RoboCode

Funktionalitäten des Roboters

Bewegung des Roboters

Christian Riess

Eva Eibenberger

Pattern Recognition Lab (Computer Science Dep. 5)

Friedrich-Alexander-University Erlangen-Nuremberg

Beispiel-Überlegungen

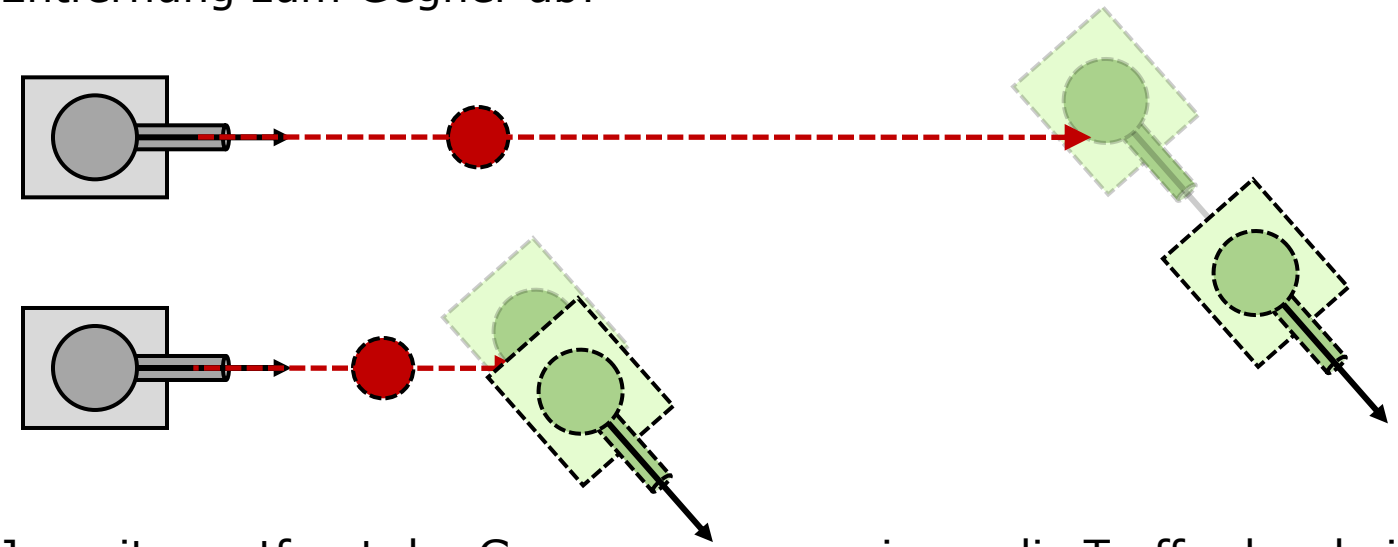
Überlegungen: Wie verhält sich der Roboter am besten wenn…

- … er sich ganz allgemein fortbewegt?
  - Wie schaut eine gute Bewegungsbahn aus?
    Stehen bleiben oder stetig bewegen?
  - Wie werden Radar und Kanone ausgerichtet?
- … ein Gegner gefunden wurde?
  - Feuert er direkt eine Kugel ab?
  - In welche Richtung feuert er den Schuss am besten?
  - Mit welcher Schussenergie sollte er schießen?
  - Soll Radarbewegung eingeschränkt werden, um Gegner zu folgen?
  - Soll die Bewegungsrichtung des Roboters angepasst werden?
- … er von einer gegnerischen Kugel getroffen wurde?
  - In welche Richtung flüchtet er am besten?
    Einfach nach hinten? Senkrecht zur Kugelbahn?
- … er einen Gegner mit einer Kugel trifft?
  - Soll die Radarbewegung eingeschränkt werden?
- … es zu einer Kollision mit der Wand kommt?
  - In welche Richtung fährt er am besten weiter?
  - Kann die Kollision generell vermieden werden?

