

Robotik

Priv.-Doz. Dr. Thomas Wiemann
Institut für Informatik
Autonome Robotik

SoSe 2021



Inhalt



Gliederung

1. Einleitung
2. Robot Operating System
3. Sensorik
4. Sensordatenverarbeitung
5. Fortbewegung
6. Lokalisierung
7. Mapping
8. Navigation
9. Ausblick

Letzte Vorlesung

- ▶ Was sind Transformationen
- ▶ Was ist ROS
- ▶ ROS-Infrastruktur
- ▶ ROS-Nachrichten

Type	Strengths	Weaknesses
Message	<ul style="list-style-type: none"> •Good for most sensors (streaming data) 	<ul style="list-style-type: none"> •Messages can be <u>dropped</u> without knowledge •Easy to overload system with too many messages
Service	<ul style="list-style-type: none"> •Knowledge of missed call •Well-defined feedback 	<ul style="list-style-type: none"> •Blocks until completion •Connection typically re-established for each service call (slows activity)
Action	<ul style="list-style-type: none"> •Monitor long-running processes •Handshaking (knowledge of missed connection) 	<ul style="list-style-type: none"> •Complicated •Mixed feedback from multiple action servers (not for SimpleAction)

Inhalt

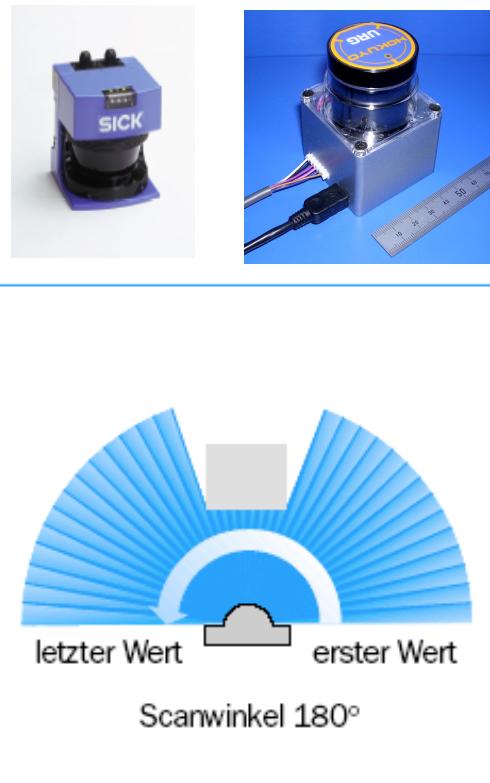


Gliederung

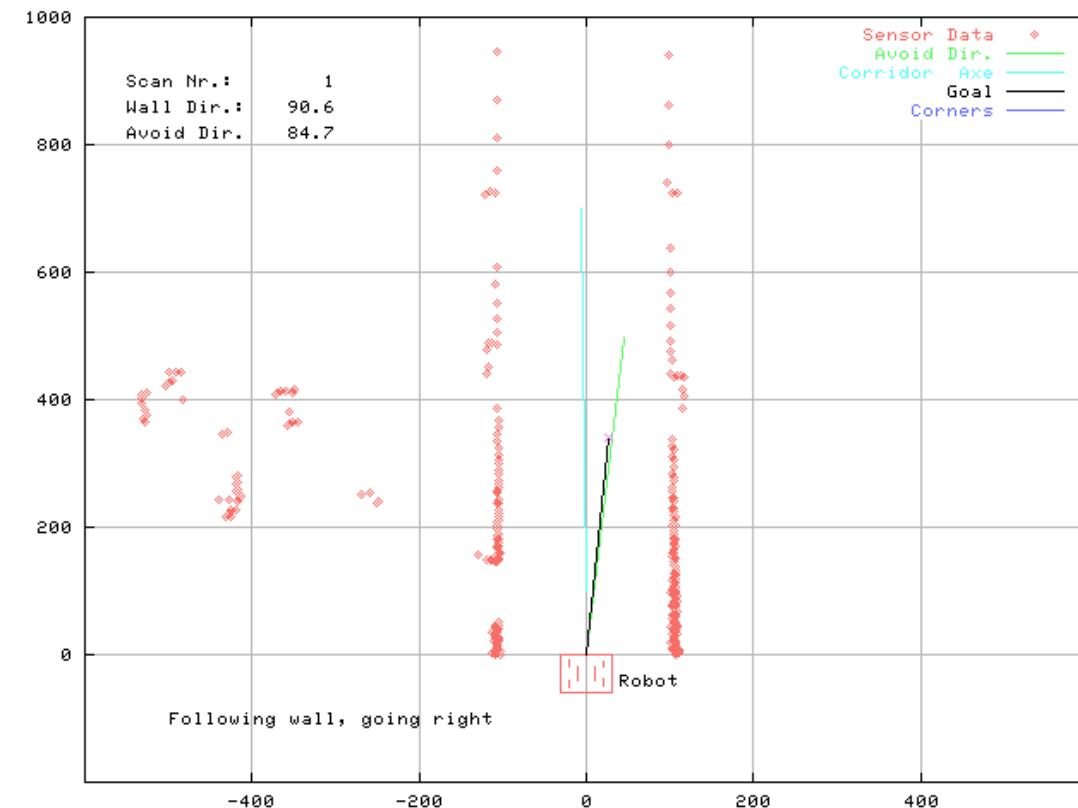
- 2. Robot Operating System
 - 1. Konventionen / Mathe
 - 2. Grundlagen
 - 3. Nachrichten
 - 4. Transformationen**
 - 5. Robotermodellierung**
 - 6. Navigation
 - 7. Ausblick

ROS - Transformationen (1)

- ▶ Jeder Sensor liefert seine Daten in einem eigenen lokalen Referenzsystem
- ▶ Typischerweise ist der Ursprung in der Mitte des Sensors



