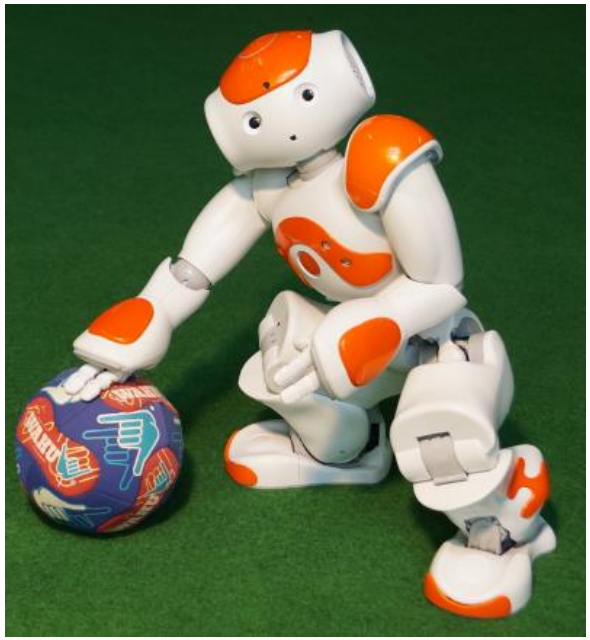
Hochschule Offenburg
University of Applied Sciences

Künstliche Intelligenz
4. Robotik

Prof. Dr. Klaus Dorer



Übersicht

- Robotik
  - Sensoren und Aktuatoren
  - Architekturen
  - Vorwärts-Kinematik
  - Inverse Kinematik
  - Laufen auf zwei Beinen

Ziele

- Grundprinzipien von Robotern kennen
- Geometrische Transformationen durchführen können

Quellen

- Huang, Qiang, Kazuhito Yokoi, Shuuji Kajita, KenjiKaneko, Hirohiko Arai, Noriho Koyachi und Kazuo Tanie (2001) Planning Walking Patterns for a Biped Robot. IEEE Transactions on Robotics and Automation,17:280289.
- Russel, Norvig (2002) Artificial Intelligence: A Modern Approach. Prentice Hall, ISBN 0137903952.
- Schindler I (2009) Laufen auf zwei Beinen in der simulierten RoboCup 3D-Umgebung, Bachelor Thesis, Hochschule Offenburg
- Ritter B (2012) Steuerung simulierter Roboter mit Kinect. Bachelor Thesis, Hochschule Offenburg
- www.wikipedia.de

Konferenzen

- Robotik: IROS (2019 Macao)
- RoboCup: Wettkampf und Konferenz (2019 Sydney)

Sensoren und Aktuatoren

- Sensoren
  - Propriozeption
  - Taktil
  - Audible
  - Vision
  - Orientierung
  - Distanz
  - Position

Encoder

Berührung Kraft

Kamera

Mikrofon

Gyroskop

IMU (Gyro + Acceleration)

Ultraschall

Infrarot

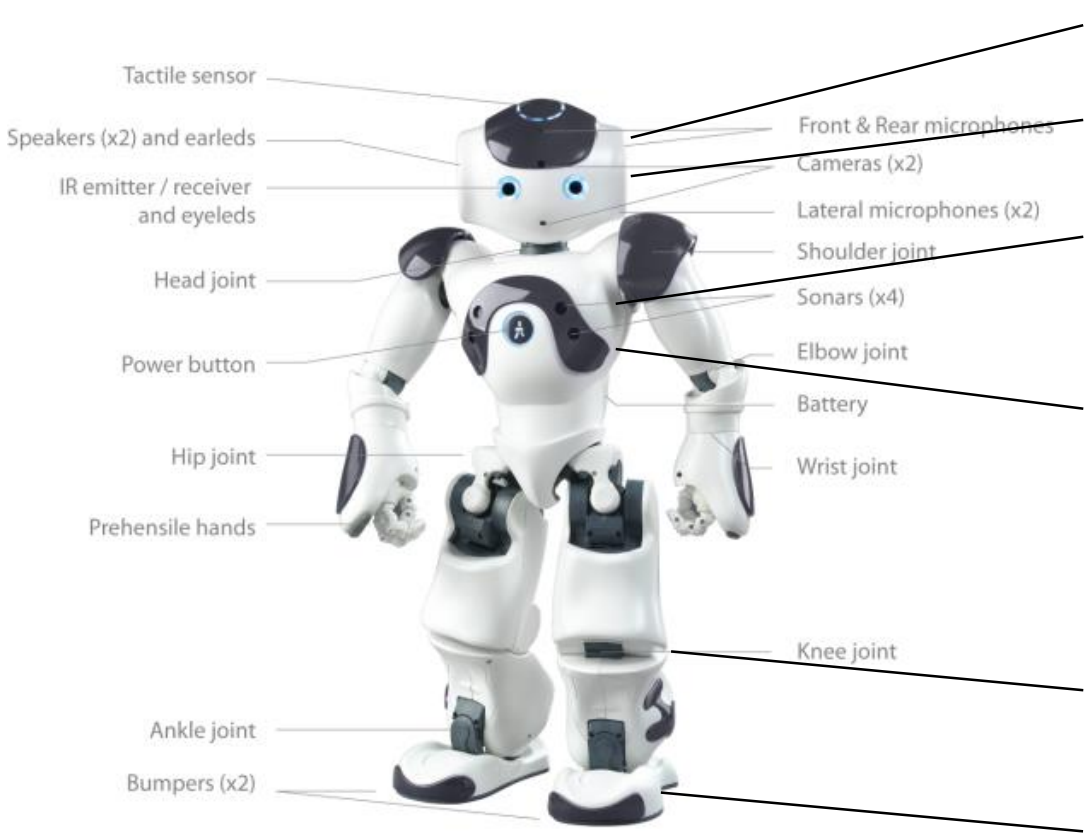
Laser Scanner

GPS Modul

Sensoren des Nao

- Echt
- Simuliert

Game State



Kamera

Mikrofon

Gyroskop

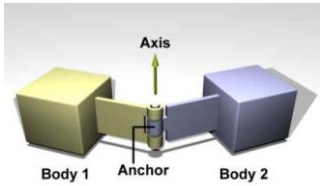
Accelerometer

Winkelgeber

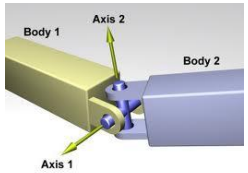
Fußdruck

Sensoren und Aktuatoren

- Aktuatoren (auch Effektoren)
  - Anzeige
    - Dioden
    - Bildschirme
  - Bewegung
    - Servos
    - Hydraulikzylinder
  - Manipulation
    - Greifer



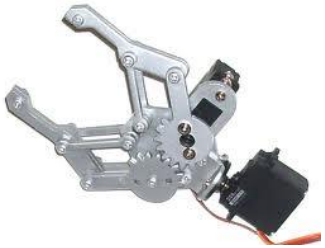
Hinge Joint



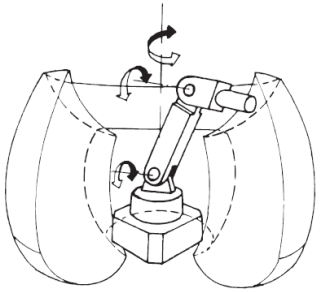
Universal Joint

Stepper

Linear Aktuator



Gripper



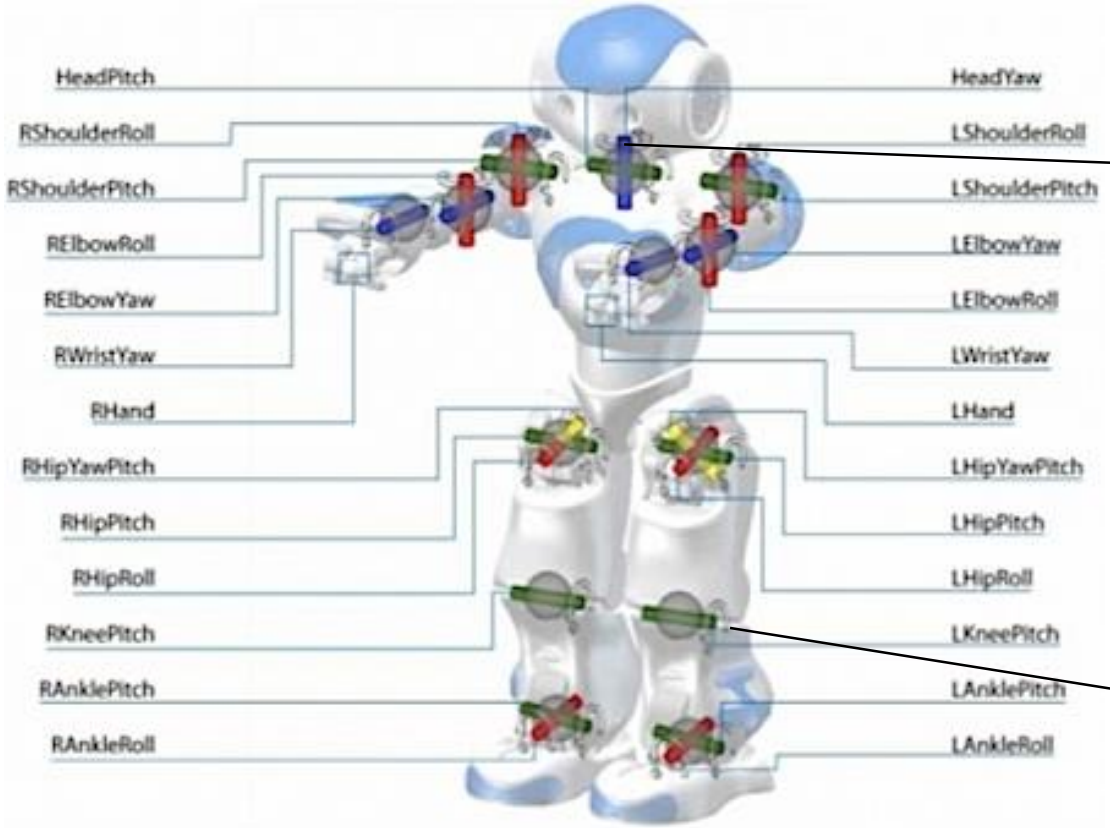
Typischer Fertigungsroboter

Aktuatoren des Nao

- Echt
- Simuliert

Init

Beam



Sprache

Hinge Joint (22x)

Sensoren und Aktuatoren

- Sensordaten kommen meist getaktet an
  - Unterschiedliche Sensoren können unterschiedliche Frequenz haben
  - Sensorfrequenz und Aktuatorfrequenz muss nicht übereinstimmen
  - Bsp: RoboCup Simulation:
    - Hinge Sensoren 50 Hz
    - Kamera 16 2/3 Hz
    - Hinge Effektoren 50 Hz
  - Lösung
    - Fusionierung der Sensordaten in ein Modell
    - Wahrnehmungen mit Zeitstempel versehen, alte Wahrnehmungen ‚irgendwann' vergessen
    - Extrapolation des Modells solange keine neuen Sensordaten vorliegen

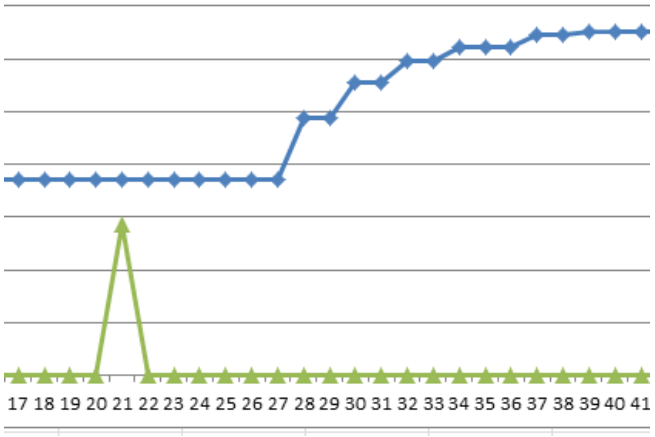
time	Ball LastSeen
1448.33	1448,33
1448.35	1448,33
1448.37	1448,33
1448.39	1448,39
1448.41	1448,39
1448.43	1448,39
1448.45	1448,45

Sensoren und Aktuatoren

- Sensordaten können Latenz haben
  - Bis die Daten ankommen kann Zeit vergehen
  - Bsp: RoboCup Simulation:
    - Hinge Sensoren 40 ms
    - D.h. eine Motoransteuerung wird erst 40 ms später gesehen
  - Lösung
    - Extrapolation
    - Anpassung der Extrapolation durch Sensorwerte

time	NeckYaw_AXIS	NeckYaw_SPEED
1448.33	7,11	-7,03
1448.35	6,74	-7,03
1448.37	-0,29	0,037
1448.39	-7,31	7,03
1448.41	-7,27	7,03
1448.43	-0,24	0,176