Ein paar Details auf den folgenden Folien zu:

Beispiel: Bewegungsstrategie 1

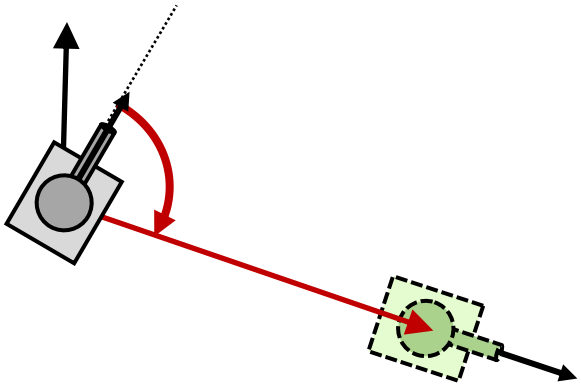
- Bewegung wenn ein Gegner gefunden wurde
- Problem:
  - Bisher wird einfach geschossen, wenn der Gegner gefunden wurde.
  - Die Wahrscheinlichkeit den Gegner zu treffen hängt aber von der Entfernung zum Gegner ab!

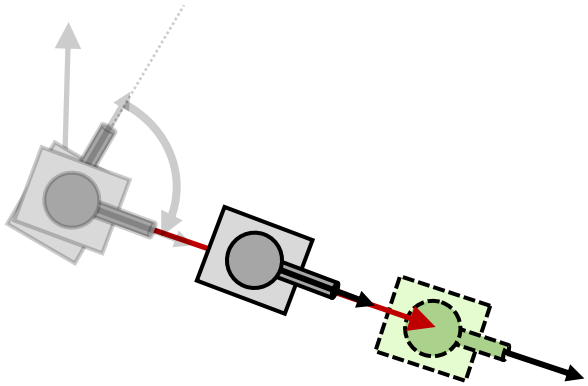


  - Je weiter entfernt der Gegner, um so geringer die Treffwahrscheinlichkeit, wenn sich der Gegner bewegt
- Ziel:
  - Die Treffwahrscheinlichkeit beim Schießen erhöhen!

Beispiel: Bewegungsstrategie 1

- Ziel: Die Treffwahrscheinlichkeit beim Schießen erhöhen!
- Idee:
  - Ansatz: Auf Gegner zu gehen, ihn verfolgen, bzw. bestimmten Abstand zu ihm halten
- Bewegungsablauf:
  - Auf Gegner zu drehen – Welcher Winkel?
  - Auf Gegner zu gehen – Welche Distanz?

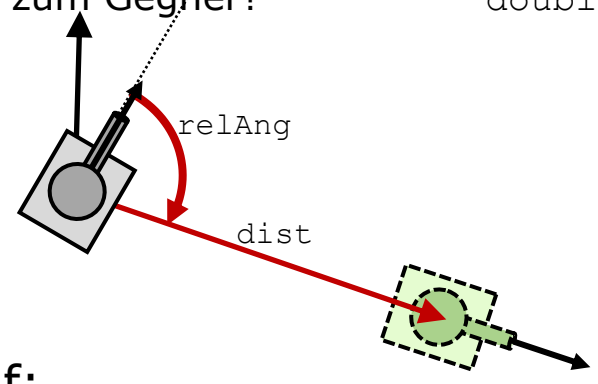




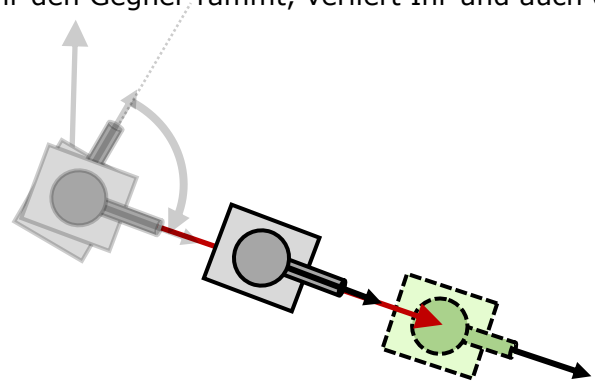
- Überlegungen:
  - Wie bekommt man rel. Winkel zum Gegner? Wie den Abstand zum Gegner?
  - Wann soll der Roboter schießen?

Beispiel: Bewegungsstrategie 1

- Überlegungen:
  - Wie bekommt man Winkel zum Gegner? `double relAng = e.getBearing();`
    Wie den Abstand zum Gegner? `double dist = e.getDistance();`



- Bewegungsablauf:
  - Auf Gegner zu drehen – Welcher Winkel? → `relAng`
  - Auf Gegner zu gehen – Welche Distanz?
    - Hier seid Ihr flexibel und könnt verschiedene Distanzen ausprobieren.
    - Vorsicht: Wenn Ihr den Gegner rammt, verliert Ihr und auch der Gegner Energie!



