RoboCode

Funktionalitäten des Roboters

Radarbewegungen

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-Strategie 2

- Bewegung des Radars und der Kanone
  - Szenario: Der Roboter bewegt sich und sucht gegnerischen Roboter
  - Fragen:
    - Wie sollte der Radar bewegt werden?
    - Wie und wann sollte die Kanone bewegt werden?
  - Hintergrund:
    - Drehen des Radar: schnell (45° pro Zeiteinheit)
    - Drehen der Kanone: langsam (20° pro Zeiteinheit)
    - Drehen des Körpers: extrem langsam
  - Idee:
    - Mit Radar stetig im Kreis scannen
    - Falls Gegner gesichtet wurde, auf kürzestem Weg Kanone auf Gegner richten und schießen

Beispiel-Strategie

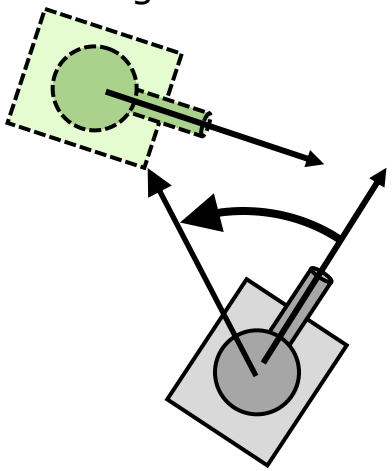
- Bewegung des Radars und der Kanone
  - Szenario: Der Roboter bewegt sich und sucht gegnerischen Roboter
  - Idee: Mit Radar stetig im Kreis scannen, bei Sichten des Gegners Kanone auf kürzesten Weg drehen
  - Vorbereitung:
    - Sicherstellen, dass sich Radar, Kanone und Körper unabhängig voneinander drehen
    - Radar sollte sich immer drehen
  - Fragen:
    - Wann sollte die Kanone bewegt werden?
    - In welche Richtung muss die Kanone gedreht werden?

```
…
public void run() {

   // unabh. Kanonen, Radar- und Körperbewegung
   setAdustGunForRobotTurn(true);
   setAdustRadarForRobotTurn(true);
   setAdustRadarForGunTurn(true);

   while(true) {
      setTurnRadarRight(360);
      setTurnRight(20);
      setAhead(30);
      waitFor(new TurnCompleteCondition(this));
   }
}
…
public void onScannedRobot(ScannedRobotEvent e) {
   setFire(1);
   execute();
}
…
```

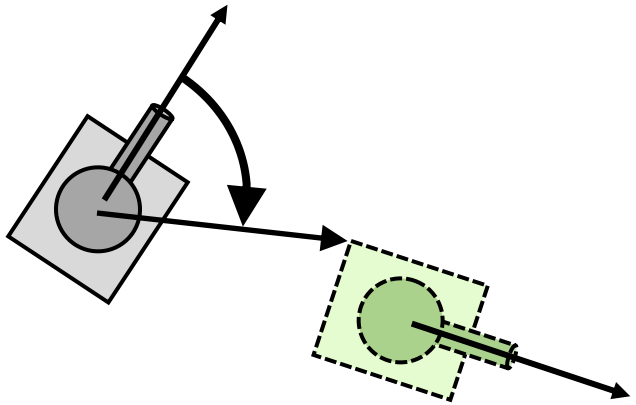
Beispiel-Strategie

- Bewegung des Radars und der Kanone
  - Idee:
    - Um schneller einen Gegner zu finden: Rotieren des Scanners anstelle der Kanone
    - Wenn Gegner gefunden wurde: Auf kürzestem Weg Kanone auf Gegner richten
  - Überlegungen:
    - In welche Methode gehört das Drehen des Scanners?
      Wohin das Ausrichten der Kanone, wenn Gegner gefunden wurde?
    - Wie soll sich die Kanone drehen?