Beispiel SICK LMS200



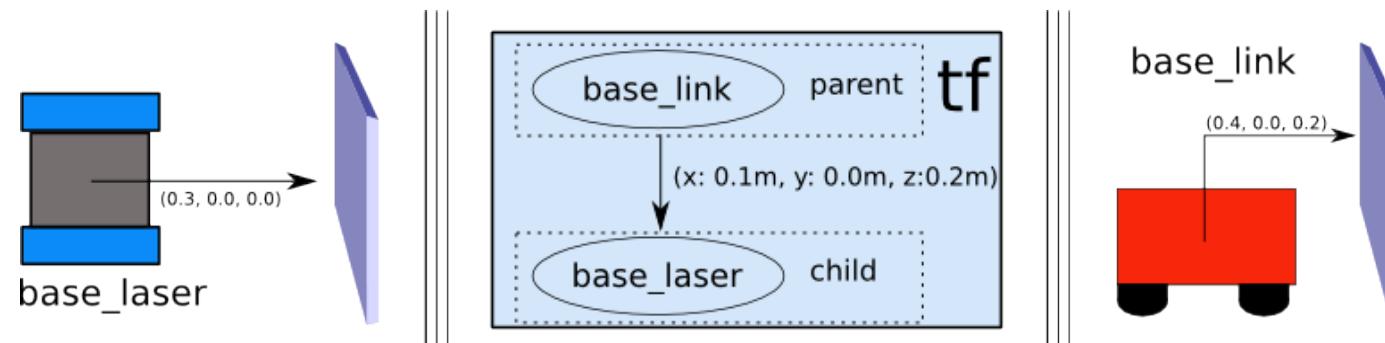
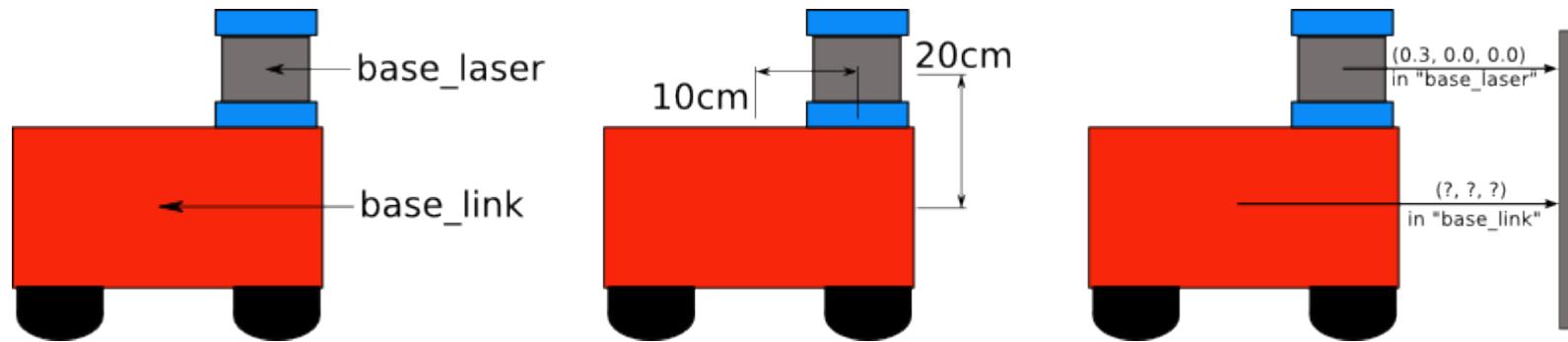
ROS - Transformationen (2)

- ▶ In der Regel sind die Sensoren nicht in der Mitte des Roboters montiert
- ▶ Oftmals braucht man für verschiedene Aufgaben unterschiedliche Referenzsysteme



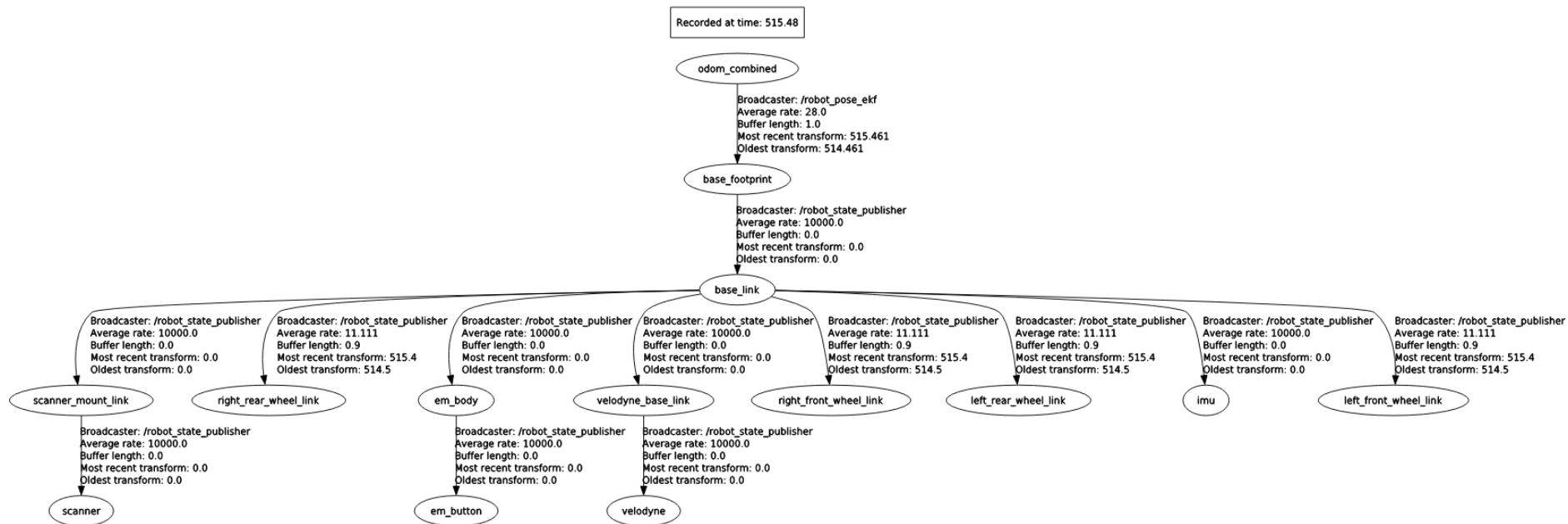
ROS Transformationen (3)

- ▶ ROS stellt via *tf* solche Transformationen zur Verfügung



ROS - Transformationen (4)

- ▶ Via tf lassen sich die Daten von einzelnen Komponenten automatisch Transformieren
- ▶ Es wird ein Transformationsbaum aufgespannt
- ▶ Es muss also immer ein Root-Koordinatensystem geben



ROS - Transformationen (5)