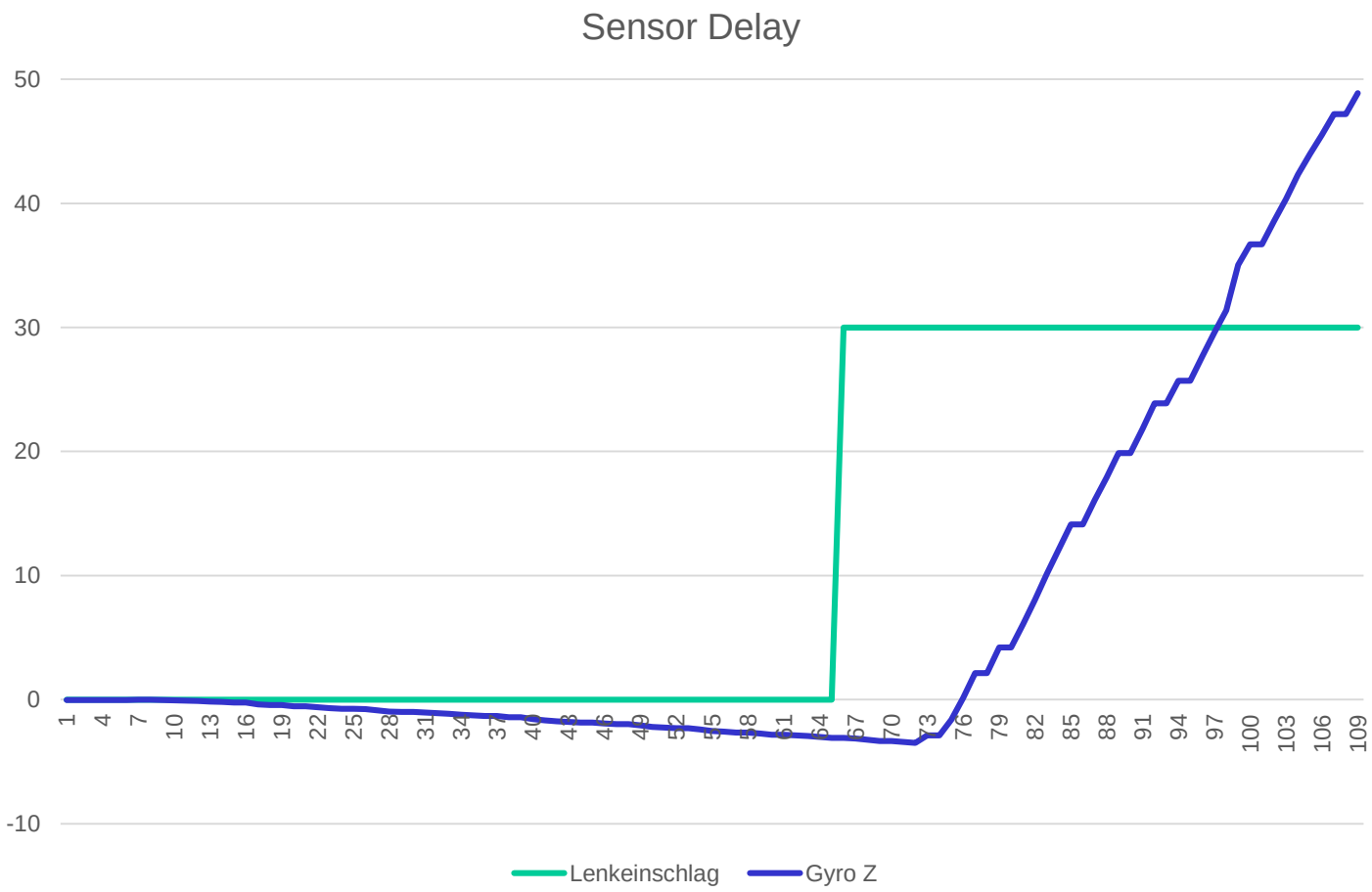
Simulation



Echter Nao

Sensoren und Aktuatoren

- Sensordaten können Latenz haben
  - Bsp: Audi Cup geradeaus fahren und dann voller Lenkeinschlag



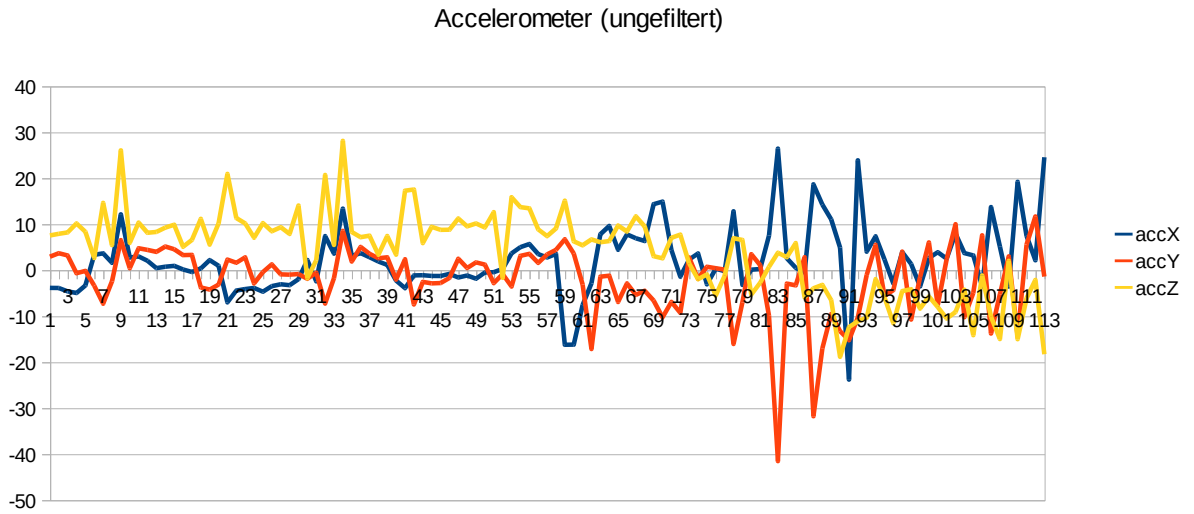
Sensoren und Aktuatoren

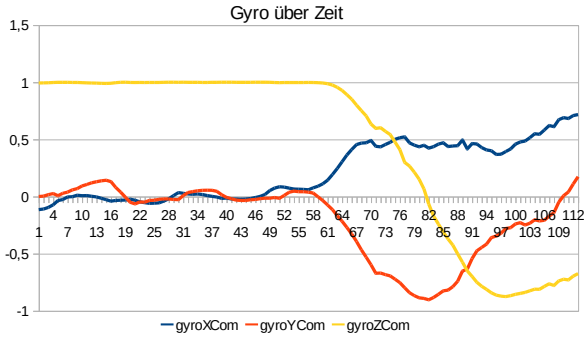
- Sensordaten können verrauscht sein
  - Bsp: RoboCup Simulation
    - Kamera liefert verrauschte Positionen der Objekte
  - Lösung
    - Filterung
      - Mittelung über n Sensorwerte
      - Gewichtete Mittelung über n Sensorwerte
  - Aber! Filter erhöhen auch die Trägheit!
    - Je höher n desto weniger Rauschen
    - Je höher n desto später wird eine tatsächliche Änderung erkannt

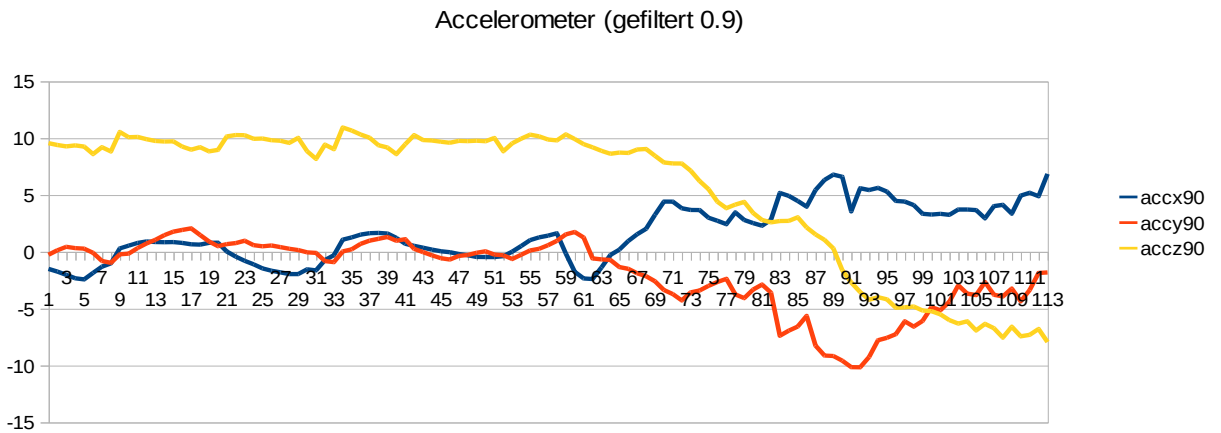
time	LastSeen	BallX	BallY	BallLocalX	BallLocalY
1448.33	1448,33	-0,01	-0,041	9,977	-0,044
1448.39	1448,39	-0,007	-0,046	9,997	-0,011
1448.45	1448,45	-0,006	-0,052	9,998	-0,007
1448.51	1448,51	-0,001	-0,031	10,03	0,057

Sensoren und Aktuatoren

- Rauschen
  - Accelerometer
  - Gyro
- Sensorfusion

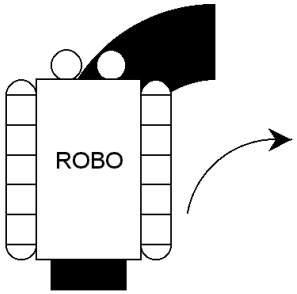


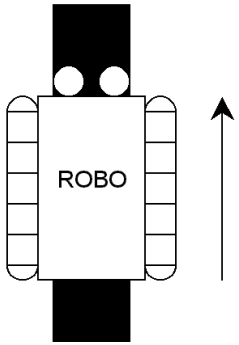


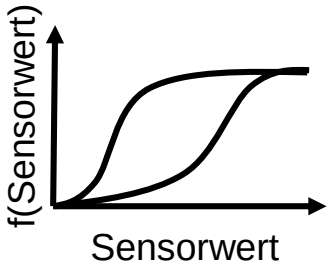


Architekturen

- Wie komme ich von Sensordaten zu einer Aktion?
- Reaktiv
  - Wenn linksHell und rechtsDunkel dann rechts
  - Wenn linksDunkel und rechtsDunkel dann geradeaus
- Vorteile
  - Sehr schnell
  - Einfach
- Nachteile
  - Flickering
    - Hysterese kann helfen
  - Schwierig höhere Fähigkeiten zu erzielen oder zielgerichtet zu handeln

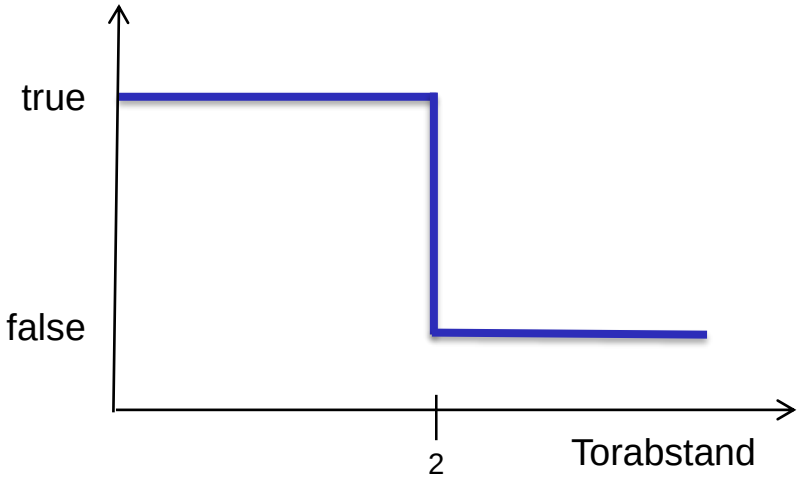


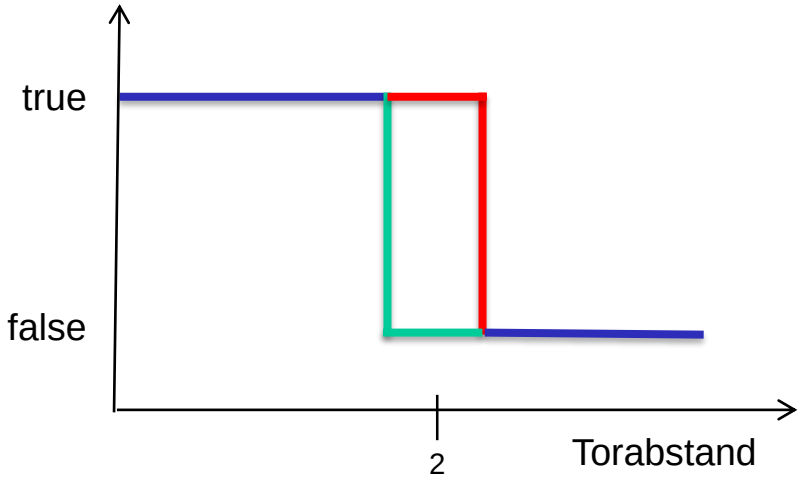




Hysterese Funktion

- Beispiel Sweaty
  - „Soll der Torwart raus oder nicht?“
  - Basierend auf wie weit der Ball vom Tor(mittelpunkt) entfernt ist
  - Klassisch: ab < 2 m true Hysterese: < 1.9 m bzw. 2.1m