Drehen nach **links**,
wenn Gegner links von Kanone



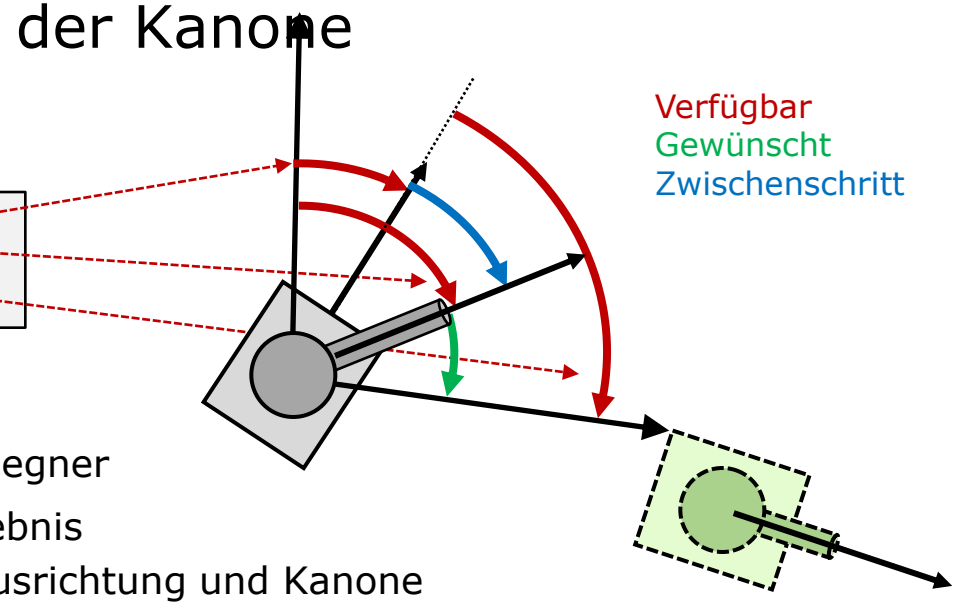
Drehen nach **rechts**,
wenn Gegner rechts von Kanone



Beispiel-Strategie

- Bewegung des Radars und der Kanone
  - Implementierungshilfe:
    - Hilfreiche Informationen z.B.:

```
double hBody = getHeading();
double hGun = getGunHeading();
double bEnemy = e.getBearing();
```



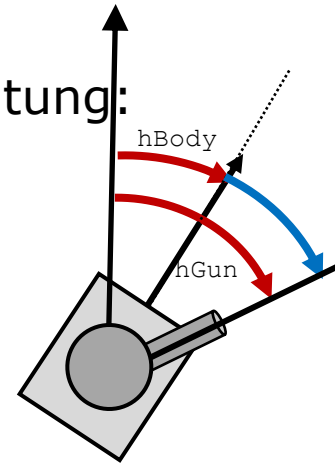
    - Gesucht:
      Winkel a zwischen Kanone und Gegner
    - Idee: Berechne als Zwischenergebnis
      den Winkel d zwischen Roboterausrichtung und Kanone
    - Daher:
      1. Berechnung von d
      2. Berechnung von a
    - Gut zu wissen:
      Um einen Winkel nach links drehen ist gleichbedeutend mit einer negativen Drehung nach rechts:
      `setTurnGunLeft (winkel) = setTurnGunRight(-1*winkel)`

Beispiel-Strategie

- Bewegung des Radars und der Kanone
  - Berechnung des Winkels der Kanone zur Roboterausrichtung:
    - Gegeben:

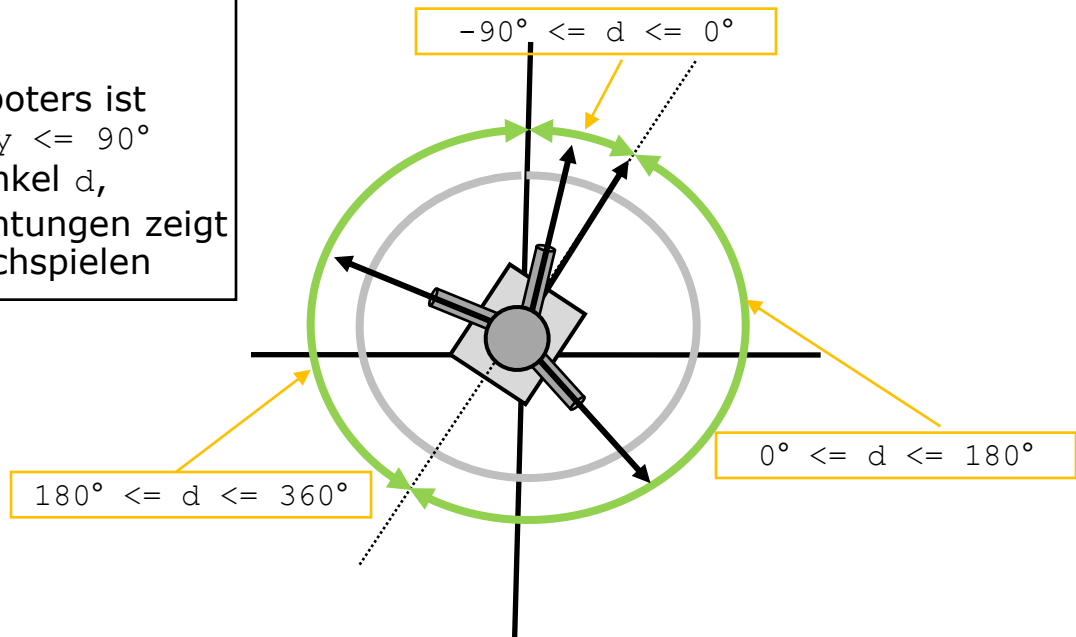
```
double hBody = getHeading();
double hGun = getGunHeading();
```

