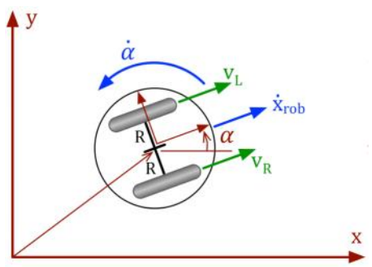
ROME 2

Kinematik mobiler Roboter

Rückwärtskinematik: Berechnung der Radgeschwindigkeit (und Auslenkung) aus der Translations- und Rotationsgeschwindigkeit

Vorwärtskinematik: Berechnung der Translations- und Rotationsgeschwindigkeit aus Radgeschwindigkeit (und Auslenkung)

Differentialantrieb



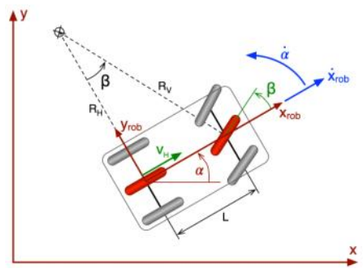
v_L, v_R = Radgeschwindigkeit
$\dot{x}_{rob}$ = Translationsgeschwindigkeit
$\dot{\alpha}$ = Rotationsgeschwindigkeit
x, y, α = Globale Lage

$$v_L = \dot{x}_{rob} - R \cdot \dot{\alpha}$$
$$v_R = \dot{x}_{rob} + R \cdot \dot{\alpha}$$

$$\dot{x}_{rob} = \frac{1}{2} \cdot (v_L + v_R)$$
$$\dot{\alpha} = \frac{1}{2 \cdot R} \cdot (v_R - v_L)$$

$$x = \int \cos(\alpha) \cdot \dot{x}_{rob} \cdot \mathrm{d}t + x_0 \qquad x_k = x_{k-1} + \cos(\alpha_{k-1} + \dot{\alpha}_{k-1} \cdot T) \cdot \dot{x}_{rob,k-1} \cdot T$$
$$y = \int \sin(\alpha) \cdot \dot{x}_{rob} \cdot \mathrm{d}t + y_0 \qquad y_k = y_{k-1} + \cos(\alpha_{k-1} + \dot{\alpha}_{k-1} \cdot T) \cdot \dot{x}_{rob,k-1} \cdot T$$
$$\alpha = \int \dot{\alpha} \cdot \mathrm{d}t + \alpha_0 \qquad \alpha_k = \alpha_{k-1} + \dot{\alpha}_{k-1} \cdot T$$

Gelenkte Räder



β = Auslenkung des gelenkten Rades
v_H = Radgeschwindigkeit ungelenkt
v_V = Radgeschwindigkeit gelenkt
$\dot{x}_{rob}$ = Translationsgeschwindigkeit
$\dot{\alpha}$ = Rotationsgeschwindigkeit
x, y, α = Globale Lage

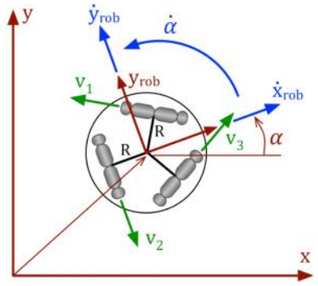
$$\beta = \mathrm{atan}\left(\frac{\dot{\alpha} \cdot L}{\dot{x}_{rob}}\right)$$
$$v_H = \dot{x}_{rob}$$
$$v_V = \frac{\dot{x}_{rob}}{\cos(\beta)}$$

$$\dot{x}_{rob} = v_H, \quad oder$$
$$\dot{x}_{rob} = v_V \cdot \cos(\beta)$$
$$\dot{\alpha} = \frac{\dot{x}_{rob}}{R_H} = \frac{\dot{x}_{rob}}{L} \cdot \tan(\beta)$$

$$x = \int \cos(\alpha) \cdot \dot{x}_{rob} \cdot \mathrm{d}t + x_0 \qquad x_k = x_{k-1} + \cos(\alpha_{k-1} + \dot{\alpha}_{k-1} \cdot T) \cdot \dot{x}_{rob,k-1} \cdot T$$
$$y = \int \sin(\alpha) \cdot \dot{x}_{rob} \cdot \mathrm{d}t + y_0 \qquad y_k = y_{k-1} + \cos(\alpha_{k-1} + \dot{\alpha}_{k-1} \cdot T) \cdot \dot{x}_{rob,k-1} \cdot T$$
$$x = \int \dot{\alpha} \cdot \mathrm{d}t + \alpha_0 \qquad \alpha_k = \alpha_{k-1} + \dot{\alpha}_{k-1} \cdot T$$

