1 Grundlagen Robotik

Robotik besteht aus den Gebieten Mechanik, Elektronik und Informatik. Wir konzentrieren uns in AFR auf die Informatik. Die wichtigen Themengebiete auf der Informatik-Seite sind Folgende:

- Kinematik
- Dynamik
- Sensoren, Karten, Lokalisierung, SLAM, Navigation
- Spezifische Probleme im autonomen Fahren (Fernsteuerung? Gesetze?)
- Generierung eines Plans und Ausführung
- Kontrolle des Verhaltens

Dafür brauchen wir Aktuatoren und dafür widerum Mechanik.

1.1 Mechanik, Physik und Aktuatoren

1.1.1 Mechanik

Ein Rigid-Body (Starrkörper) Modell besteht aus Links und Joints und hat Aktuatoren und eine Anzahl an Freiheitsgraden. Es besteht aus:

- Connectivity Graph
- Link and Joint Geometrie Parameter
- Link-Trägheits Parameter
- Menge an Joint Models

1.1.2 Physik

meh.

1.1.3 Aktuatoren

hydraulisch, pneumatisch, elektromagnetisch ... dazu später mehr!

1.2 Und was ist jetzt ein Roboter?

Roboter sind physische, bewegliche Maschinen, die Aufgaben erfüllen können?
Roboter haben Sensoren und Aktoren, die mittels Software kommunizieren.

Roboter:

- Automatisierungstechnik, Roboterarme, Manipulator
- Marsroboter
- Medizinische Chirurgie Roboter
- Saugroboter, Rasenmähroboter

kein Roboter:

- Baumaschinen, vom Menschen kontrollierte Maschinen

- Teilautomatisierte Maschinen wie das momentane Auto

1.3 Robotik Frameworks

In Frameworks sind die üblichen Probleme schon gelöst und Algorithmen implementiert. Man kann schneller neue Dinge 'implementieren'.
Robotics Toolkit (Rock), Robot Operating System (ROS).

1.3.1 ROS

ROS ist als eine publish-subscriber Architektur aufgebaut. Verschiedene Nodes (laufende Programme) sind verbunden durch den ROS master. Dieser (mit dem parameter server zusammen) speichert und verbreitet public/private Parameter. Die Knoten können Nachrichten über Topics versenden, andere Knoten können sich diesen Topics subscriben. Neben dem PubSub Prinzip gibt es auch noch das Client-Server Modell, wodurch Knoten auch Services bereitstellen können und andere Knoten diese Services anfragen können. Services sind Funktionen, die aufgerufen werden, wenn ein Client eine Anfrage sendet.
ROS2 hat halt spezifische moderne Qualitätsanforderungen wie Security, Safety, Real Time, Readiness. Für den Nutzer ändert sich aber nicht viel, grundlegende Prinzipien bleiben gleich.

Wie sieht eine Message genau aus?

2 Kinematik und Dynamik

optional: **Joint Kinematics (Details)**:
optional: **Dynamics**:

2.0.1 Modelling Robot Kinematics:

URDF: Unified Robot Description Format. Baum-Struktur(?)
SDF: Simulation Descr. Format: part of Gazebo. Darstellung als Graph.
???

3 Grundlagen und Diskussion Autonomes Fahren

3.1 Diskussion

3.1.1 Was ist autonomes Fahren?

Das Fahrzeug kann ohne menschlichen Fahrer fahren, nämlich auch in unbekannten Umgebungen und unbekannten Hindernissen. Das Fahrzeug kann selber die Umgebung wahrnehmen und Steuern und Navigieren.

3.1.2 Wieso wurde autonomes Fahren entwickelt?

- Logistik Sektor
- Nachfrage Privatkäufer
- Mehr Komplexität – teurer verkaufen
- Verkehrssicherheit (als Vorwand)

3.1.3 Warum wurden die Fahrzeuge nicht schon (schneller) entwickelt?

- Rechenleistung
- KI Entwicklung
- fehlende Gesetzt (USA?)
- soziale Kritik ()
- nicht gut genug, nicht skalierbar

3.2 Levels der Autonomie, Gesetze

autonomes fahren vs. Assistenzsysteme! lvl 0 - lvl 2 : Nur Assistenz (das, was wir schon haben)
lvl 3 - lvl 5 : Die automatischen Fahrsysteme brauchen keine menschliche Aufmerksamkeit - Nur nach Aufforderung in lvl 3.

