

# Intelligente Systeme

Masterprojekt: Robot Navigation in a Crowd

Philipp Schulz, Nihal Nallari, Jan Paul Wessendorf, Qiwei Jiang, Utku Karadeniz

Betreut von Martin Moder

Ziel:

Roboter soll sich in Menschenmenge kollisionsfrei bewegen können



[https://github.com/srl-freiburg/pedsim\\_ros](https://github.com/srl-freiburg/pedsim_ros)

Ziel:

Roboter soll sich in Menschenmenge kollisionsfrei bewegen können

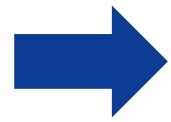
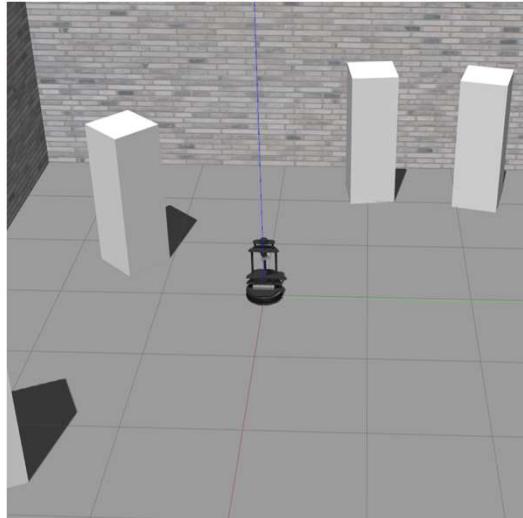


Roboter: Locobot mit ROS

- Lidar: erkennt Hindernisse
- Kamera: erkennt Personen
  - Prediction Model für Personenbewegungen
- DWA: berechnet Wege

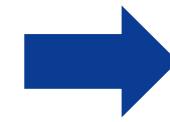
# Vorgehen

Simulation  
aufbauen



Daten  
sammeln

```
Start_pos: [26.67186895310854, 26.97870277638416] Goal_pos: [0 0]
59614983244133, 25.3010550833549276], [22.884399771552765, 25.24
716078, 19.15573887399417], [20.250185405900563, 19.3112319266;
547, 14.449456501686727], [16.26577205332588, 14.2554405823061;
68781623], [10.98688575944926, 8.940342882350329], [10.76700963;
147201635466], [3.5053612941244334, 7.354413925308031], [3.2307;
1.4800524188640567, 1.6712578820081847], [0.3807651194467038
Start_pos: [0.04674959862981569, 0.33061933646842795] Goal_pos: [
94, 2.7671212523502877], [1.7280444295961936, 3.03270051356627;
67480152775969, 5.733087741268227], [8.052383780738385, 5.9612;
022466], [12.895511933358508, 11.159647066132004], [12.96610814;
395], [15.267144749094019, 17.261535791860396], [15.48249851697;
492296905, 22.425714335529936], [19.371421555051725, 22.689840
26.52351806284505], [25.54387218656172, 26.61017541062822], [2;
Start_pos: [26.665444499290924, 26.92859552104784] Goal_pos: [0 0];
77], [25.267633858047148, 21.702574556779165], [25.128636676656;
0497907743, 17.30782866675052], [20.086305854549778, 17.512986(
14.225293534689445], [14.256763077046513, 14.259093177163676],
19071529], [10.900916338124498, 8.843528803756936], [10.6761429-
65596078032], [3.7788574012932097, 7.587088615168983], [3.49059-
3577704849, 1.5167397298956293], [0.7613915058570097, 1.187696-
8244, 1.7713538934314723], [3.105259146175492, 1.8063586947582;
90099756, 1.0155067795211663], [-0.06081491123387798, 0.7117594
Start_pos: [-0.04887092164125904, 0.3377781964586669] Goal_pos: [
.7104881709780548], [5.160371879140614, 1.5862010374138946], [5.
```

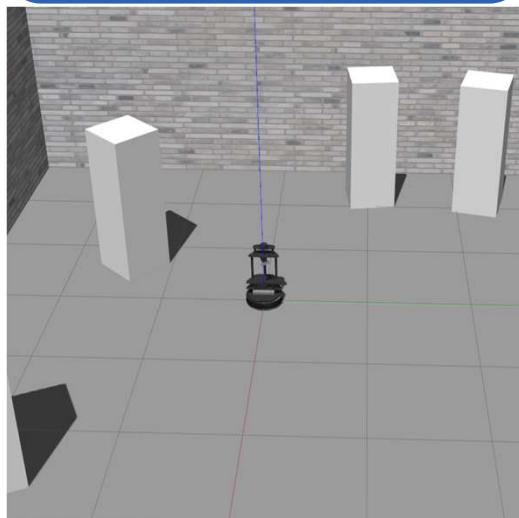


Echte  
Umgebung



# Vorgehen

## Simulation aufbauen



- Locobot in Gazebo simulieren
- Pedestrian Simulator mit Gazebo Plugin
- DWA zur Steuerung (+ Prediction Model)

# Vorgehen

## Daten sammeln

```
Start_pos: [26.67186895310854, 26.97870277638416] Goal_pos: [0 0]
59614983244133, 25.301055083549276], [22.884399771552765, 25.24
716078, 19.15573887399417], [20.250185405900563, 19.31123192669
547, 14.449456501686727], [16.26577205332588, 14.25544058230611
68781623], [10.98688575944926, 8.940342882350329], [10.767009636
147201635466], [3.5053612941244334, 7.354413925308031], [3.23078
, [0.4800524188640567, 1.6712578820081847], [0.3807651194467038
Start_pos: [0.04674959862981569, 0.33061933646842795] Goal_pos: [
94, 2.7671212523502877], [1.7280444295961936, 3.032700513566275
67480152775969, 5.733087741268227], [8.052383780738385, 5.96121
024466], [12.895511933358508, 11.159647066132004], [12.966108140
395], [15.267144749094019, 17.261535791860396], [15.48249851697
492296905, 22.42571435529936], [19.371421555051725, 22.6898840
, 26.52351806284505], [25.54387218656172, 26.61017541062822], [2
Start_pos: [26.6654444499290924, 26.92859552104784] Goal_pos: [0 0
77], [25.267633858047148, 21.702574556779165], [25.1286366766560
0497907743, 17.30782866675052], [20.086305854549778, 17.5129860
14.225293534689445], [14.256763077046513, 14.259093177163676],
19071529], [10.900916338124498, 8.843528803756936], [10.6761429
65596078032], [3.7788574012932097, 7.587088615168983], [3.49059
3577704849, 1.5167397298956293], [0.7613915058570097, 1.1876964
8244, 1.7713538934314723], [3.105259146175492, 1.80635869475822
90099756, 1.0155067795211663], [-0.06081491123387798, 0.7117594
Start_pos: [-0.04887092164125904, 0.3377781964586669] Goal_pos: [
.7104881709780548], [5.160371879140614, 1.5862010374138946], [5.
```

- Test-Welt erstellen
- Simulationen (viele!)
- Daten auswerten

# Vorgehen

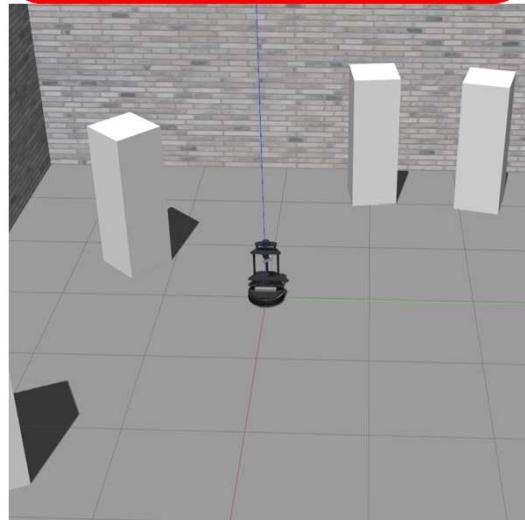
Echte  
Umgebung



- Portierung auf realen Roboter
- Gibt es Unterschiede zur Simulation?

# Vorgehen

Simulation  
aufbauen



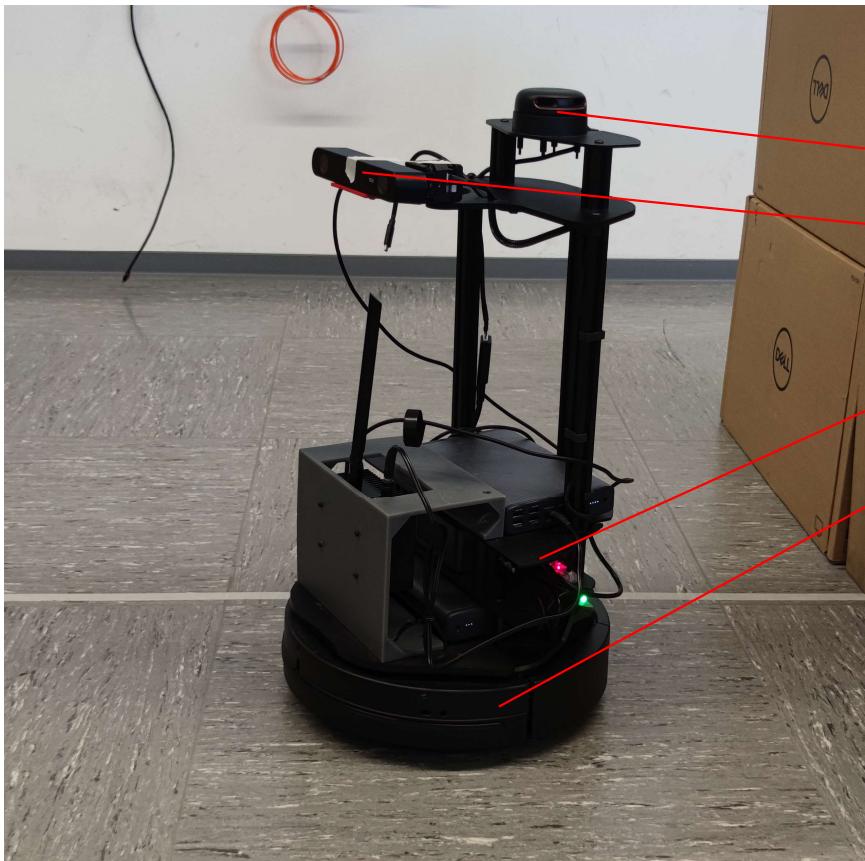
Daten  
sammeln

```
Start_pos: [26.67186895310854, 26.97870277638416] Goal_pos: [0 0]
59614983244133, 25.3010550833549276], [22.884399771552765, 25.24
716078, 19.15573887399417], [20.250185405900563, 19.3112319266;
547, 14.449456501686727], [16.26577205332588, 14.2554405823061;
68781623], [10.98688575944926, 8.940342882350329], [10.76700963;
147201635466], [3.5053612941244334, 7.354413925308031], [3.2307;
1.4800524188640567, 1.6712578820081847], [0.3807651194467038
Start_pos: [0.04674959862981569, 0.33061933646842795] Goal_pos: [
94, 2.7671212523502877], [1.7280444295961936, 3.03270051356627;
67480152775969, 5.733087741268227], [8.052383780738385, 5.9612;
022466], [12.895511933358508, 11.159647066132004], [12.96610814;
395], [15.267144749094019, 17.261535791860396], [15.48249851697;
492296905, 22.425714335529936], [19.371421555051725, 22.689840;
26.52351806284505], [25.54387218656172, 26.61017541062822], [2;
Start_pos: [26.665444499290924, 26.92859552104784] Goal_pos: [0 0];
77], [25.267633858047148, 21.702574556779165], [25.128636676656;
0497907743, 17.30782866675052], [20.086305854549778, 17.512986;
14.225293534689445], [14.256763077046513, 14.259093177163676],
19071529], [10.900916338124498, 8.843528803756936], [10.6761429;
65596078032], [3.7788574012932097, 7.587088615168983], [3.49059;
3577704849, 1.5167397298956293], [0.7613915058570097, 1.187696;
8244, 1.7713538934314723], [3.105259146175492, 1.8063586947582;
90099756, 1.0155067795211663], [-0.06081491123387798, 0.7117594
Start_pos: [-0.04887092164125904, 0.3377781964586669] Goal_pos: [
.7104881709780548], [5.160371879140614, 1.5862010374138946], [5.
```

Echte  
Umgebung



# Der Locobot



- LiDAR
- Stereolabs ZED 2 Stereo Camera
- Mini PC
- Kobuki platform

# Hardware im Detail

- LiDAR:
  - Scannt die Umgebung in kleinem Radius um den Roboter
  - Umgebungsdaten für den DWA Algorithmus
- ZED 2 Stereo Camera:
  - Erkennt Menschen in der Umgebung
  - Umgebungsdaten für das Prediction Model
- Mini PC:
  - Verarbeitung der Daten
  - Steuerung des Roboters
- Kobuki:
  - Plattform für die Bewegung
  - Akkurate Odometrie



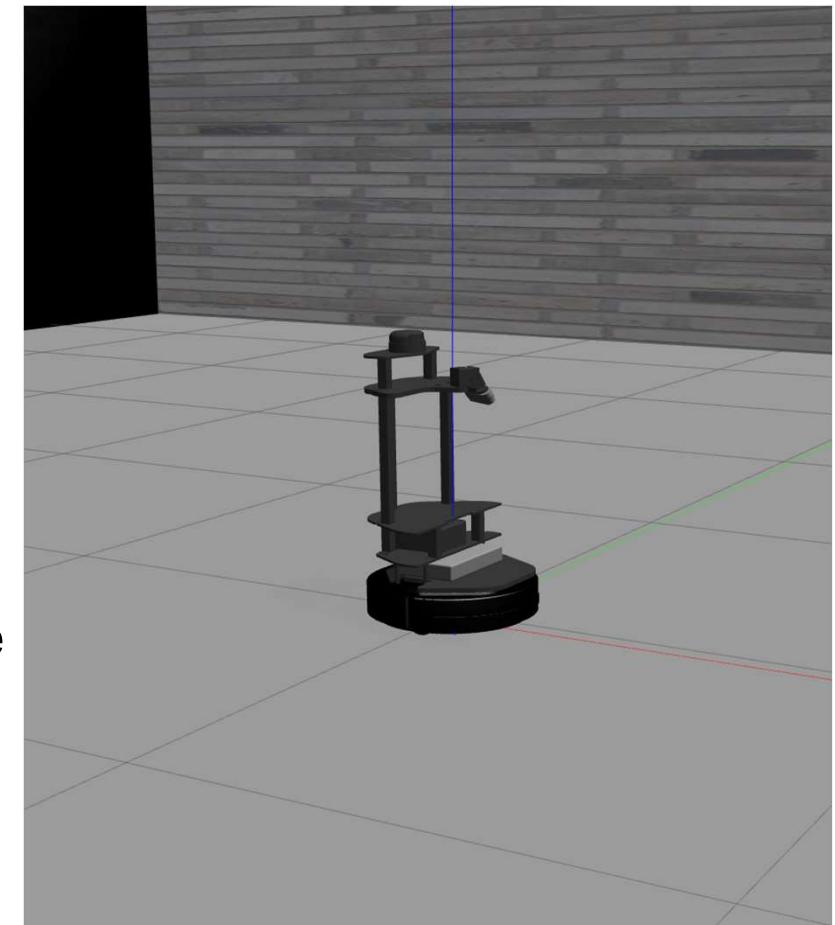
# Gazebo

- In Gazebo können realistische Szenarien entworfen und getestet werden (z.B. Roboter)
- Komplexe Innen- und Außenumgebungen auf eigener Karte mit beliebigen Variablen
- Elemente wie Roboter, Sensoren, statische Objekte, usw.
- Gazebo-Plugins bieten den URDF-Modellen eine größere Funktionalität
- Zum Beispiel:
  - `gazebo_ros` : Dieses Plugin bietet Nachrichten- und Service-Publisher für die Verwendung von Gazebo mit ROS

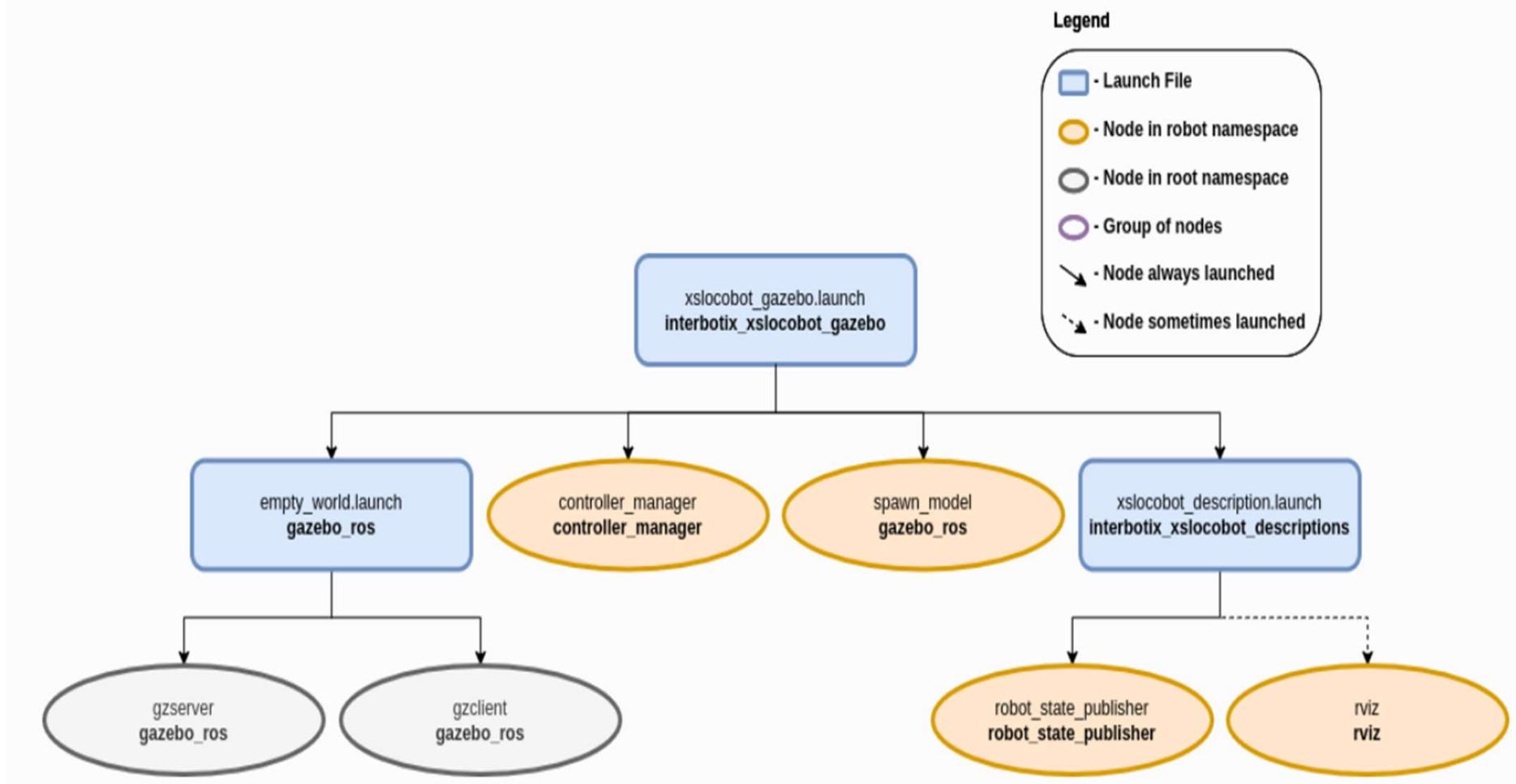
# Locobot in Gazebo

## Interbotix\_xslocobot\_gazebo Packet:

- Basiert auf interbotix\_xslocobot\_descriptions and gazebo\_ros packages
- Simuliert Locobot in Gazebo mit Physik (Reibung, ungenaue Motoren usw.)
- Kann (wie echter Roboter) Bewegungsbefehle ausführen