Inhalt

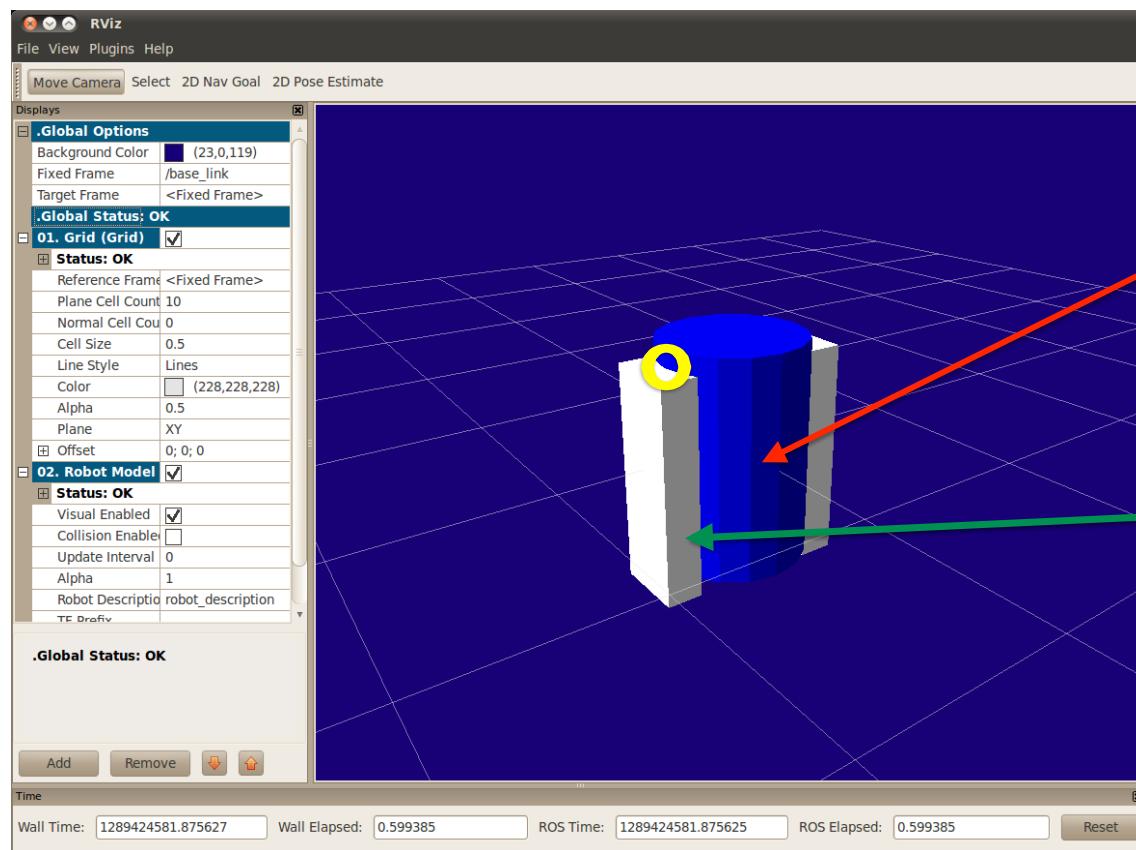


Gliederung

- 2. Robot Operating System
 - 1. Konventionen / Mathe
 - 2. Grundlagen
 - 3. Nachrichten
 - 4. Transformationen
 - 65. Robotermodellierung**
 - 76. Tools
 - 8. Ausblick

Robotmodellierung (1)

- ROS stellt mit URDF (Unified Robot Description Format) eine Metasprache zur Modellierung von Robotern bereit



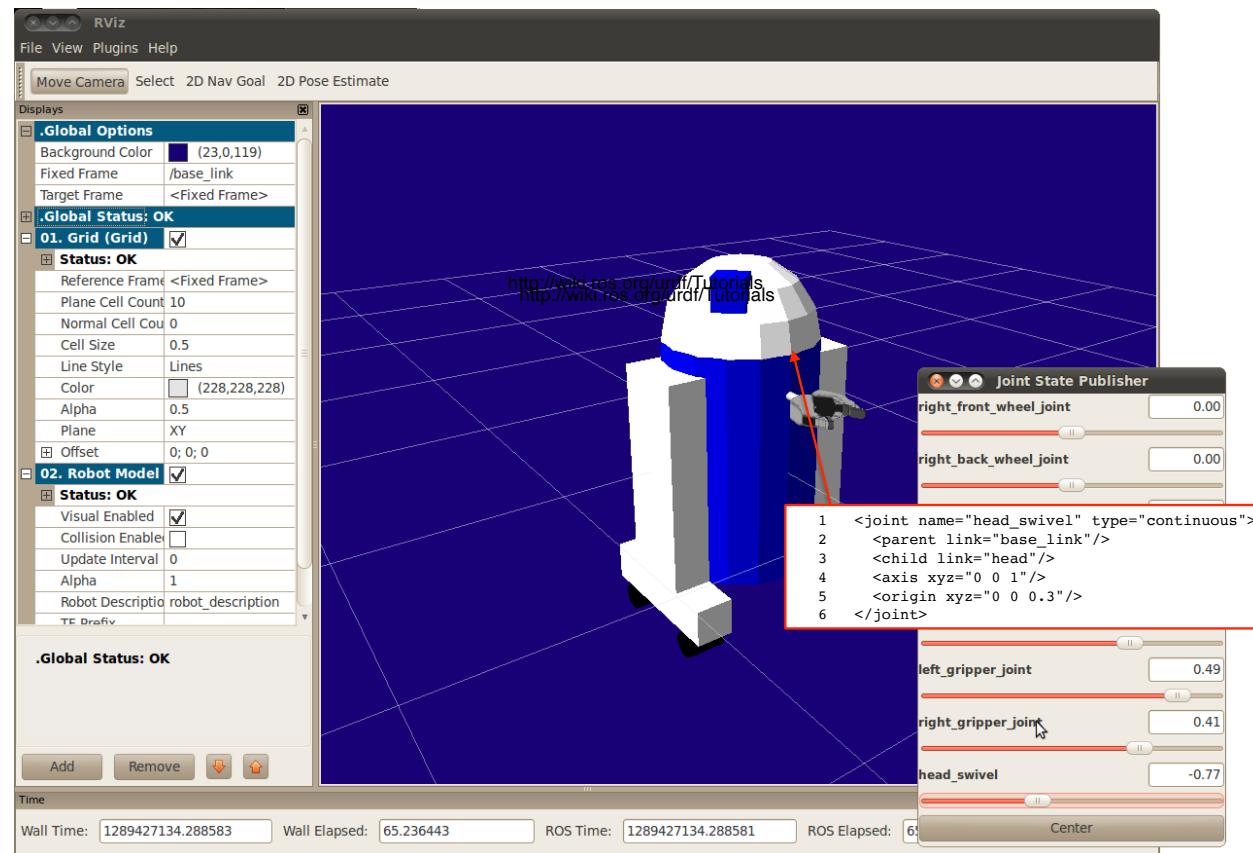
```

<?xml version="1.0"?>
 2 <robot name="materials">
 3   <link name="base_link">
 4     <visual>
 5       <geometry>
 6         <cylinder length="0.6" radius="0.2"/>
 7       </geometry>
 8       <material name="blue">
 9         <color rgba="0 0 .8 1"/>
10       </material>
11     </visual>
12   </link>
13
14   <link name="right_leg">
15     <visual>
16       <geometry>
17         <box size="0.6 .2 .1"/>
18       </geometry>
19       <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
20       <material name="white">
21         <color rgba="1 1 1 1"/>
22       </material>
23     </visual>
24   </link>
25
26   <joint name="base_to_right_leg" type="fixed">
27     <parent link="base_link"/>
28     <child link="right_leg"/>
29     <origin xyz="0.22 0 .25"/>
30   </joint>
31...