Holonome Räder

Omni-Wheels



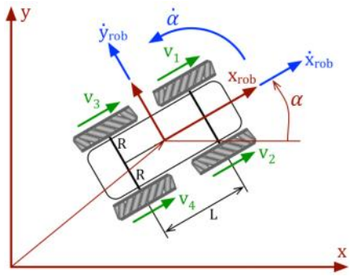
v_1, v_2, v_3 = Radgeschwindigkeit
$\dot{x}_{rob}, \dot{y}_{rob}$ = Translationsgeschwindigkeit
$\dot{\alpha}$ = Rotationsgeschwindigkeit
x, y, α = Globale Lage

$$v_1 = -\frac{\sqrt{3}}{2} \cdot \dot{x}_{rob} + \frac{1}{2} \cdot \dot{y}_{rob} + R \cdot \dot{\alpha}$$
$$v_1 = -\dot{y}_{rob} + R \cdot \dot{\alpha}$$
$$v_1 = \frac{\sqrt{3}}{2} \cdot \dot{x}_{rob} + \frac{1}{2} \cdot \dot{y}_{rob} + R \cdot \dot{\alpha}$$

$$\dot{x}_{rob} = \frac{1}{\sqrt{3}} \cdot (v_3 - v_1)$$
$$\dot{y}_{rob} = \frac{1}{3} \cdot (v_1 - 2 \cdot v_2 + v_3)$$
$$\dot{\alpha} = \frac{1}{3 \cdot R} \cdot (v_1 + v_2 + v_3)$$

$$\dot{x} = \cos(\alpha) \cdot \dot{x}_{rob} - \sin(\alpha) \cdot \dot{y}_{rob}$$
$$\dot{y} = \sin(\alpha) \cdot \dot{x}_{rob} + \cos(\alpha) \cdot \dot{y}_{rob}$$
$$x = \int \dot{x} \cdot \mathrm{d}t + x_0$$
$$y = \int \dot{y} \cdot \mathrm{d}t + y_0$$
$$\alpha = \int \dot{\alpha} \cdot \mathrm{d}t + \alpha_0$$

Mecanum-Wheels

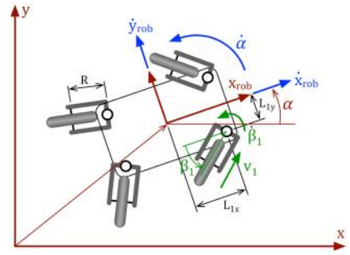


v_1, v_2, v_3, v_4 = Radgeschwindigkeit
$\dot{x}_{rob}, \dot{y}_{rob}$ = Translationsgeschwindigkeit
$\dot{\alpha}$ = Rotationsgeschwindigkeit
x, y, α = Globale Lage

$$v_1 = \dot{x}_{rob} - \dot{y}_{rob} - \left(\frac{L}{2} + R\right) \cdot \dot{\alpha}$$
$$v_2 = \dot{x}_{rob} + \dot{y}_{rob} + \left(\frac{L}{2} + R\right) \cdot \dot{\alpha}$$
$$v_3 = \dot{x}_{rob} + \dot{y}_{rob} - \left(\frac{L}{2} + R\right) \cdot \dot{\alpha}$$
$$v_4 = \dot{x}_{rob} - \dot{y}_{rob} + \left(\frac{L}{2} + R\right) \cdot \dot{\alpha}$$

$$\dot{x}_{rob} = \frac{v_1 + v_2 + v_3 + v_4}{4}$$
$$\dot{y}_{rob} = \frac{v_2 + v_3 - v_1 - v_4}{4}$$
$$\dot{\alpha} = \frac{v_2 + v_3 - v_1 - v_4}{4\left(\frac{L}{2} + R\right)}$$

Castor-Räder



$\dot{\beta}_1, \ldots$ = Auslenkung Castorrad
$v_1, \ldots$ = Radgeschwindigkeit Castorrad
$\dot{x}_{rob}, \dot{y}_{rob}$ = Translationsgeschwindigkeit
$\dot{\alpha}$ = Rotationsgeschwindigkeit
x, y, α = Globale Lage

Berechnungen für ein Rad:

$$x_{rob,1} = L_{1x} - R \cdot \cos(\beta_1)$$
$$y_{rob,1} = -L_{1x} - R \cdot \sin(\beta_1)$$

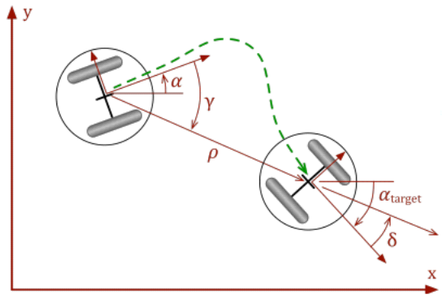
$$\dot{x}_{rob,1} = \dot{x}_{rob} - y_{rob,1} \cdot \dot{\alpha}$$
$$\dot{y}_{rob,1} = \dot{y}_{rob} + x_{rob,1} \cdot \dot{\alpha}$$

$$v_1 = \dot{x}_{rob,1} \cdot \cos(\beta_1) + \dot{y}_{rob,1} \cdot \sin(\beta_1)$$
$$\dot{\beta}_1 = -\frac{1}{R} \cdot (-\dot{x}_{rob,1} \cdot \sin(\beta_1) + \dot{y}_{rob,1} \cdot \cos(\beta_1))$$