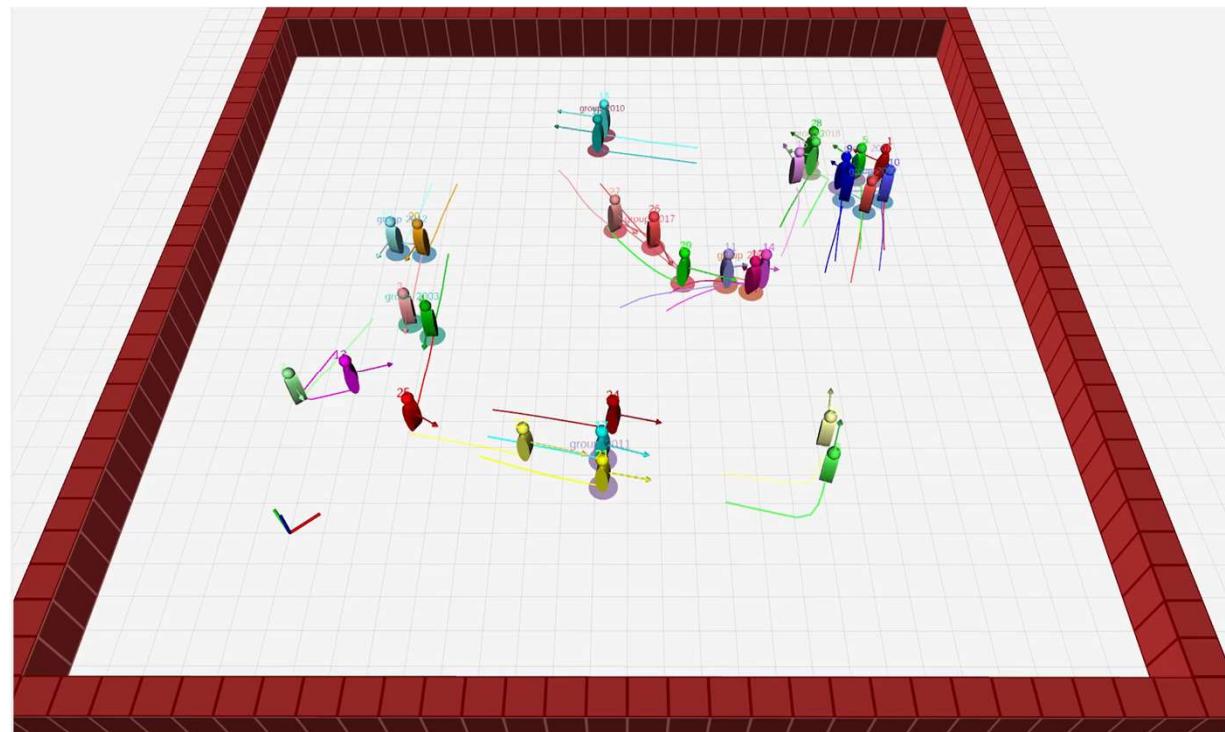


# Locobot in Gazebo



# Pedestrian Simulator - Übersicht

- Projekt der SRL-Freiburg (Self-Regulated Learning)
- Nutzt “Social Force Modell” nach Helbing et. al.
- D.h. die einzelnen Personen werden von verschiedenen Kräften beeinflusst:
  - Goal Force
  - Obstacle Force
  - Sozial Force
  - Group Force
  - Random Force
  - Wall Force
- Simuliert Roboter  
→ Personen reagieren “realistisch”



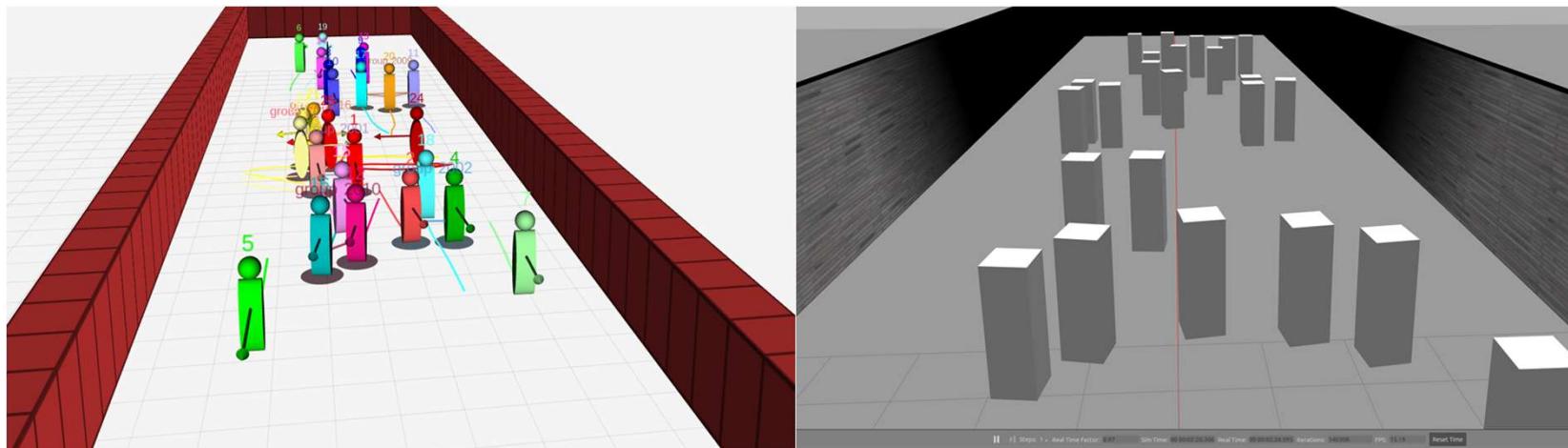
# Pedestrian Simulator - Übersicht

- Szenarien XML basiert
  - Hindernisse
  - Wegpunkte & Schlangen
  - Personen(gruppen)
  - Roboter
- C++ und Python Code
- Problem:
  - Keine Updates → Teilweise Überarbeitung nötig
  - Wenig Dokumentation

```
1 ?xml version="1.0" encoding="UTF-8"?>
2 <!-- This scenario file was created by SGDicoP on 2014-04-16T15:14:48-->
3 <scenario>
4   <!--Obstacles-->
5   <obstacle x1="-10.0" y1="-10.0" x2="29.5" y2="-10.0"/>
6   <obstacle x1="-10.0" y1="-10.0" x2="-10.0" y2="29.5"/>
7   <obstacle x1="-10.0" y1="29.5" x2="29.5" y2="29.5"/>
8     <obstacle x1="29.5" y1="29.5" x2="29.5" y2="-10.0"/>
9
10  <!--Waypoints (incl. WaitingQueues)-->
11  <waypoint id="bookshop_entry" x="25" y="5" r="5"/>
12  <waypoint id="bookshop_exit" x="25" y="25" r="5"/>
13
14  <waypoint id="coffee_entry" x="5" y="25" r="5"/>
15  <waypoint id="coffee_exit" x="5" y="5" r="5"/>
16
17  <waypoint id="robot_goal" x="22" y="27" r="2"/>
18  <waypoint id="robot_start" x="4" y="4" r="2"/>
19
20  <queue id="info_desk" x="20" y="15" direction="0"/>
21
22  <!--Agents-->
23  <agent x="0" y="0" n="1" dx="0" dy="0" type="2">
24    <addwaypoint id="robot_start"/>
25    <addwaypoint id="robot_goal"/>
26  </agent>
27  <!--AgentClusters-->
28  <agent x="6" y="5" n="10" dx="3" dy="3" type="0">
29    <addwaypoint id="bookshop_entry"/>
30    <addwaypoint id="bookshop_exit"/>
31    <addwaypoint id="coffee_entry"/>
32    <addwaypoint id="coffee_exit"/>
33  </agent>
34  <agent x="7" y="5" n="5" dx="3" dy="3" type="1">
35    <addqueue id="info_desk"/>
36    <addwaypoint id="bookshop_exit"/>
37    <addwaypoint id="coffee_entry"/>
38    <addwaypoint id="coffee_exit"/>
39  </agent>
40  <agent x="24" y="25" n="10" dx="4" dy="4" type="0">
41    <addwaypoint id="coffee_entry"/>
42    <addwaypoint id="coffee_exit"/>
43    <addwaypoint id="bookshop_entry"/>
44    <addwaypoint id="bookshop_exit"/>
45  </agent>
46  <agent x="23" y="25" n="5" dx="3" dy="3" type="1">
47    <addwaypoint id="coffee_entry"/>
48    <addqueue id="info_desk"/>
49    <addwaypoint id="bookshop_exit"/>
50  </agent>
51</scenario>
```

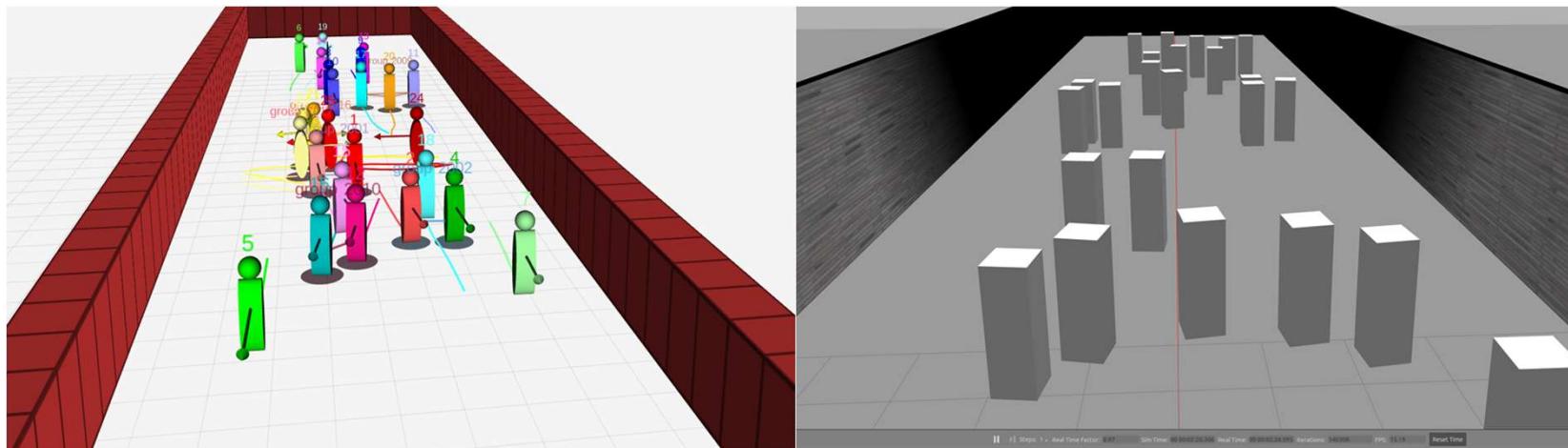
# Pedestrian Simulator - Übersicht

- Unterstützt Gazebo:
  - Erstellt Gazebo Welt
  - Pedsim läuft im Hintergrund
  - Spawnt Personen in Gazebo Simulation & updated Positionen
- Idee:
  - Locobot in Gazebo für Simulation
  - Pedsim Robot in Pedsim für (realistische) Personenbewegungen

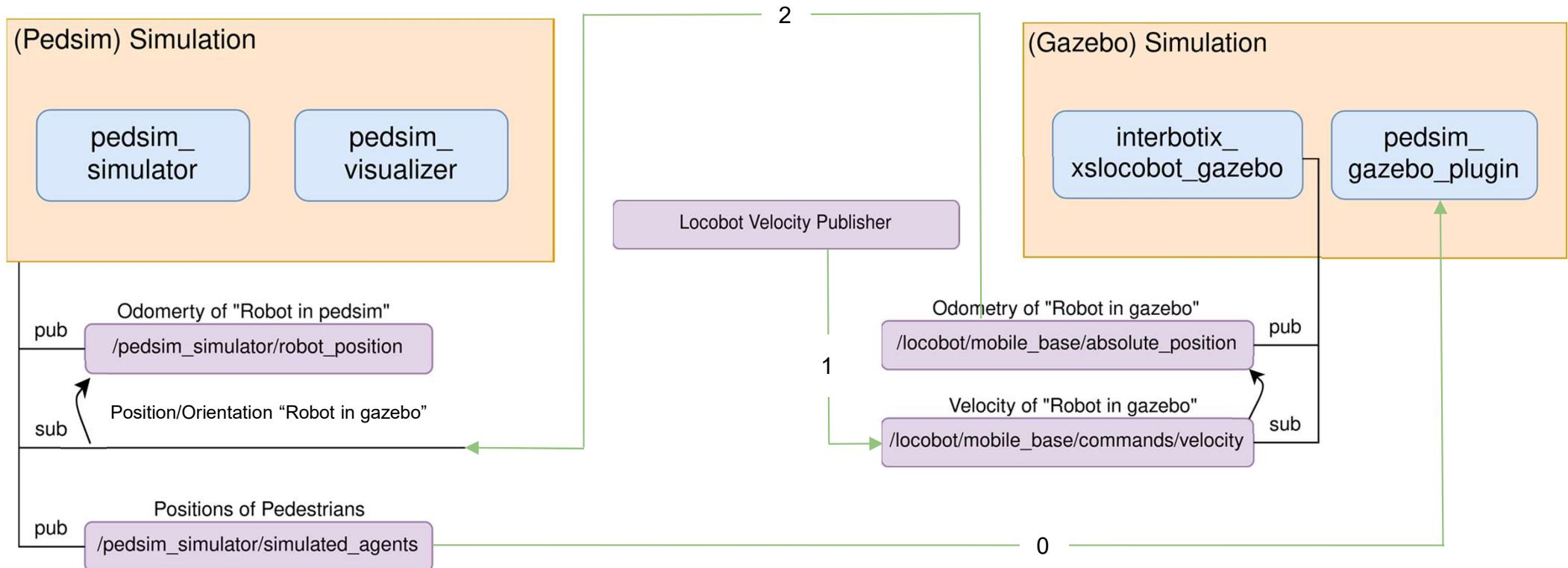


# Pedestrian Simulator für unser Projekt

- Simultanes Ansteuern vom Locobot in Gazebo und Pedsim Roboter
- Problem: Gazebo unterstützt Physik
  - Roboter reagieren nicht gleich auf Bewegungsbefehle
  - Umschreiben von Pedsim, um absolute Koordinaten zu benutzen



# Pedsim + Gazebo + Locobot



# Lidar



Entfernungen &  
Winkel

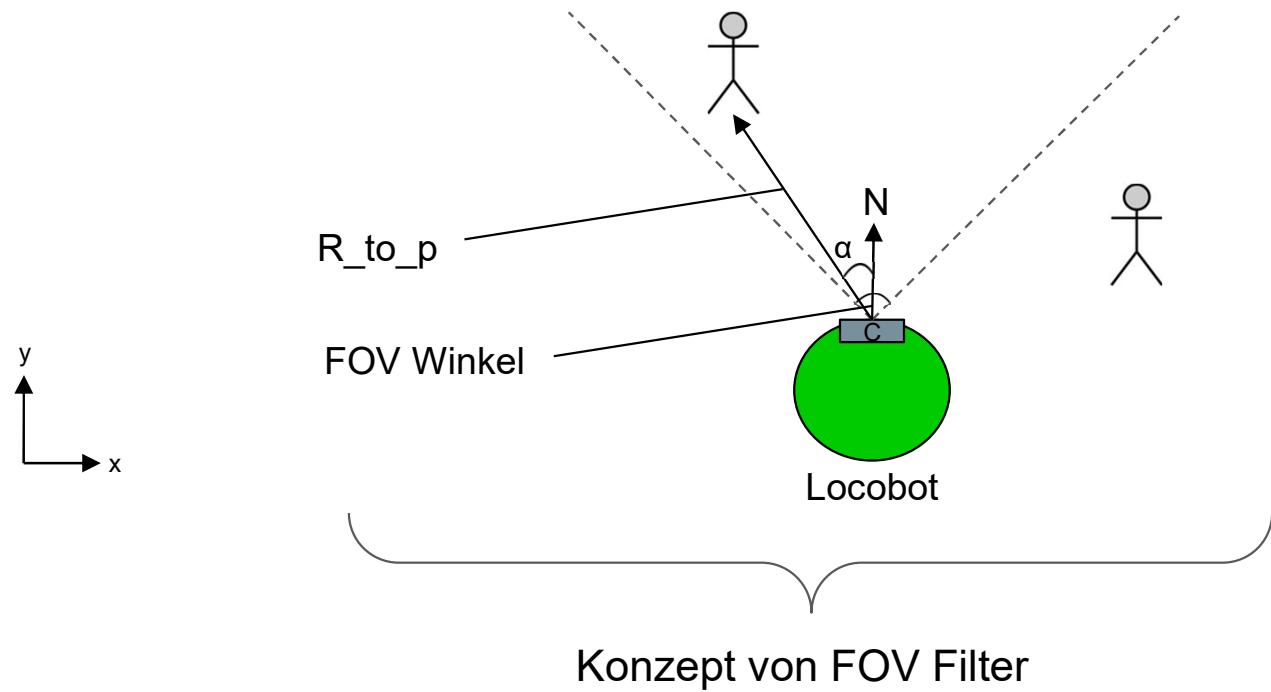
laserToAbs()

