermann. „tolles Musterthema“. Studienarbeit. Musterstadt, 2012.

[5] Open Robotics. *ROS Introduction*. 2018. URL: http://wiki.ros.org/ROS/Introduction.

[6] Open Robotics. *ROS Topics*. 2021. URL: http://wiki.ros.org/Topics.

[7] Robert Eisele. *Ackerman Steering*. 2021. URL: https://www.xarg.org/book/kinematics/ackerman-steering/.

Glossar

Glossareintrag

Ein Glossar beschreibt verschiedenste Dinge in kurzen Worten.

Anhang

(Beispielhafter Anhang)

A. Assignment

B. List of CD Contents

C. CD

B. List of CD Contents

```
├ Literature/
|    ├ Citavi-Project(incl pdfs)/   ⇒ Citavi (bibliography software) project with
|    |                               almost all found sources relating to this report.
|    |                               The PDFs linked to bibliography items therein
|    |                               are in the sub-directory 'CitaviFiles'
|    |    – bibliography.bib        ⇒ Exported Bibliography file with all sources
|    |    – Studienarbeit.ctv4      ⇒ Citavi Project file
|    |    ├ CitaviCovers/           ⇒ Images of bibliography cover pages
|    |    ├ CitaviFiles/            ⇒ Cited and most other found PDF resources
|    ├ eBooks/
|    ├ JournalArticles/
|    ├ Standards/
|    ├ Websites/
|
├ Presentation/
|    –presentation.pptx
|    –presentation.pdf
|
├ Report/
     – Aufgabenstellung.pdf
     – Studienarbeit2.pdf
     ├ Latex-Files/ ⇒ editable LaTeX files and other included files for this report
          ├ ads/                   ⇒ Front- and Backmatter
          ├ content/               ⇒ Main part
          ├ images/                ⇒ All used images
          ├ lang/                  ⇒ Language files for LaTeX template
```