Im deutschen Gesetz sind vollständig autonome Fahrzeuge schon vorgesehen; Lediglich die technische Umsetzung fehlt noch.

3.3 Geschichte

Seit 20 Jahren werden autonome Autos entwickelt. Vorher war die Technologie halt noch nicht weit genug dafür. Angestoßen wurde die Entwicklung natürlich vom Militär Sektor in den 80ern. In den 90ern konnten die civilen Fahrzeuge und Straßen getestet werden. Fahrzeuge konnten eta 95% der Zeit autonom fahren. Aber halt auf der Autobahn. Es konnte auch nur die Fahrbahn erkennen, sich aber nicht global lokalisieren. Das war so mit Computer im Kofferraum für die Bildverarbeitung.
In den USA hängt die Entwicklung an Gesetzen. Die Hersteller setzen nur auf ADAS.
In 2005 gab es einen Contest in den USA. Der Aufbau war so: Die Autos fahren durch die Wüste, müssen nicht wirklich navigieren sondern fahren von GPS Punkt zu GPS Punkt. Die Hindernisse waren auch bekannt quasi und Verkehr gab es nicht. Also war halt auch nicht so ganz krass.

3.4 Verkehrssimulation

- Gazebo (with ROS, general 3D simulator incl. physics)
- Carla (focus on street env. + vehicles, ROS bridge)
- Udacity (self-driving-car-sim: focus on street etc, uses Unity)

- Unity (game engine, 3D and physics)
- TORCS (3D racing simulator)
- Commercial simulators (for FEM and CFD (comp. fluid dynamics)

4 Sensortechnik

Situation: Das Fahrzeug bewegt sich allein von einer Position zur nächsten. Dafür braucht es:

- moving base ?
- Modell der Umgebung
- Lokalisation
- Pfadplanung

Für das alles braucht es Sensordaten. Roboter haben meistens mehrere Sensoren. Die lassen sich in zwei Klassen unterteilen:

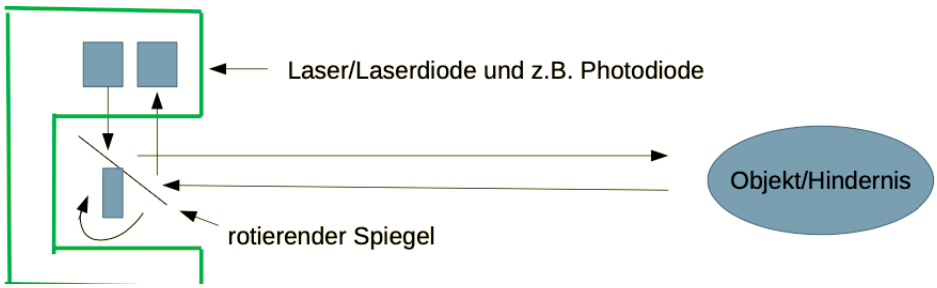
1. **Propriocepive Sensoren:** Um den Zustand des Roboters zu berechnen.
   - Beschleunigung und Rotationen
   - Motorstrom messen
   - Odometry (Raddrehsensoren, Winkelmessung, Drehratenmessung)
2. **exteroceprive sensors** messen externe Dinge zwischen Roboter und Umwelt, um den Zustand der äußeren Welt zu berechnen.
   - Kameras
   - Sensoren basierend auf elektromagnetischen Wellen (LiDAR, Radar)
   - Ultraschall Abstandssensoren
   - GPS
   - Kompass

Da kann man noch auf Seite 87ff lesen ... keine Lust gerade.

4.1 Sensoren

4.1.1 LiDAR

Light Detection And Ranging.



LiDAR und Radar können sogar die Geschwindingeit gemessen, durch aussenden und empfangen von Wellen in GH. (Wenn die wellen im selben GH wiederkommen, gibt es keine Geschwindingkeit.) Es gibt auch 3D Lidar. Vorteile:

- Hohe Weitwinkel Auflösung
- weite Distanz

Nachteile:

- Anfällig gegen Regen, Schnee
- Probleme bei reflektierenden Oberflächen
- kleine, weit entfernte Objekte schwierig erkennbar

4.1.2 Radar

Radion Detection and Ranging.
zB 1D Park-Sensoren. 2D vorne, oder 3D.
Ist insgesamt etwas besser in den Nachteilen vom LiDAR. Und auch in den Vorteilen.