Weltkoordinaten  
(x, y)

```
header:  
  seq: 1933  
  stamp:  
    secs: 76  
    nsecs: 244000000  
  frame_id: "locobot/laser_frame_link"  
angle_min: -3.141590118408203  
angle_max: 3.141590118408203  
angle_increment: 0.015747318044304848  
time_increment: 0.0  
scan_time: 0.0  
range_min: 0.20000000298023224  
range_max: 16.0  
ranges: [11.99010181427002, 12.015121459960938, 12.026434898376465, 12.013851165771484, 12.036181449890137, 12.030182838439941, 12.05628490447998, 12.06802749633789, 12.083757400512695, 12.107967376708984, 12.154698371887207, 12.156437873840332, 12.194140434265137, 12.251670837402344, 12.308550834655762, 12.345300674438477, 12.373998641967773, 12.444868087768555, 12.497284889221191, 12.553483963012695, 12.596541404724121, 12.669074058532715, 12.723958015441895, 12.818258285522461, 9.133149147033691, 8.27815055847168, 8.145909309387207, 8.169034004]
```

# Erkennung von Menschen (Kamera)

- Simulation: FOV(Field-of-View) Filter



$\alpha$ -Calculation:  
Skalar-Produkt & arccos

Überprüfen:  
 $\alpha \leq \text{FOV Winkel} / 2$

# Prediction Model

- Autoregressive(AR) Modell
  - nutzt vorherigen Zustand, um Vorhersage zu treffen
  - geht von einer linearen Beziehung aus
- Verwendet Transformer
- Pre-Trained mit den ETH\* und UCY\*\* Datensätzen  
→ bekannte Datensätze für menschliche Bewegungen

```
TrajectoryGeneratorAR(  
    (inputLayer_encoder): Linear(in_features=2, out_features=16, bias=True)  
    (inputLayer_decoder): Linear(in_features=16, out_features=16, bias=True)  
    (pl_net): Pooling_net(  
        (attn): Linear(in_features=16, out_features=1, bias=True)  
        (spatial_embedding): Linear(in_features=2, out_features=32, bias=True)  
        (mlp_pre_pool): Sequential(  
            (0): Linear(in_features=64, out_features=64, bias=True)  
            (1): ReLU()  
            (2): Linear(in_features=64, out_features=16, bias=True)  
            (3): ReLU()  
        )  
    )  
    (hist_encoder): Hist_Encoder(  
        (transformer_encoder): TransformerEncoder(  
            (layers): ModuleList(  
                (0): TransformerEncoderLayer(  
                    (self_attn): MultiheadAttention(  
                        (out_proj): NonDynamicallyQuantizableLinear(in_features=16, out_features=16, bias=True)  
                    )  
                    (linear1): Linear(in_features=16, out_features=32, bias=True)  
                    (dropout): Dropout(p=0.0, inplace=False)  
                    (linear2): Linear(in_features=32, out_features=16, bias=True)  
                    (norm1): LayerNorm((16,), eps=1e-05, elementwise_affine=True)  
                    (norm2): LayerNorm((16,), eps=1e-05, elementwise_affine=True)  
                    (dropout1): Dropout(p=0.0, inplace=False)  
                    (dropout2): Dropout(p=0.0, inplace=False)  
                )  
            )  
            (dropout): Dropout(p=0.0, inplace=False)  
            (fc): Linear(in_features=32, out_features=16, bias=True)  
            (input_fc): Linear(in_features=2, out_features=16, bias=True)  
        )  
        (pred_lstm_model): LSTMCell(16, 16)  
        (pred_hidden2pos): Linear(in_features=16, out_features=4, bias=True)  
        (dropout): Dropout(p=0.0, inplace=False)  
    )  
)
```

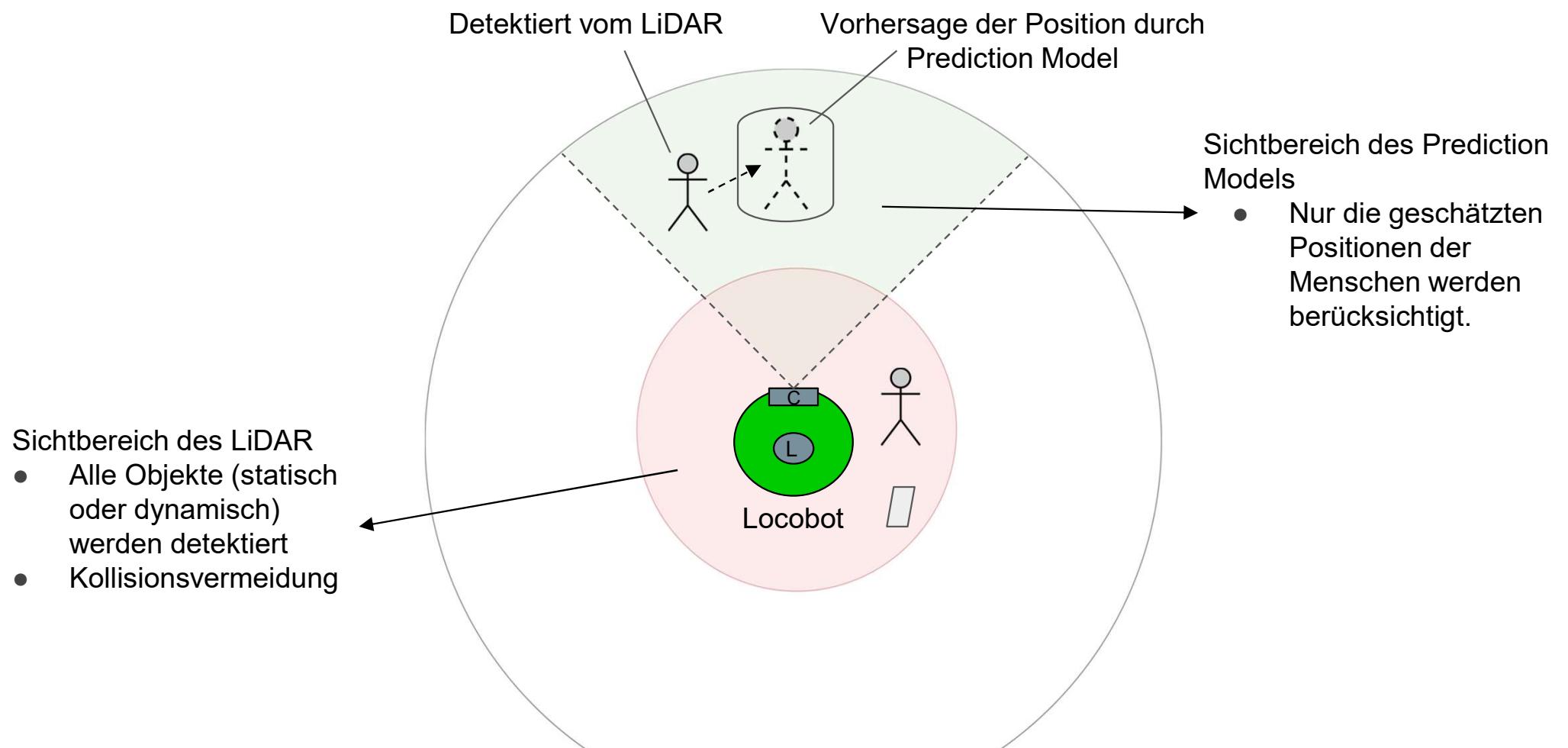
\* "You'll never walk alone: Modeling social behavior for multi-target tracking", Pellegrini, S. et. al. (2009)

\*\* Crowds by example", Lerner, A. et. al. (2007)

# Prediction Model

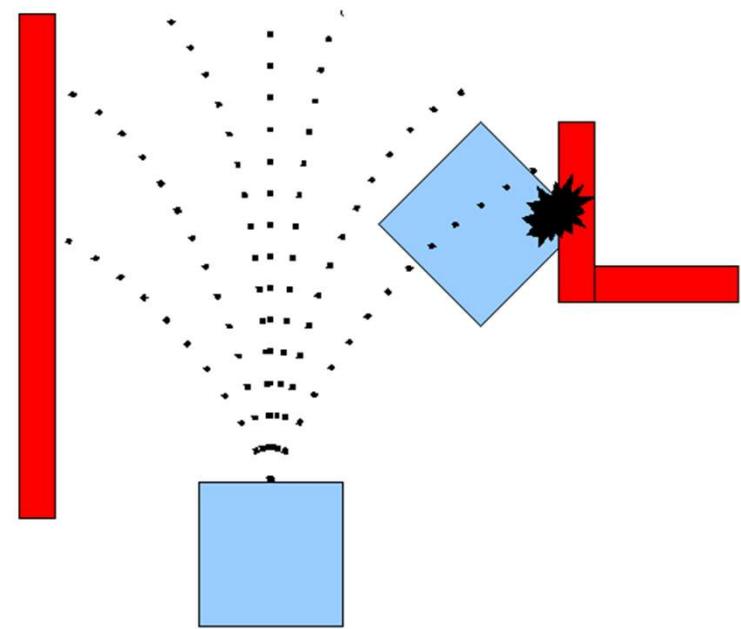
- $n$  sei die Anzahl der Personen, die wir verfolgen
- **Eingabe:**
  - Nachbarschaft Matrix:  $n \times n$  Matrix
  - Position der Menschen in X-Koordinaten:  $8 \times n \times 2$  Matrix
  - Relative Trajektorien der Menschen:  $8 \times n \times 2$  Matrix
- **Ausgabe:**
  - Vorhergesagte Trajektorien der Menschen (3 Ticks in die Zukunft) in X-Koordinaten:  $3 \times n \times 2$  Matrix
- **Änderung des bestehenden Modells:**
  - Keine Vorhersage der Roboterposition bei der Vorwärtspropagierung  
→ für die aktuelle Aufgabe nicht erforderlich

# Prediction Model + Kamera + LiDAR



# DWA

- Dynamic Window Approach (DWA)
- Local path planning Algorithmus
- Eingabedaten vom LiDAR Sensor
  - Winkel und Distanz zum Objekt
  - Transformation zu Point Cloud Map
    - Kartesische Koordinaten (X, Y, Z)
    - Point cloud Koordinaten
- Eingabedaten vom Prediction Model
  - Zukünftige Position der Menschen
    - Kartesische Koordinaten (X, Y, Z)

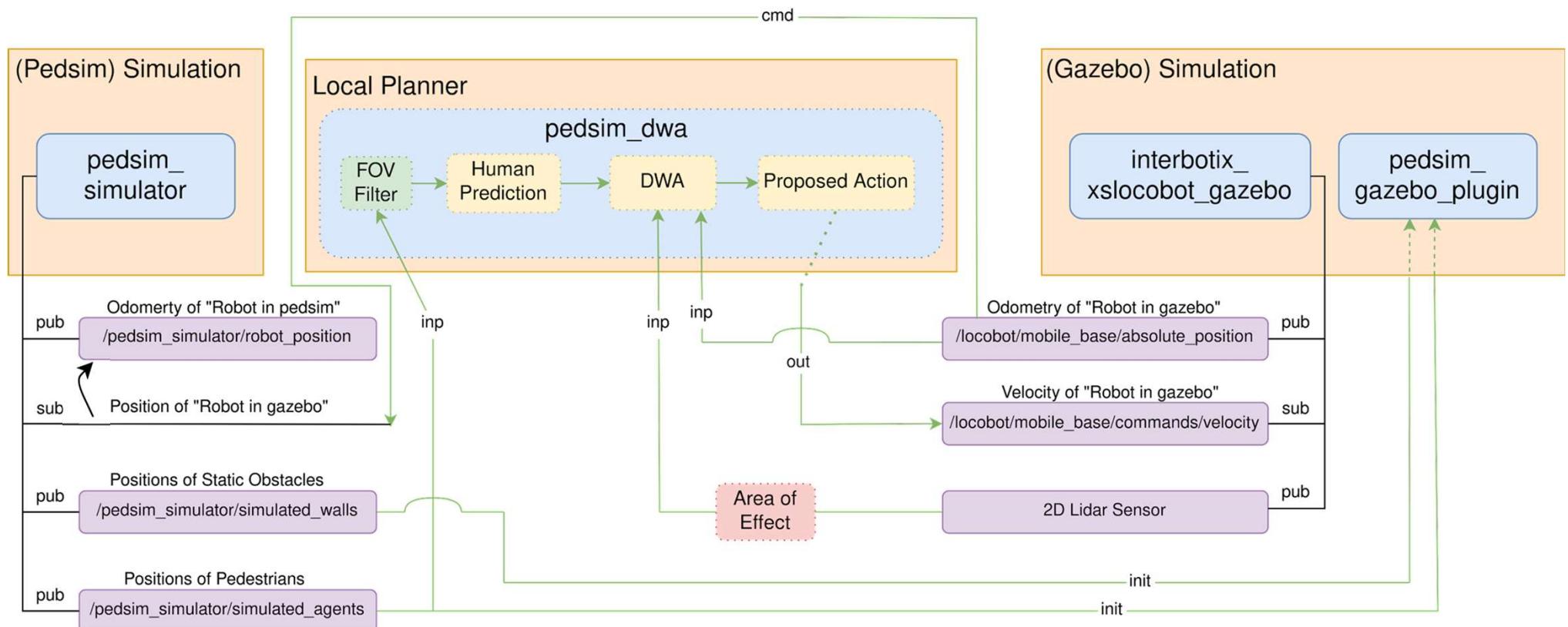


[http://wiki.ros.org/base\\_local\\_planner](http://wiki.ros.org/base_local_planner)

# DWA

- Vorteile der Dynamik-Fenster-Methode
  - Online-Daten
  - Nutzbar bei limitierte Geschwindigkeit und Beschleunigung
  - Optimalen Regelung um dynamischen und statischen Hindernissen auszuweichen
- Finaler Stand:
  - In der Simulation:
    - Lidar + DWA + Perception + Prediction Model funktionieren kombiniert
    - 1 Lauch File möglich (2 für Nutzerfreundlichkeit)

# Datenfluss



\* **init:** Datenpfad für die Initialisierung der Gazebo-Welt

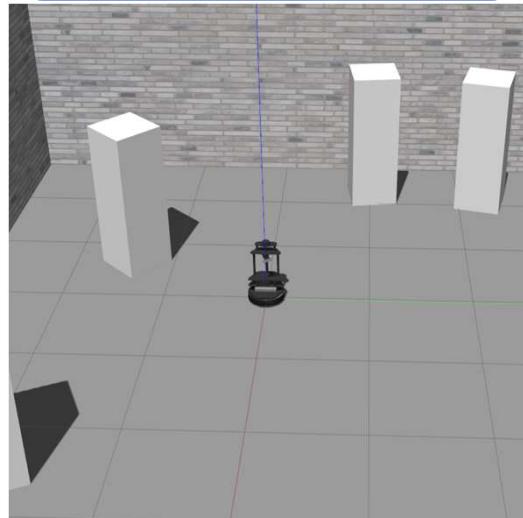
\* **cmd:** aktuelle (absolute) Position des Roboters im Gazebo, die an Pedsim gesendet wird

\* **inp:** Eingabe von Daten in den lokalen Planer

\* **out:** Ausgabe der lokalen Planer

# Vorgehen

Simulation  
aufbauen



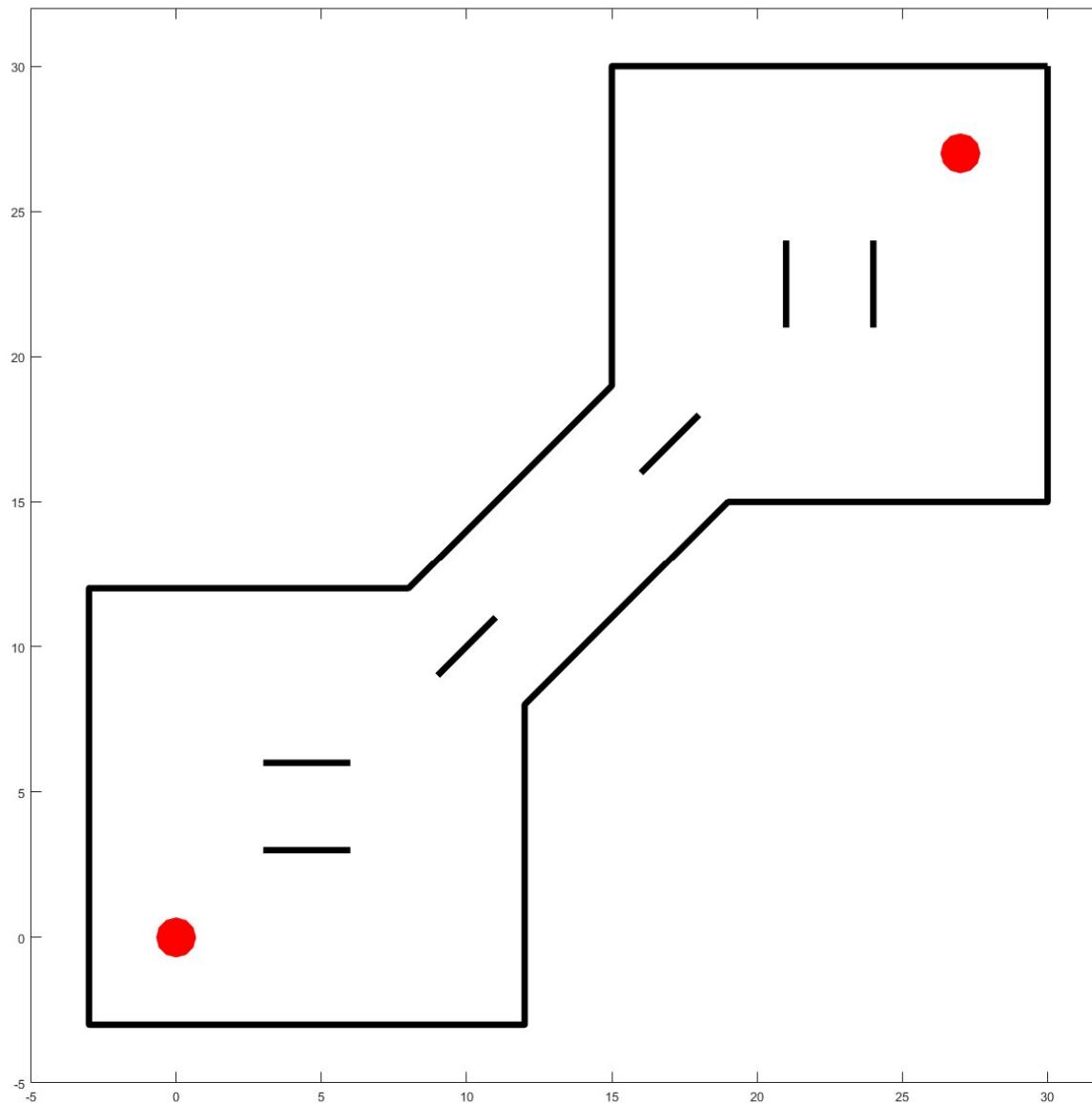
Daten  
sammeln

```
Start_pos: [26.67186895310854, 26.97870277638416] Goal_pos: [0 0]
59614983244133, 25.3010550833549276], [22.884399771552765, 25.24
716078, 19.15573887399417], [20.250185405900563, 19.3112319266;
547, 14.449456501686727], [16.26577205332588, 14.2554405823061;
68781623], [10.98688575944926, 8.940342882350329], [10.76700963;
147201635466], [3.5053612941244334, 7.354413925308031], [3.2307;
1.04800524188640567, 1.6712578820081847], [0.3807651194467038
Start_pos: [0.04674959862981569, 0.33061933646842795] Goal_pos: [
94, 2.7671212523502877], [1.7280444295961936, 3.03270051356627;
67480152775969, 5.733087741268227], [8.052388780738385, 5.9612;
022466], [12.895511933358508, 11.159647066132004], [12.96610814;
395], [15.267144749094019, 17.261535791860396], [15.48249851697;
492296905, 22.425714335529936], [19.371421555051725, 22.689840
26.52351806284505], [25.54387218656172, 26.61017541062822], [2;
Start_pos: [26.665444499290924, 26.92859552104784] Goal_pos: [0 0];
77], [25.267633858047148, 21.702574556779165], [25.128636676656;
0497907743, 17.30782866675052], [20.086305854549778, 17.512986(
14.225293534689445], [14.256763077046513, 14.259093177163676],
19071529], [10.900916338124498, 8.843528803756936], [10.6761429-
65596078032], [3.7788574012932097, 7.587088615168983], [3.49059-
3577704849, 1.5167397298956299], [0.7613915058570097, 1.187696-
8244, 1.7713538934314723], [3.105259146175492, 1.8063586947582;
90099756, 1.0155067795211663], [-0.06081491123387798, 0.7117594
Start_pos: [-0.04887092164125904, 0.3377781964586669] Goal_pos: [
.7104881709780548], [5.160371879140614, 1.5862010374138946], [5.
```