Beispiel: Bewegungsstrategie 1

- Überlegungen:
  - Wann sollte der Roboter schießen?
    - Der Roboter sollte nur dann schießen, wenn er möglichst nahe am Gegner ist.
    - Auch hier können beliebige Szenarios ausprobiert werden!
  - Beispiel 1: Zulaufen und schießen
  - Beispiel 2: Zulaufen oder schießen
- Übrigens:
  - Indem der Roboter dem Gegner hinterher fährt, wird i.d.R. sehr gut vermieden, dass der Roboter die Wand rammt (und somit zusätzliche Energie verliert).
  - Wer die Wand allerdings extra berücksichtigen möchte, kann eine Blick auf Folien 8 bis Ende werfen – nur für Leidensfähige ☺

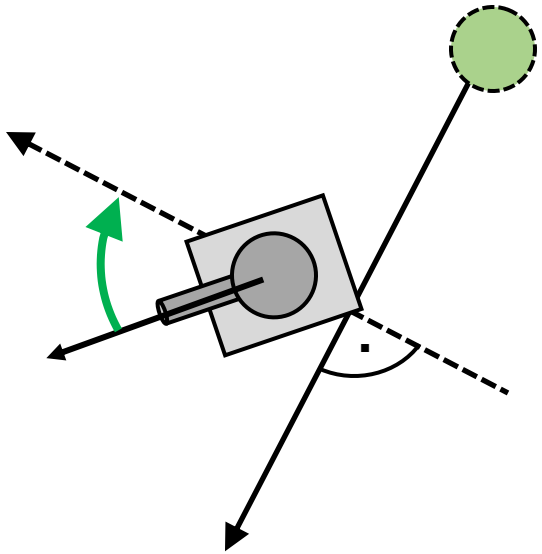
```
…
public void run() {
   while(true) {
      // hier ist keine Bewegung nötig
      setTurnRadarRight(360);
      waitFor(new TurnCompleteCondition(this));
   }
}
```

```
public void onScannedRobot(ScannedRobotEvent e) {
   double enDist = e.getDistance();
   double enAng = e.getBearing();

   setTurnRight(eAng);      // auf Gegner zu drehen
   setAhead(enDist - 100); // 100 pix von Gegner wegbleiben
   waitFor(new TurnCompleteCondition(this));

   setFire(2.0);  // schießen
   execute();
}
```

```
public void onScannedRobot(ScannedRobotEvent e) {
   double enDist = e.getDistance();
   double enAng = e.getBearing();

   if (endDist < 200) { // nahe beim Gegner?
      setFire(2.0);     // schießen
      execute();
   } else { // andernfalls
      setTurnRight(eAng);      // auf Gegner zu drehen
      setAhead(enDist - 100); // 100 pix von Gegner wegbleiben
      waitFor(new TurnCompleteCondition(this));
   }
}
```

Beispiel: Bewegungstrategie 2

- Bewegung nachdem der eigene Roboter getroffen wurde
- Idee:
  - Der Roboter sollte sich umgehend möglichst weit von der Schussbahn der Kugel entfernen



  - Der größte Abstand wird dadurch erreicht, dass der Roboter sich senkrecht zur Kugelbewegung bewegt!
  - Suche in der RoboCode API nach dem Ereignis `HitByBulletEvent`.
    Hier findest du heraus, mit welcher Methode die Orientierung der Kugel ermittelt werden kann.