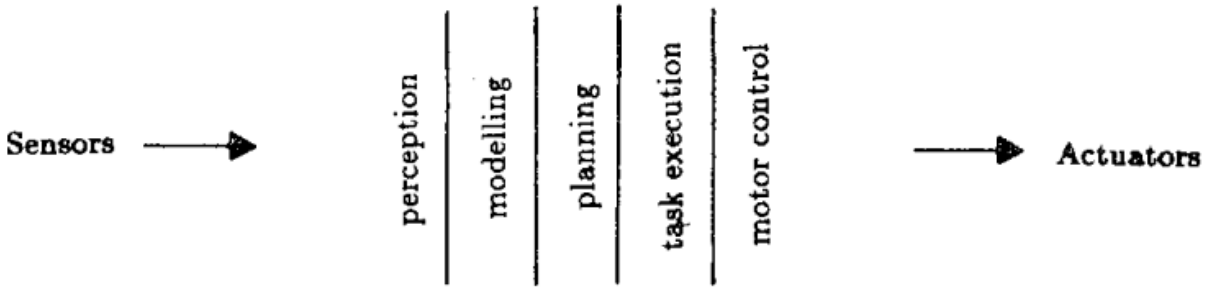


Falls gerade true

Falls gerade false

Architekturen

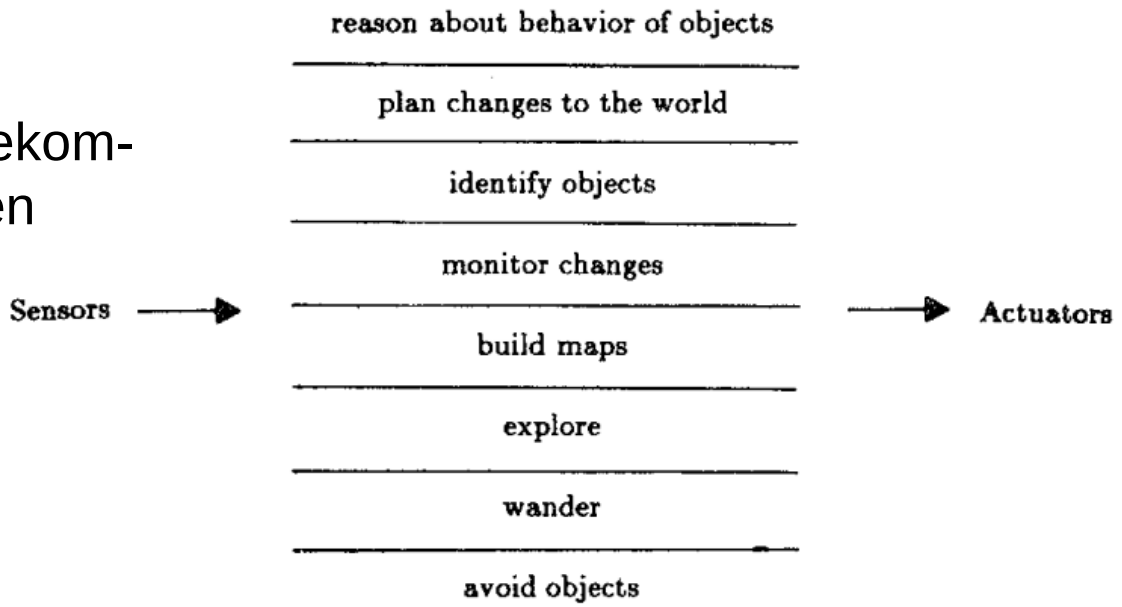
- Sequentielles Modell
  - Ebenen werden nacheinander durchlaufen
  - Verwendet von magmaOffenburg



- Vorteile
  - Einfach
  - Modell erlaubt komplexe Zusammenhänge zu modellieren
- Nachteile
  - Beim Fehlschlag eines Teils fallen alle Teile aus
  - Reaktionsgeschwindigkeit hängt vom kompletten Pfad ab und damit von der langsamsten Komponente

Architekturen

- Subsumption
  - Schichtenbasierte Dekom-position von Verhalten
  - Höhere Ebenen können niedere Ebenen hemmen oder subsumieren

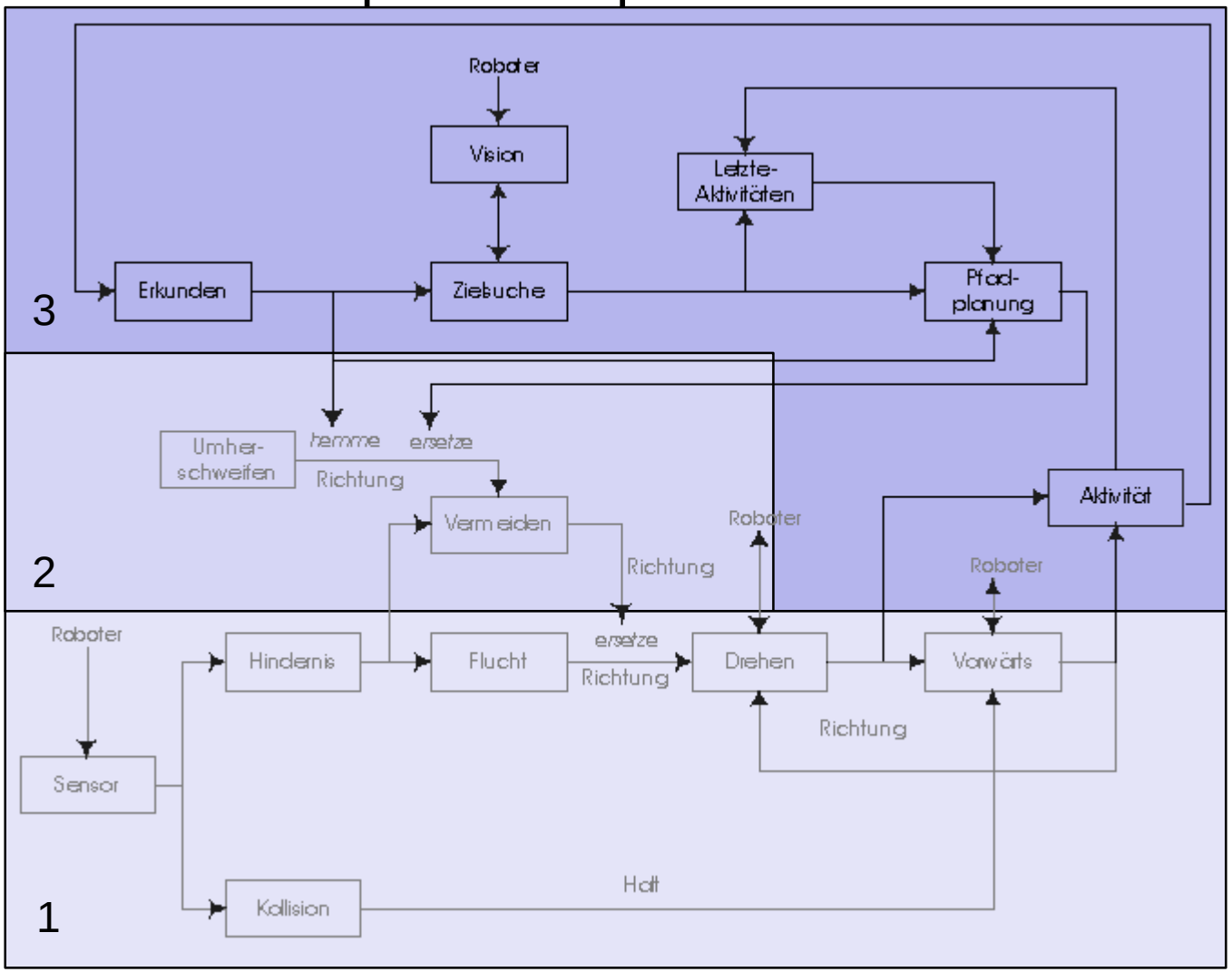


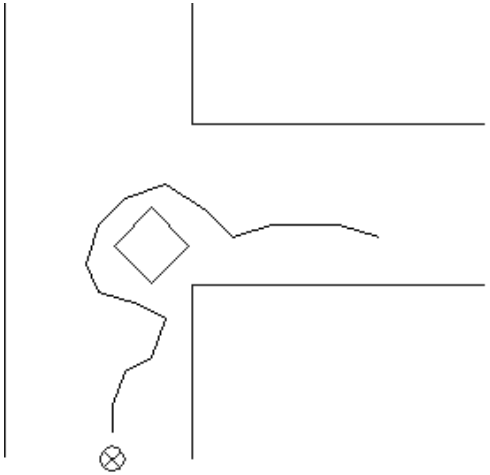
- Vorteile
  - Wenn höhere Schichten ausfallen funktioniert z.B. immer noch die Obstacle Avoidance
  - Schnelle Reaktionen möglich, auch wenn höhere Schichten lange rechnen müssen
- Nachteile
  - Es wird immer schwieriger, je mehr Schichten dazu kommen

Brooks, R. (1986). "A robust layered control system for a mobile robot". Robotics and Automation, IEEE Journal of Robotics and Automation 2 (1): 14–23

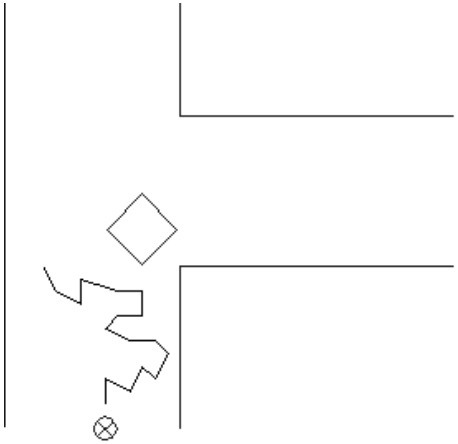
Architekturen

- Subsumption Beispiel





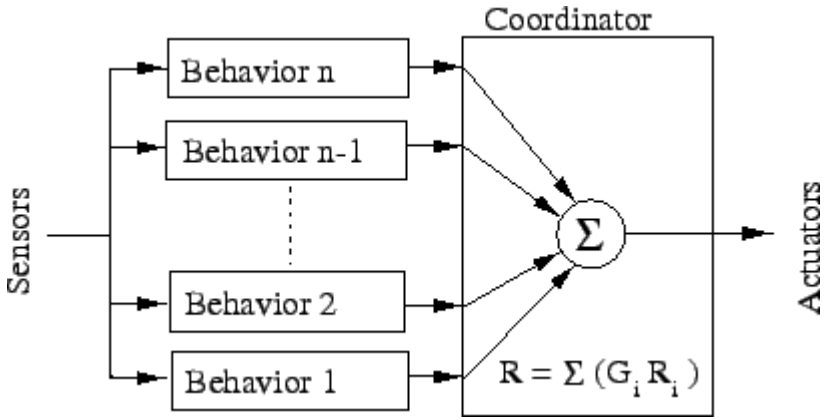
Ebene 1 + 2 + 3



Ebene 1 + 2

Architekturen

- Motor Schemas
  - Biologisch inspiriert
  - Benutzt Kraftfelder oder Potenzialfelder für die Abbildung Sensorraum -> Motorraum
  - Koordinierung der Behaviors als Summierung der Felder

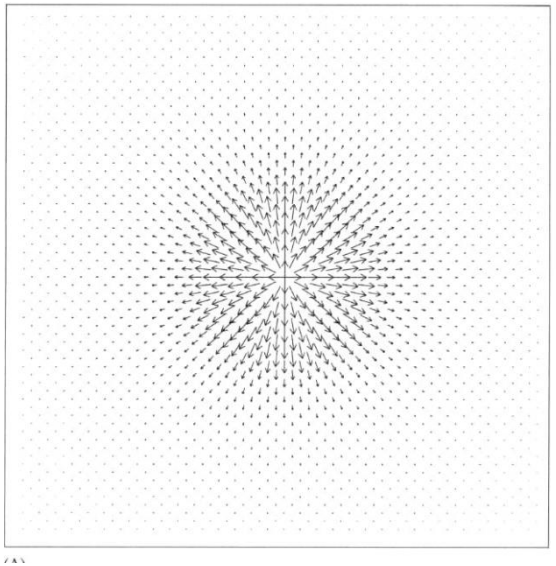


- Vorteile
  - Einfache Überlagerung von Verhalten
- Nachteile
  - Lokale Minima
    - Noise hinzufügen
  - Schwierig, in jeder Situation gewünschtes Verhalten zu erzielen