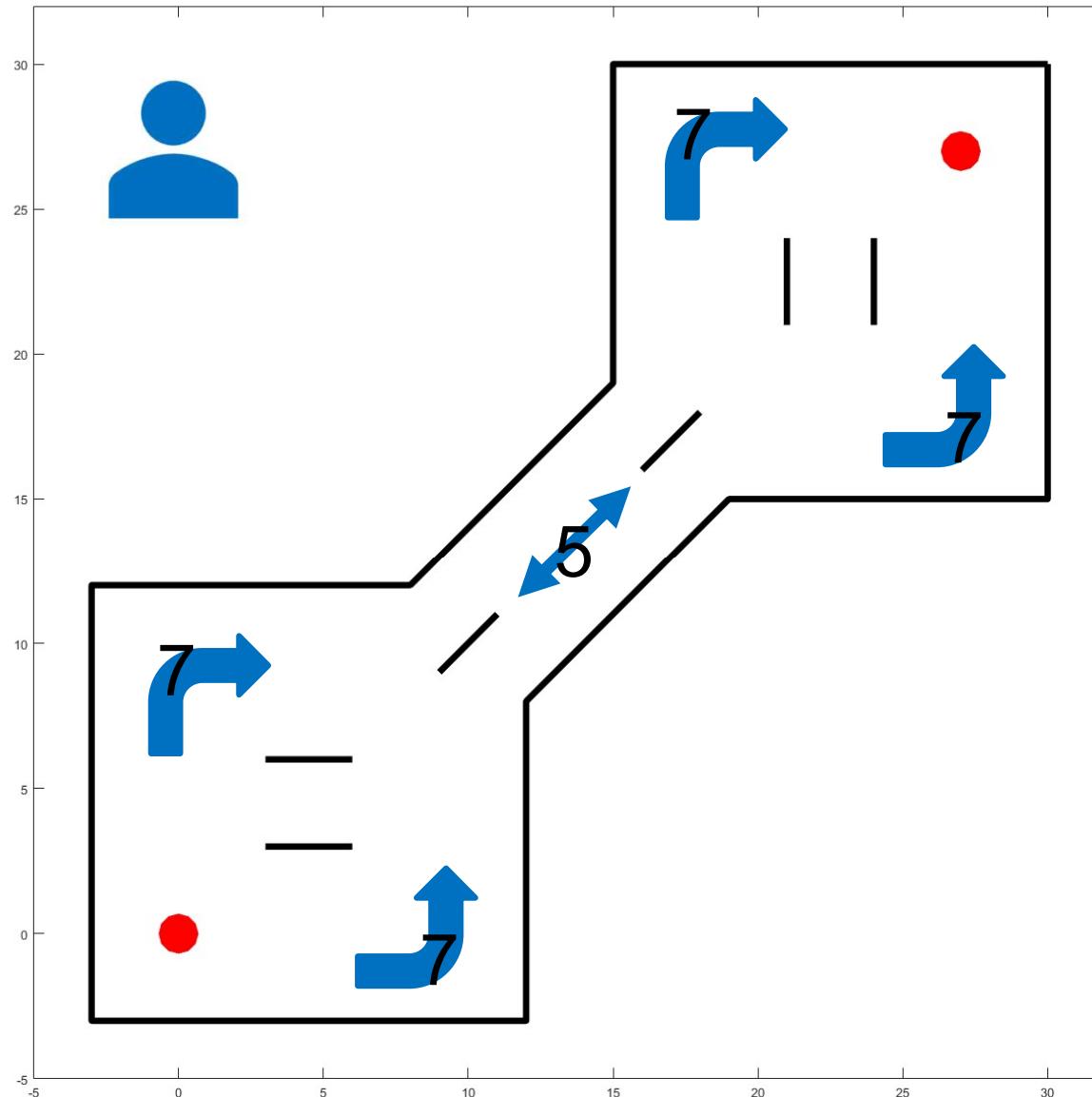
Echte  
Umgebung



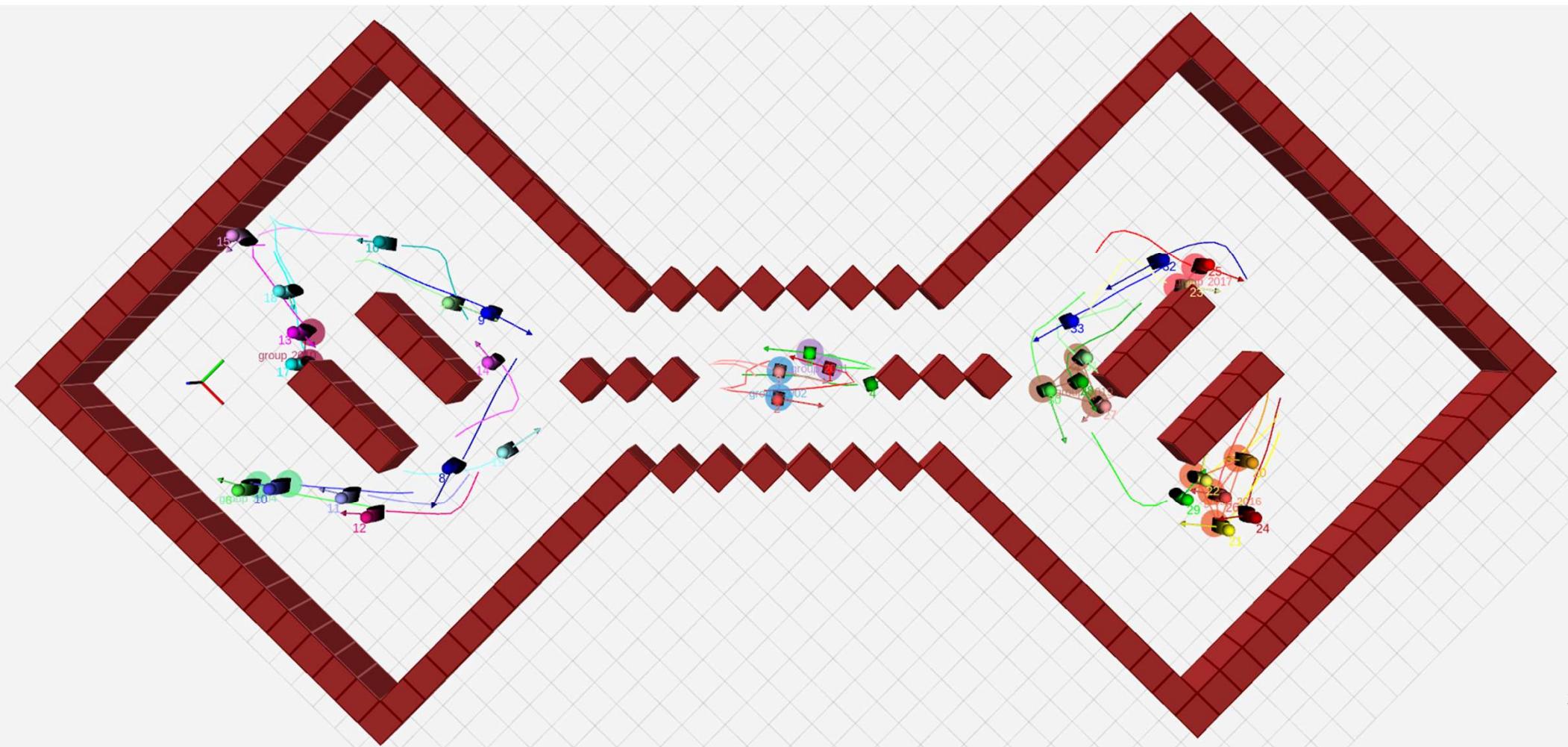
# Die Karte



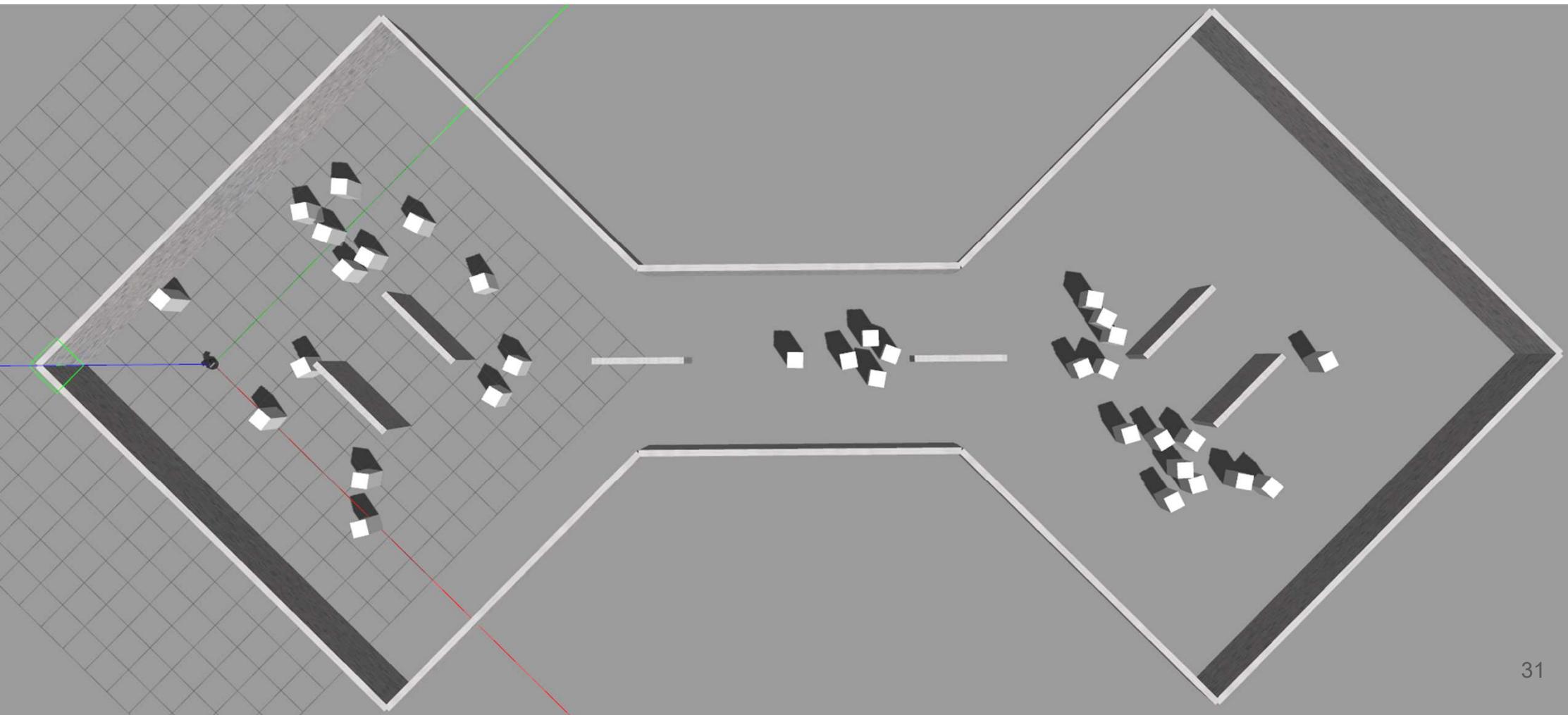
# Die Karte

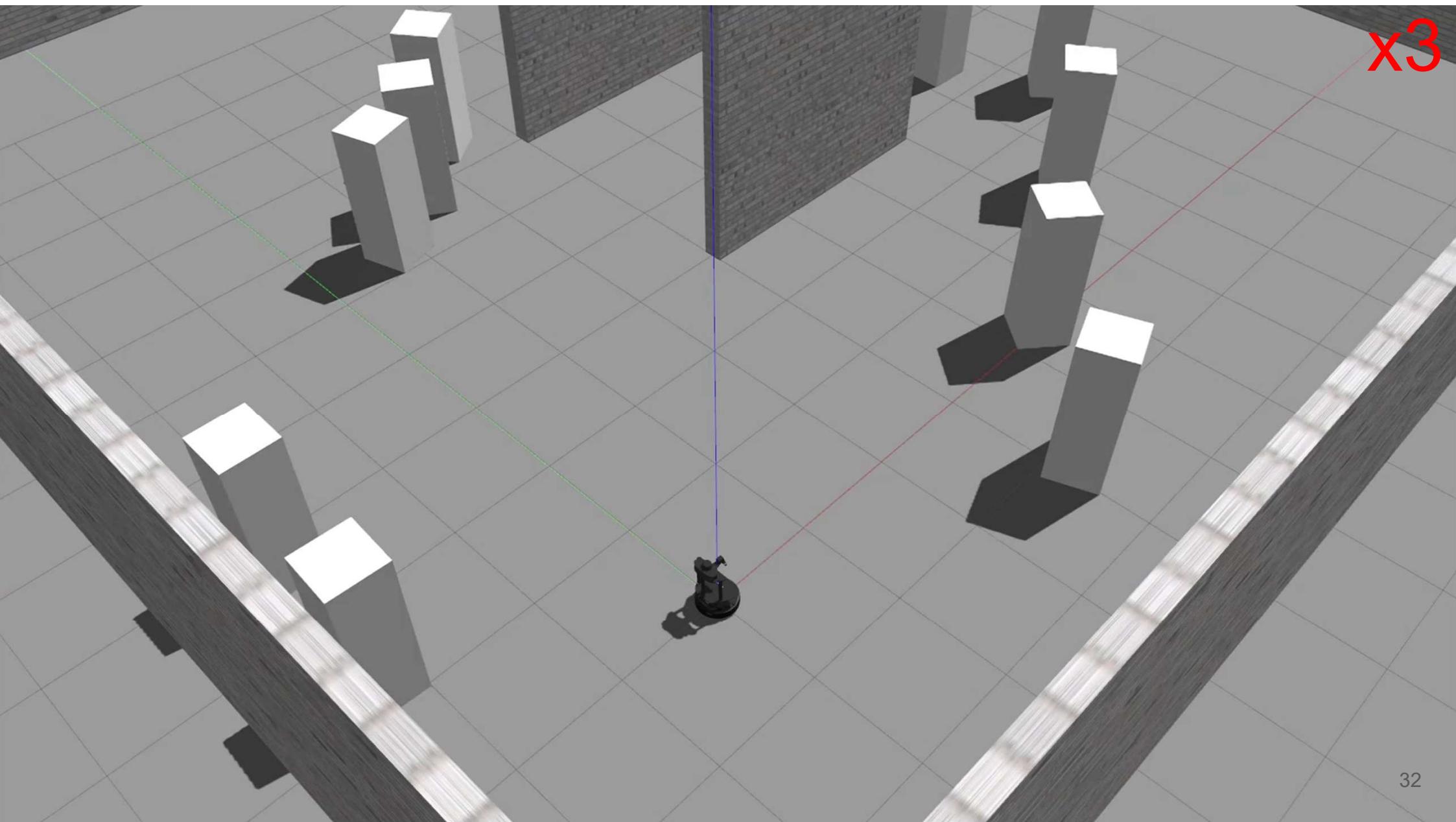


# Die Karte

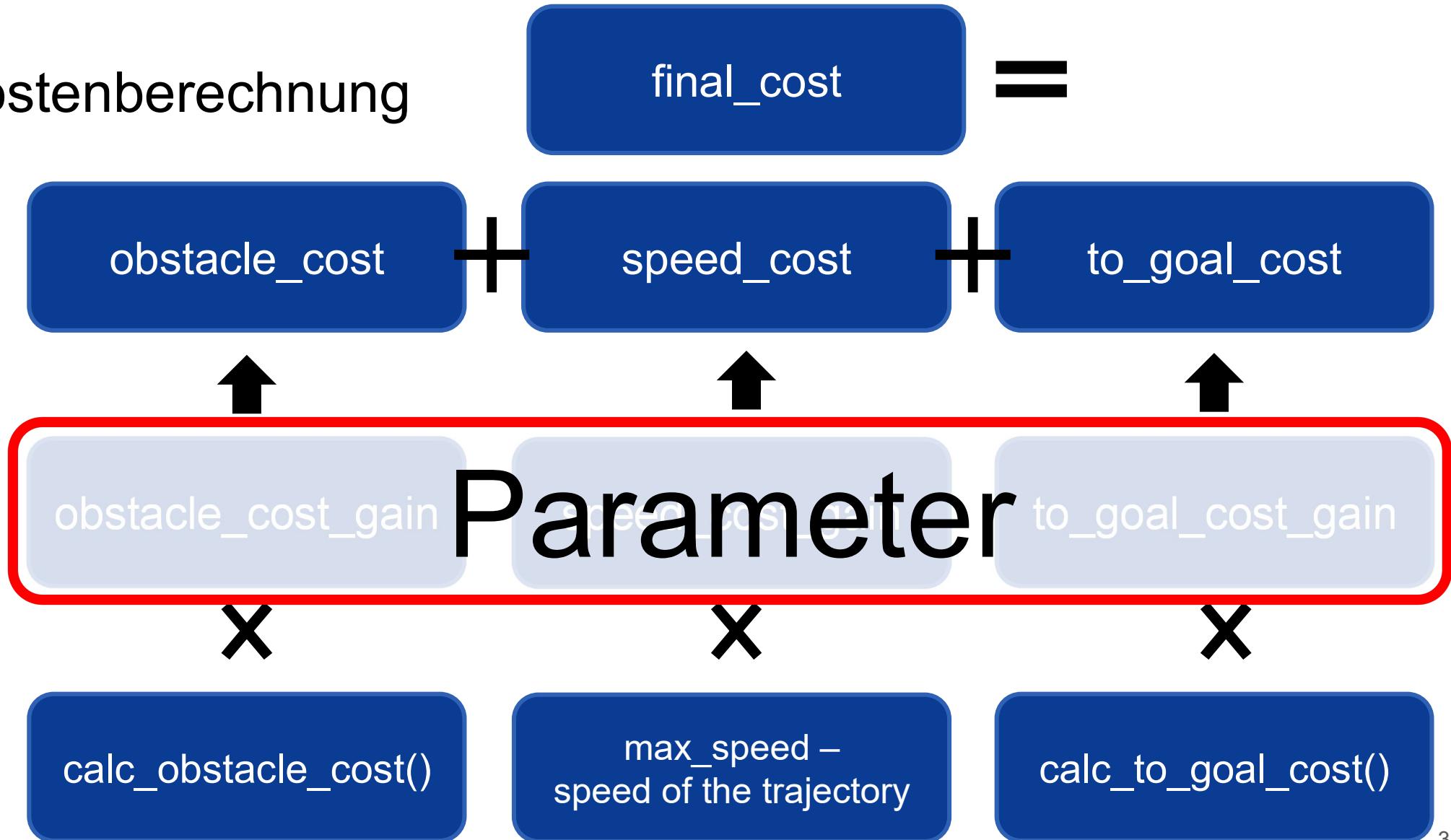


# Die Karte





## Kostenberechnung



# Roboter Parameter

Zu hoch

Große Umwege

Wird nie langsamer

Umfährt keine  
Hindernisse

obstacle\_cost\_gain

speed\_cost\_gain

to\_goal\_cost\_gain

Zu klein

Kollisionen

Umfährt keine  
Hindernisse

Ziel unwichtig

# Roboter Parameter

Zu hoch

Große Umwege

Wird nie langsamer

Umfährt keine  
Hindernisse