Beispiel: Bewegungsstrategie 3

Die übrigen Folien sind nur für Leute,

die viel Zeit und

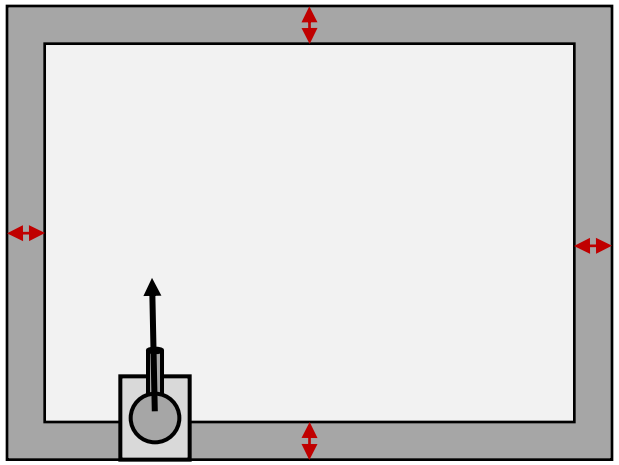
Leidensfähigkeit mitbringen ☺

Alle übrigen wenden sich am Besten

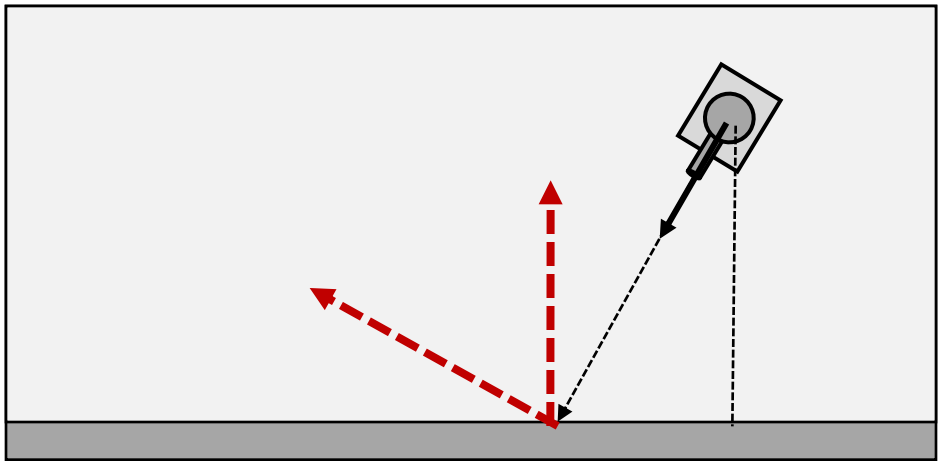
einem anderen Foliensatz zu!

Beispiel: Bewegungsstrategie 3

- Kollisionen mit der Wand sind schlecht
  - Sie kosten Energie
  - Der Roboter bewegt sich kurz nicht und ist somit leichtes Ziel für Gegner
- Das Spielfeld hat 4 Wände
  - In der Regel: gesonderte Betrachtung alle vier Wände
- Achtung:
  - Der Körper des Roboters benötigt Platz.
  - Daher: Um Kollision zu vermeiden sollte der Roboter nicht näher als 18 Pixel an die Wand kommen!

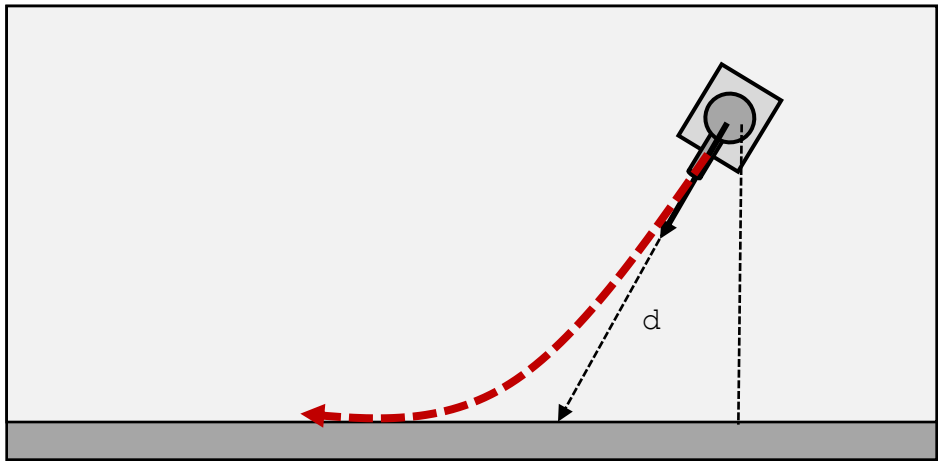


Beispiel: Bewegungsstrategie 3(a)



- Kollisionsvermeidung
  - Um Energieverlust zu vermeiden
- Idee:
  - Wann:
    Wenn Roboter nur noch 18 Pixel von der Wand entfernt ist
  - Was:
    Ändere Bewegungsrichtung (z.B. senkrecht von der Wand weg oder in einem anderen Winkel)
- Überlegungen:
  - Man untersucht am besten alle 4 Wände getrennt.
    Zudem auch Annähern an die Wand von links bzw. von rechts.
  - Hilfreiche Methoden:
    `getHeading()`, `getX()`, `getY()`, `battleFieldWidth()`, `battleFieldHeight()`
  - Zu ermitteln:
    - Mit welchem Winkel nähert sich Roboter der Wand an?
    - Wie weit muss er sich drehen, um im gewünschten Winkel weiter zu fahren?