```

Robotmodellierung (2)

- In URDF lassen sich verschiedene Arten von Gelenkverbindungen modellieren: Continuous, Revolute, Prismatic



ROS - Robot State Publisher (1)

- Mittels URDF und `robot_state_publisher` kann der Zustand des Roboters aktualisiert werden

```
#include <string>
 2 #include <ros/ros.h>
 3 #include <sensor_msgs/JointState.h>
 4 #include <tf/transform_broadcaster.h>
 5
 6 int main(int argc, char** argv) {
 7     ros::init(argc, argv, "state_publisher");
 8     ros::NodeHandle n;
 9     ros::Publisher joint_pub = n.advertise<sensor_msgs::JointState>("joint_states", 1);
10     tf::TransformBroadcaster broadcaster;
11     ros::Rate loop_rate(30);
12
13     const double degree = M_PI/180;
14
15     // robot state
16     double tilt = 0, tinc = degree, swivel=0, angle=0, height=0, hinc=0.005;
17
18     // message declarations
19     geometry_msgs::TransformStamped odom_trans;
20     sensor_msgs::JointState joint_state;
21     odom_trans.header.frame_id = "odom";
22     odom_trans.child_frame_id = "axis";
```

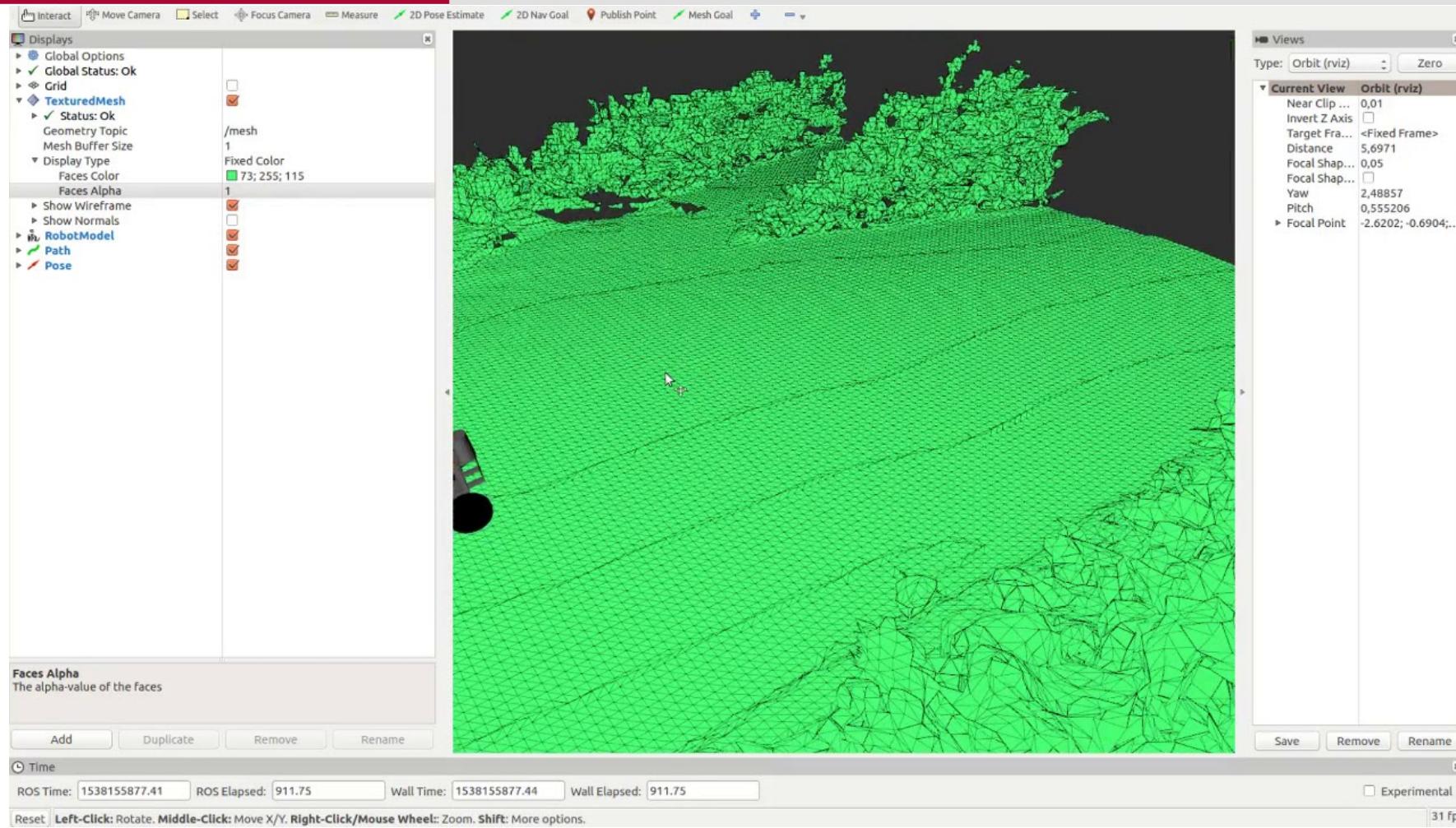
ROS - Robot State Publisher (2)

```
while (ros::ok()) {
 25      //update joint_state
 26      joint_state.header.stamp = ros::Time::now();
 27      joint_state.name.resize(3);
 28      joint_state.position.resize(3);
 29      joint_state.name[0] = "swivel";
 30      joint_state.position[0] = swivel;
 31      joint_state.name[1] = "tilt";
 32      joint_state.position[1] = tilt;
 33      joint_state.name[2] = "periscope";
 34      joint_state.position[2] = height;
 35
 36
 37      // update transform
 38      // (moving in a circle with radius=2)
 39      odom_trans.header.stamp = ros::Time::now();
 40      odom_trans.transform.translation.x = cos(angle)*2;
 41      odom_trans.transform.translation.y = sin(angle)*2;
 42      odom_trans.transform.translation.z = .7;
 43      odom_trans.transform.rotation = tf::createQuaternionMsgFromYaw(angle+M_PI/2);
 44
 45      //send the joint state and transform
 46      joint_pub.publish(joint_state);
 47      broadcaster.sendTransform(odom_trans);
```

ROS - Robot State Publisher (3)

```
49      // Create new robot state
50      tilt += tinc;
51      if (tilt<-.5 || tilt>0) tinc *= -1;
52      height += hinc;
53      if (height>.2 || height<0) hinc *= -1;
54      swivel += degree;
55      angle += degree/4;
56
57      // This will adjust as needed per iteration
58      loop_rate.sleep();
59  }
60
61
62  return 0;
63 }
```

ROS Tools - RViz



<https://youtu.be/ir4kZif5FS8>

Inhalt



Gliederung

1. Einleitung
2. Robot Operating System
- 3. Sensorik**
4. Sensordatenverarbeitung
5. Fortbewegung
6. Lokalisierung
7. Mapping
8. Navigation
9. Ausblick

Inhalt



Gliederung

- 3. Sensorik**
- 1. Allgemeines
- 2. Bewegungsmessung
- 3. Ausrichtungsmessung
- 4. Globale Positionsbestimmung
- 5. Entfernungsmessung
- 6. Kamerarund Kameramodelle
- 7. Farbsensoren
- 8. Ausblick