4.1.3 Ultraschall

nothing.

4.1.4 Vision (Kamera?)

Normale Kameras, visuelles Spektrum.
Typen:
-2D monocular,
-stereo(3D), catadioptric,
-intelligent cameras,
-TOF mit Distanz, ...
-Weitwinkel !!Sensing: Siciliano page 92, 1179 (PDF-page 147, 1205), section 4

4.1.5 GNSS

global navigation satellite systems
whiteboard??

4.1.6 IMU

inertial measurement unit
combination of 2/3 of these 3D sensors:

- 3D accelerometer (Beschleunigungssensor)
- 3D gyroscope (rotations geschwindigkeit)
- magnetometer (magnetisches feld orientierung)

4.2 Aufgaben der Sensoren im Auto

Abstandssensoren

- Ultraschall (1D)
- elektromagnetische Wellen Sensoren
- in 1D, 2D und 3D
- Kamera

Objekterkennung

- Kamera

Interne Daten ... (keine Odometrie :( )

- IMU:
- Beschleunigungssensor
- Gyroskop/Rotoationsgeschwindigkeit
- Magnetometer

Navigation GPS

5 Kinematik

— same as before — ?

5.1 Grundlagen?

pose : Position und Orientierung der Links und Joints.
kinematik : Pose, Geschwindigkeit, Beschleunigung (und alle weiteren Ableitungen). study of different ways of representing the pose of a body.
serial chain : system of rigid bodies in which each member is connected to two others (except the first and last member)
fully parallel mechanism : two members are connected by multiple joints.

Handbuch: 1.2.1.
6 Koordinaten werden benötigt, um einen Körper im eukl. Raum darzustellen. Es gibt euklidische, sphärische oder cylindrische Koordinatendarstellung. Eine Position wird mit einem 3D Vektor dargestellt.
frame : Origin + set of 3 basis vectors

5.1.1 Translation und Rotation

Eine **Translation** ist ein Versatz, bei dem die Orientierung gleich bleibt, aber die Position sich ändert. Eine **Rotation** ist ein Versatz, bei dem die Position von mindestens einem Punkt des Körpers gleich bleibt. **Rotationsmatrix**: Die Orientierung von Vektor1 relativ zu Vektor2 kann beschrieben werden in einer Matrix. **Homogene Transformation**:

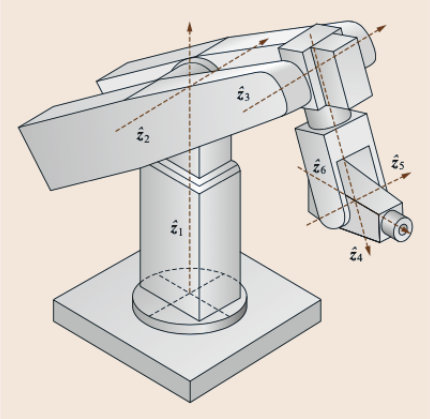
5.2 Joint Kinematics

- kinematische beschreibungen benutzen meistens Idealisierungen: absolute Starrheit oder gemetrisch perfekt
- Freiheitsgrade der Joints
- zwei Körper, die in Kontakt stehen, bilden einen Joint.
- **lower pair joints** : Kontakt nur über Oberfläche (zB Rad zu Kugelgelenk/Fassung)
- **higher pair joints** : Kontakt über Punkte oder Linien. (zB Rad zu Boden)

5.3 Geometrische Repräsentation

Khalil and Dombre: Die Konvention braucht nur vier (anstatt sechs) Parameter, um ein frame (relativ zu einem anderen) zu lokalisieren. Sofern der Roboter nach bestimmten Regeln konstruiert wurde.

i	α_i	a_i	d_i	θ_i
1	0	0	0	θ_1
2	$-\frac{\pi}{2}$	0	0	θ_2
3	0	a_3	0	θ_3
4	$-\frac{\pi}{2}$	0	d_4	θ_4
5	$\frac{\pi}{2}$	0	0	θ_5
6	$-\frac{\pi}{2}$	0	0	θ_6