Baumaschinen

Positionsregelung

Differentialantrieb



ρ =Abstand zum Ziel
γ =Winkel zum Ziel im lokalen Koordinatensystem des Roboters
δ =Abweichung der Zielorientierung von der Gerade ρ

$$\rho = \sqrt{(x_{target} - x)^2 + (y_{target} - y)^2}$$

$$\gamma = atan2(y_{target} - y,\ x_{target} - x) - \alpha$$

$$\delta = \gamma + \alpha - \alpha_{target}$$

$$\dot{x}_{rob} = k_1 \cdot \rho \cdot cos(\gamma)$$

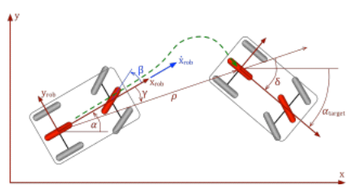
$$\dot{\alpha} = k_2 \cdot \gamma + k_1 \cdot sin(\gamma) \cdot cos(\gamma) \cdot \frac{\gamma + k_3 \cdot \delta}{\gamma}$$

k_1: Skaliert die Translationsgeschwindigkeit abhängig vom Abstand und Winkel γ; bei seitlicher Zielposition ($\gamma = \pm 90°$) ist $v = 0$ wegen $\cos(\gamma)$.
k_2: Bestimmt, wie schnell sich der Roboter in Richtung des Ziels dreht (Einfluss von γ).
k_3: Steuert die Ausrichtung auf die Zielorientierung (Einfluss von δ); höhere Werte sorgen für exakteres Einfahren mit korrekter Endausrichtung.

Implementierungsaspekte:

- Bei **ρ = 0** sind **γ** und **δ** undefiniert; der Regler wird nicht mehr benötigt.
- Bei **γ = 0** tritt Division durch Null auf; **ω** kann dann einfach auf 0 gesetzt werden.
- **γ** und **δ** müssen im Bereich $(-\pi,\pi]$ liegen; ggf. durch Addition oder Subtraktion von 2π normalisieren.

Ackermann-Steuerung



$$\rho = \sqrt{(x_{target} - x)^2 + (y_{target} - y)^2}$$

$$\gamma = atan2(y_{target} - y,\ x_{target} - x) - \alpha$$

$$\delta = \gamma + \alpha - \alpha_{target}$$

$$\dot{x}_{rob} = k_1 \cdot \rho$$

$$\beta = k_2 \cdot \gamma + k_3 \cdot \delta$$

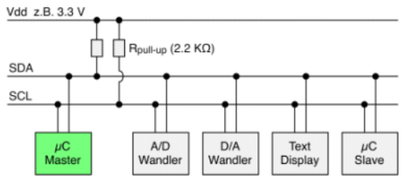
wobei

$k_1 > 0$
$k_3 < 0$
$k_2 > k_1$

IO-Peripherie

I2C (Inter-Integrated Circuit)

Jahr	Modus	Übertragungsrate
1982	Standard Mode	100 kBit/s
1992	Fast Mode	400 kBit/s
2007	Fast Mode Plus	1 MBit/s
1998	High Speed Mode	3.4 MBit/s
2012	Ultra Fast-mode	5.0 MBit/s

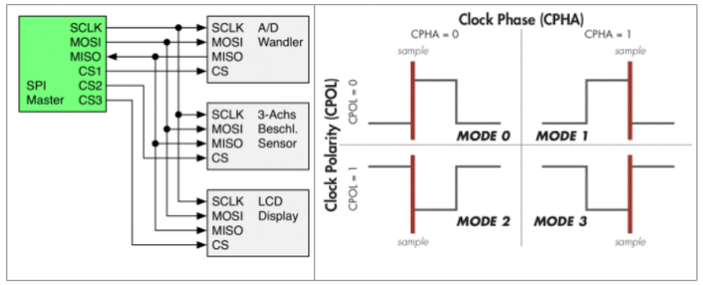


Master (grün) sendet

Slave (grau) sendet(nach Aufforderung von Master)

Vorteile	Nachteile
+ Nur 2 Leitungen, 112 Geräte möglich + Gute Mikrocontroller- und Sensorunterstützung + Einfache Adressierung	– Langsam (max. 3,4 MHz) – Störanfällig bei längeren Leitungen – Keine Fehlererkennung – Halb-Duplex

SPI (Serial Peripheral Interface)