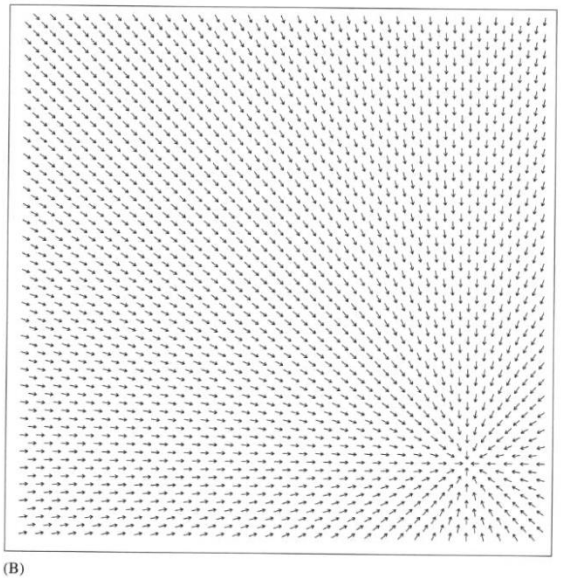
R.C. Arkin. Motor schema-based mobile robot navigation. International Journal of Robotics research, 8(4):92-112, 1989

Architekturen

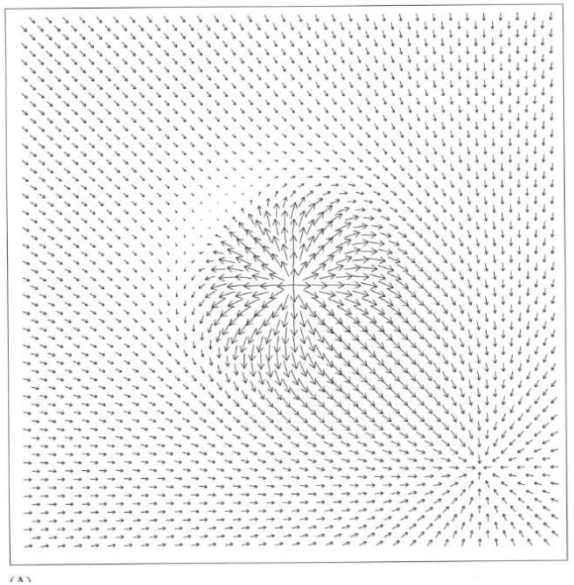
- Motor Schemas Beispiel



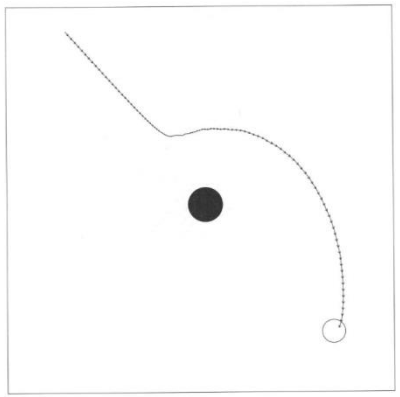
Hindernis



Ziel



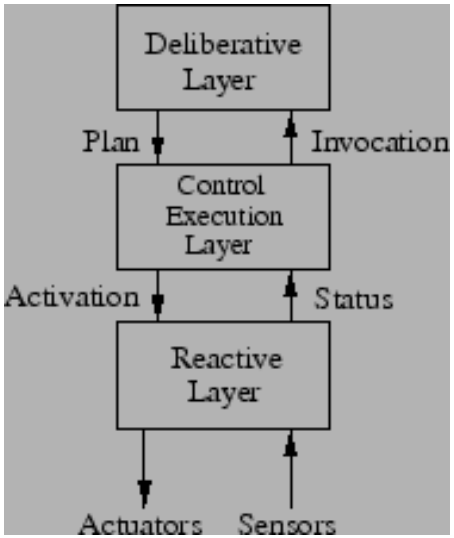
Beides zusammen



Verfolgter Pfad

Architekturen

- Hybride Ansätze
  - Deliberative Layer: High level planning für die Zielerreichung
  - Control Execution Layer: Zerlegung des High Level Plans in Unteraufgaben
  - Reactive Layer: Ausführung der Behaviors und Reaktion auf Ereignisse
  - Verwendet in JPL Rovern

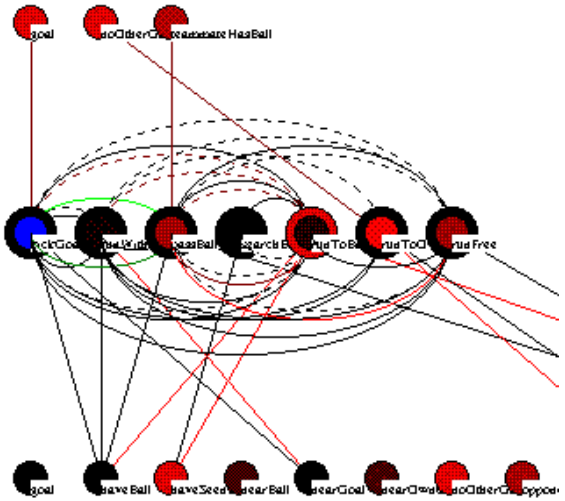


- Vorteile
  - Entkopplung erlaubt längere Rechenzeiten fürs Planen
- Nachteile
  - Zusammenspiel zwischen Control Execution und Reactive Layer ist nicht trivial

E. Gat. Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, 1991

Architekturen

- Verhaltensnetzwerke
  - Beliefs steuern die Ausführbarkeit
  - Ziele steuern die Aktivierung von Verhalten
  - Ausführbares Verhalten mit der höchsten Aktivierung wird ausgewählt
  - Eigenes Kapitel

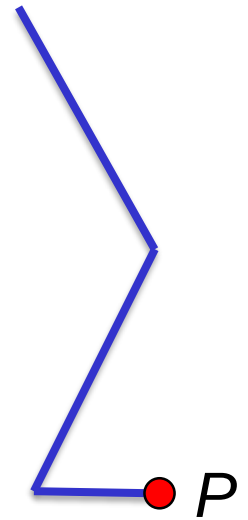


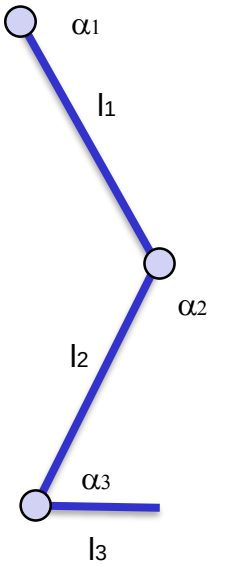
- Vorteile
  - Kombination aus proaktivem und reaktivem Verhalten
  - Symbolische Repräsentation (numerische Verarbeitung)
  - Schnell
- Nachteile
  - Nicht immer einfach, gewünschtes Verhalten zu erzielen
  - Objekte Adressieren (Lösung: indexical functional objects)

Maes, P.: The agent network architecture (ANA). In: SIGART Bulletin, 2 (4), pages 115–120 (1991)

Vorwärts-Kinematik

- Frage
  - Wo ist meine Fußspitze?
  - Genauer, was ist deren Pose (Position und Orientierung)
- Gesucht
  - Punkt P
- Bekannt
  - Geometrie der Köperteile (hier Längen)
  - Aufhängungspunkte
  - Winkel
- Problem
  - Winkel sind im lokalen Koordinatensystem angegeben





Vorwärts-Kinematik

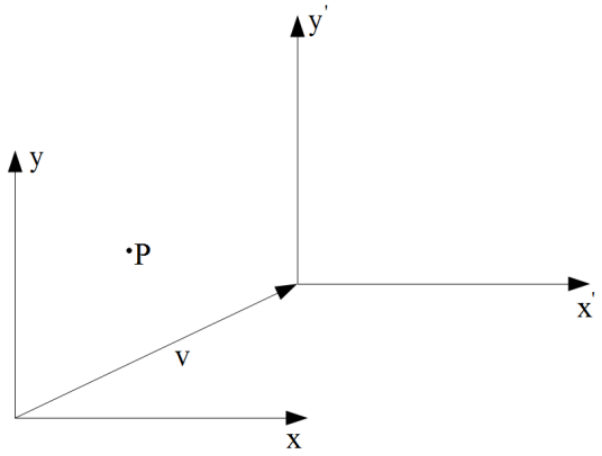
- Berechnung der Position erfolgt durch geometrische Transformationen
- Translation als Vektoraddition

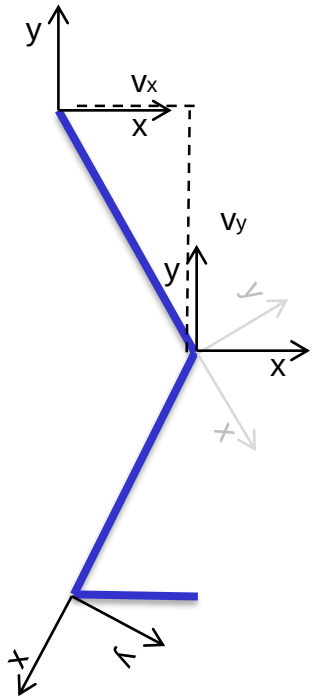
$$P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

$$T = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

$$P' = P + T$$

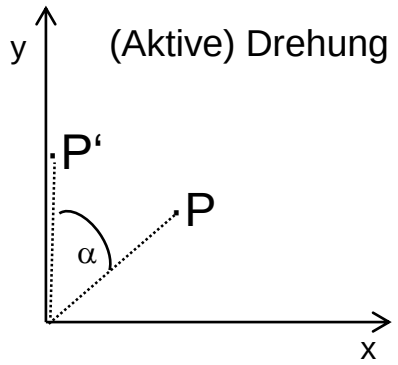
$$P' = \begin{bmatrix} P_x + v_x \\ P_y + v_y \\ P_z + v_z \end{bmatrix}$$

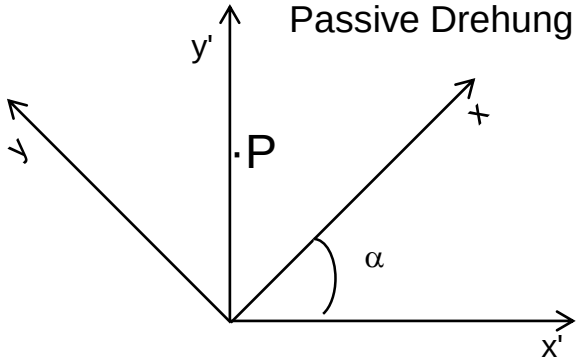




Vorwärts-Kinematik

- Rotation mit orthogonalen Rotationsmatrizen in 2D

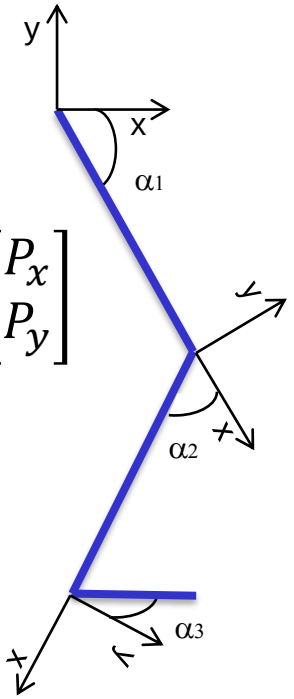




$$R = \begin{bmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{bmatrix}$$

$$R = \begin{bmatrix} \cos(-\alpha) & -\sin(-\alpha) \\ \sin(-\alpha) & \cos(-\alpha) \end{bmatrix}$$

$$P = \begin{bmatrix} P_x \\ P_y \end{bmatrix}$$



$$P' = RP$$

$$P' = \begin{bmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix}$$

$$P' = \begin{bmatrix} P_x cos\alpha - P_y sin\alpha \\ P_x sin\alpha + P_y cos\alpha \end{bmatrix}$$

Vorwärts-Kinematik

- 3D

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix} \quad R_y(\alpha) = \begin{bmatrix} cos\alpha & 0 & sin\alpha \\ 0 & 1 & 0 \\ -sin\alpha & 0 & cos\alpha \end{bmatrix} \quad R_z(\alpha) = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P' = RP$$

  - Mehrere Rotationen

$$P' = R_1 R_2 P$$

  - Die Reihenfolge ist wichtig!