In der Tabelle sind die geometrischen Parameter

- a_i : Länge des Links: Abstand der Rotationsachsen(?)
- α_i : Winkel zwischen den Rotationsachsen z
- d_i : Joint offset ??
- θ_i : Joint Winkel

Der Parameter α_i ist immer $(0, +\frac{\pi}{2}, -\frac{\pi}{2})$ = (0, orthogonal , orthogonal. Dadurch werden in T_i einige Terme Null. Die Matrix T_i repräsentiert eine homogene Transformation. Es ist eine Konkatenation von Rotation- und Translation. (siehe auch Forward-Kinematik)

5.4 Workspace

(reachable) Workspace Das Volumen, das der End-Effektor erreichen kann.
dextrous Workspace Das Volumen, das der Endeffector erreichen kann in beliebiger Ausrichtung.

Es gibt also bestimmten Dead Space quasi "innerhalb" des volumens, das der Roboter nicht erreichen kann, zB da wo er selbst steht oder so wie man selbst nicht seinen Ellenbogen ablecken kann.

5.5 Forward Kinematik

Problem: Finden der Position und Orientation des Endeffectors relativ zur Basis, wobei die Positionen aller Joints und die Werte aller Link Parameter gegeben sind. Wir wollen also die relative Pos/Ori finden. Es ist wichtig, um die Koordinationsalgorithmen für den Manipulator zu entwickeln. Wir kennen also die Parameter aller Joints und Links und wollen daraus die Position und Or. des Endeffektors berechnen.

- Tool frame : Endeffektor Center Point
- Station frame : Basis des Koordinatensystems

- In den Formeln in Table 1.8 steckt die form des Roboters drin (länge der arme) und die Orientierungen
- Theta 4-6 kommen nicht in den p Thermen vor. —¿ wieso? ???

$$ {}^0\boldsymbol{T}_6 = \begin{pmatrix} r_{11} & r_{12} & r_{13} & {}^0p_6^x \\ r_{21} & r_{22} & r_{23} & {}^0p_6^y \\ r_{31} & r_{32} & r_{33} & {}^0p_6^z \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Berechnung eines Vorwärts Kinematik Problems: Die Transformation T_i zwischen dem **tool frame** und dem **station frame** wird berechnet. Für die Transformation ${}^0T_6$ des 6DOF Roboterarm werden alle Transformationen von Joint zu Joint auf-konkateniert (multipliziert?).

$$ {}^0T_6 = {}^0\,T_1 \cdot \ldots \cdot {}^5\,T_6 \tag{1} $$

In wirklichkeit werden die Position und Orientierung in der Transformation aber separiert um die Berechnung zu vereinfachen. Und irgendwas mit Bäumen...

5.6 Inverse Kinematik

- Wie komme ich zu einer bestimmten Pose?
- Ich kenne also die Position und Orientierung des Endeffektors und möchte die Möglichkeiten der **Joint Positionen herausfinden**.
- Finden aller Joint Positionen auf Basis der homogenen Transformationsmatrix T_i zwischen den zwei Members.
- für unseren 6DOF manipulator haben wir ja T_6.
- Lösung inverser Probleme braucht Lösung nichtlinearer Gleichungen.
- 3 Gleichungen jeweils für Position und Rotation
- Es kann keine oder mehrere Lösungen geben
- Eine Lösung muss innerhalb des Workspaces liegen.
- es gibt verschiedene Darstellungsarten(?)

5.6.1 Lösungen in geschlossener Form

- Lösungen in geschlossener Form sind schneller als numerische Lösungen!
- Nachteil ist, dass diese für jeden Roboter einzeln gefunden werden muss. der Numerische Ansatz(?) ist generell.
- Wie findet man geschlossene Form? :

- im Allgemeinen können geschlossene Formen nur für 6DOF Roboter mit einer speziellen Kinematischen Struktur gefunden werden (wenn viele Parameter Null sind)
- Für jeden Joint θ_i habe ich eine Gleichung mit Lösungen
- geschlossene Formen lassen sich unterteilen in Algebraische Methode und Geometrische Methode:

Algebraische Methode

- identifizieren der signifikanten Gleichungen ...
- C1, C2, C3 finden durch Nullsetzen ... joa