obstacle\_cost\_gain  
=3

speed\_cost\_gain  
=4

to\_goal\_cost\_gain  
=0.5

Zu klein

Kollisionen

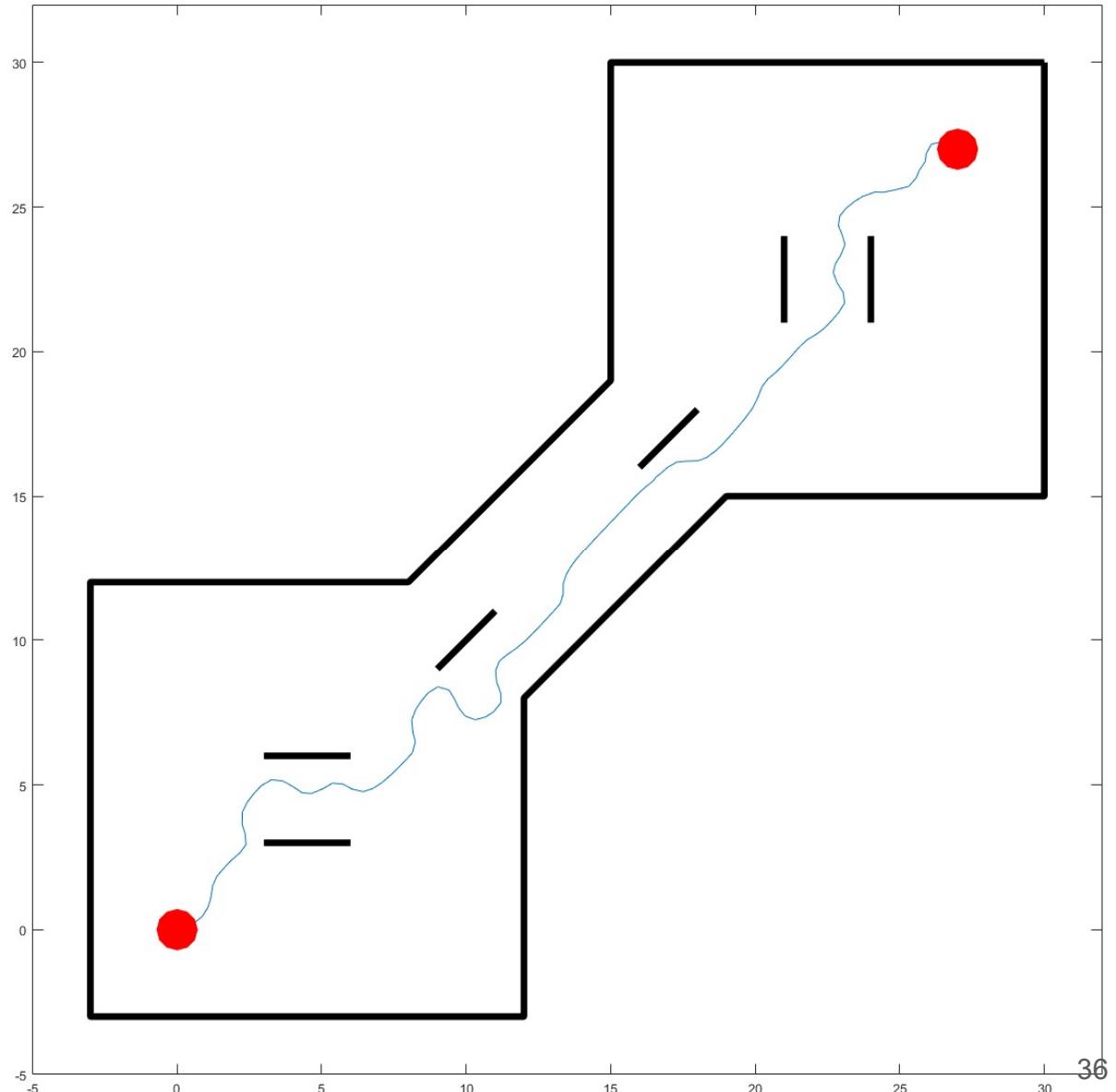
Umfährt keine  
Hindernisse

Ziel unwichtig

# Daten

- Start und Ziel Position
- Minimale Lidar Entfernung
- Zeit
- Roboter Position

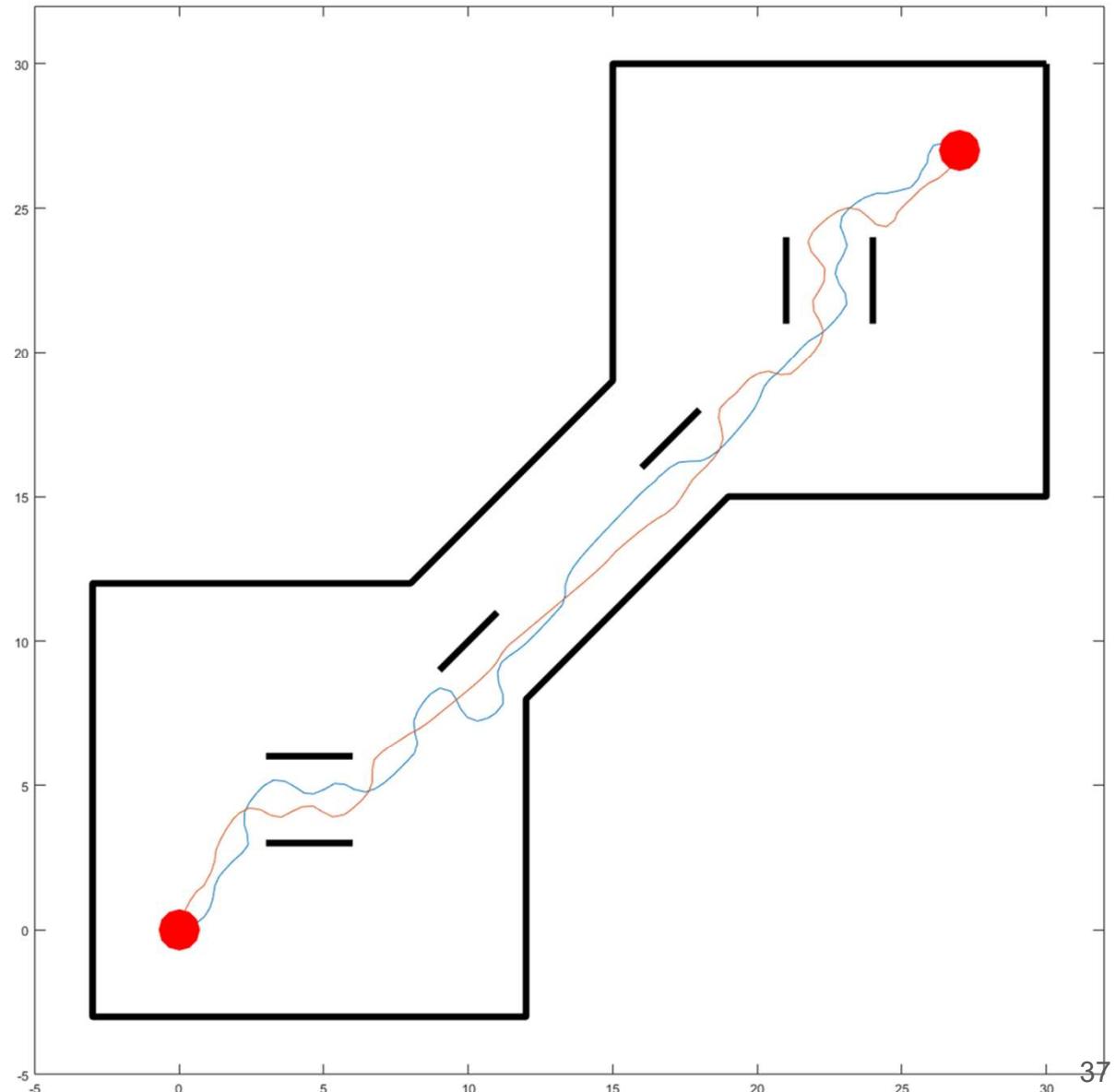
→ Automatisiert in .txt Datei



# Daten

- Start und Ziel Position
- Minimale Lidar Entfernung
- Zeit
- Roboter Position

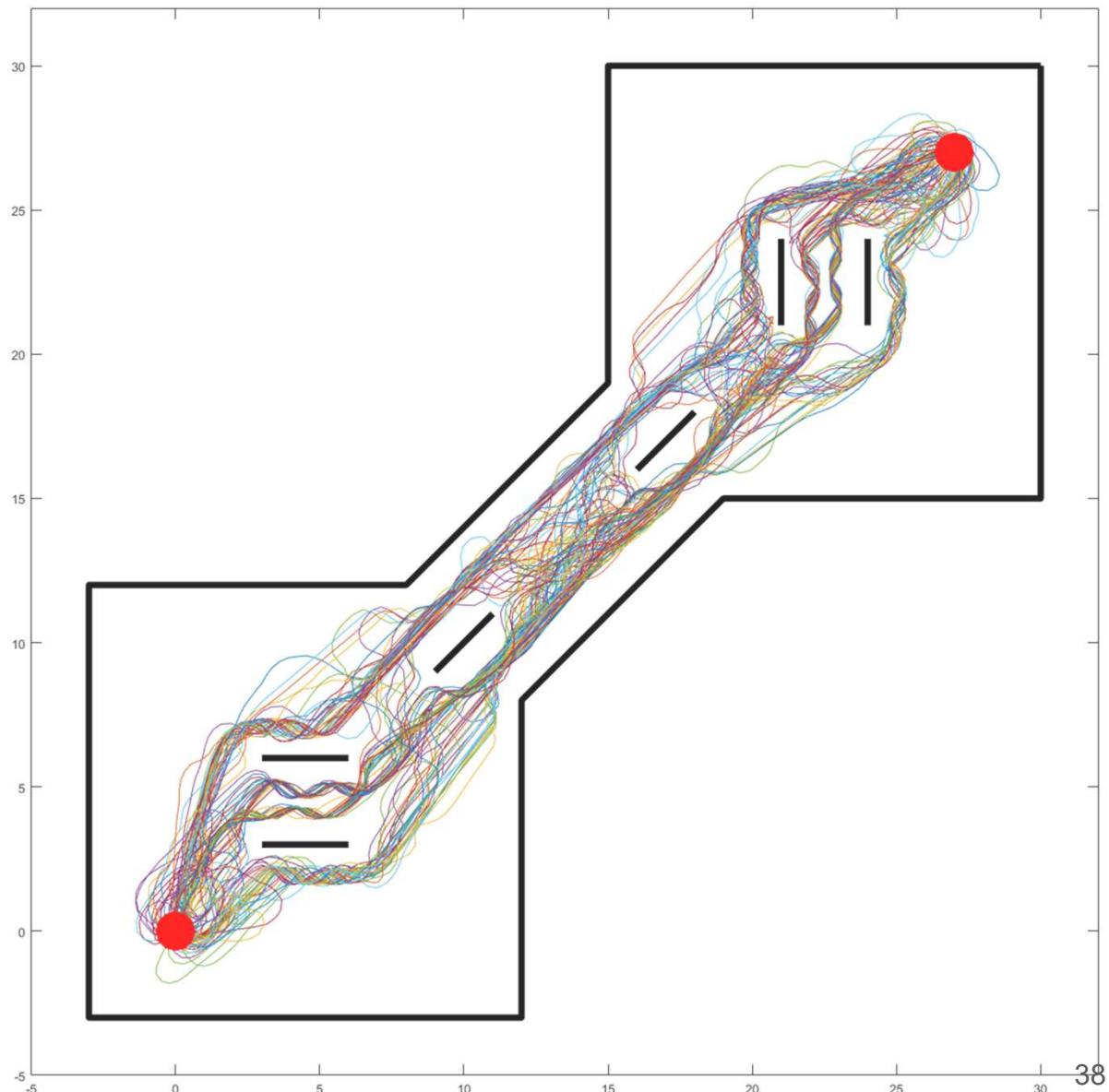
→ Automatisiert in .txt Datei



# Daten

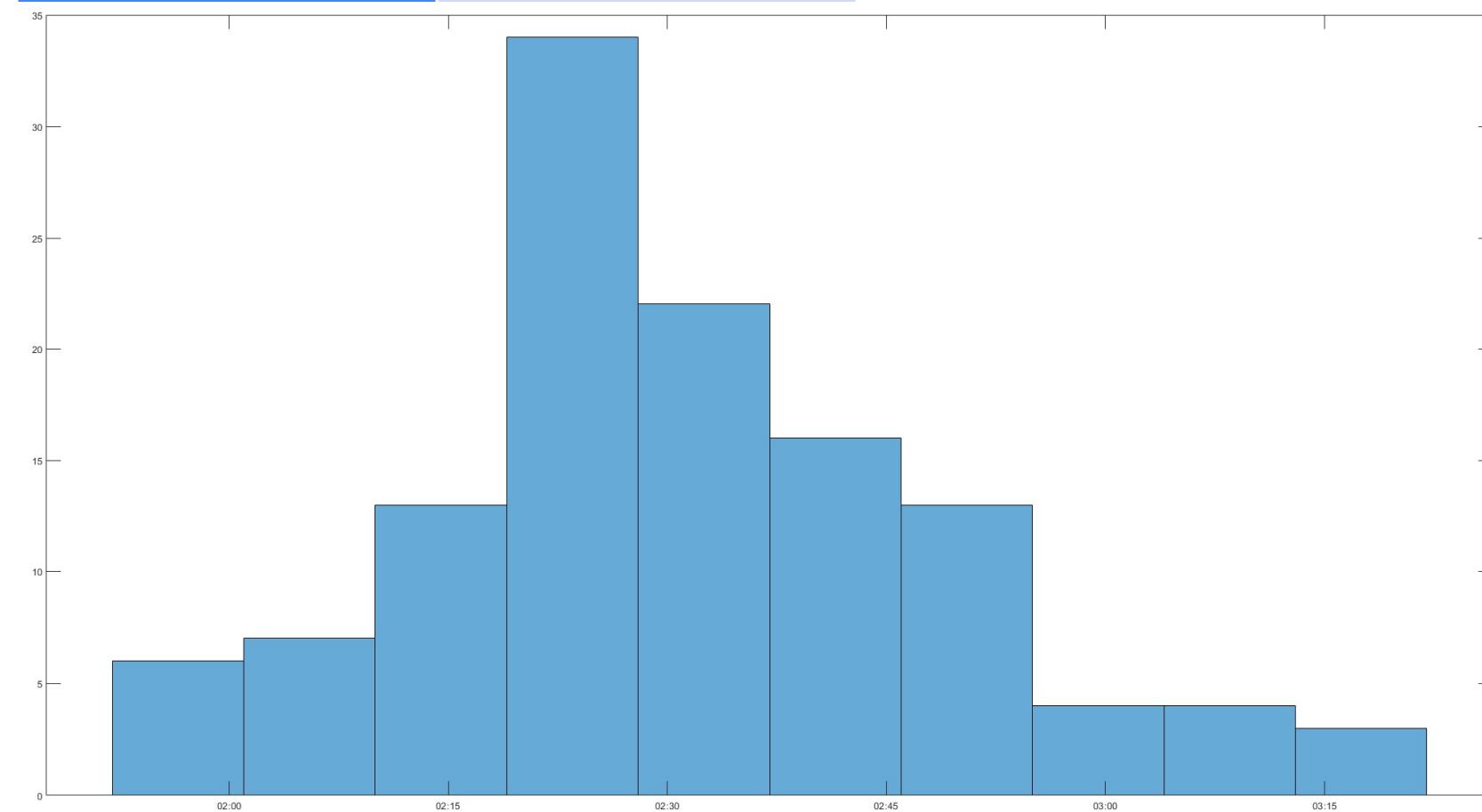
- Start und Ziel Position
- Minimale Lidar Entfernung
- Zeit
- Roboter Position