$$R_x(\frac{\pi}{4}) \cdot R_y(\frac{\pi}{4}) = \begin{bmatrix} 0.71 & 0 & 0.71 \\ 0.5 & 0.71 & -0.5 \\ -0.5 & 0.71 & 0.5 \end{bmatrix} \qquad R_x(\frac{\pi}{4}) \cdot R_y(\frac{\pi}{4}) \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2.83 \\ 0.41 \\ 2.41 \end{bmatrix}$$

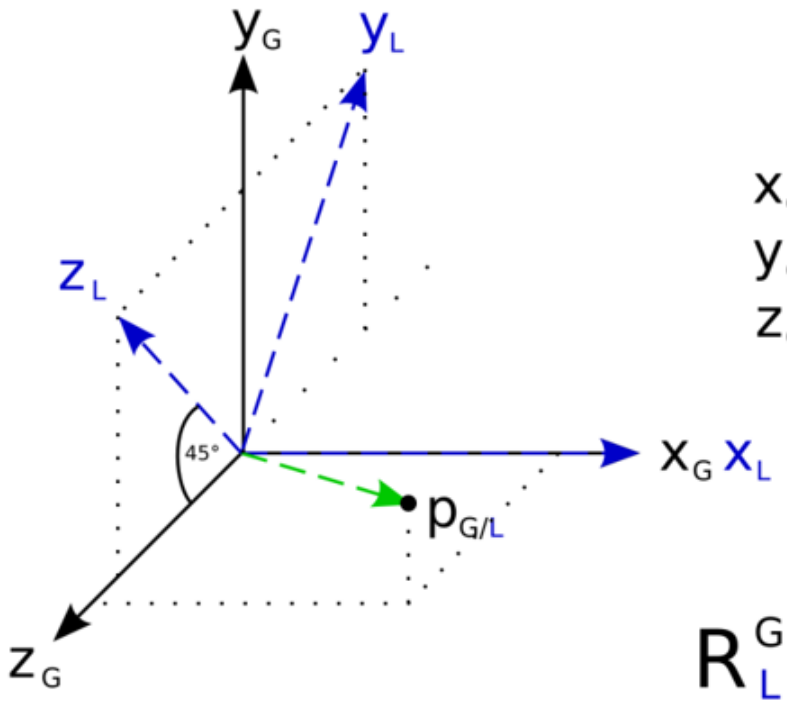
$$R_y(\frac{\pi}{4}) \cdot R_x(\frac{\pi}{4}) = \begin{bmatrix} 0.71 & 0.5 & 0.5 \\ 0 & 0.71 & -0.71 \\ -0.71 & 0.5 & 0.5 \end{bmatrix} \qquad R_y(\frac{\pi}{4}) \cdot R_x(\frac{\pi}{4}) \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 3.21 \\ -0.71 \\ 1.79 \end{bmatrix}$$

Vorwärts-Kinematik

- Orthogonale Rotationsmatrizen
  - Erster Spaltenvektor ist Richtung der x-Achse, zweiter y-Achse, dritter z-Achse des rotierten Koordinatensystems im Original-Koordinatensystem
  - Erster Zeilenvektor ist Richtung der x-Achse, zweiter y-Achse, dritter z-Achse des Original-Koordinatensystems im rotierten Koordinatensystem
  - Inverse Rotation ist transponierte der Originalmatrix
  - Rechtshändig, orthogonal, normiert
    - Rechte Hand Regel für die Stellung der Achsen
    - Rechte Hand Regel für die Richtung positiver Rotation

Vorwärts Kinematik

- Beispiel
  - Rotation um x mit -45°



(x-rotation by -45°)

$$\begin{array}{c} \\ x_G \rightarrow \\ y_G \rightarrow \\ z_G \rightarrow \end{array} \begin{array}{c} \begin{array}{ccc} x_L & y_L & z_L \\ \downarrow & \downarrow & \downarrow \\ x & y & z \end{array} \\ \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix} \end{array}$$

$$R_L^G \times p_L = p_G$$

$$\begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix} \times \begin{bmatrix} p_{L_x} \\ p_{L_y} \\ p_{L_z} \end{bmatrix} = \begin{bmatrix} p_{G_x} \\ p_{G_y} \\ p_{G_z} \end{bmatrix}$$

Vorwärts-Kinematik

- Beispiel
  - Gegeben

$sei\ \alpha_1 = -60, \alpha_2 = -55, \alpha_3 = 115$

$sei\ a_1 = 2.5,\ a_2 = 2.5, a_3 = 1$

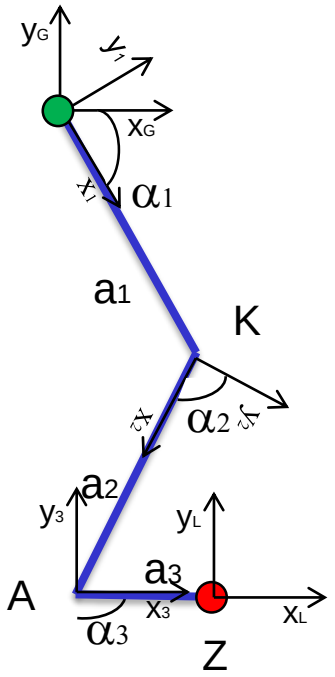
$$H_{Global} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad T_1 = \begin{bmatrix} 2.5 \\ 0 \\ 0 \end{bmatrix}$$

  - Gesucht: Koordinate der Fußspitze im Hüft-Koord.

$$K_{Hüfte} = H_{Global} + R_{Knie}^{Hüfte}\, T_1 = \begin{bmatrix} 1.25 \\ -2.17 \\ 0 \end{bmatrix}$$

$$A_{Hüfte} = K_{Hüfte} + (R_{Knie}^{Hüfte} R_{Knöchel}^{Knie}) T_2 = \begin{bmatrix} 0.19 \\ -4.43 \\ 0 \end{bmatrix}$$

$$Z_{Hüfte} = A_{Hüfte} + (R_{Knie}^{Hüfte} R_{Knöchel}^{Knie} R_{Zeh}^{Knöchel}) T_3 = \begin{bmatrix} 1.19 \\ -4.43 \\ 0 \end{bmatrix}$$



Vorwärts Kinematik

- Pose
  - Lage und Richtung eines Körperteils (bzw. Koordinatensystems) wird als Pose bezeichnet
    - Position des Ursprungs (im Bezug auf das Referenz-Koordinatensystem)
    - Rotation (im Bezug auf das Referenz-Koordinatensystem)

$$Z_{Hüfte} = P_{Knie}^{Hüfte} P_{Knöchel}^{Knie} P_{Zeh}^{Knöchel} * Z_{Zeh}$$

  - In Java

```
Pose3D pose1 = new Pose3D(Vector3D.ZERO, rot1);
Pose3D pose2 = pose1.applyTo(new Pose3D(trans1, rot2));
Pose3D pose3 = pose2.applyTo(new Pose3D(trans2, rot3));
Pose3D poseL = pose3.applyTo(new Pose3D(trans3, Rotation.IDENTITY));
Vector3D zehHüfte = poseL.applyTo(zehZeh);
```

    - `poseL` ist also die Pose des Koordinatensystems der Fußspitze bezogen auf das Hüftkoordinatensystem
    - `zehHüfte` ist die Lage der Fußspitze im Hüft-Koordinatensystem

Vorwärts-Kinematik

- Homogene Transformationsmatrizen
  - 4x4 Matrix für 3D Transformationen
  - Erlauben, alle Transformationen gleich zu behandeln und zu verknüpfen
  - $H = \begin{bmatrix} \boxed{Rotation} & \boxed{Translation} \\ Perspektive & Skalierung \end{bmatrix}$

- Beispiel
  - Rotation um x Achse und Translation

$$H = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha & 0 \\ 0 & sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & cos\alpha & -sin\alpha & y \\ 0 & sin\alpha & cos\alpha & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation Rotation Kombiniert

Vorwärts-Kinematik

- Beispiel

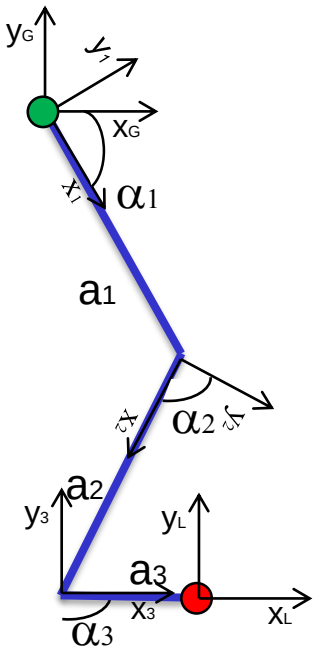
$$R_1 = \begin{bmatrix} \cos\alpha_1 & -\sin\alpha_1 & 0 & 0 \\ \sin\alpha_1 & \cos\alpha_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_1 = \begin{bmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} \cos\alpha_2 & -\sin\alpha_2 & 0 & 0 \\ \sin\alpha_2 & \cos\alpha_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_2 = \begin{bmatrix} 1 & 0 & 0 & a_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

… …

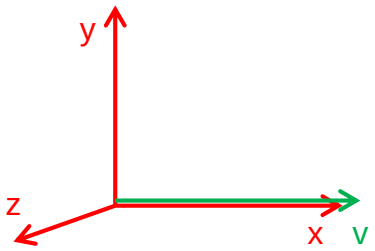
$$M = R_1 T_1 R_2 T_2 R_3 T_3$$

$$P_L = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad P_G = MP_L = \begin{bmatrix} 1.19 \\ -4.43 \\ 0 \\ 1 \end{bmatrix}$$



Vorwärts-Kinematik

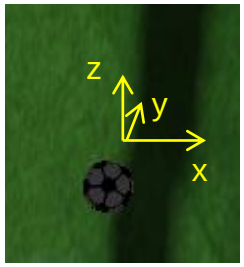
- Rotation 3D: Quaternione
  - Komplexe Zahlen mit 3 Imaginär-Anteilen
    - $q = q_0 + q_1 i + q_2 j + q_3 k$
  - Polardarstellung $q = \cos -\frac{\alpha}{2} + v \sin -\frac{\alpha}{2}$ (v Einheitsquaternion)
  - Geometrische Deutung
    - Imaginär Anteil ist Richtung der Rotationsachse
    - q_0 repräsentiert Rotationswinkel
    - Beispiel: -45° Rotation um x-Achse
    - $q_0 = 0.924$, $q_1 = 0.383$, $q_2 = 0$, $q_3 = 0$

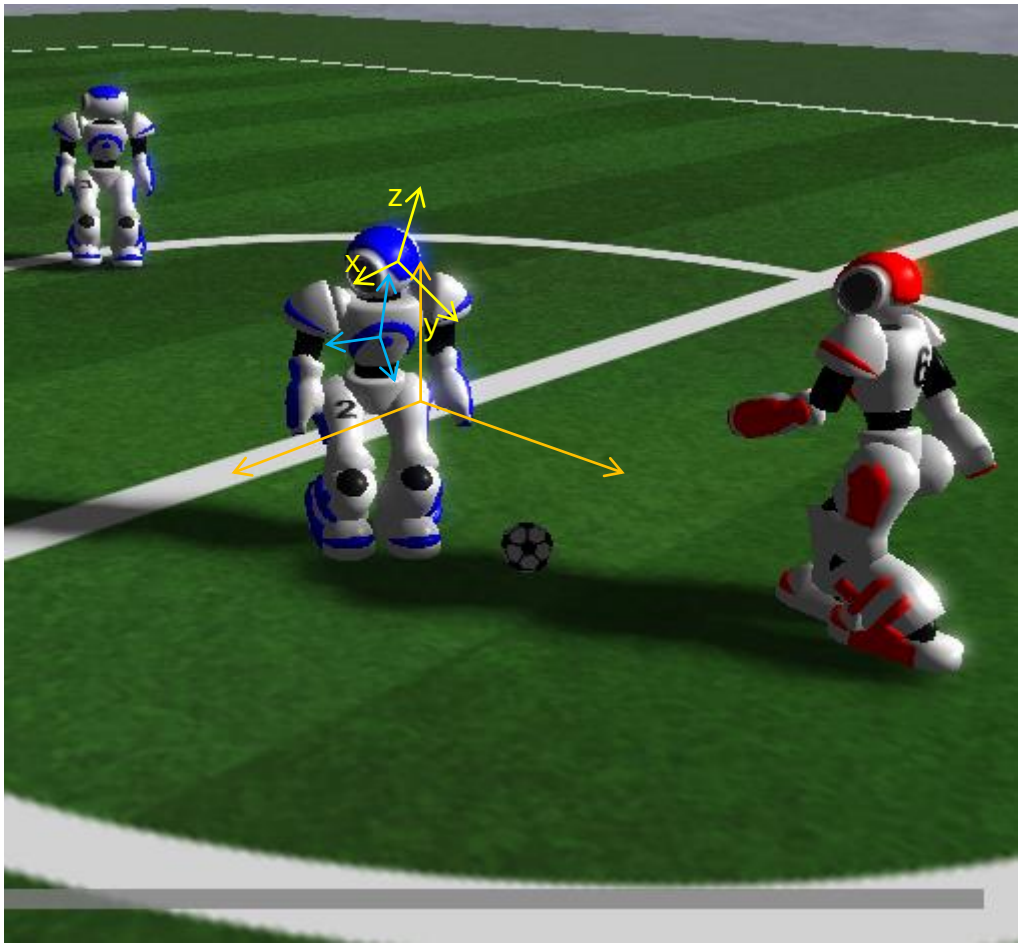


  - Rotation:
    - x'= 2 * (q_0 * (x * q_0 - (q_2 * z - q_3 * y)) + s * q_1) - x
    - y'= 2 * (q_0 * (y * q_0 - (q_3 * x - q_1 * z)) + s * q_2) - y
    - z'= 2 * (q_0 * (z * q_0 - (q_1 * y - q_2 * x)) + s * q_3) - z
    - Mit s = q_1 * x + q_2 * y + q_3 * z