Geometrische Methoden

- identifizieren von punkten am roboter ...
- Dekomposition von räumlichen Problem in separate **planare Probleme**
- die Gleichungen werden dann mittels algebraischer Manipulation ? gelöst
- Dekomposition des Problems in **inverse Position Kinematik** und **inverse Orientierung Kinematik**
-

$$ {}^0T_6^6T_5^5T_4^4T_3 = {}^0\ T_1^1T_2^2T_3 \tag{2}$$

Was genau sagt ${}^0T_6$ eigentlich aus? T ist die Transformation, die die Position (und orientierung) des End-Links beschreibt. Die anderen Ts beschreiben demnach die einzelnen Positionen der Links relativ zueinander.
Die Gleichung sagt aus, dass die Position 3 beschrieben werden kann durch a) ausgehend von der Endposition mit rückwärts-Transformationen oder b) ausgehend vom ursprung vorwärts-Transformationen.
Warum berechnen wir nur Thetas (θ_i)?

- Geometrisch: $\theta_{(1-3)}$: Position
- $\theta_{(4-6)}$: Orientierung.
- i bezeichnet den Joint
- Ich vermute, dass Theta der Variable Parameter des Roboters ist. D.h. der Roboter lässt sich durch vier Parameter beschreiben, 3 sind fest durch die Konstruktion des Roboters, und Theta beschreibt den Winkel der beweglichen Arme. Daher wollen wir θ berechnen für jeden Joint.

Die geschlossenen Formen reichen nicht aus weil

1. diese nur für max. 6DOF Roboter geeignet sind (warum eigentlich genau? –¿ irgendwas mit weil sie nur polynome 4. Grades maximal berechnen kann (?)
2. diese für jeden Roboter einzeln gefunden werden müssen, nicht allgemeingültig
3. in den Formen der geometrischen Berechnungen es Terme gibt, die nicht lösbar sind (durch Null teilen, Wurzel von negativer Zahl)

Dafür gibt es die numerischen Methoden

5.6.2 Numerische Methoden

- unabhängig von Roboter

- Nachteil: langsamer, können nicht immer alle Lösungen berechnen
- es gibt symbolic elim., continuation methods, iterative methods.

5.7 Instantane Kinematik

handelt von der Geschwindigkeit der Joints. Positionen der Joints müssen bekannt sein.

5.7.1 Forward Instantaneous Kinematics

Berechnung der totalen Geschwindigkeit des Endeffektors. Gegeben sind alle Positionen aller Joints. –¿ wichtig für dynamik

- das forward oder inverse position kinematik Problem muss vorher berechnet werden.

Jacobi Matrix Die zeitliche Differenzierung der forward position Gleichungen ergibt eine Menge an Gleichungen, die die Jacobi Matrix erstellen in der Form

$$v_N = J(q)\dot{q} \tag{3}$$

- v_n : spatial velocity end-effector
- q : vector composed of joint variablen (welche sind die genau?)
- $\dot{q}$: vektor mit aktuellen Winkelgeschwindigkeiten
- $J(q)$: $6 \times N$ matrix, non-linear functions $q_1 ... q_N$

N ist die Anzahl der Gelenkwinkel. Ich muss vorher die Gelenkwinkel kennen weil es wichtig ist, ob der Arm ausgestreckt ist oder nicht.
Was bedeutet q? Warum hängt q von J ab?
Ich denke, q bezeichnet erst mal die Gelenke. Genauer gesagt, q ist der Vektor der Joint Variablen

5.7.2 Inverse Instantane Kinematik

Wichtiges Thema; Problem: Finde die Geschwindigkeiten/Bewegung aller Joints, bei gegebener Endgeschwindigkeit und allen Positionen der Joints.

- Neben der berechnung der Endposition ist auch eine smoothe "Flugbahn" wichtig.
- wir suchen also $\dot{q}$, den Vektor mit allen Joint Geschwindigkeiten.
- Um $\dot{q}$ zu berechnen, wird die Jacobi Matrix invertiert
- $\dot{q} = J^{-1}(q)v_N$
- die meisten industriellen roboter sind simpel, sodass analytisch explizite gleichungen gefunden werden können (vs numerisches Invertieren)
- für komplexe roboter ist numerisches Invertieren die einzige Möglichkeit

5.8 Holonomes und Nicht-holonomes System

Holonomität? ist eine Eingenschaft von mechanischen Systemen **Holonom** : Reihenfolge egal, um an einen Punkt zu kommen (Hafenkran)
Nur die Joint Positions Variablen müssen beachtet werden.