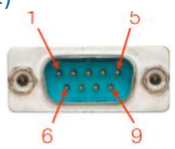


CPOL = Polarität von SCLK
CPHA = Phasenverschiebung von SCLK

Vorteile	Nachteile
+ Sehr schnell (mehrere 10 MHz) + Vollduplex, geringe Latenz + Stabil und zuverlässig	– Viele Leitungen nötig (1 SS pro Slave) – Keine Norm, keine Fehlererkennung

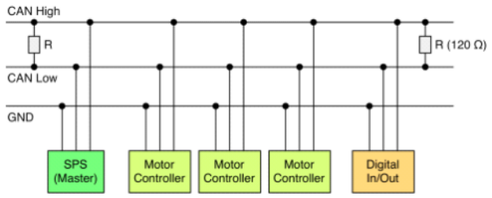
RS232 (Recommended Standard 232)

Pin	Funktion	
2	RxD	Receive Data
3	TxD	Transmit Data
5	Gnd	Signal Ground
7	RTS	Request to Send
8	CTS	Clear to Send



Vorteile	Nachteile
+ Einfach, weit verbreitet + Gute PC-Kompatibilität + Reichweite bis 15 m	– Langsam – Nur Punkt-zu-Punkt – Unübliche Pegel, keine Fehlerkorrektur

CAN (Controller Area Network)



Genormt

Baudrate	Leitungslänge
1 MBit/s	40 m
500 kBit/s	100 m
250 kBit/s	200 m
125 kBit/s	500 m
10 kBit/s	6 km

Vorteile	Nachteile
+ Robust, störsicher + Fehlererkennung integriert + Echtzeitfähig, viele Teilnehmer	– Komplex – Zusätzliche Hardware nötig – Datenrate und Nutzlast begrenzt

Vergleich

Merkmal	I²C	SPI	RS232	CAN
Bus oder Punkt-zu-Punkt	Bus	Punkt-zu-Punkt	Punkt-zu-Punkt	Bus
Anzahl Leitungen	2	≥4	≥3	3
Geschwindigkeit	Meist ≤400kHz	Bis zu 40MHz	Bis zu 115200Hz	Bis zu 1 MHz
Kabellänge	Kurz (~1 m)	Sehr kurz	Mittel (~15 m)	Lang (>40 m)
Fehlererkennung	Nein	Nein	Nein	Ja
Multimaster	Möglich	Nein	Nein	Ja
Adressierung	Ja	Hardware	Hardware	Ja (ID-basiert)

Lokalisierung

Relative Lagemessung (Koppelnavigation)

Die relative Lagemessung mobiler Systeme wird als **Dead Reckoning** bezeichnet und basiert auf der Bestimmung der Position durch Messung von Bewegungsrichtung, Geschwindigkeit und Zeit. Ein typisches Beispiel ist die **Odometrie** mittels Drehzahlsensoren an den Rädern von Robotern oder Fahrzeugen.

Sensoren

- Encoder & Drehzahlsensoren
- IMU (Inertial Measurement Unit)
  - Beschleunigungssensoren
  - Gyrosensoren (Drehraten)
- Doppler Sensoren
- Kameras / Visionsysteme

Vorteile

- Kostengünstig (Sensoren oft schon vorhanden)
- Hohe Messgeschwindigkeit (z. B. 1 kHz Abtastrate)

Nachteile

- Startposition muss manuell gesetzt werden
- Positionsdrift durch systematische und dynamische Fehler
  - Systematisch: abweichende Masse, beschränkte Auflösung Winkelencoder
  - Dynamisch: Schlupf oder schleudern

Absolute Lagemessung

Sensoren

- GPS
- Ultraschall, IR-Distanzsensoren
- LIDAR / Laserscanner
- 3D & TOF Kameras
- Radio Beacons (Bluethooth oder UWB)
- 2D Kameras / Visionsysteme
- Kompass

Vorteile

- Werte direkt brauchbar für Navigation

Nachteile

- Oft Langsam
- Oft nicht verfügbar
  - GPS in Gebäuden
  - Beacons sind verdeckt durch Hindernisse

Abs. Lagemessung mit LIDAR

Position in Roboter Koordinatensystem:

$${}^{rob}P_{1,x} = R_1 \cdot cos(\gamma_1)$$
$${}^{rob}P_{1,y} = R_1 \cdot sin(\gamma_1)$$

R = Abstand des Markers
γ = Winkel zum Marker

Position in globalem Koordinatensystem:

$$\begin{bmatrix} {}^{0}P_{1,x} \\ {}^{0}P_{1,y} \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\alpha) & -sin(\alpha) & {}^{0}x \\ sin(\alpha) & cos(\alpha) & {}^{0}y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^{rob}P_{1,x} \\ {}^{rob}P_{1,y} \\ 1 \end{bmatrix}$$