Vorwärts-Kinematik

- Weiteres Beispiel
  - Wo ist der Ball?
  - Blick eines Beobachters
  - Lokaler Blick von Blau 2

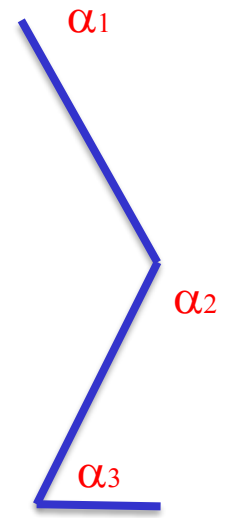


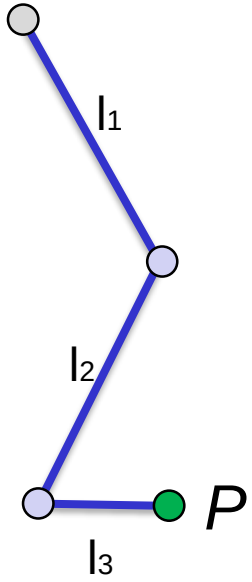


$$P' = R_G T_G R_T T_T P$$

Inverse Kinematik

- Frage
  - Wie kriege ich meine Fußspitze an den Punkt P?
- Gesucht
  - Winkel
- Bekannt
  - Geometrie der Köperteile (hier Längen)
  - Aufhängungspunkte
  - Zielpunkt (eines Köperteils)
- Problem
  - Lösung ist meist nicht eindeutig
  - Unzulässige Konfigurationen
  - Es gibt nicht immer eine Lösung
    - Workspace: erreichbare Punkte
    - Dexterous Workspace: in jeder Orientierung des Endglieds erreichbare Punkte

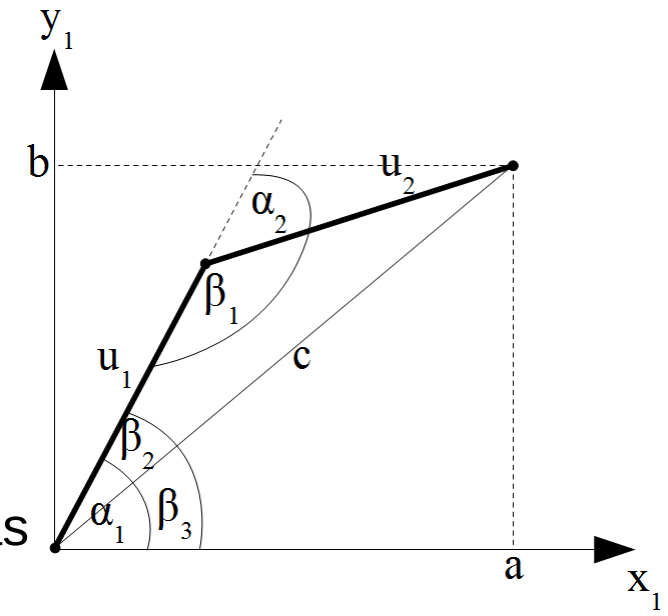




Inverse Kinematik

- Geometrische Methode
  - Beispiel zwei Drehgelenke in 2D
  - Für α_2

$$\cos\beta_1 = \frac{u_1^2 + u_2^2 - c^2}{2u_1u_2} \quad *$$



Da c unbekannt ist, Satz des Pythagoras

$$\cos\beta_1 = \frac{u_1^2 + u_2^2 - a^2 - b^2}{2u_1u_2}$$

Und somit

$$\alpha_2 = 180 - \cos^{-1}\left(\frac{a^2 + b^2 - u_1^2 - u_2^2}{2u_1u_2}\right)$$

* Kosinussatz $\quad c^2 = a^2 + b^2 - 2ab\cos\alpha$

Inverse Kinematik

- Geometrische Methode
  - Für α_1

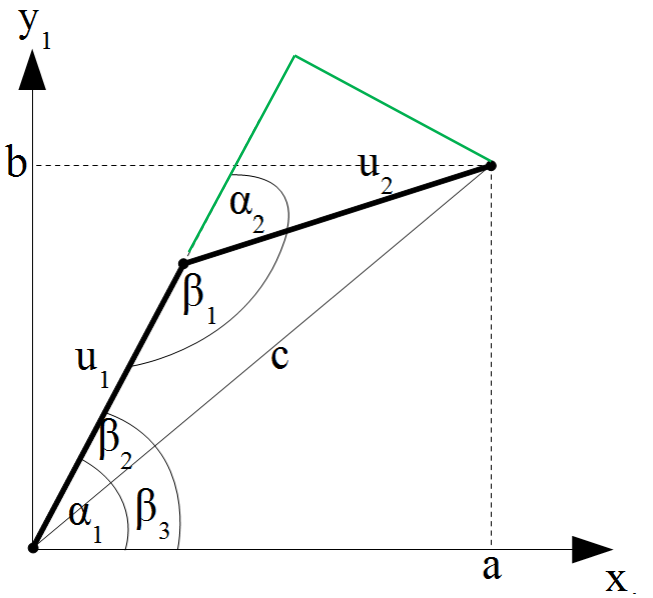
$$\beta_3 = \tan^{-1}\left(\frac{b}{a}\right)$$

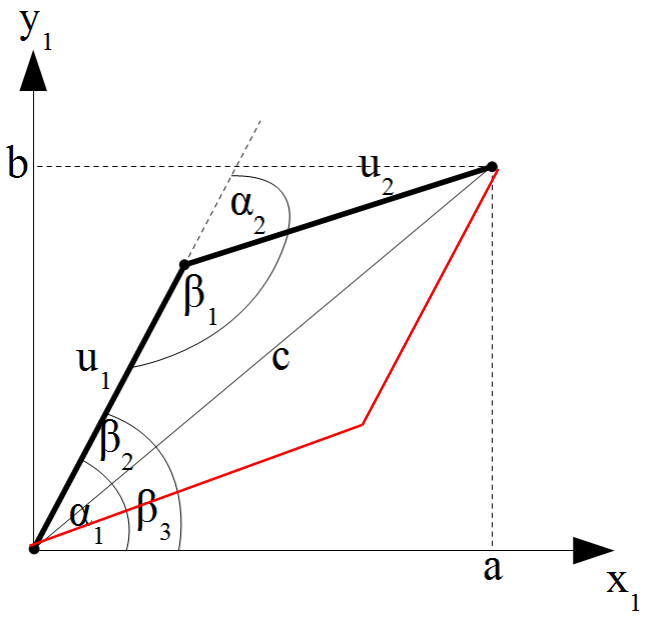
Mit Hilfe des grünen Dreiecks

$$\beta_2 = \tan^{-1}\left(\frac{u_2 \sin\alpha_2}{u_1 + u_2 \cos\alpha_2}\right)$$

Und somit

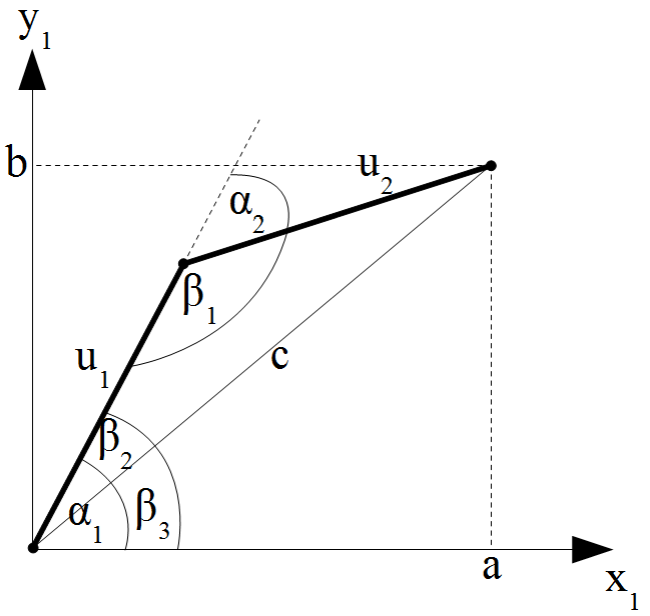
$$\alpha_1 = \beta_2 + \beta_3$$





- Alternative Lösung
  - Falls die Gelenke diese Stellung zulassen

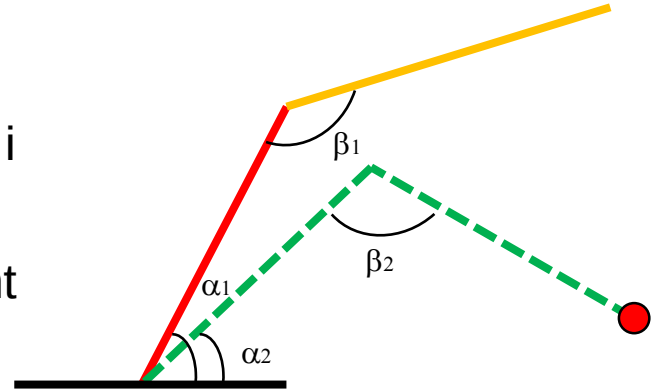
Inverse Kinematik



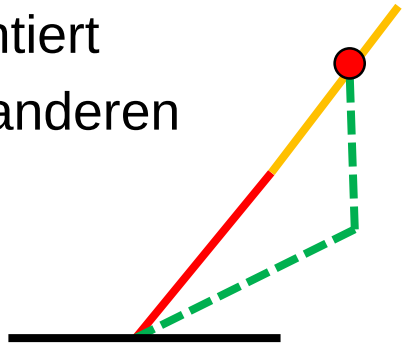
- Algebraische Methode
  - Vorwärts Kinematik
    - $x = u_1 \cos \alpha_1 + u_2 \cos(\alpha_1 + \alpha_2)$
    - $y = u_1 \sin \alpha_1 + u_2 \sin(\alpha_1 + \alpha_2)$
  - Inverse Kinematik
    - Gesucht α_1 und α_2
    - x und y sind gegeben
    - Zwei Gleichungen mit zwei Unbekannten
    - Auflösen nach α_1 und α_2
  - Meist schwierig, da trigonometrische Funktionen involviert sind
  - Für jedes Robotermodell spezifische Lösung notwendig

Inverse Kinematik

- Numerische Methode
  - Z.B. Cyclic Coordinate Descent
    - For i = n to 1 do verändere Gelenk i
    - -> Minimiere Zielfunktion
    - Bis Zielposition hinreichend erreicht
  - Andere Verfahren
    - Jacobian Transpose
    - Pseudo Inverse
    - Damped Least Squares



- Probleme
  - Iterativ, Konvergenzgeschwindigkeit nicht garantiert
  - Resultierende Bewegung weniger rund als bei anderen Verfahren
  - Liefert keine Lösung, wenn keine einzelne Bewegung näher an das Ziel führt

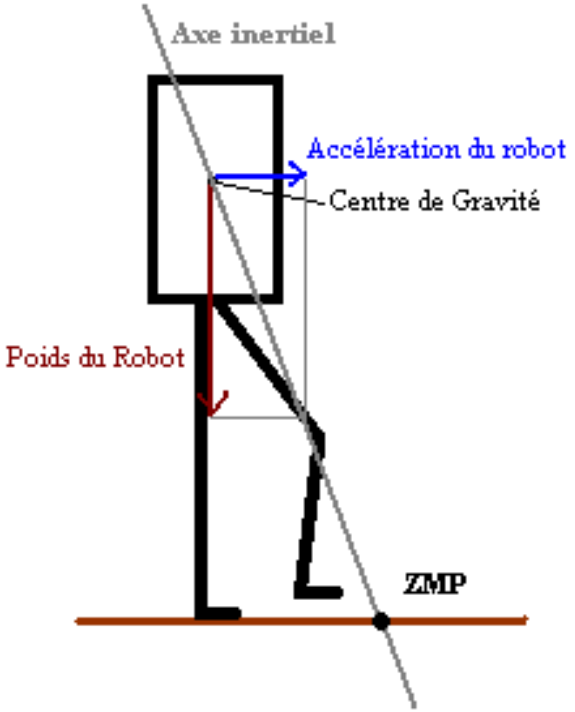


Inverse Kinematik

- Geschlossene Form:
  - Geometrische Lösung
    - \+ : Fallunterscheidung der Roboter-Konfigurationen
    - − : Robotertypen-spezifisch
  - Algebraische Lösung
    - \+ : korrekte Lösungen aus Gleichungen
    - − : geometrisch nicht anschaulich
    - − : Robotertypen-spezifisch
- Numerische Form:
  - Iterative Verfahren
    - \+ : Verfahren übertragbar auf beliebige Robotermodelle
    - − : rechenintensiv, nicht garantierte Konvergenz bei Sonderfällen