Nicht-Holonom : Drehachse und Lenkachse, Reihenfolge der Aktionen wichtig!!
Also zeitliche Komponente muss beachtet werden

6 Fahrzeug Kinematik/Intelligent Vehicles

6.1 Arten von Rädern

Standard Rad vs. Special Rad.

6.1.1 Standard Rad

- passive fixed wheel
- off-centered orientable wheel
- active orientable wheel without
Standard Räder unterliegen dem nonholonomic velocity constraint. Eingeschränkte Beweglichkeit. **Schlepprad (Caster wheel)**

6.1.2 Special Rad

ist nun holonom durch schlaue sachen die mehr oder weniger gut funktionieren! **Schwedisches Rad** : Rad mit Rädern
Leg-like Wheel
Sphärische Räder

6.2 Ackerman achse

Das Model besagt, dass die zwei vorderen Räder im Fall des Autos nicht im selben Winkel geneigt werden, sondern ein Rad eine Orthogonale hinsichtlich der Linie zum Zentrum des Wendekreises bildet. D'accord? Weil beide Räder ja einen unterschiedlichen Wendekreis haben.

7 Path Tracking / Lokale Navigation

Angenommen, wir haben einen Pfad, und auch den ganzen Kinematik Kram des Fahrzeugs! Wie folgen wir dem Pfad? Dafür gibt es Regler, die anhand von sensordaten das Fahrzeug auf der Straße halten.
Wir müssen nebenbei aber auch noch auf Hindernisse und Straßenverkehrsregeln achten. Hier aber erst mal die grundlegende Führung des Fahrzeugs hinsichtlich Lenkung und Geschwindigkeit:

7.1 Lateral control / Querführung

Lateral control = Querführung ist das Problem, das Fahzeug au
Es gibt drei Arten für die Querführung: Pure pursuit, Stanely und MPC combined with sth.
Das Problem wird durch **image processing** erkannt.

Es gibt den **Cross-Track error**, der den Abstand zwischen Fahrzeug und Pfad beschreibt. Der **heading error** ist der Winkel zwischen Fahrbahn/Pfad und der aktuellen Fahrtrichtung.

Pure Pursuit

- Geometric path tracking Controller.
- **minimiert nur den Cross-Track error**
- using only the geometry of the vehicle kinematics and reference path.
- benutzt look-ahead/target point in fixed distance auf dem Pfad
- benutzt hintere Achse als referenzpunkt für das fahrzeug
- need to compute a steering angle.
- Proportionaler Controller.

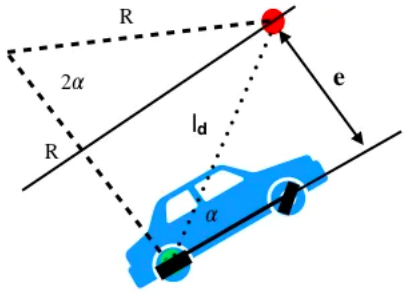


Fig4. Cross-track error

Stanely Controller

- auch geometrischer Controller
- Berechnet Winkel zwischen aktueller Fahrtrichtung und eigentlichem Pfad.
- **minimiert Cross-Track und Heading-error**
- Cross-Track: kürzester Abstand Vorderachse zu Pfad.
- smoother als PP

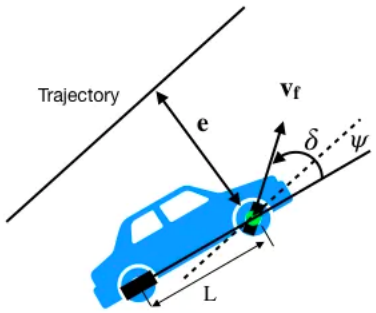


Fig5. Stanly geometric relationship

MPC (Model Predictive Controller)

- Wir haben eine Cost Function
-

7.2 Longitudinal Control/ Längsführung

- Speed/Cruis control
- Distance control
- ACC (adaptive cruise control)

Die Längsführung sprich die Geschwindigkeit (?) kann geregelt werden durch zB einen PID Controller. Die Daten für die Regelung kommen von Abstandssensoren bzw auch vom Lateral Controller.