Von mehreren Punkten:

$$\begin{bmatrix} {}^{0}P_{1,x} & {}^{0}P_{1,y} \\ {}^{0}P_{2,x} & {}^{0}P_{2,y} \\ \vdots & \vdots \\ {}^{0}P_{n,x} & {}^{0}P_{n,y} \end{bmatrix} = \begin{bmatrix} {}^{rob}P_{1,x} & {}^{rob}P_{1,y} & 1 \\ {}^{rob}P_{2,x} & {}^{rob}P_{2,y} & 1 \\ \vdots & \vdots & \vdots \\ {}^{rob}P_{n,x} & {}^{rob}P_{n,y} & 1 \end{bmatrix} \cdot \begin{bmatrix} cos(\alpha) & sin(\alpha) \\ -sin(\alpha) & cos(\alpha) \\ {}^{0}x & {}^{0}y \end{bmatrix}$$

$$\Rightarrow P_0 = P_{rob} \cdot T$$

Berechnen der eigenen Position

Aus $P_0 = P_{rob} \cdot T$ erhält man: $T = P_{rob}^{-1} \cdot P_0$

respektive für n > 3: $T = P_{rob}^{\#} \cdot P_0$

wobei: $P_{rob}^{\#} = P_{rob}^{T} \cdot (P_{rob} \cdot P_{rob}^{T})^{-1}$

$${}^{0}x = T_{3,1}$$
$${}^{0}y = T_{3,2}$$
$$\alpha = atan2(T_{1,2}, T_{1,1})$$

Filtern von Sensorsignalen

Tiefpassfilter

1. Ordnung

$$G(s) = \frac{\omega_c}{s + \omega_C} \quad \omega_c = 2\pi \cdot f_c, \qquad f_c = eckfrequenz$$

Zeitdiskret:

$$y_k = s_f \cdot x_k + \left(1 - s_f\right) \cdot y_{k-1}, \qquad s_f = smoothing\ factor$$

s_f beschreibt den Anteil des Einganges. Je kleiner desto mehr Filterwirkung.

$$s_f = \frac{2\pi \cdot f_C \cdot T}{1 + 2\pi \cdot f_C \cdot T}$$

$$\Rightarrow f_C = \frac{s_f}{2\pi \cdot \left(1 - s_f\right) \cdot T}$$

2. Ordnung

$$G(s) = \frac{\omega_C^2}{s^2 + 2 \cdot D \cdot \omega_C \cdot s + \omega_c^2}$$

Zeitdiskret:

$$z_k = \begin{bmatrix} 1 + \omega_c T & T \\ -\omega_c^2 T & 1 - \omega_c T \end{bmatrix} \cdot \mathrm{e}^{-\omega_c T} \cdot z_{k-1} + \begin{bmatrix} \frac{1 - \left(1 + \omega_c T\right) \cdot e^{-\omega_c T}}{\omega_C^2} \\ T\mathrm{e}^{-\omega_C T} \end{bmatrix} \cdot x_k$$

$$y_k = \begin{bmatrix} w_C^2 & 0 \end{bmatrix} \cdot z_k$$

Digitale Filter

Median Filter

Den Median über die letzten *n* Werte

Moving Average

Der Mittelwert über die letzten *n* Werte

$$y_k = \frac{\sum_{i=0}^{n-1} x_{k-i}}{n}$$

Finite Impulse Response (FIR) Filter

Gewichteter Mittelwert der letzten *n* Werte

$$y_k = \sum_{i=0}^{n-1} b_i \cdot x_{k-i}$$

Wobei die Summe der $b_i = 1$ sein muss

Modelbasierte Filter

Klassische Filter wie Tiefpass- oder FIR-Filter **dämpfen** und **verzögern** stark verrauschte Signale, wenn sie effektiv filtern sollen. **Modellbasierte** Filter umgehen dieses Problem, indem sie mithilfe eines mathematischen Modells die **erwarteten Messwerte schätzen** und so bessere Ergebnisse liefern.

- Zustandsregler mit Luenberger Beobachter
- Geschwindigkeitsbeobachter
- Kalman-Filter

Kalman-Filter

Kalman-Filter sind zeitdiskrete Filter für lineare Systeme. Sie bestehen aus zwei Schritten: Prädiktion (Vorhersage des nächsten Zustands mit einem Modell) und Korrektur (Anpassung mit Messwerten). Sie berücksichtigen dabei statistische Unsicherheiten im Modell **und** den Messungen. Für nichtlineare Systeme gibt es erweiterte Versionen wie den Extended Kalman-Filter.