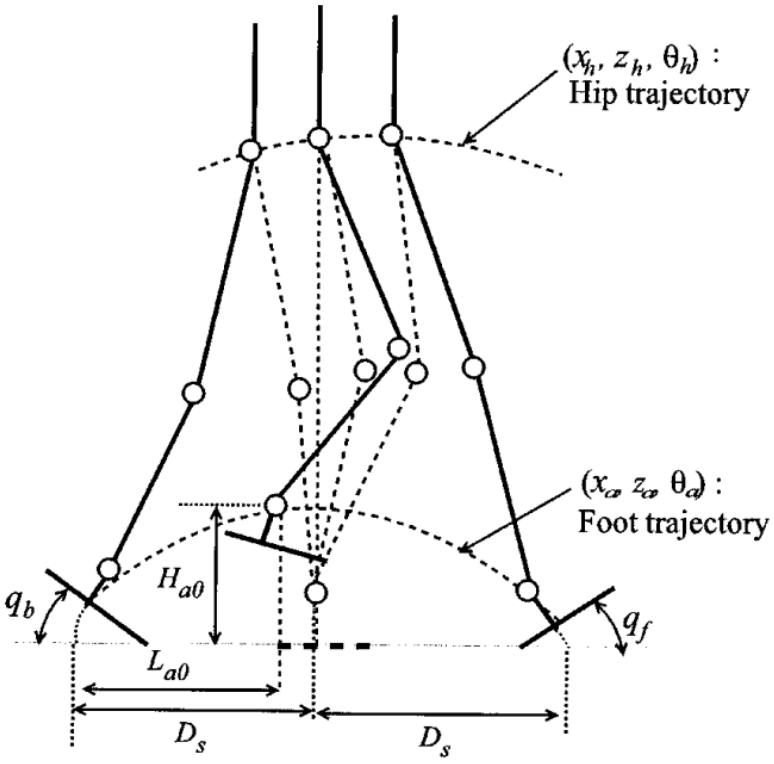
Laufen auf zwei Beinen

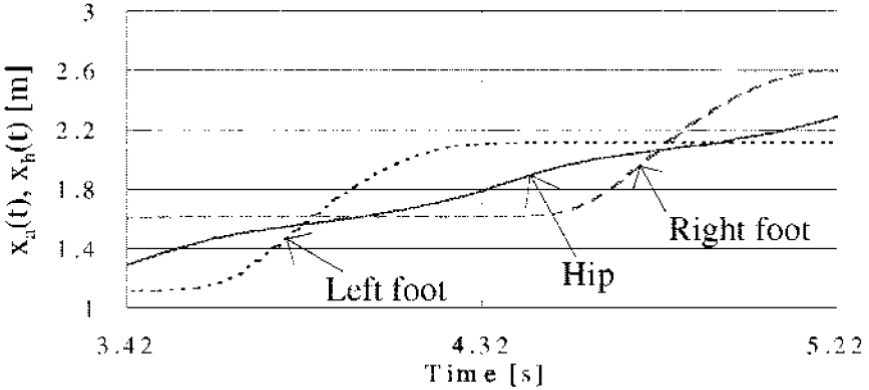
- Begriffe
  - Center of Mass (CoM)
    - Masseschwerpunkt
  - Zero Moment Point (ZMP)
    - Punkt auf dem Boden, der keine horizontale Kraft erzeugt
  - Supportpolygon (SP)
    Befindet sich der ZMP innerhalb des SP, fällt er nicht um
  - Statische Stabilität
    - CoM immer über den Füßen
  - Dynamische Stabilität
    - CoM auch außerhalb der Füße
    - Bewegungsende kann bedeuten, dass der Roboter umfällt



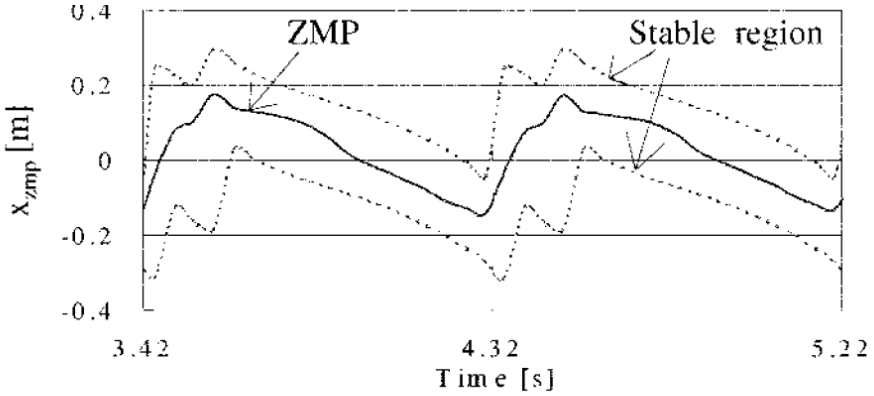
Laufen auf zwei Beinen

- Wie müssen die Beine bewegt werden, damit Laufen in eine gewünschte Richtung entsteht?





(d) Foot motion and hip motion in x-axis



(e) ZMP trajectory in the hip coordinate system

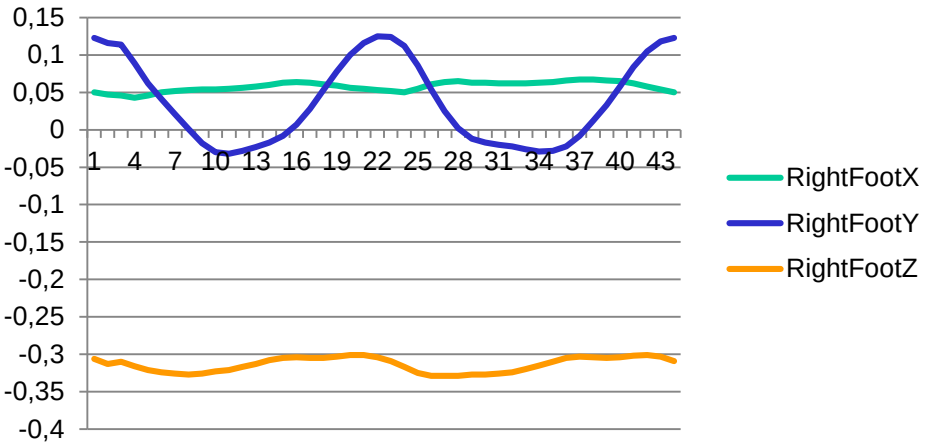
Huang, Qiang, Kazuhito Yokoi, Shuuji Kajita, Kenji Kaneko, Hirohiko Arai, Noriho Koyachi und Kazuo Tanie: Planning Walking Patterns for a Biped Robot. IEEE Transactions on Robotics and Automation,17:280-289, 2001.

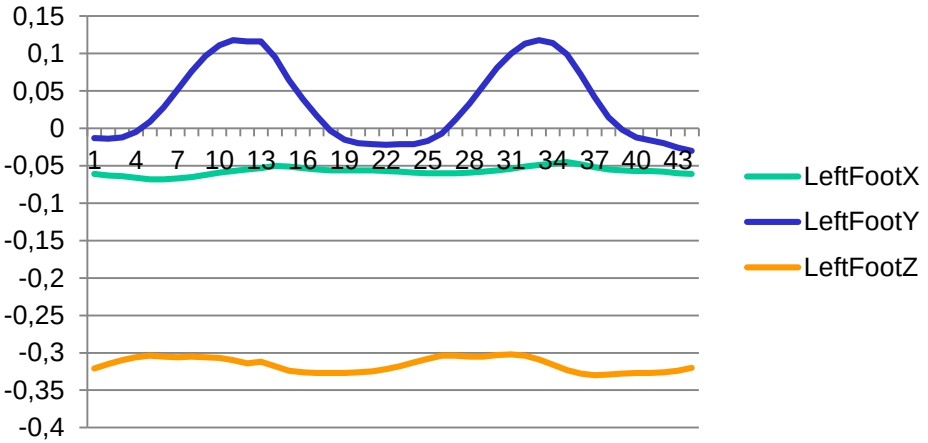
Laufen auf zwei Beinen

- Parameter definieren die Kurven von Fuß und Hüfte
  - Tc = duration of the whole motion (Td < Tm < Tc)
  - Tm = time when the foot should touch the ground again
  - Td = middle of the motion
  - h_gs = Height of Ground when leaving the Ground (hip height)
  - h_ge = Height of Ground when touching the Ground again (hip height)
  - Ds * 2 = Step-Length
  - Hao = Highest Point of the Foot during a step
  - Lao = Distance of Hao relative to the Point the foot leaves the Ground
  - qgs = Angle of the Ground when leaving the Ground
  - qb = Foot-Angle when leaving the Ground
  - qf = Foot-Angle when touching the Ground again
  - qge = Angle of the Ground when touching the Ground again
  - x_ed = Hip to Ankle distance at the start of the single support phase
  - x_sd = Hip to Ankle distance at the end of the single support phase
  - l_ab = Distance Ankle->Heel
  - l_af = Distance Ankle->Tiptoe
  - l_an = Distance Ankle->Sole

Laufen auf zwei Beinen

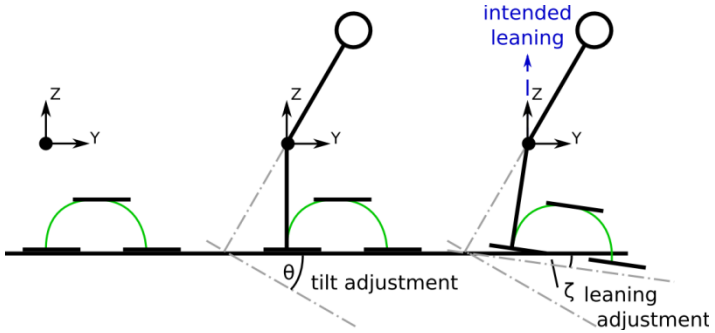
- Aus den Parametern werden Splines berechnet
- Parameter wurden gelernt
  - Tabu Suche
  - Genetisch
  - CMA-ES
- Probleme
  - Ursprüngliches Modell ist fürs geradeaus Laufen
  - Wir wollen in jede Richtung laufen und uns drehen können
  - Zu langsam

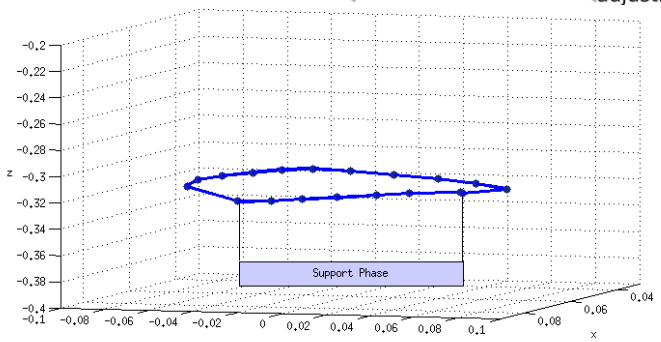


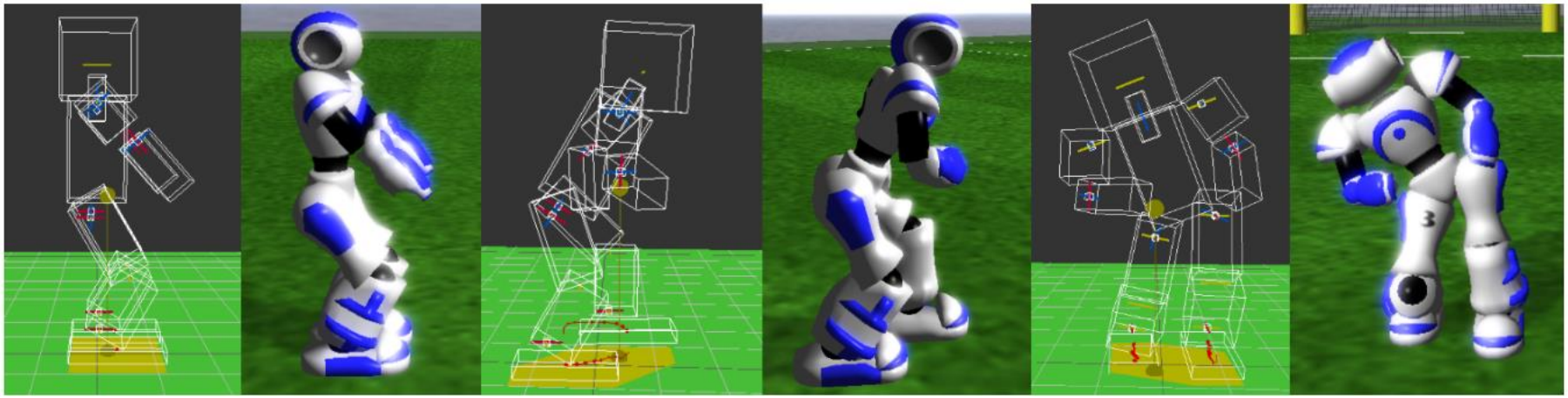


Laufen auf zwei Beinen

- Dynamische Anpassung der Bewegung an Lage des Torso
  - Invers kinematisch
  - Gekapselt in einem Framework
- Spezifikation
  - Fußtrajektorie
  - Gewünschte Torsoorientierung