7.3 Enabling Technologies

Fahrzeuge können entweder Markern folgen, wie Antennen, Magneten, Kabeln im Boden, Fahrbahnmarkierung, optischen Linien.
Aber ich denke das ist ja nicht unser Ziel, wir wollen uns ja frei bewegen. Dafür brauchen wir:

- Position (kinematischer und dynamischer State) wie ist dynamisch hier gemeint?
- Zustand der Umgebung
- Zugang zu digitalen Karten und Satelliten Daten
- (Zustand des Fahrers/Mitfahrer)
- (Kommunikation mit Roadside infrastructure)

Das Fahrzeug braucht weiterhin die Kontrolle über die Geschwindigkeit und Lenkung. Moderne Fahrzeuge: elektrisches Lenken, steering by wire und elektrische Bremsen. Problematik!

8 Pfadplanung und Navigation

8.1 Grundlegend...

Wir brauchen

1. Umgebungsmodell
2. Lokalisierung
3. Pfadplanung

Im Gegensatz zur Navigation und Pfadplanung steht die "Motion Planung", die Regelung der Geschwindigkeit und Beschleunigung, also sehr lokal.
In der Navigation wollen wir nicht alles auf einmal planen, sondern Stück für Stück.
Es gibt verschiedene Algorithmen,

- Random Walk (siehe Babou)
- Dijkstra
- A
- ... die kommen dann noch?

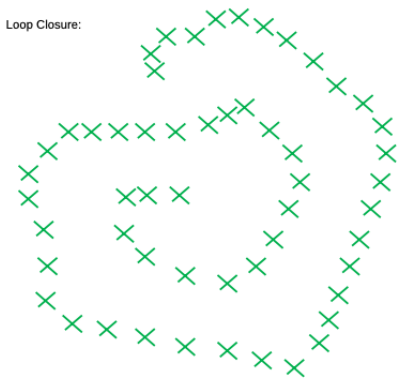
Es gibt halt lokale und globale Planung – lokal ist wie schon besprochen das Ausweichen von Hindernissen, das Fahren auf der Vorgesehenen linie, das Verhalten in Situationen, Verkehr, ... oder?

8.2 Odometrie

Odometrie beschreibt Verfahren zur Lokalisation anhand der Roboter-intern erfassbaren Daten. Die Daten werden von den Aktuatoren (Rädern) erfasst. Eine Strecke wird versucht zu messen anhand der Anzahl der Rad-Umdrehungen. Das Verfahren ist aber sehr ungenau und kann maximal zB für kurze Strecken im Tunnel ohne GPS empfang eingesetzt werden.
Gründe für Abweichung können Schlupf oder komischer Untergrund sein.
In der Seefahrt wurde das Verfahren verwendet (Knoten gezählt).

9 Lokalisation / Erstellung einer Karte

Ok in diesem Kapitel geht es um die Erstellung einer Karte damit der Roboter sich richtig Lokalisieren kann. Der Roboter muss irgendwie durch die Gegend fahren und seine Umgebung wahrnehmen können, um eine Karte zu erstellen. Das Schwierige dabei ist, die selben Orte auf der Karte richtig darzustellen, also zu erkennen, wann man am selben Ort ist, den man schon ein mal gesehen hat. Mit dem Random Walk Algorithmus könnte man ja zB die Gegend erkunden und Wände auf einer Karte Malen. Durch Odometrie könnte erfasst werden, wie weit der Roboter fährt und im Welchen Winkel er sich gedreht hat. Aber weil die Methode gar nicht gut ist, gibt es bestimmt andere Dinge die man tun kann die hoffentlich hier besprochen werden.
Generell kann halt das Problem des Loop closure passieren: dass der roboter genau im Kreis fährt, aber es nicht checkt. Auf der Karte denkt er, er wäre woanders. Wrong wrong wrong!
S. 533, Model Matching. Man möchte die beobachteten Daten mit gespeicherten Daten vergleichen, um möglicherweise Orte wiederzuerkennen. SLAM kann zB Karten erstellen und lokalisieren gleichzeitig. Der eingesetzte Algorithmus fürs Matching hängt ab von der Komplexität der Struktur. Für einfache geometrische Strukturen wie 3D Punkte oder Dreiecke, Algorithmen wie ICP kann benutzt werden. (Und für komplexere geom. Strukturen dann SLAM?)