Inhalt



Gliederung

- 3. Sensorik
 - 1. Allgemeines
 - 2. Bewegungsmessung
 - 3. Ausrichtungsmessung
 - 4. Globale Positionsbestimmung
 - 5. Entfernungsmessung
 - 6. Kamerarund Kameramodelle
 - 7. Ausblick

Grobklassifikation

► Propriozeptiv vs. exterozeptiv („nach innen“ – „nach außen“)

- Propriozeptiv: Misst Aspekte des Roboterzustands
z.B. Batterieladung, Gelenkwinkel, Radumdrehungen
- Exterozeptiv: Miss Aspekte der Umgebung oder des Verhältnisses Roboter/Umgebung
z.B. Außentemperatur, Fahrgeschwindigkeit, Wandabstand
- keine scharfe Unterscheidung
z.B. Radumdrehung / $s \times$ Radumfang \sim Fahrgeschwindigkeit

► Aktiv vs. Passiv

- Aktiv: Miss Umgebungsantwort auf gesendete „Energie“
z.B. Laserscanner, Ultraschallsensoren
- Passiv: Miss Signale/Werte, die „von allein“ kommen
z.B. Kamera, Mikrofon
- praktisch ebenfalls unscharfe Unterscheidung
z.B. „Anregung“ der Umgebung durch Blitzlicht

Sensorklassen (1)

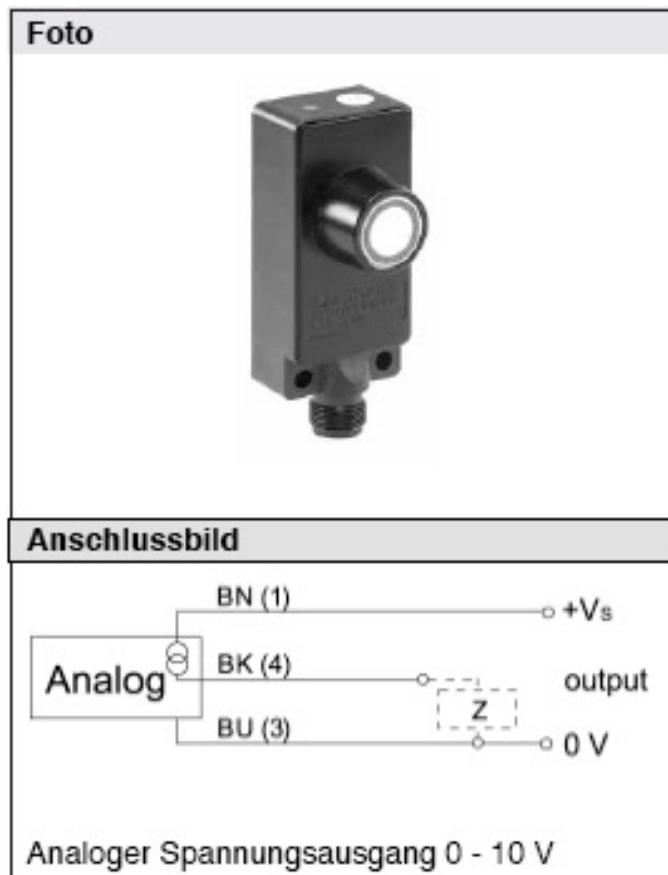
Sensorklasse	Beispiele	PC / EC	Aktiv / Passiv
<u>Kontaktsensoren</u> Detektion von physischen Kontakt oder Nähe, Sicherheitsschalter	Schalter, Bumper Lichtschranken Entfernungssensoren	EC EC EC	P A A
<u>Radsensoren / Motorsensoren</u> Messung der Drehgeschwindigkeit der Räder oder des Motors	Bürstengeber ("Brush Encoder") Potentiometer Synchros / Drehmelder Optische Encoder Magnetische Encoder Induktionsencoder Kapazitive Encoder	PC PC PC PC PC PC PC	P P A A A A A
<u>Lagesensoren</u> Orientierung des Roboters relativ zu einem festen Referenzsystem	Kompass Gyroscope Inklinometer	EC PC EC	P P A / P

Sensorklassen (2)

Sensorklasse	Beispiele	PC / EC	Aktiv / Passiv
<u>Beacons</u> Lokalisierung in einem Bezugssystem anhand von künstlichen Landmarken	GPS Lichtquellen oder Funkstationen Ultraschall-Beacons Reflektoren	EC EC EC EC	A A A A
<u>Aktive Entfernungsmessung</u> Reflektivität, Time-of-Flight, Triangulation	Infrarotsensoren Ultraschallsensoren LiDAR Optische Triangulation Strukturiertes Licht	EC EC EC EC EC	A A A A A
<u>Bewegung und Geschwindigkeit</u> Geschwindigkeit relativ zu einem festen oder sich bewegenden Objekt	Dopplerradar Doppler Sound	EC EC	A A
<u>Bildgebende Sensoren</u> Visual Ranging, Bildanalyse, Segmentierung, Objekterkennung	CCD / CMOS Kameras “Blackbox” Systeme	EC	P

Beispiel: Ultraschallsensor

► Datenblatt Baumer UNDK 30U6103:



Technische Daten	
Erfassungsbereich Sd	100 - 1000 mm
Auflösung	0,3 mm
Ausgangsschaltung	Analog Spannungsausgang
Bauform	quaderförmig
Breite / Durchmesser	30 mm
Höhe	65 mm
Tiefe	31 mm
Gehäusematerial	Polyester / Zink Druckguss
Anschlussart	Stecker
Erfassungsbereich Startwert Sdc (Teach-in)	100 - 1000 mm
Erfassungsbereich Endwert Sde (Teach-in)	100 - 1000 mm
Öffnungswinkel typ.	10 °
Schallfrequenz	240 kHz
Betriebsspannungsbereich +Vs	15 - 30 VDC
Restwelligkeit	< 10 % Vs
Stromaufnahme	< 35 mA
max. Laststrom	< 20 mA
Ausgang	0 - 10 V/ 10 - 0 V
Lastwiderstand +Vs>15V	-
Lastwiderstand +Vs>24V	-
kurzschlussfest	ja
verpolungsfest	ja
Reaktionszeit ton / toff	< 80 ms
Linearität L	-
Wiederholgenauigkeit	< 0,5 mm
Temperaturdrift	< 2% von Objektdistanz So
Arbeitstemperaturbereich	-10 ... +60 °C
Schutzklasse	IP 67
Empfangsanzeige	LED gelb/ LED rot
Einstellung	Teach-in
Einstellhilfe	Objektanzeige blinkt
Anschlussoption	-
mögl. Kabeldosen	ESW 33, ESG 34