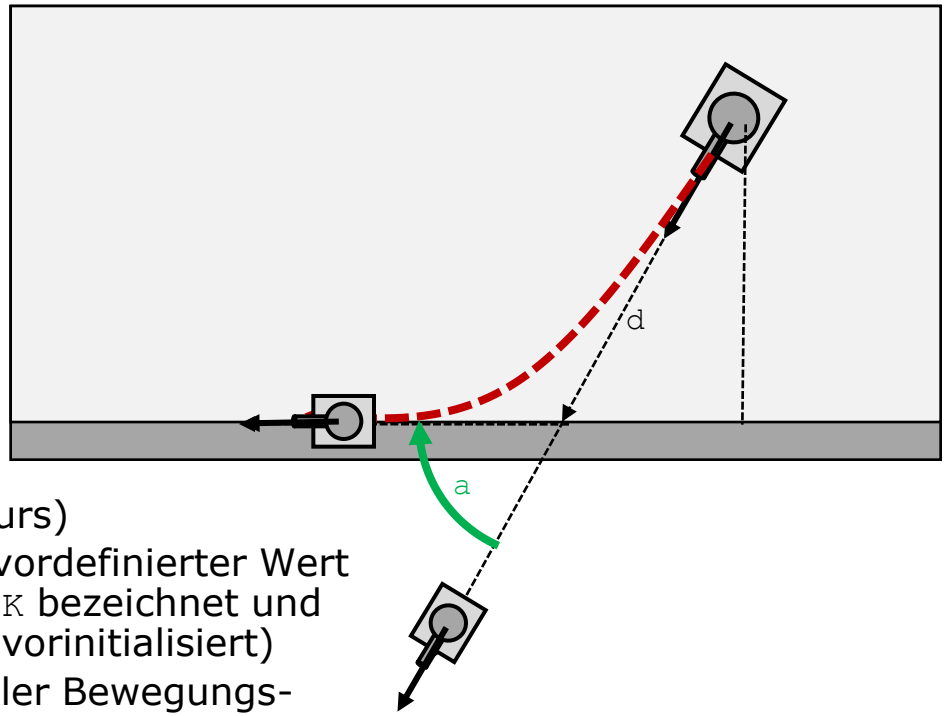
Beispiel: Bewegungsstrategie 3(b)



- „Wall-Smoothing“
  - Kollision mit Wand und „Still-Stehen“ zu vermeiden
  - Besser geeignet für AdvancedRobots, kann aber auch mit Robots gemacht werden
- Idee:
  - Wann:
    Wenn Fahrtweg zu einer Wand kleiner als bestimmter Wert ist
  - Was:
    Die Bewegungsbahn soll dann in einer Kurve verlaufen, die sich quasi an die Wand anschmiegt
- Informationen:
  - Es wird der Abstand `d` vom Roboter in Fahrtrichtung zur Wand benötigt.
    Dieser wird am besten in einer separaten Methode ermittelt. Dabei kann auch gleich der Abstand nach hinten zur nächsten Wand ermittelt werden, da dieser auch oft hilfreich sein kann (siehe Folie 29 für ein paar mehr Details)

Beispiel: Bewegungsstrategie 3(b)

- „Wall-Smoothing“
  - Wir zeigen hier eine simple Strategie, die nur vom Abstand `d` und dem Auftreffwinkel des Roboters abhängt
  - Es gibt kompliziertere Ansätze, die z.B. auch die Position des Gegners berücksichtigen