9.1 Karten

Kapitel 36, "World Models"

Occupancy grids / Belegungskarte

- reine LiDAR Daten, Punktwolke, ist noch keine Belegungskarte.
- eine Abbildung einer 3D Punktwolke auf 2,5D Quaderdarstellung ist eine Belegungskarte.
- Auf welcher Höhe das Hindernis ist, spielt keine Rolle.

Line Maps

- Aus Messpunkten werden Supporterlinien gefunden, nacheinander
- höhere Auflösung im Vergleich zu Metrischen Karten
- Speichereffizient

Metrische Karten

- meist 2D - Grid
- Auflösung in Meter
- Speicherintensiv

Topologische Karte

- Graphenbasiert
- Speichereffizient
- Voronoi Diagramm
- einfache Entscheidungsfindung
- Pfad führt immer in der Mitte entlang

Landmarken-basierte Karte

- ???

Cost Maps

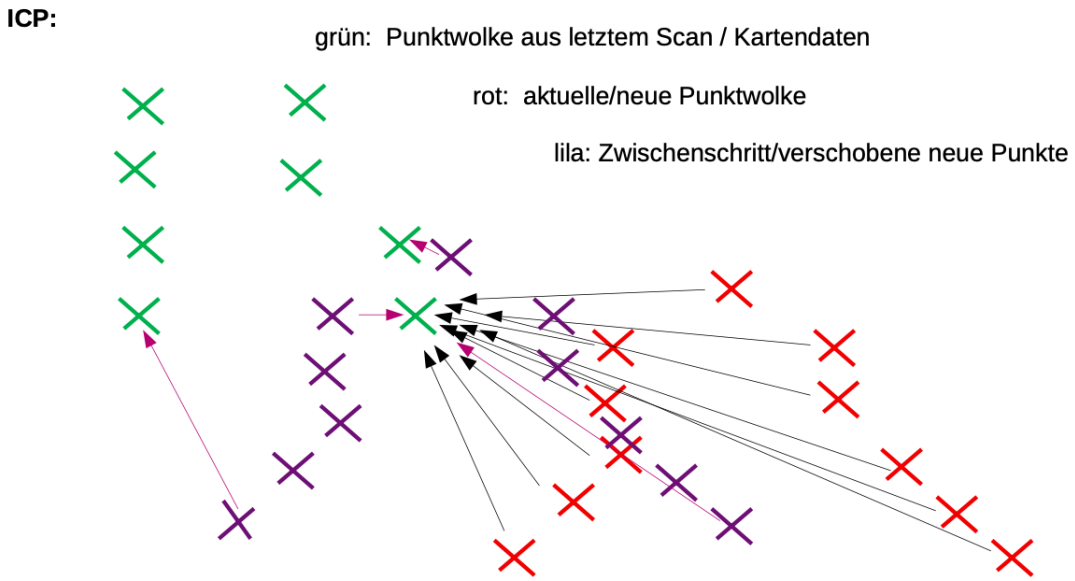
- Einfärbung, um Traversierbarkeit darzustellen.

9.2 Algorithmen

9.2.1 Grundlegende Algorithmen

Kalman Filter Mathematisches Verfahren zur iterativen Schätzung von Systemzuständen. Im autonomen Fahren wird der Algorithmus verwendet, um Daten von mehreren Sensoren zu vereinigen? "Data Fusion". Wird für mapping und localization verwendet.
Ich finde noch keine gute Quelle dafür..

ICP (Iterative Colosest Point) Das ist ein Algorithmus, um "Orte" wiederzuerkennen. Er findet Parameter, die eine point cloud auf eine andere matcht? ICP minimiert den Abstand zwischen zwei point clouds. Also es findet (möglicherweise?) die Transformation (Translation, Rotation) von Point Cloud A (aktuell), sodass sie auf Point Cloud B (Referenz)



passt. Iteriere:

1. Finde für jeden Punkt aus A den nahesten Nachbarn aus B
2. Schätze die Transformation, sodass diese Menge an Distanzen minimiert wird
3. Transformiere A. (A nähert sich B also an)

RANSAC Eine Methode um Outliers zu identifizieren und rauszufiltern.
Particle Filter

9.3 Grundlagen Algorithmen?

Random Walk, AStern, Dijkstra, ... durch Odometry,