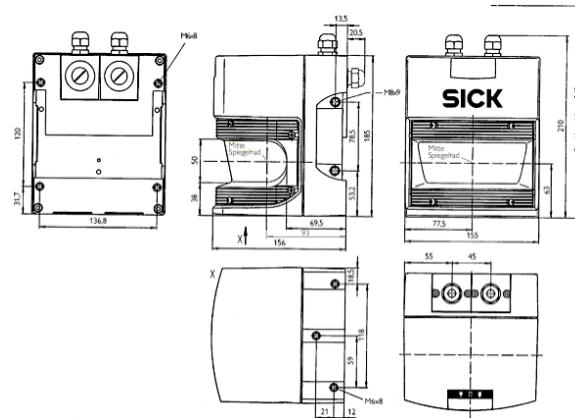
Beispiel: Laserscanner

► SICK LMS 200

Typ: LMS 200-30106
Bestell-Nr.: 1015850

Technische Daten

Öffnungswinkel:	180 °	Reichweite:	80 m
Winkelauflösung:	1 ... 0,25 °	Datenschnittstelle:	RS-232, RS-422
Ansprechzeit:	13 ... 53 ms	Datenübertragungsrate:	9,6 / 19,2 / 38,4 / 500 kBaud
Auflösung:	10 mm	Schaltausgänge:	3 x PNP
Systematischer Fehler:	+/- 15 mm	Versorgungsspannung:	24 V DC +/- 15%
Statistischer Fehler (1 Sigma):	5 mm	Leistungsaufnahme:	20 W
Laserklasse:	1	Lagertemperatur:	-30 °C ... +70 °C
Schutzart:	IP 65	Gewicht:	4,5 kg
Betriebstemperatur:	0 °C ... +50 °C	Abmessungen (L x B x H):	156 x 155 x 210 mm



Sensoreigenschaften (1)

► **Messbereich** (“*Range*”)

- Untere und obere Grenze valider Messwerte
- Beispiel: Entfernungsmessung
 - Baumer US: 100-1000mm;
 - SICK 5cm-80m

► **Dynamik** (“*Dynamic Range*”)

- Verhältnis von Ober- zu Untergrenze der Messwerte
- Beispiele:
 - Baumer US: 10;
 - SICK 1600

Dynamik in *logarithmischer Form in dB (relativ)*:

$$10 \log_{10} \left(\frac{\text{Obergrenze}}{\text{Untergrenze}} \right) = \text{Dynamik} [\text{dB}]$$

Beispiel: Baumer 10, Sick ~32

Sensoreigenschaften (2)

► Auflösung (“*Resolution*”)

- Eigentlich: Minimalunterscheid zweier Messwerte
 - Baumer US: 0,3 mm
 - Sick: 10 mm
 - Webcams: 640 x 480 ... 1920 x 1080 Pixel
- Manchmal: Untergrenze des Messbereichs
- Manchmal: Diskretisierungsfehler bei A/D Wandlung
 - z.B. Spannungsmesser 0 - 5V bei 8 Bit Auflösung $\Rightarrow 5V / 255 = 19,6 \text{ mV}$

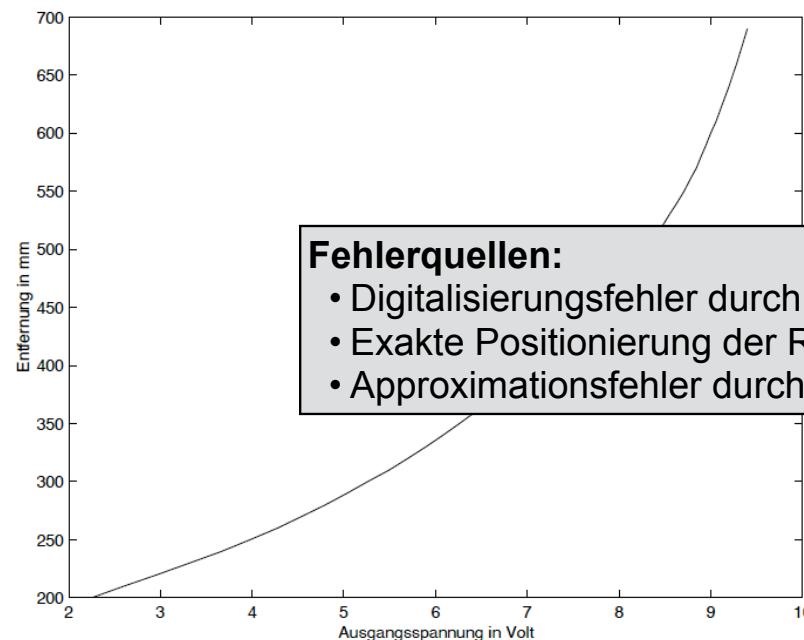
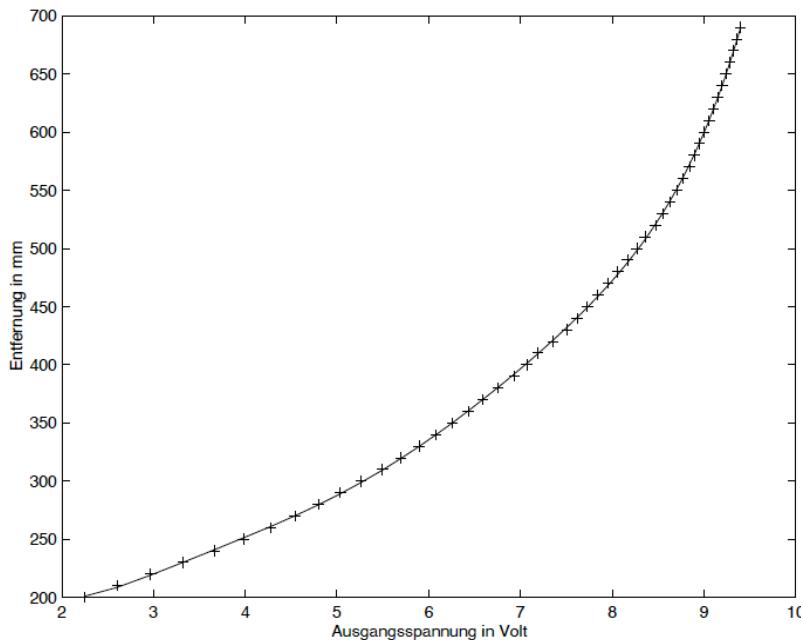
► Linearität (“*Linearity*”)

- Abhängigkeit von tatsächlicher Messgröße und anliegendem Messwert
- Bei vorhandenen Messwerten meist “mit gekauft”
 - z.B: Abstandssensoren

Aber: Je nach Messprinzip nichtlineare Überlagerungen durch systematische Fehler!