→ Automatisiert in .txt Datei



# Erstes Simulationsergebnis

Zeit (Durchschnitt) 2:30 Standardabweichung: 18 Sekunden

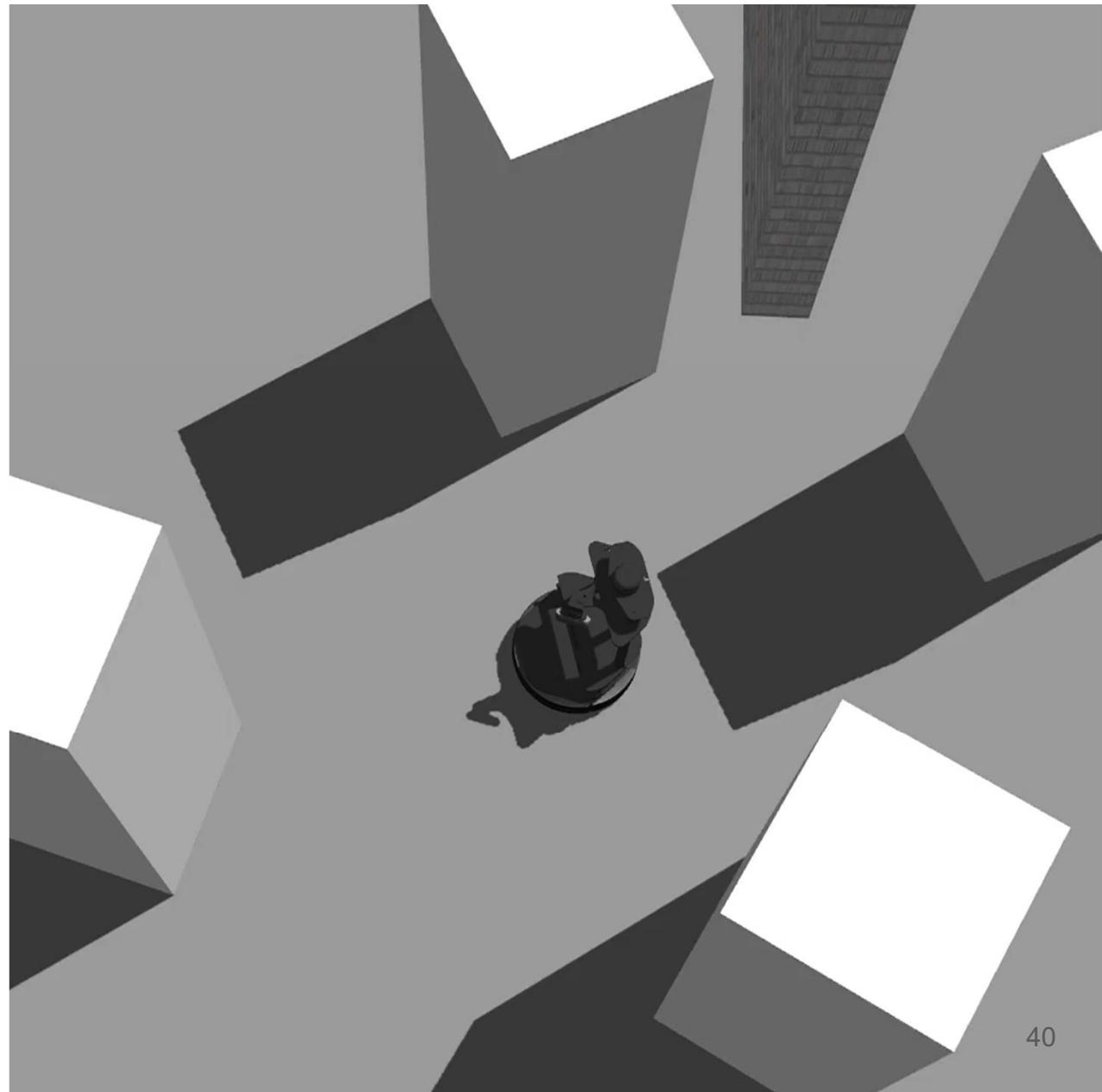


# Erstes Ergebnis

|                                     |              |
|-------------------------------------|--------------|
| Zeit (Durchschnitt)                 | 2:30         |
| #Minimale Lidar Entfernung erreicht | 60% (75/126) |
| #Crashes (Roboter umgefallen)       | 4% (5/126)   |

## CRASHES

## Warum?



# Menschen oder Roboter verantwortlich?

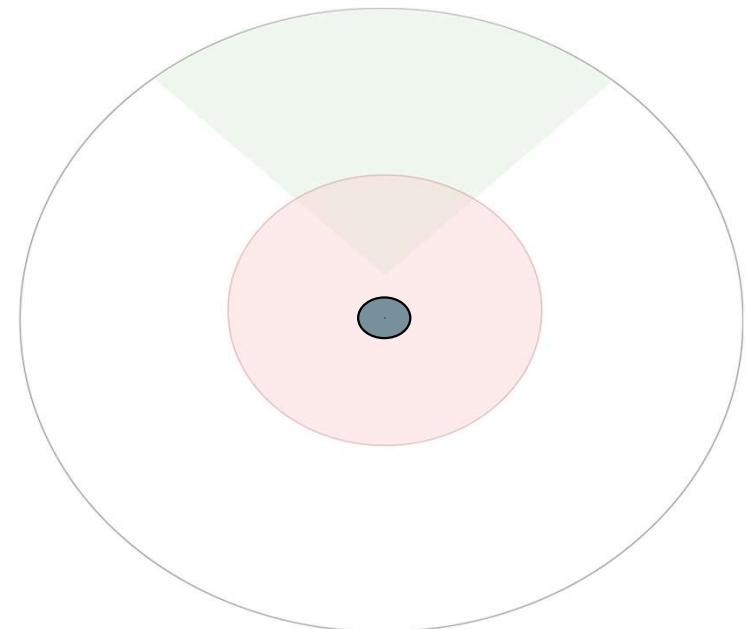
Test: Notstopp → Roboter kann Menschen nicht mehr aktiv anstoßen

|  | Original     | Notstopp    |
|--|--------------|-------------|
| <b>Zeit (Durchschnitt)</b>                 | 2:30         | 2:35        |
| <b>#Minimale Lidar Entfernung erreicht</b> | 60% (75/126) | 94% (47/50) |
| <b>#Crashes (Roboter umgefallen)</b>       | 4% (5/126)   | 20% (10/50) |

→ Menschen verantwortlich

# Funktioniert das Prediction Model?

Test: Veränderung der Lidar Reichweite  
→ mehr/weniger „Verantwortung“  
für Prediction Model



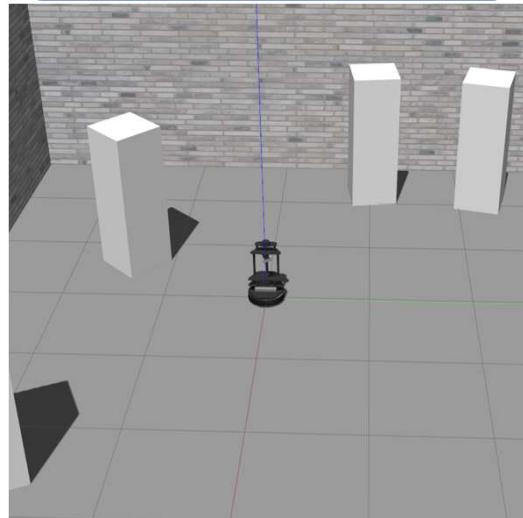
|                               | Original<br>(Lidar range 2) | Lidar range 1 | Lidar range 3 |
|-------------------------------|-----------------------------|---------------|---------------|
| #Crashes (Roboter umgefallen) | 4% (5/126)                  | 2.6% (3/117)  | 5% (6/122)    |

# Zusammenfassung

- Funktionierende Simulationsumgebung  
→ Automatische Datensammlung
- Ergebnisse vielversprechend

# Vorgehen

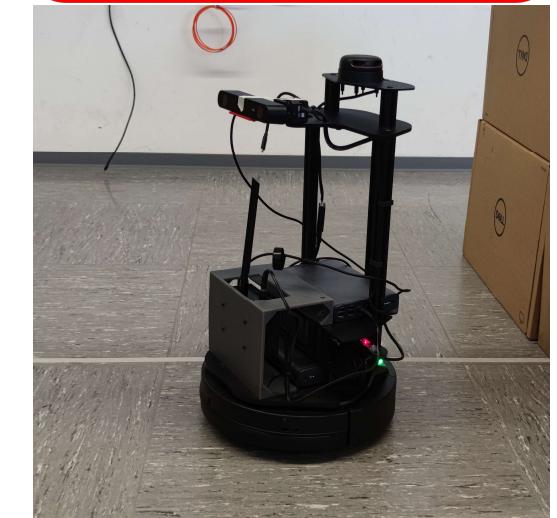
Simulation  
aufbauen



Daten  
sammeln

```
Start_pos: [26.67186895310854, 26.97870277638416] Goal_pos: [0 0]
59614983244133, 25.3010550833549276], [22.884399771552765, 25.24
716078, 19.1553887399417], [20.250185405900563, 19.3112319266;
547, 14.449456501686727], [16.26577205332588, 14.2554405823061;
68781623], [10.98688575944926, 8.940342882350329], [10.76700963;
147201635466], [3.5053612941244334, 7.354413925308031], [3.2307;
1.04800524188640567, 1.6712578820081847], [0.3807651194467038
Start_pos: [0.04674959862981569, 0.33061933646842795] Goal_pos: [
94, 2.767121252302877], [1.7280444295961936, 3.03270051356627;
67480152775969, 5.733087741268227], [8.052383780738385, 5.9612;
022466], [12.89551193335808, 11.159647066132004], [12.96610814;
395], [15.267144749094019, 17.261535791860396], [15.48249851697;
492296905, 22.425714335529936], [19.371421555051725, 22.689840
26.52351806284505], [25.54387218656172, 26.61017541062822], [2;
Start_pos: [26.665444499290924, 26.92859552104784] Goal_pos: [0 0];
77], [25.267633858047148, 21.702574556779165], [25.128636676656;
0497907743, 17.30782866675052], [20.086305854549778, 17.512986(
14.225293534689445], [14.256763077046513, 14.259093177163676],
19071529], [10.900916338124498, 8.843528803756936], [10.6761429-
65596078032], [3.7788574012932097, 7.587088615168983], [3.49059-
3577704849, 1.5167397298956293], [0.7613915058570097, 1.187696-
8244, 1.7713538934314723], [3.105259146175492, 1.8063586947582;
90099756, 1.0155067795211663], [-0.06081491123387798, 0.7117594
Start_pos: [-0.04887092164125904, 0.3377781964586669] Goal_pos: [
.7104881709780548], [5.160371879140614, 1.5862010374138946], [5.
```

Echte  
Umgebung



# Änderungen für den echten Roboter

- Zwei Frames:
    - berechnete Robotertrajektorie aus DWA
    - LiDAR Data
- Welt Koordinaten
- interpolierte Position der Menschen
- Roboter-Referenzrahmen

- Eine Lösung:

$$\begin{array}{c} \text{Rotationsmatrix}(R) \quad \text{erforderliche Translation}(T) \\ \left[ \begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{array} \right] * \begin{pmatrix} x_{rf} \\ y_{rf} \\ z_{rf} \\ 1 \end{pmatrix} = \begin{pmatrix} x_{wf} \\ y_{wf} \\ z_{wf} \\ 1 \end{pmatrix} \end{array}$$

R und T berechnet durch Transform Listener in ROS zwischen /map und /locobot/tf\_base\_link

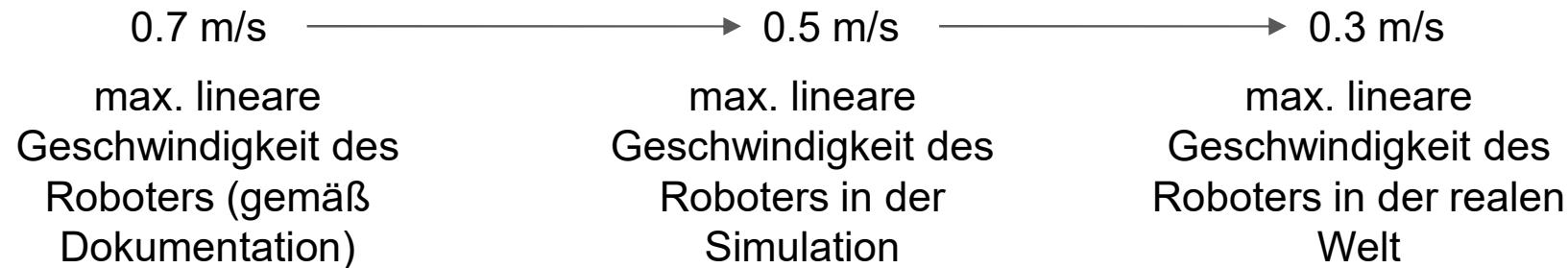
Transformationsmatrix für den Roboter-Referenzrahmen(rf) zum Weltrahmen(wf)

Positionsvektor in Roboter-Referenzrahmen

Positionsvektor in Weltkoordinaten

# Änderungen für den echten Roboter

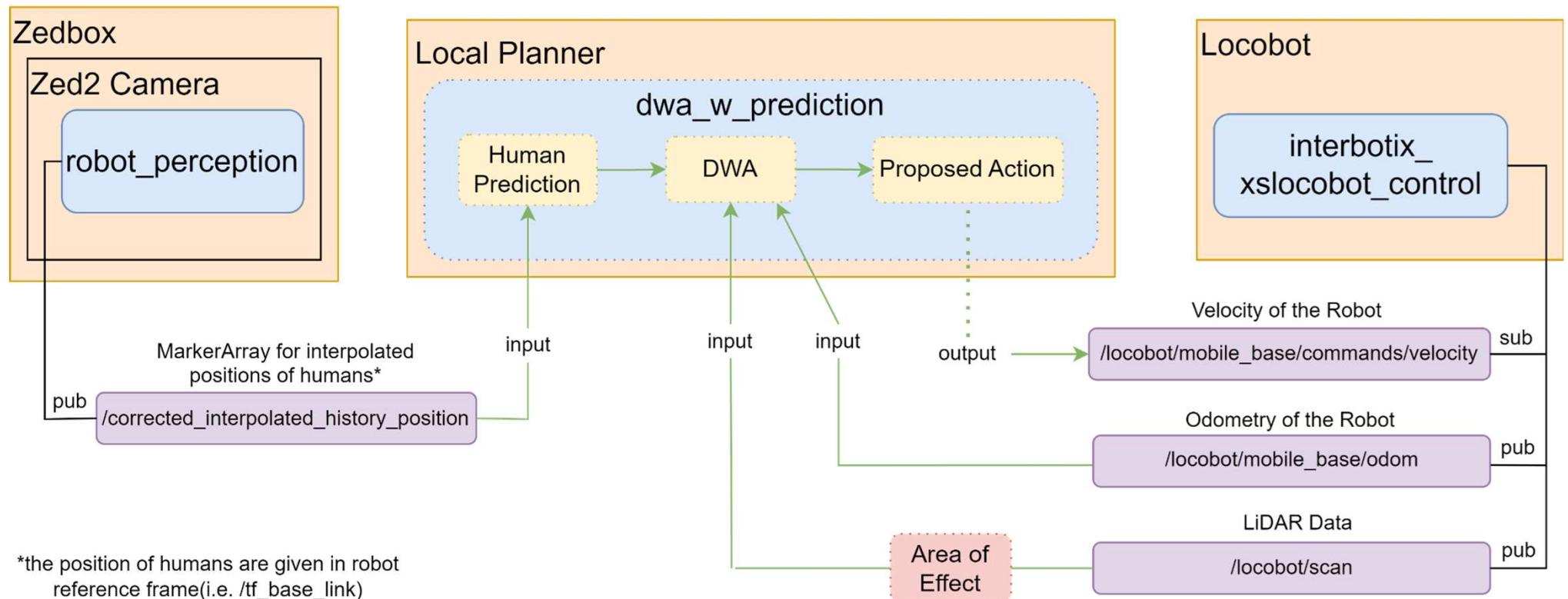
- Anpassen der Maximal Schwierigkeit:

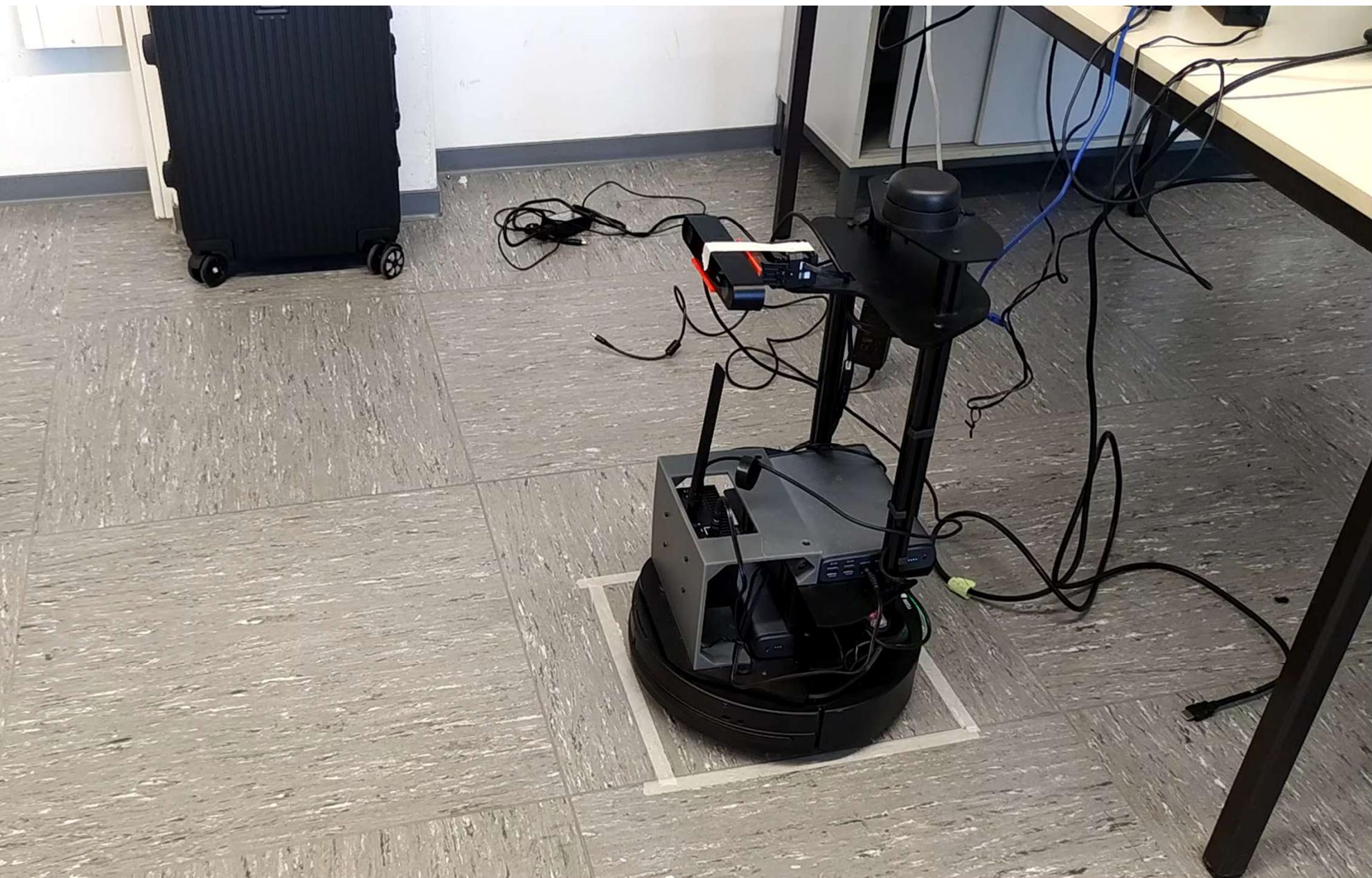


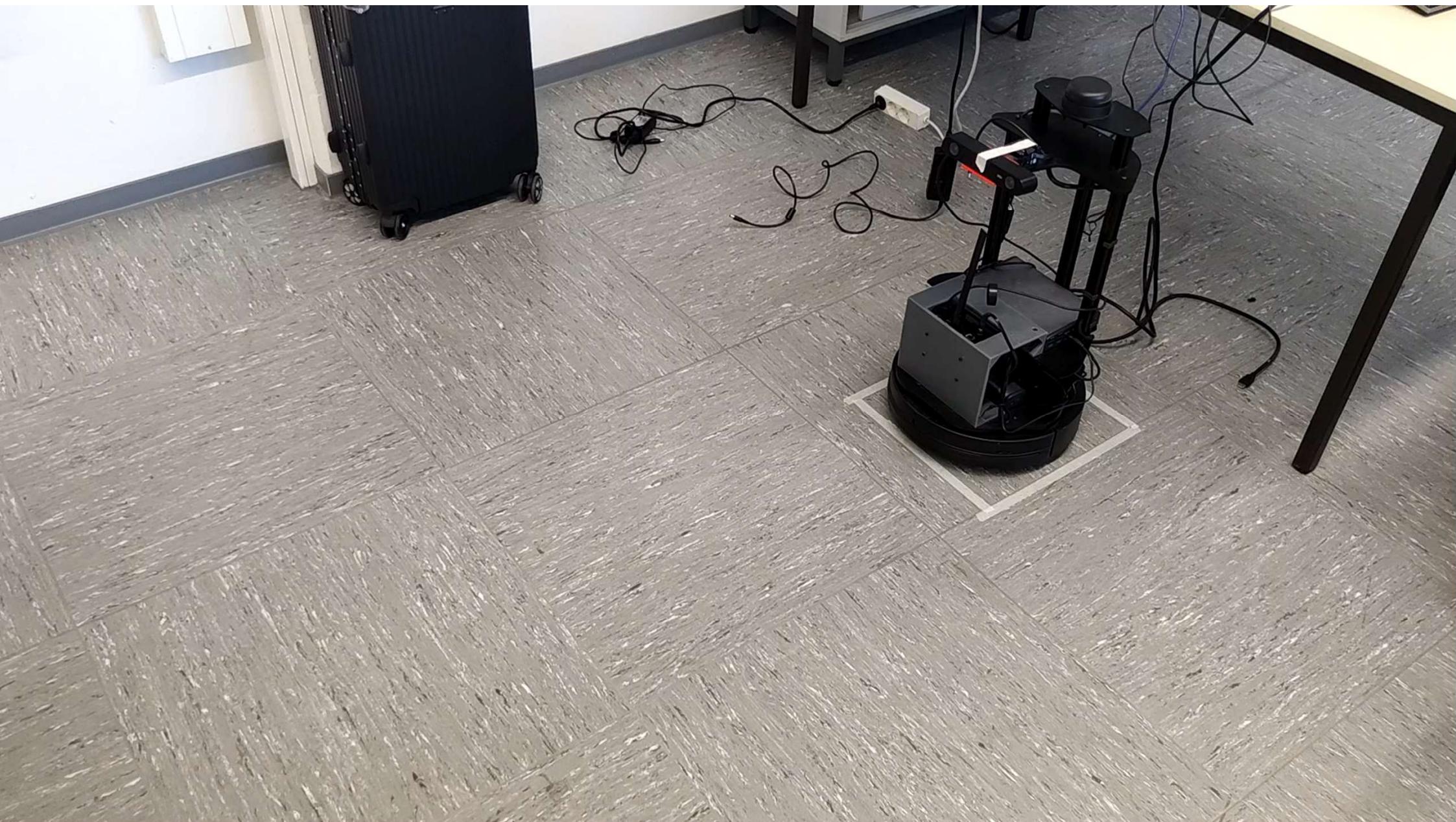
- Anpassen der anderen Parameter:

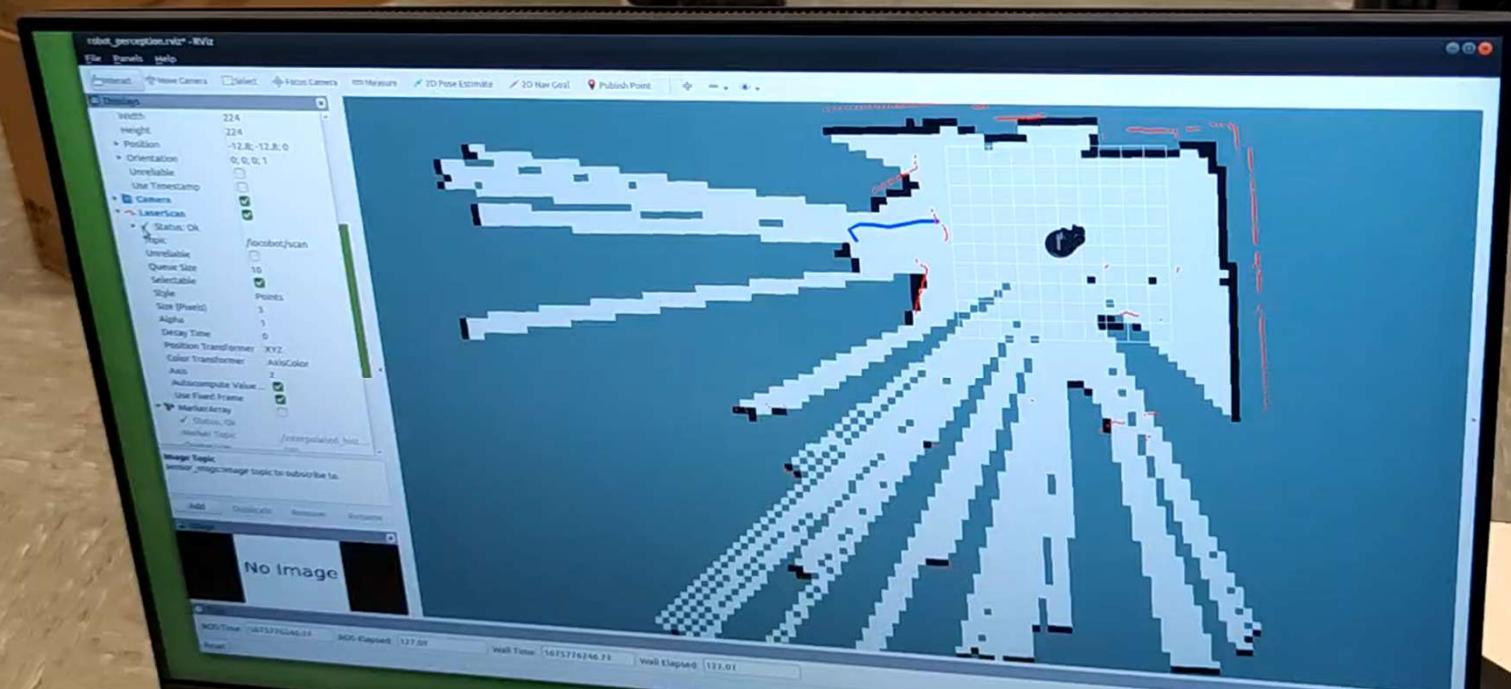
- `to_goal_cost = 0.5`
- `speed_cost_gain = 10`
- `obstacle_cost_gain = 3`
- `pred_time_steps = 4`

# Datenfluss des realen Roboter









# Mapping mit Rviz

- Locobo kann Karte generieren, um sich zu orientieren
- Rviz kann beim Mappen helfen
- Locobot manuell mit Tastatur bewegen → Karte erstellen

# SLAM(Gmapping)

- Package: Gmapping
- Sensor: Lidar, Odometry
- Anwendung: Lokalisierung und Mapping(gleichzeitig)
- Subscribed Topics: tf (tf/tfMessage); scan (sensor\_msgs/LaserScan)
- Published Topics: map\_metadata (nav\_msgs/MapMetaData); map (nav\_msgs/OccupancyGrid)
- Algorithmus: Rao-Blackwellized Particle Filter

# Algorithmus: Rao-Blackwellized Particle Filter(RBPF)

Frage: Was macht der Roboter zuerst?  
Lokalisierung oder Mapping?

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = \\ p(m \mid x_{1:t}, z_{1:t}) \cdot p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$$

$x_{1:t}$ : Trajektorie des Roboters,  $x_{1:t}=x_1, \dots, x_t$

$x_1$ : originaler Zustand des Roboters

m: Map der Umgebung

$z_{1:t}$ : Observation von Lidar

$u_{1:t}$ : Messung von Odometry

$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$  : Berechnung Pose des Roboters(Odometry) und Maps durch Observation und Messung

$p(m \mid x_{1:t}, z_{1:t})$  : Berechnung Maps durch Pose des Roboters(Odometry) und Observation

$p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$  : Berechnung Pose des Roboters(Odometry) durch Observation und Messung

Nachteil: Ein genaue Karte



mehr Teilchen(Particles)

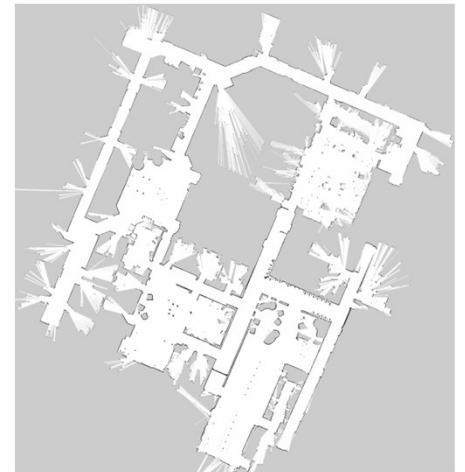


Erhöhte Belastung der Rechnung

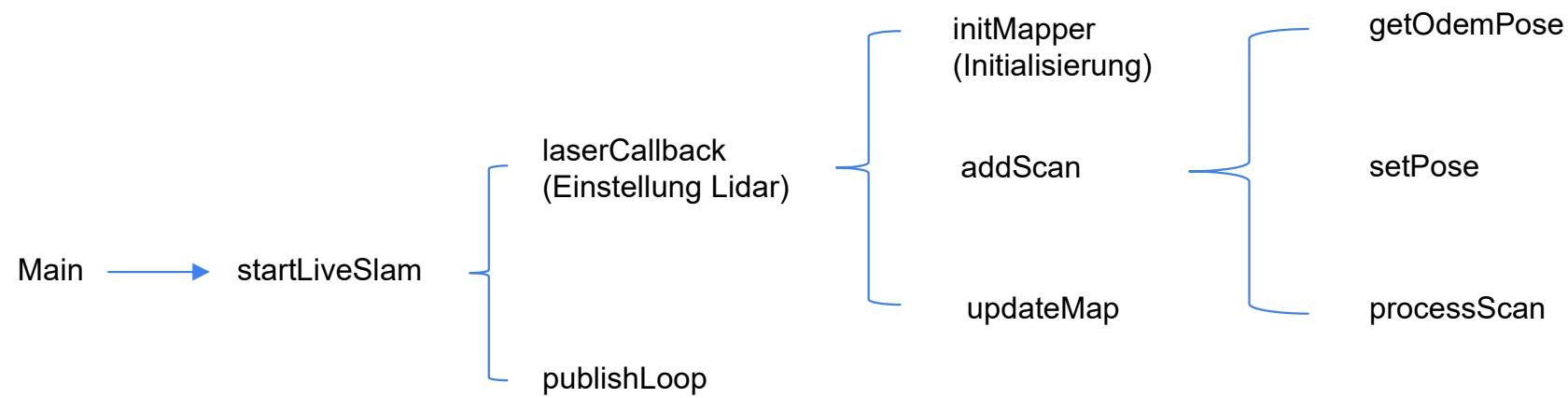
# Prozess des Mappings

1. Sampling: Vorhersage der aktuellen Position(Trajectorie) nach letzter Karte und Bewegungsmodel(Odometry),  
(Teilch:  $x_{1:t}, z_{1:t}$ )
2. Importance Weighting: Vergabe des Gewichts jedes Teilchens
3. Resampling: Sammlung der Teilchen nach ihrem Gewicht
4. Map Estimation: Abschätzung der Karte nach ausgewählten Teilchen( $x_{1:t}, z_{1:t}$ ); Erstellung und Aktualisierung der Karte

zum Beispiel:



# Rahmen Gmapping



# Git Archiv

- Setup Guides
- 3 Branches:
  - Simulationscode
  - Code für echten Roboter (ohne Prediction Model)
  - Code für echten Roboter (mit Prediction Model)

The screenshot shows a GitHub repository interface. At the top, there are buttons for 'Go to file', 'Add file', and 'Code'. Below that, it says 'This branch is 13 commits ahead of main.' On the right, there's a 'Contribute' button. The commit list shows 14 commits from 'janwes' made yesterday. The commits are:

- Update README.md
- Update pedsim\_dwa.py
- Update README.md
- Update README.md
- Create requirements.txt

Below the commits is the 'README.md' file content:

```
Masterprojekt-Intelligente-Systeme

This is the final version of the code that was used for the simulation

Steps for setup:
Operating system: Ubuntu 20.04
For prediction model: compatible GPU with CUDA drivers
```

Vielen Dank für  
Ihre Aufmerksamkeit

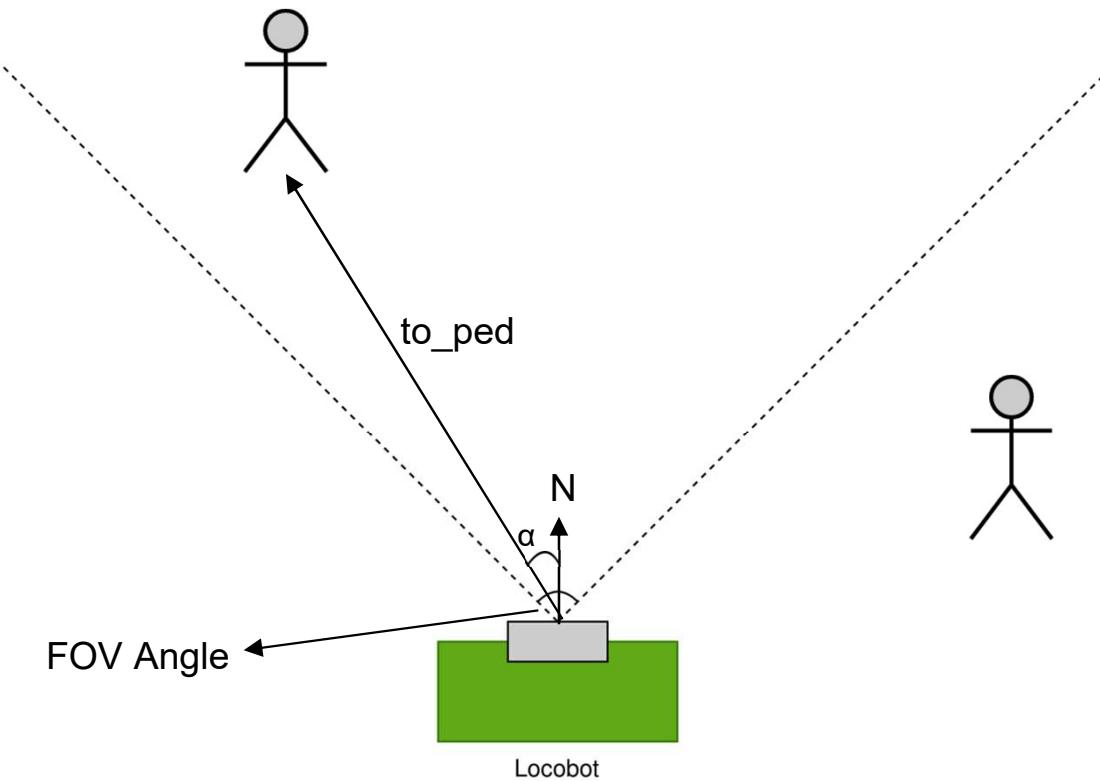
# Bonus Folien

# Aktuelle Zustand von Realer Robot Code

- Es gibt ein ROS Packet für DWA+LiDAR+Perception+Prediction
- Dieser Packet hat derzeit eine große Verzögerung (Sekunden) und funktioniert nicht perfekt.
  - ~ 20-50ms für DWA
  - ~ 20-40ms für Perception
  - Totale Zeit für einen Zyklus (berechnet im Code): ~40-100ms
  - Reaktionszeit des Roboters in Echtzeit: 3-4 Sekunden