    - Es müssen verschiedene Fälle unterschieden werden:
      Überlege dir für jeden Fall, welchen Wertebereich
      der Winkel `d = hGun - hBody` annimmt.



Beispielüberlegung:

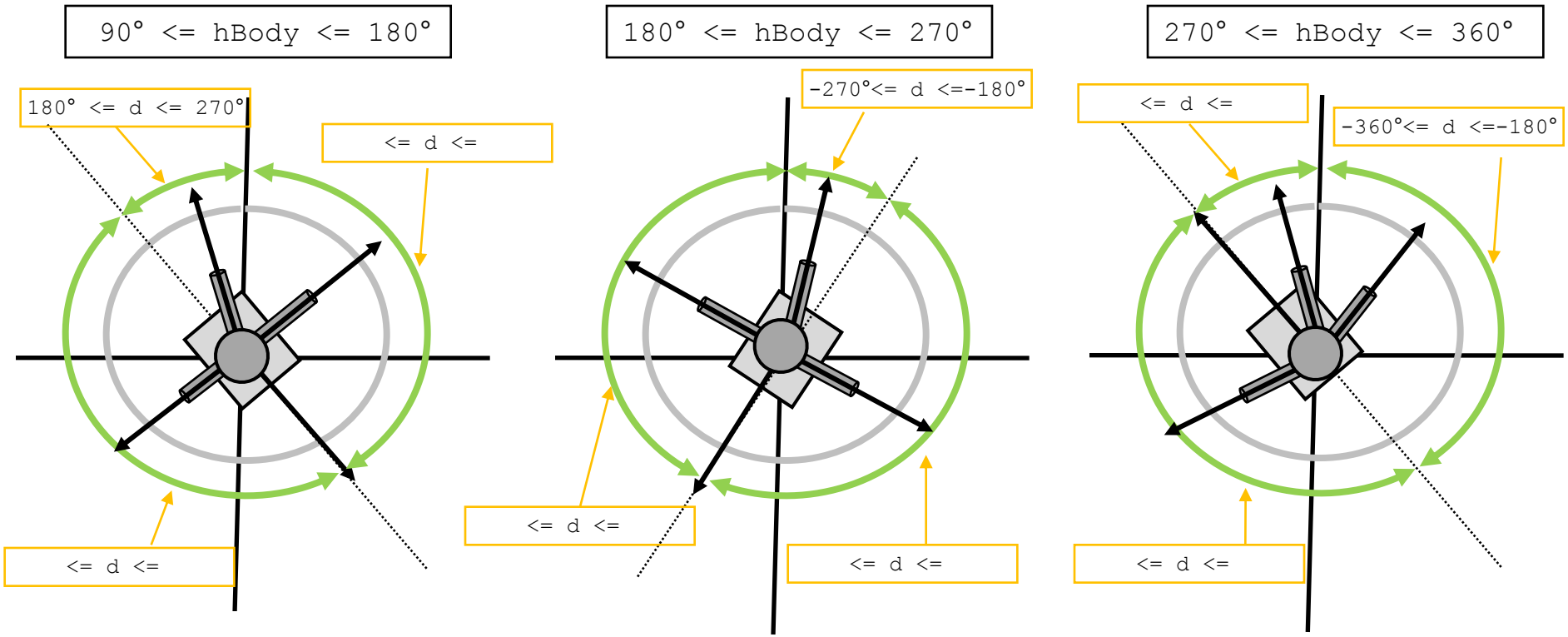
Angenommen: Absolute Ausrichtung des Roboters ist
im 1. Quadranten: `0° <= hBody <= 90°`
Frage: Welchen Wertebereich hat der Winkel `d`,
falls die Kanone in bestimmte Richtungen zeigt
Lösung: Überlegen und alle Szenarien durchspielen