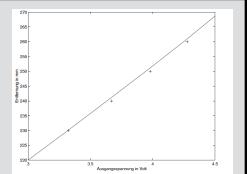
Sensoreigenschaften (3)

► Beispiel: Analoger Infrarot-Entfernungssensor



Fehlerquellen:

- Digitalisierungsfehler durch D/A Wandler
- Exakte Positionierung der Referenz
- Approximationsfehler durch Polynom



- Kalibrierung: Messung gegen eine Referenzplatte, Aufnahme der Kurve
- Anfitten eines Polygons Grad n (hier: 6)
 - Funktionale Approximation des Datenverlaufs
 - Eingabe Spannung, Ergebnis: Entfernung

Sensoreigenschaften (4)

- ▶ **Frequenz / Zyklus- / Ansprechzeit** (“Frequency” / “Cycle Time”)
 - Zahl valider Messungen pro Zeiteinheit/Sekunde [Hz]
 - In der Regel abhängig von Messprinzip, ggf. von Datenraten
 - Baumer US ~13 Hz (80ms);
 - SICK ~77–19 Hz (13–53ms, abhängig von Winkelauflösung & Übertragungsprotokoll)
 - Webcam 25 Hz (fps)
- ▶ **Weitere Charakteristika:**
 - Baugröße
 - Gewicht
 - Spannungsversorgung
 - Leistungsaufnahme
 - Kosten
 - Verarbeitungsaufwand
 - ...

Sensorfehler (1)

► **Empfindlichkeit** (“Sensitivity”, ~Auflösung)

- Änderung des angezeigten Messwerts in Abhängigkeit der tatsächlichen Parameteränderung
- Ist generell gewünscht
- Geht meist mit Störanfälligkeit einher...

► **Störanfälligkeit** (“Fault Liability”)

- Änderung des angezeigten Messwerts durch (nichtkontrollierbare) Umwelteinflüsse
 - z.B. Elektromagnetischer (Flux-Gate) Kompass:
 - Empfindlichkeit / Auflösung $0,1^\circ$ bis $0,5^\circ$ erzielbar (empfindlich)
 - Messfehler durch Umgebung (Stahlkonstruktionen, Motorströme)
 - Praktisch unbegrenzt störanfällig
 - z.B. Laserscanner
 - Störunanfällig, da Messung durch monochromatisches (Infrarot-)Licht
 - Aber Vorsicht bei hartem Sonnenlicht im Freien und spiegelnden Flächen

Sensorfehler (2)

► Messfehler / Genauigkeit (“Error / Accuracy”)

v : tatsächlicher Wert
 m : gemessener Wert

Messfehler

$$\text{Genauigkeit} := 1 - \frac{|m - v|}{v}$$

► Systematische Messfehler (“Systematic Errors”)

- Deterministische Messfehler, d.h. im Prinzip modellierbar und daher behandelbar
 - Temperaturdrift in Ultraschallsensoren (s. Baumer)
 - Linsenverzerrung (Kalibrierung)
 - SICK-Datenblatt: “Systematischer Fehler +/- 15 mm”

► Zufällige Messfehler (“Random Errors”), Reproduzierbarkeit (“Precision”)

- Rauschen tritt in allen technischen Sensoren auf!
- Modelliert durch Gaußverteilung (μ, σ)
 - SICK-Datenblatt: “Statistischer Fehler σ : 5mm”

$$\text{Reproduzierbarkeit} := \frac{\text{Messbereich}}{\sigma}$$

Leben mit Sensorfehlern

► Aus Sicht des Roboters:

- Systematische und zufällige Fehler oft ununterscheidbar
 - ➡ arbeite mit 1 Fehlermodell!
- Dieses Fehlermodell ist i.d.R. monomodal und symmetrisch (~Gaußverteilung, genauer: Normalverteilung um v);
- Die tatsächlichen Fehler sind multimodal und asymmetrisch (Bewegung vs. Stillstand, Reflexionen, Übersprechen, ...)
 - ➡ Sensorfehler sind praktisch unvermeidbar!

Reduziere Sensorfehler so gut wie technisch möglich!

**Die Roboterprogrammierung muss aber dennoch davon ausgehen,
dass die Sensordaten fehlerbehaftet sind!**

Inhalt



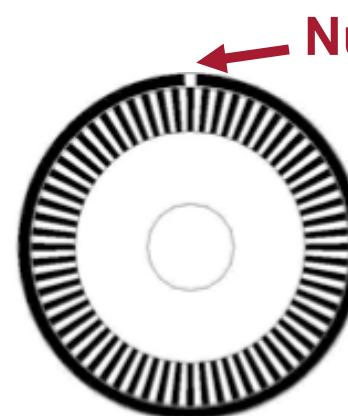
Gliederung

- 3. Sensorik
 - 1. Allgemeines
 - 2. Bewegungsmessung**
 - 3. Ausrichtungsmessung
 - 4. Globale Positionsbestimmung
 - 5. Entfernungsmessung
 - 6. Kamerarund Kameramodelle
 - 7. Ausblick
- 8. Ausblick

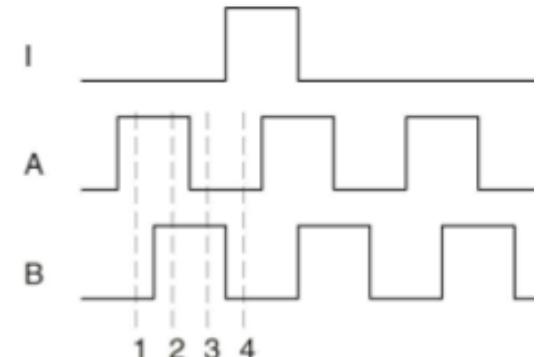
Bewegungsmessung

- ▶ Grundidee: Mit den Achsen rotieren Messscheiben, die optisch ausgelesen werden.
- ▶ Typische Auflösung (“Counts per Revolution”, CPR):
 - ▶ Mobile Robotik ~2000 CPR
 - ▶ Industrieroboter ~10.000 CPR (counts per revolution).
- ▶ Bei bekannter Übersetzung: Anzahl Umdrehungen x Radumfang = Wegstrecke
- ▶ **Odometrie**

Viele Ablesepunkte
phasenversetzt
(höhere Auflösung,
Drehrichtung)



Nulldurchgang



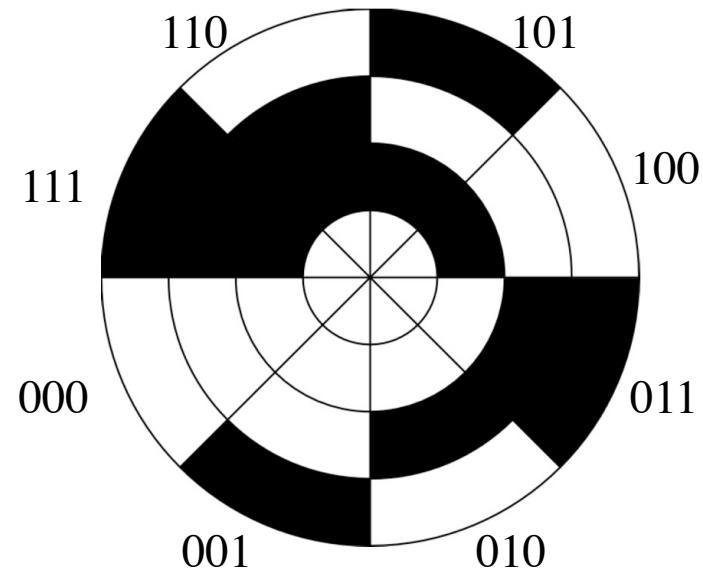
Inkrementalgeber

Nachteil: Absolute Gelenkstellung nicht messbar.