# Kamera(in Simulation) durch FOV-Filter



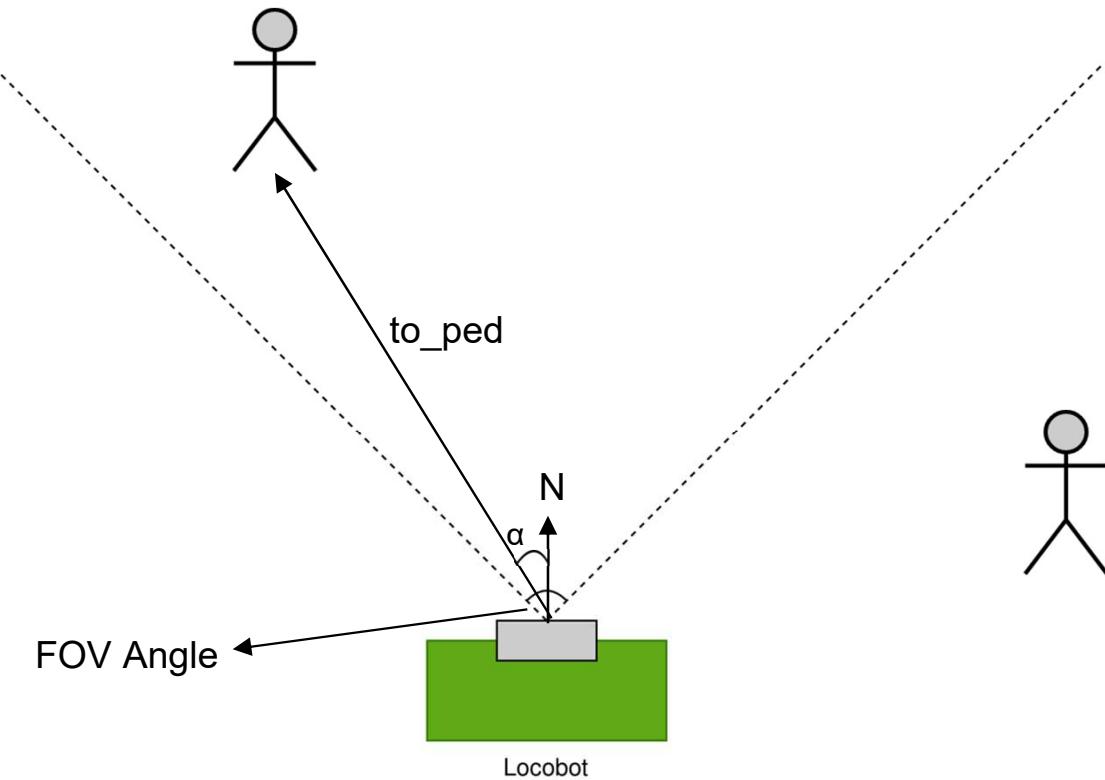
- known: position of the robot (`r_pos`) and pedestrians (`obs`)
- given: field of view (`fov`)

Pseudo Code:

```
N = unit_vector_at(r_pos, theta)  
for p_pos in obs:  
    to_ped = r_pos - ob_pos  
    dot_product = to_ped * N  
    alpha = arcos(dot_product)  
    if alpha <= fov / 2:  
        obs_in_fov.append(p_pos)
```

<sub>61</sub>

# Kamera(in Simulation) durch FOV-Filter



Limitation:

- 2D Coordinate System
- (so far) no Occlusion

Possible Improvement:

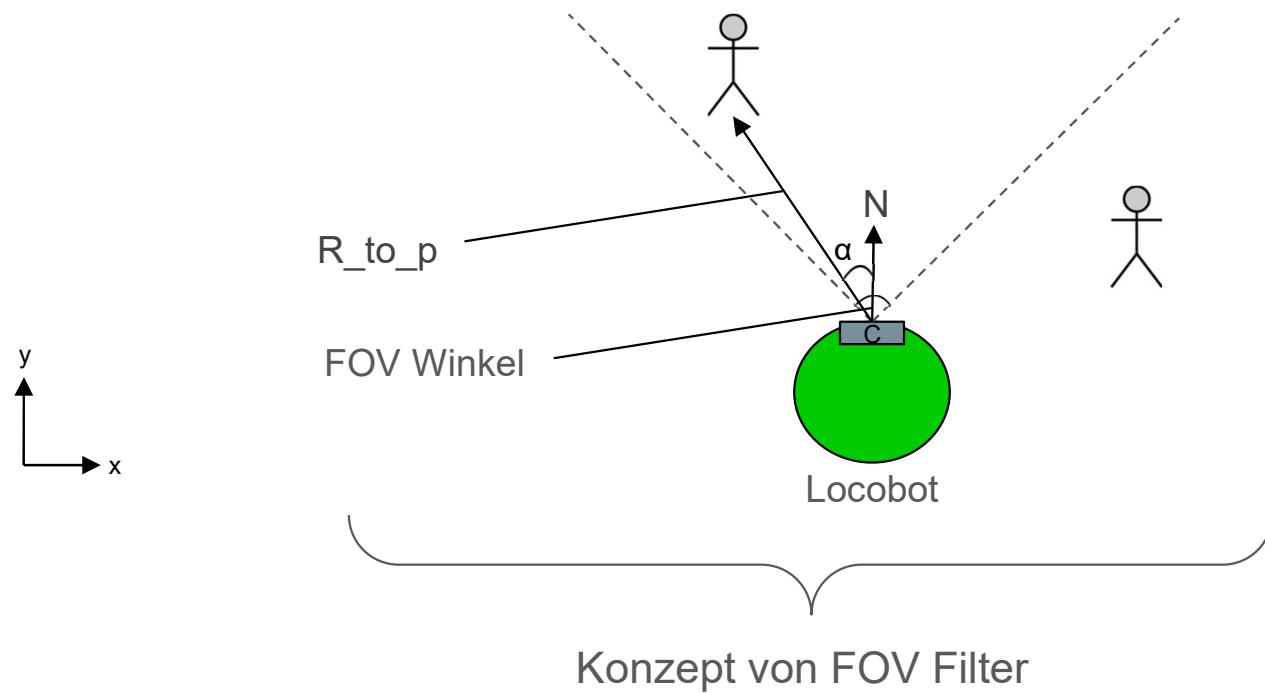
- Cater to Occlusion

# From here onwards Final Presentation Slides

- the following are my(Utku's) slides for the final presentation
- these cover the following topics:
  - FOV Filter
  - Prediction Model
  - Dataflow in Simulation(copied for reference only)/Real Robot
  - Changes made for the real robot

# Erkennung von Menschen(Kamera)

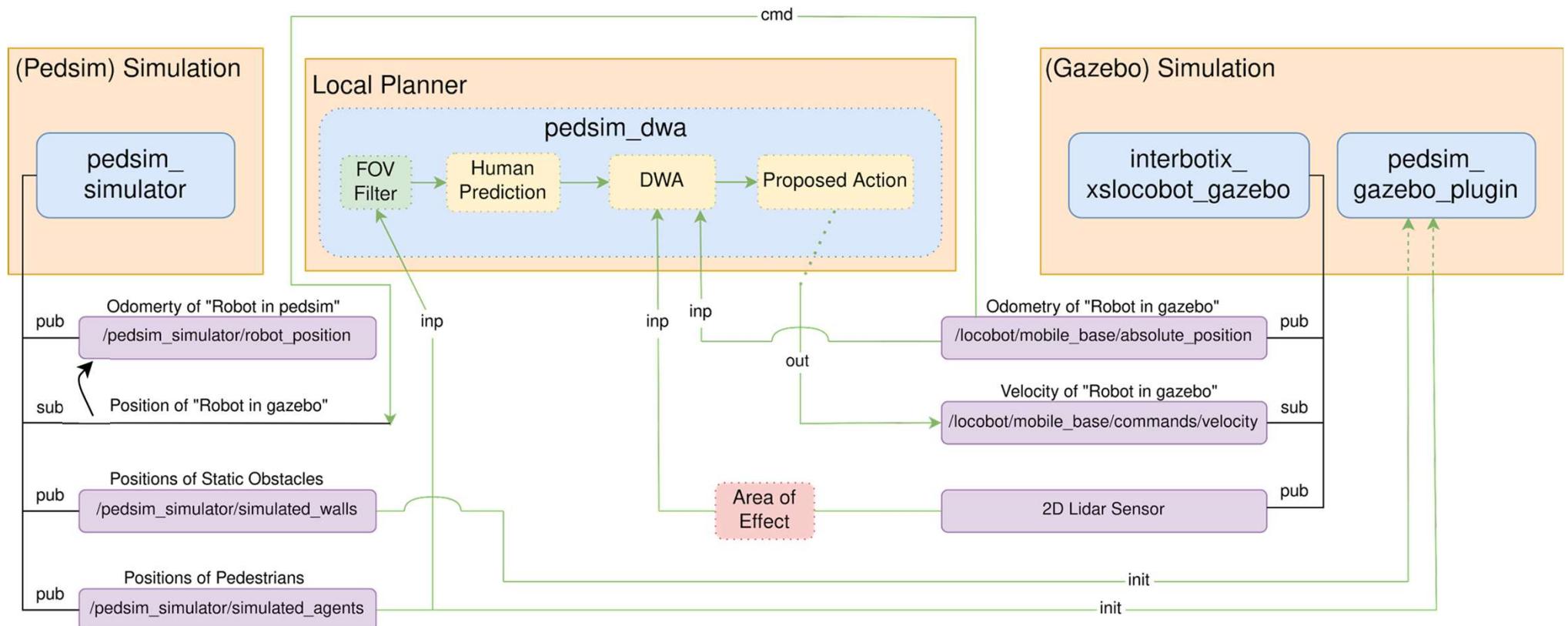
- Simulation: FOV(Field-of-View) Filter



$\alpha$ -Calculation:  
Skalar-Produkt & arccos

Überprüfen:  
 $\alpha \leq \text{FOV Winkel} / 2$

# Datenfluss in der Simulation



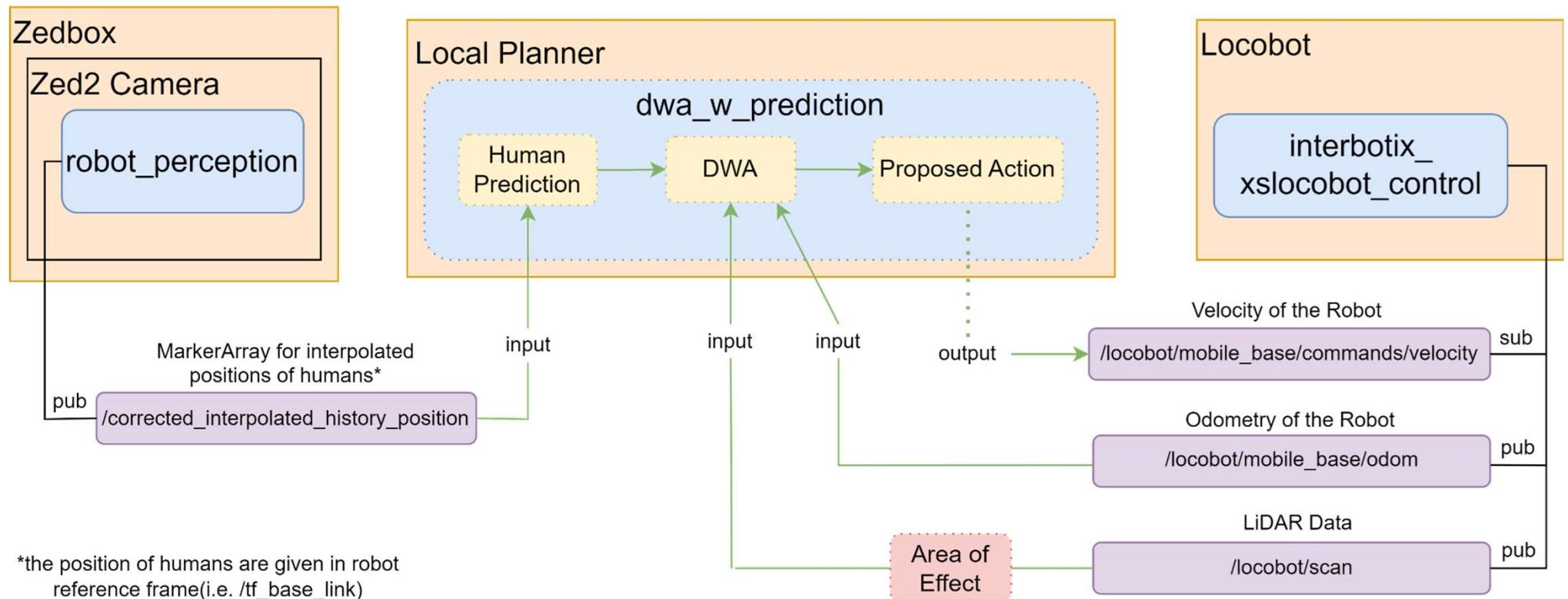
\* **init:** Datenpfad für die Initialisierung der Gazebo-Welt

\* **cmd:** aktuelle (absolute) Position des Roboters im Gazebo, die an Pedsim gesendet wird

\* **inp:** Eingabe von Daten in den lokalen Planer

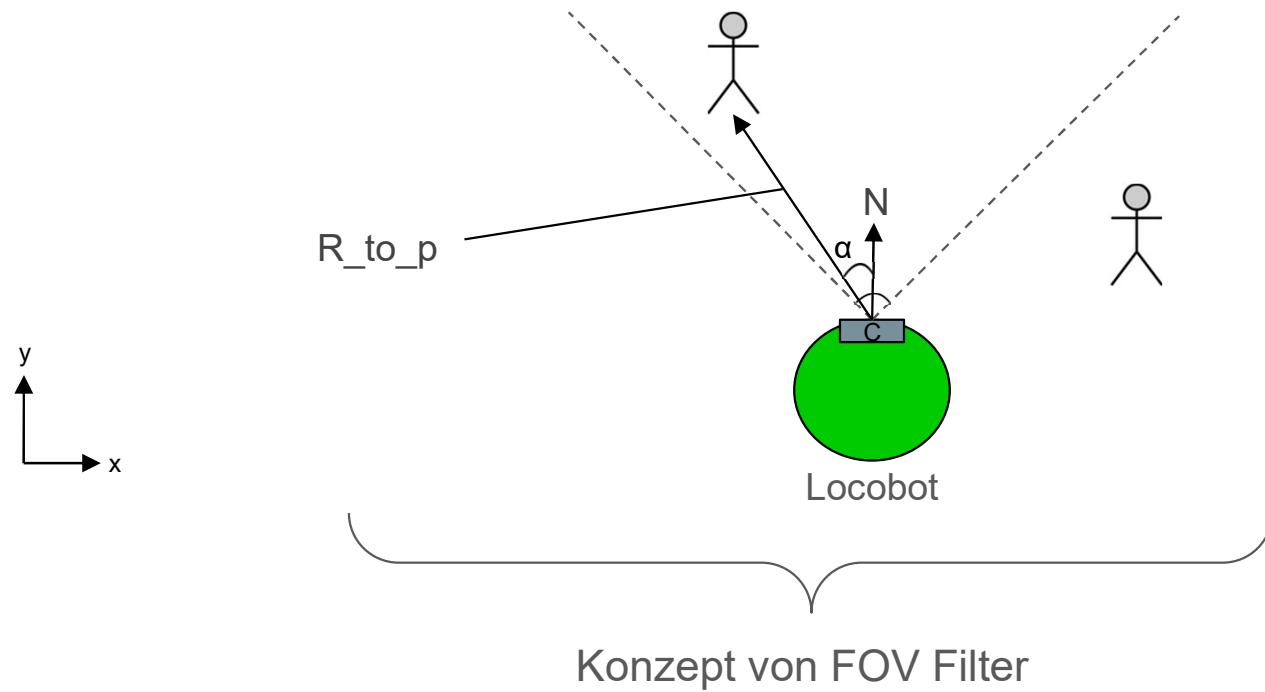
\* **out:** Ausgabe der lokalen Planer

# Datenfluss des realen Roboter



# Erkennung von Menschen(Kamera)

- Simulation: FOV(Field-of-View) Filter



$\alpha$ -Calculation:  
Skalar-Produkt & arccos

Überprüfen:  
 $\alpha \leq \text{FOV Winkel} / 2$

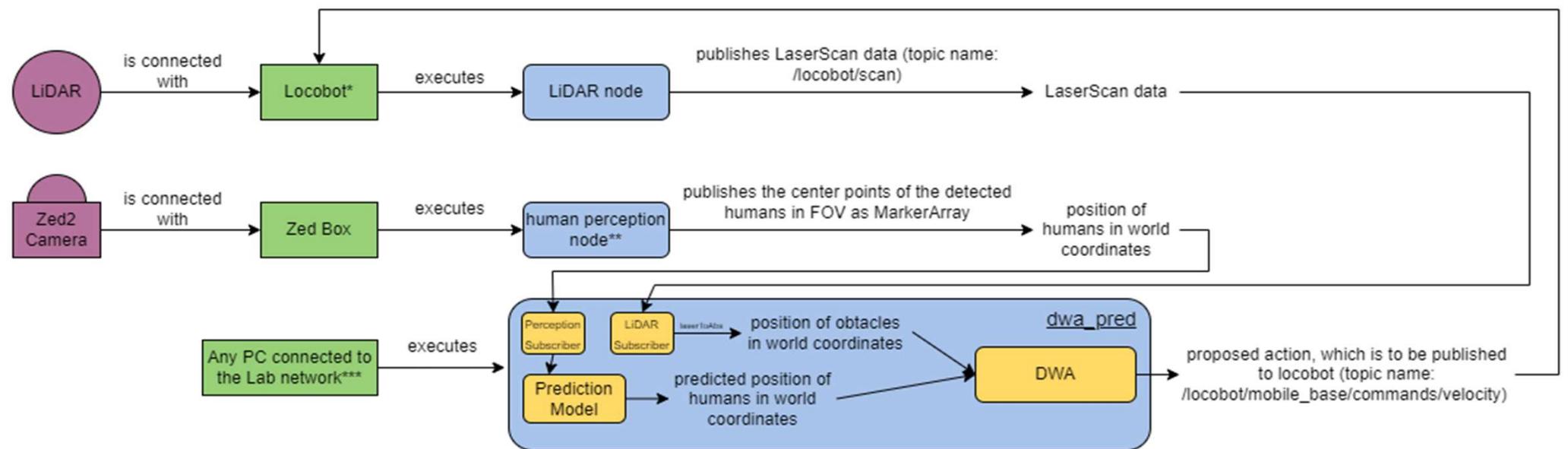
# Example Odometry Message

```
---
```

```
header:  
  seq: 6372  
  stamp:  
    secs: 1437847480  
    nsecs: 632620096  
  frame_id: odom  
child_frame_id: base_link  
pose:  
  pose:  
    position:  
      x: 0.00577289803386  
      y: 1.55462132978e-06  
      z: 0.0  
    orientation:  
      x: 0.0  
      y: 0.0  
      z: 0.000444339384108  
      w: 0.999999901281  
  covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  
  , 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  
.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  
0.0, 0.0, 0.0, 0.0]  
twist:  
  twist:  
    linear:  
      x: 0.0  
      y: 0.0  
      z: 0.0  
    angular:  
      x: 0.0  
      y: 0.0  
      z: 0.0  
    covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  
  , 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  
.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  
0.0, 0.0, 0.0, 0.0]
```

```
--
```

# Dataflow in Real Robot-2



\*The ROS Master is prob. best run from the Locobot

\*\*technically called "robot perception", since the camera is mounted on top of the locobot

\*\*\*possibly one of the computers at the lab with a GPU(instead of a personal laptop), since the PC will have to run the Prediction Model which is an AI model