Ergebnis:
Kanone rechts von Roboter-Orientierung falls:
`0° <= d <= 180°`
Kanone links von Roboter-Orientierung falls:
`-90° <= d <= 0°` oder
`180° <= d <= 360°`

Beispiel-Strategie

- Bewegung des Radars und der Kanone
  - Gleiche Überlegungen für die Fälle: Absolute Ausrichtung des Roboters ist im 2., 3. oder 4. Quadranten



Beispiel-Strategie

- Bewegung des Radars und der Kanone
  - Kombiniert man alle 4 Fallunterscheidungen, ergibt sich:

Falls `hBody <= 180°`: Kanone **rechts** von Roboter-Orientierung falls:

`0° <= d <= 180°`		(1. Fall)
`0° <= d <= 180°`		(2. Fall)

Kanone **links** von Roboter-Orientierung falls:

`-90° <= d <= 0°`	oder `180° <= d <= 360°`	(1. Fall)
`-180° <= d <= 0°`	oder `180° <= d <= 270°`	(2. Fall)

Falls `hBody > 180°`: Kanone **rechts** von Roboter-Orientierung falls:

`-270° <= d <= -180°`	oder `0° <= d <= 180°`	(3. Fall)
`-360° <= d <= -180°`	oder `0° <= d <= 90°`	(4. Fall)

Kanone **links** von Roboter-Orientierung falls:

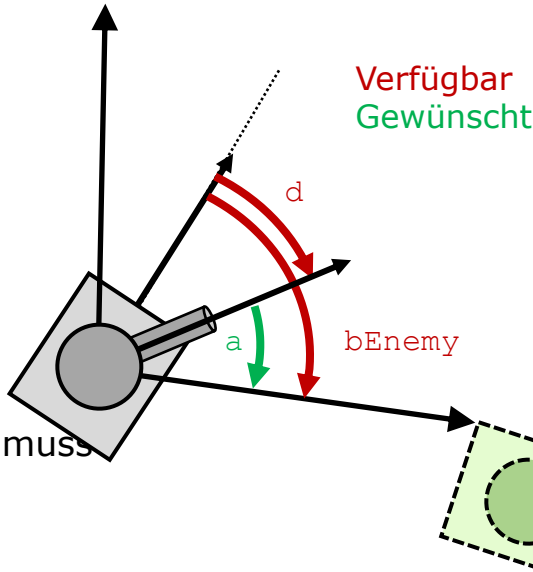
`-180° <= d <= 0°`		(3. Fall)
`-180° <= d <= 0°`		(4. Fall)

  - Daher: Zu Wissen, ob Kanone links/rechts ist, reicht aus:

Falls `hBody <= 180°`:
Kanone **rechts** von Roboter-Orientierung falls: `0° <= d <= 180°`
Sonst ist sie **links** (Finales `d`: falls `d > 0°` muss `d = d-360°` gerechnet werden).

Falls `hBody > 180°`:
Kanone **links** von Roboter-Orientierung falls: `-180° <= d <= 0°`
Sonst ist sie **rechts** (Finales `d`: falls `d < 0°` muss `d = d+360°` gerechnet werden).

Beispiel-Strategie



- Bewegung des Radars und der Kanone
  - Gesucht: Winkel `a` zwischen Kanone und Gegner
  - Gegeben:
    - Rel. Winkel `d` zw. Kanone und Roboter-Orientierung
    - Rel. Winkel `bEnemy` zw. Gegner und Roboter-O.
  - Berechne die Winkeldifferenz `a = bEnemy - d`
    - Gibt genau an, um wie viel Grad die Kanone gedreht werden muss
    - Winkel d Werte zwischen -360° und +360° annehmen
  - Kanone soll auf kürzesten Weg gedreht werden
    - Daher ist maximaler Drehwinkel in beide Richtungen 180°
    - Überlegungen:
      - `setTurnGunLeft(winkel) == setTurnGunRight(-winkel)`
      - Drehung um `200°` nach rechts entspricht Drehung um `160°` nach links
  - Daher:

```
double a = e.getBearing() - d;
if (a < -180) {
          a += 360;
} else if( a > 180) {
          a -= 360;
}
setTurnGunRight(a);
```