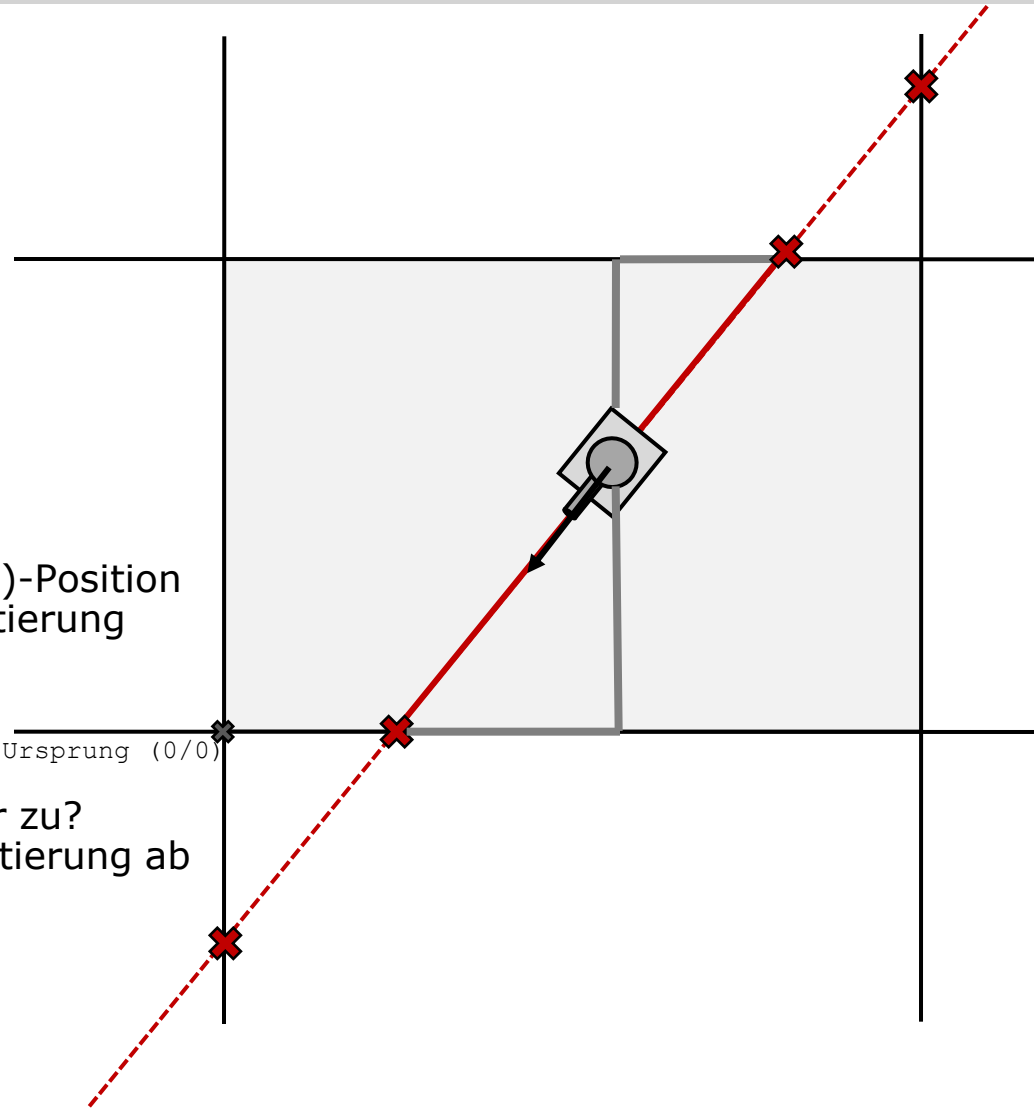
Vorgehen:

- Berechne Abstand `d` auf der Bewegungsbahn zur Wand (siehe Exkurs)
- Falls `d` kleiner ist als ein bestimmter, vordefinierter Wert (diese Variable wird oft als `WALL_STICK` bezeichnet und auf einen Wert zwischen `120` und `160` vorinitialisiert)
- Ermittle den Winkel `a` zwischen aktueller Bewegungs-richtung und Zielrichtung
- Nun implementiere eine While-Schleife:
  Solange der Abstand zur Wand noch nicht kleiner 18 Pixel ist:
  Drehe Roboter noch ein wenig in Richtung Zielrichtung

Wichtig: Separate Behandlung aller 4 Wände

```
double a = ...;
double distanceToWall = ...;
while(distaneToWall > 18) {
   setTurnRight(a*0.1)
}
```

Beispiel: Bewegungsstrategie 3 – Exkurs



- Berechnung des Abstandes nach vorne und hinten zur nächsten Wand
- Ansatz:
  - Wichtig ist hierbei die aktuelle (x,y)-Position des Roboters und seine abs. Orientierung
  - Viele Fallunterscheidungen nötig.
  - Auf welche Wand fährt der Roboter zu? Dies hängt nicht nur von der Orientierung ab sondern auch von seiner Position!