Zustandsmodell:

$$x_k = A \cdot x_{k-1} + B \cdot u_{k-1} + w_{k-1}$$
$$z_k = H \cdot x_k + v_k$$

x_k = Zustandsvektor u_k = Steuergrösse w_k = Prozessrauschen

z_k = Messungen v_k = Messrauschen H = C Ausgangsmatrix

Kovarianz Matrizen:

$$Q = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$
$$R = \begin{bmatrix} \sigma_3^2 \end{bmatrix}$$

Q = Kovarianz Matrix Prozessrauschen *R* = Kovarianz Matrix Messrausche

Standartabweichungen:

σ_1 = von Zustand 1 σ_2 = von Zustand 2 σ_3 = von Messgrösse 1

Prädiktion:

$$\hat{x}_k = A \cdot \hat{x}_{k-1} + B \cdot u_{k-1}$$
$$P_k = A \cdot P_{k-1} \cdot A^T + Q$$

$\hat{x}_k$ = Schätzung des Zustandes P_k = Kovarianz-Matrix des Zustandes

Korrektur

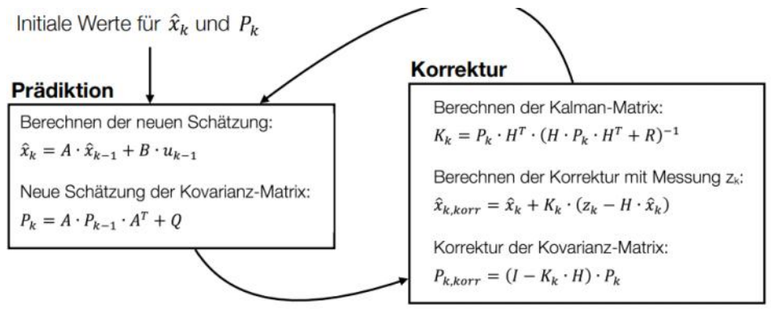
$$K_k = P_k \cdot H^T \cdot \left(H \cdot P_k \cdot H^T + R\right)^{-1}$$

K_k = Kalman-Matrix (Korrekturmatrix)

Geschätzen Zustand plus messung-schätzung * Korrekturmatrix

$$\hat{x}_{k,korr} = \hat{x}_k + K_k \cdot \left(z_k - H \cdot \hat{x}_k\right)$$
$$P_{k,korr} = \left(I - K_k \cdot H\right) \cdot P_k$$

$\hat{x}_{k,korr}$ = Korrektur der Schätzung des Zustandes	$P_{k,korr}$ = korrigierte Kovarianz Matrix des Zustandes



Einstellen des Filters über *Q* und *R*

Sensor Fusion

Sensorfusion bedeutet, mehrere Sensorsignale zu kombinieren, um ein genaueres, robusteres oder vollständigeres Bild eines Zustands zu erhalten, als es ein einzelner Sensor liefern könnte.

Komplementärfilter

Ein Komplementärfilter kombiniert zwei Sensorsignale, indem er hochfrequente Anteile vom einen (z. B. Gyroskop) und niedrigfrequente Anteile vom anderen (z. B. Beschleunigungssensor) verwendet, um eine stabile und schnelle Schätzung zu erzeugen.

Mit Kalman-Filter

Es ändert sich hier die Anzahl an Messgrössen.

$$\begin{bmatrix}\alpha_{y,k}\\ \omega_{y,k}\end{bmatrix} = \begin{bmatrix}1 & T\\ 0 & 1\end{bmatrix} \cdot \begin{bmatrix}\alpha_{y,k-1}\\ \omega_{y,k-1}\end{bmatrix} + w_{k-1}$$

$$z_k = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \cdot \begin{bmatrix}\alpha_{y,k}\\ \omega_{y,k}\end{bmatrix} + v_k$$

$$Q = \begin{bmatrix}\sigma_{q,\alpha}^2 & 0\\ 0 & \sigma_{q,\omega}^2\end{bmatrix} \qquad R = \begin{bmatrix}\sigma_{r,\alpha}^2 & 0\\ 0 & \sigma_{r,\omega}^2\end{bmatrix}$$

Ansonsten wie gehabt beim Kalman-Filter

Navigation

Ohne Karte

Die **Reaktive Navigation** ist die einfachste Form der Navigation für einen Roboter ohne Karte. Hierbei reagiert der Roboter auf Hindernisse und folgt dann seinem Vorgeschriebenen Protokoll, z.B. drehe nach rechts und fahre weiter geradeaus. Es kann auch eine Position angefahren werden wenn diese Markiert ist (z.B. Eine Lichtquelle).

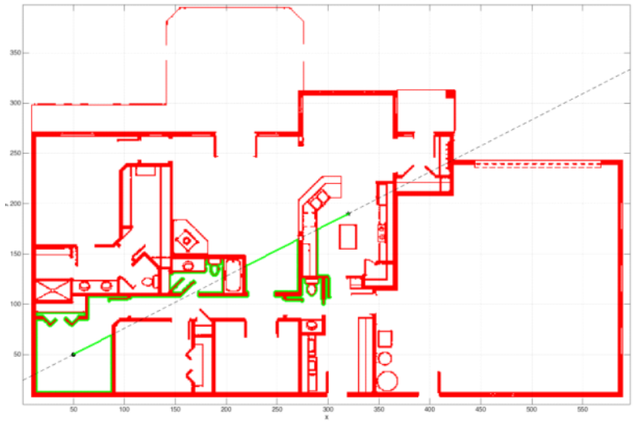
Bugalgorithmus

Fortgeschrittener als allg. Reaktive Navigation.