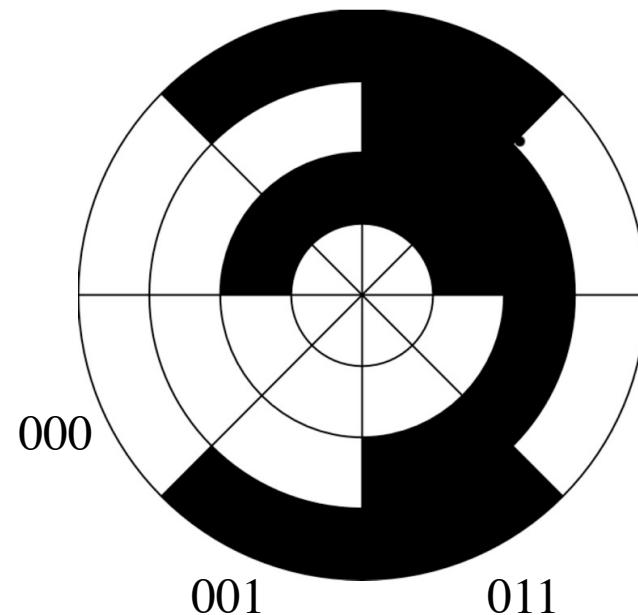
- ▶ Beim Einschalten erst „Räkeln“ bis zu einer Absolut-Marke

Absolutgeber

- ▶ Erlauben direktes Ablesen des Gelenkwinkels (auch bei Stillstand)



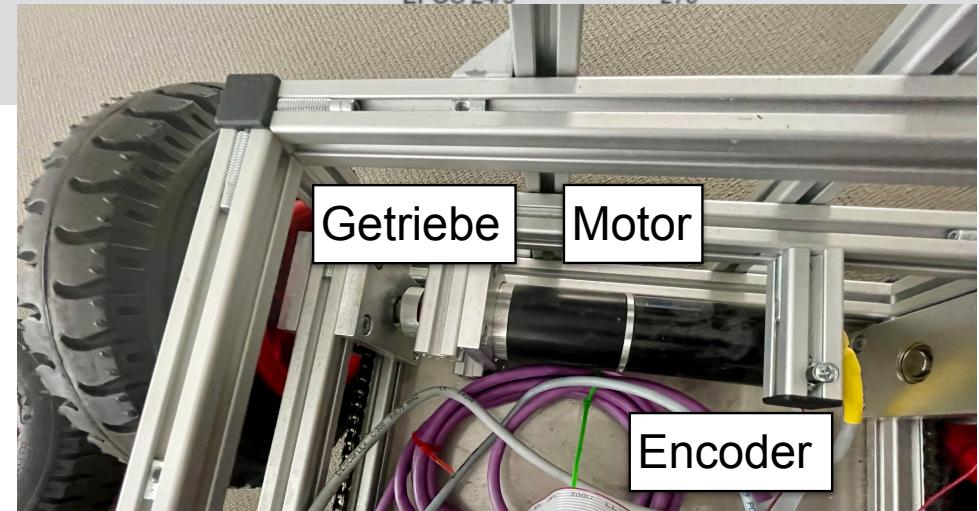
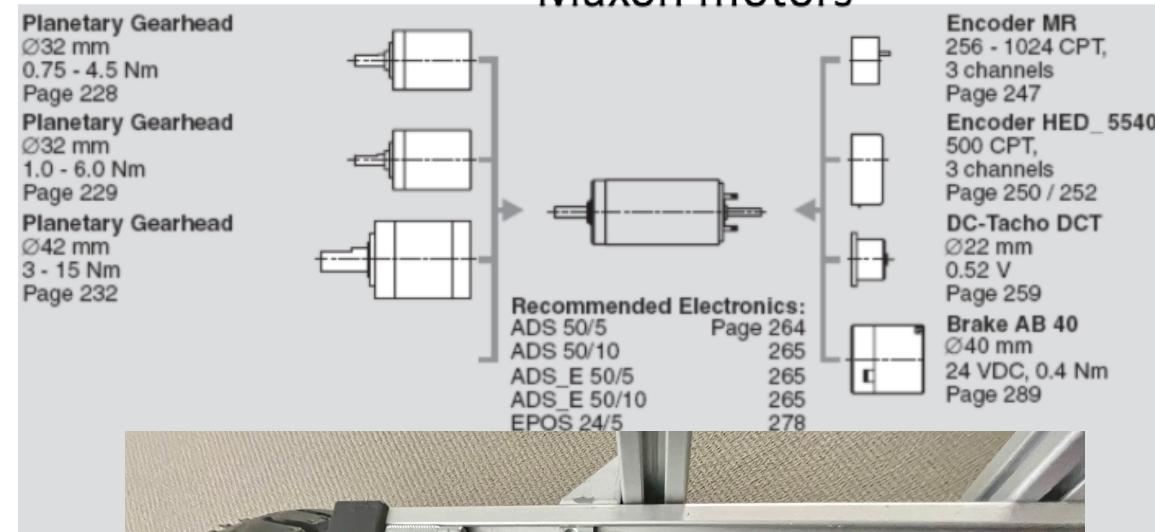
Binärer Absolutgeber (3 Bit)



Graycode-Absolutgeber (3 Bit)
(Robuster gegen Ablesefehler beim
Sektorübergang)

Encoder beim Volksbot

- ▶ Inkrementalgeber sind direkt am Motor angebracht
- ▶ Vorteile:
 - Einfach und robust
 - Eine Baueinheit
 - Günstig
- ▶ Nachteile:
 - Getriebe zwischen Motor und Rädern erzeugen Ungenauigkeiten
 - Getriebetoleranzen
- ▶ CPR: 5.000 bis 20.000
- ▶ Wie kann man die CPR-Anzahl bestimmen?
 - Empirisch!



- ▶ Für **Gleichstrommotoren**:
 - hohe Spannung am Motor ➔ schnell
 - niedrige Spannung ➔ langsam
 - Wechsel \oplus, \ominus ➔ Drehrichtungswechsel
 - Verbinde \oplus, \ominus (Motor wird Generator) ➔ Bremsen

- ▶ Wie reguliert man Motorspannung?
 - über Widerstand ➔ Verlustleistung (Wärme)

- ▶ **PWM (Pulsweitenmodulation)**
 - schalte Spannung periodisch voll und null
 - dadurch Regelung des Mittelwerts/Integrals pro T
 - feste Auflösung über festem Zeitintervall
 - KURT-Roboter: 0 – 1024, 0 entspricht Vollgas