- Berechne gerade Linie zwischen aktuellen Startposition und Zielposition
- Roboter bewegt sich in kleinen Inkrementen entlang Linie
- Wenn er Auf Hindernis trifft folge diesem im Uhrzeigersinn bis er wieder zur Linie findet.
    - Wenn näher am Ziel als bei verlassen folge der Linie

Einschränkungen

- Der Roboter bewegt sich in einem diskreten Gitter.
- Er kann sich frei in alle Richtungen bewegen.
- Seine aktuelle Position ist bekannt.
- Er erkennt nur direkte Hindernisse und das Ziel.
- Die Karte dient nur zur Visualisierung, nicht zur Planung.



Mit Gitterkarte

Gitterkarten unterteilen die Umgebung in Zellen, wobei Hindernisse markiert werden. Sie bieten eine einfache Darstellung, erfordern aber bei hoher Auflösung viel Speicher und Rechenleistung. Für kleine Mikrocontroller, etwa in günstigen Staubsaugerrobotern, kann das problematisch sein.

Distanztransformation

Die Distanztransformation berechnet den Weg zum Ziel, indem sie jeder befahrbaren Zelle einen Distanzwert zuweist.

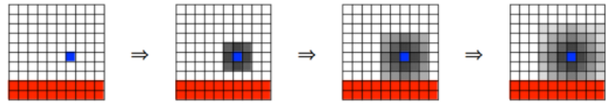
- Start ist das Ziel: Benachbarte Zellen erhalten Distanz 1.0 (direkt) bzw. 1.41 (diagonal).
- Diese Werte werden schrittweise auf weitere Nachbarn übertragen, jeweils um 1.0 oder 1.41 erhöht.
- Der Vorgang wird wiederholt, bis alle Zellen bewertet sind.

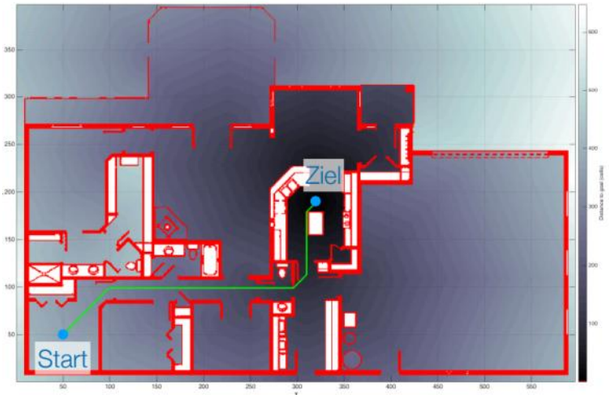
Vorteile:

- Einfache Umsetzung
- Übersichtliche Darstellung der Distanzen
- Ermöglicht robuste Wegplanung

Nachteile:

- Rechenaufwändig bei großen Karten
- Benötigt Omnidirektionaler Roboter
- Hindernisränder werden oft zu nah passiert





Probabilistic Road map Method (PRM)

Die PRM-Methode (Probabilistic Roadmap) reduziert den Rechenaufwand der Distanztransformation und ermöglicht trotzdem optimale Wege.
Dazu wird eine Netzkarte mit zufällig gewählten, hindernisfreien Knoten erstellt. Je nach Kartengröße werden z. B. 200–300 Knoten gewählt und miteinander verbunden, wenn kein Hindernis dazwischenliegt. Die Distanzen zwischen verbundenen Knoten werden berechnet.

Der Weg wird in drei Schritten berechnet:

- Direkter Pfad vom Start zum nächsten Knoten
- Kürzester Weg im Netzwerk bis zum Zielknoten
- Direkter Pfad vom Zielknoten zur Zielposition

Vorteile:

- Deutlich weniger Rechenaufwand
- Netzkarte muss nur einmal erzeugt werden
- Wegsuche danach sehr schnell

Nachteile:

- Wege meist weniger optimal als bei der Distanztransformation
- Roboterkinematik wird nicht berücksichtigt (freie Beweglichkeit vorausgesetzt)

Rapidly-exploring Random Tree (RRT)

Die RRT-Methode ähnelt PRM, berücksichtigt aber die Kinematik des Roboters: Bewegungen erfolgen nur entlang von Kreissegmenten.
Knoten enthalten Position und Orientierung und werden nur verbunden, wenn eine kreisförmige Verbindung möglich ist.
So entsteht eine baumartige Netzstruktur.
Die Verbindung vom Start zum Ziel wird wie bei PRM im Netz gesucht, kann je nach Knotenverteilung und Auflösung stark variieren.

Vorteile:

- Geringer Rechenaufwand nach einmaliger Kartenerstellung
- Gut geeignet für statische Umgebungen
- Einfach erweiterbar auf große Karten
- Beachtet nicht Omni-direktionale Kinematiken

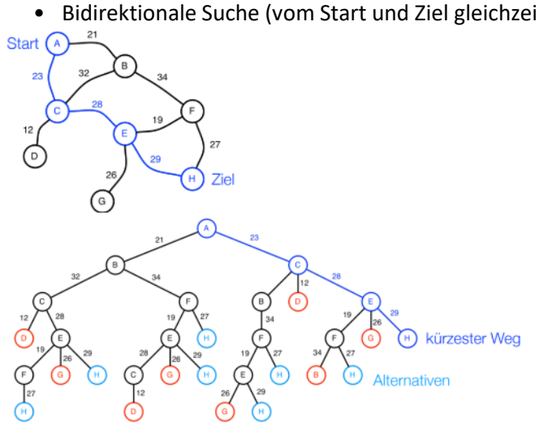
Nachteile:

- Wege meist nicht optimal
- Funktioniert schlecht in engen oder stark strukturierten Umgebungen

Mit Topologischen Karten

Navigation mit topologischen Karten nutzt Knoten und deren Verbindungen zur Wegplanung. Die Verbindungen können gewichtet sein (z. B. Distanz, Zeit, Energie). Ziel ist es, den besten Weg vom Start- zum Zielknoten zu finden.
Zur Wegsuche wird ein Graph aufgebaut, der von einem Startknoten aus Äste zu allen erreichbaren Knoten bildet. Äste, die in Sackgassen enden, werden verworfen. Durch Bewertung aller möglichen Wege (z. B. per Distanzen) kann die optimale Route bestimmt werden.
Bei großen Karten wird das rechenintensiv – deshalb gibt es Optimierungen wie:

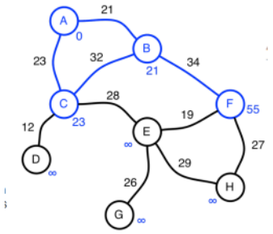
- Entfernen von Sackgassen
- Tiefensuche statt Breitensuche
- Bidirektionale Suche (vom Start und Ziel gleichzeitig)



Dijkstra-Algorithmus

Berechnet den **kürzesten Weg** im Graphen vom Start- zum Zielknoten.
Ablauf:

- Startknoten = Distanz 0, alle anderen = ∞
- Wiederhole: Wähle Knoten mit kleinster Distanz
- Aktualisiere Distanzen zu Nachbarn
- Beende, wenn Ziel erreicht ist



Koordination mehrerer Roboter

Um Blockaden zu vermeiden, gibt es drei Ansätze:

- **Dynamische Hindernisse**: Roboter weichen einander aus, z. **B. mit Verkehrsregeln.**
- **Mutual Exclusion**: Umgebung in Blöcke unterteilt, pro Block nur ein Roboter erlaubt.
- **Zentrale Steuerung**: Eine zentrale Instanz koordiniert alle Roboterwege.

1. Definition am rechtwinkligen Dreieck

Für einen Winkel θ in einem rechtwinkligen Dreieck gilt:

- **Sinus:**

$$\sin(\theta) = \frac{\text{Gegenkathete}}{\text{Hypotenuse}}$$

- **Kosinus:**

$$\cos(\theta) = \frac{\text{Ankathete}}{\text{Hypotenuse}}$$

- **Tangens:**

$$\tan(\theta) = \frac{\text{Gegenkathete}}{\text{Ankathete}} = \frac{\sin(\theta)}{\cos(\theta)}$$

2. Trigonometrische Beziehungen im Einheitskreis

Im Einheitskreis ($r = 1$):

- Punkt auf Kreis bei Winkel θ:

$$(\cos(\theta), \sin(\theta))$$

- Wichtig:

$$\sin^2(\theta) + \cos^2(\theta) = 1$$

- Weitere Identitäten:

$$1 + \tan^2(\theta) = \frac{1}{\cos^2(\theta)}$$

3. Wichtige Winkelsymmetrien

- $\sin(-\theta) = -\sin(\theta)$ (ungerade)
- $\cos(-\theta) = \cos(\theta)$ (gerade)
- $\tan(-\theta) = -\tan(\theta)$
- $\sin(\theta + \pi) = -\sin(\theta)$
- $\cos(\theta + \pi) = -\cos(\theta)$

5. Additionstheoreme

- $\sin(a \pm b) = \sin a \cos b \pm \cos a \sin b$
- $\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$
- $\tan(a \pm b) = \frac{\tan a \pm \tan b}{1 \mp \tan a \tan b}$

7. Umformungen / Identitäten

- $\cos^2\theta = 1 - \sin^2\theta$
- $\sin(2\theta) = 2\sin(\theta)\cos(\theta)$
- $\cos(2\theta) = \cos^2(\theta) - \sin^2(\theta)$