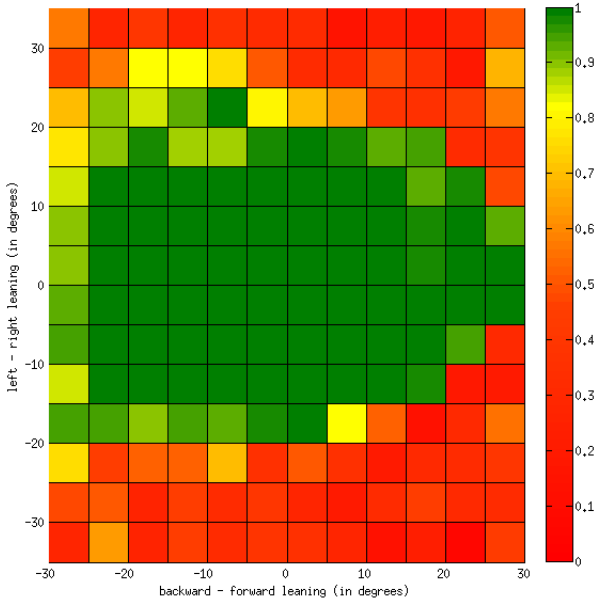




Glaser S and Dorer K (2013) Trunk Controlled Motion Framework. In Proceedings of the 8th Workshop on Humanoid Soccer Robots, IEEE-RAS International Conference on Humanoid Robots, Atlanta, October 2013

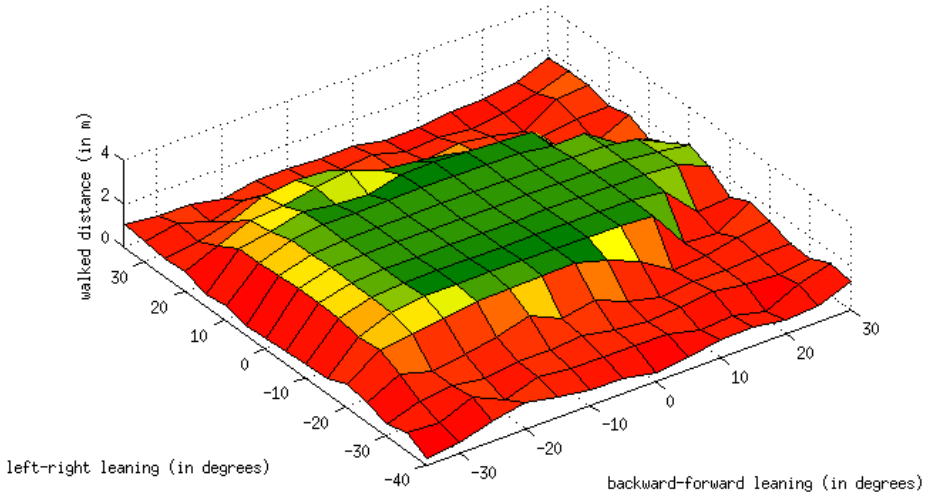
Laufen auf zwei Beinen

- Zuverlässigkeit
  - Variation von vorwärts und seitwärts Lage



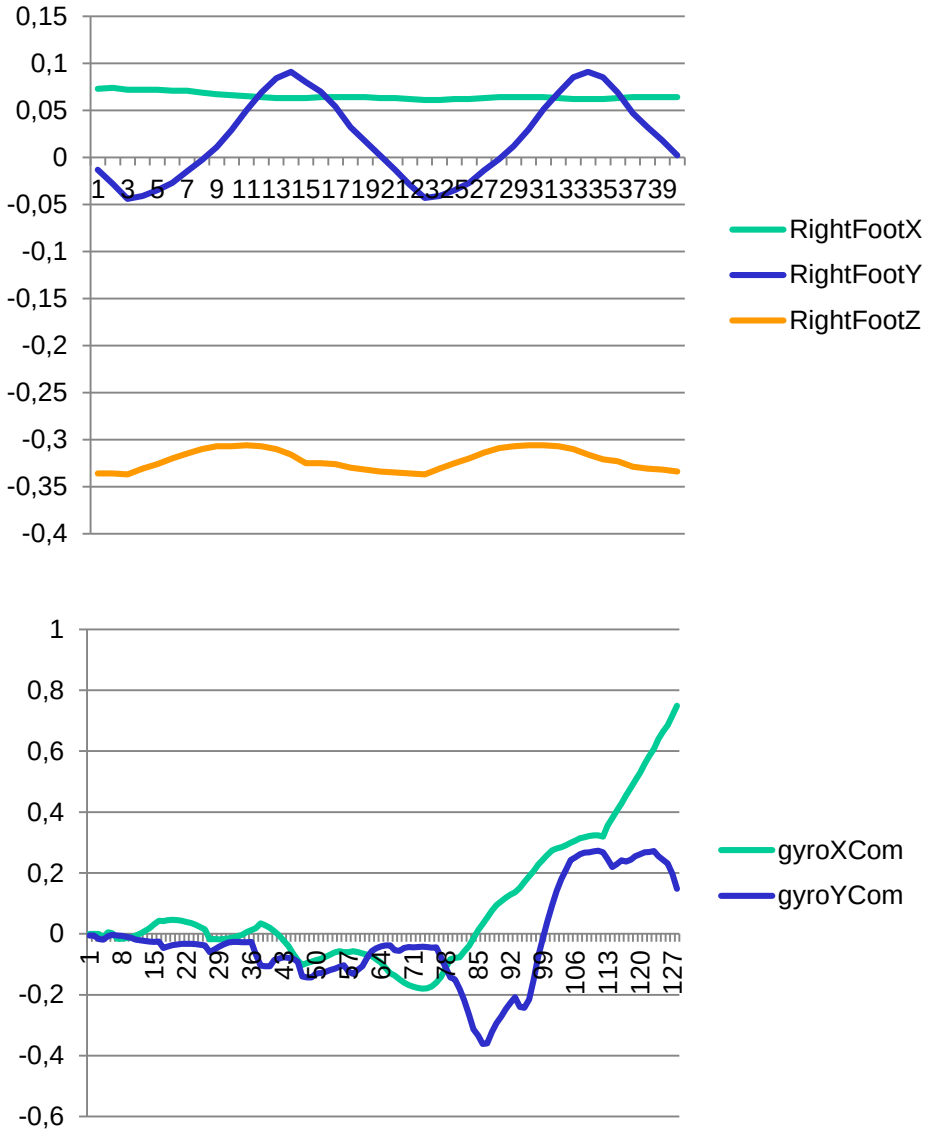
- Geschwindigkeit

Benchmark	Classic	Trunk controlled
Forward (m/s)	0.85	0.84
Backward (m/s)	0.75	**0.78**
Sideward (m/s)	0.48	**0.56**
Diagonal (m/s)	0.62	**0.73**
Turning (deg/s)	158.1	**179.4**



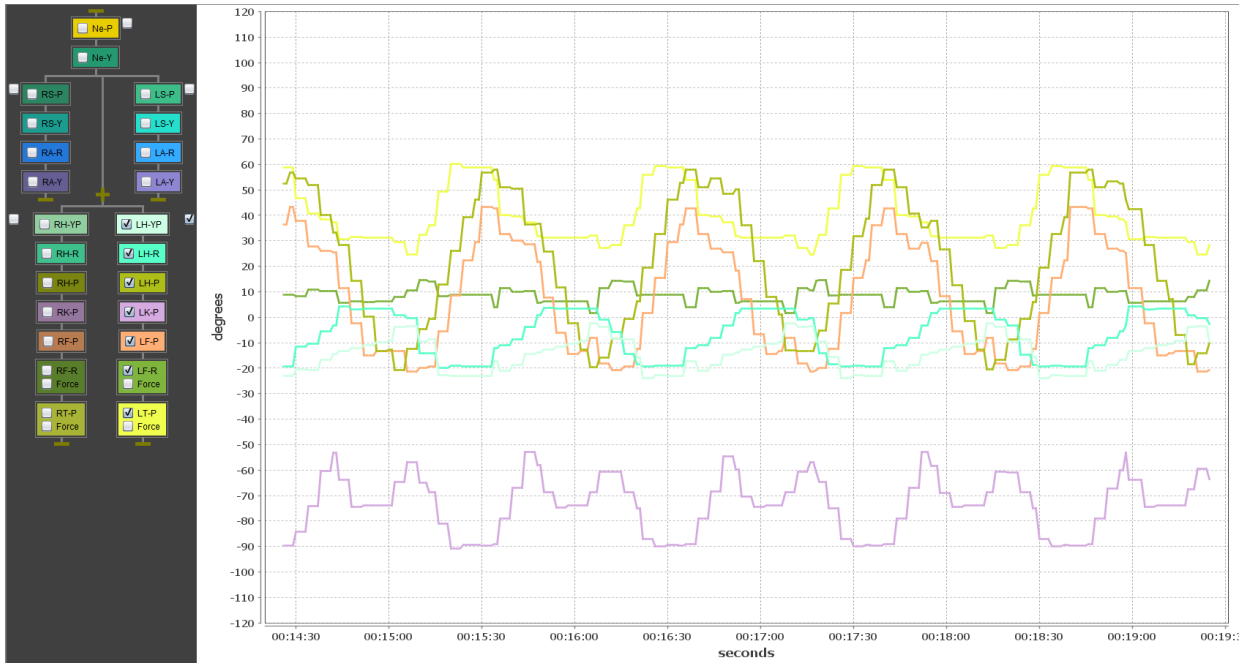
Laufen auf zwei Beinen

- Vorteile
  - Schneller (4:0)
  - Einfachere Kontrolle
  - Ermöglicht Schießen aus dem Lauf
- Probleme
  - Open Loop: schnelle statische Version berücksichtigt Lage und Beschleunigung des Roboters nicht -> fällt noch zu oft hin



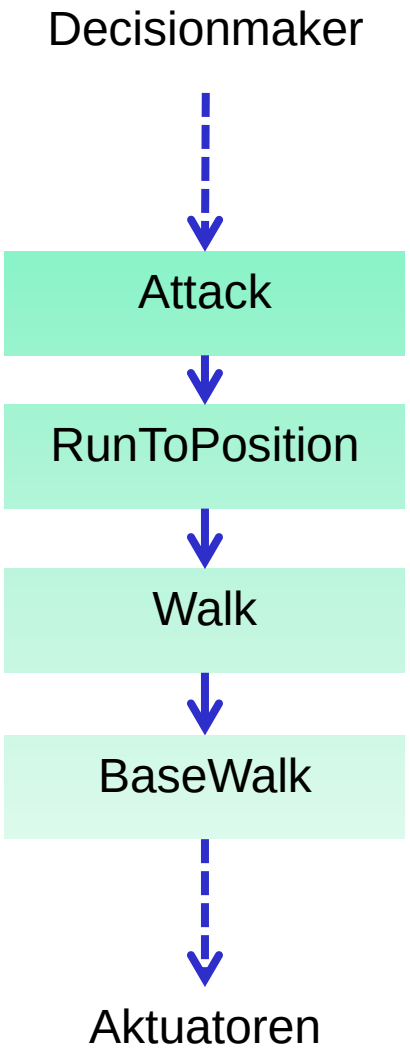
Laufen im Fußball

- Lernen einer Laufbewegung im Gelenkraum
  - 116 freie Parameter
  - Angle und Speed von sieben Gelenken pro Bein (28)
  - Für insgesamt vier Bewegungsphasen (112)
  - Dauer jeder Phase (116)
  - Modellfrei genetisch gelernt (Bewegung und Roboter)



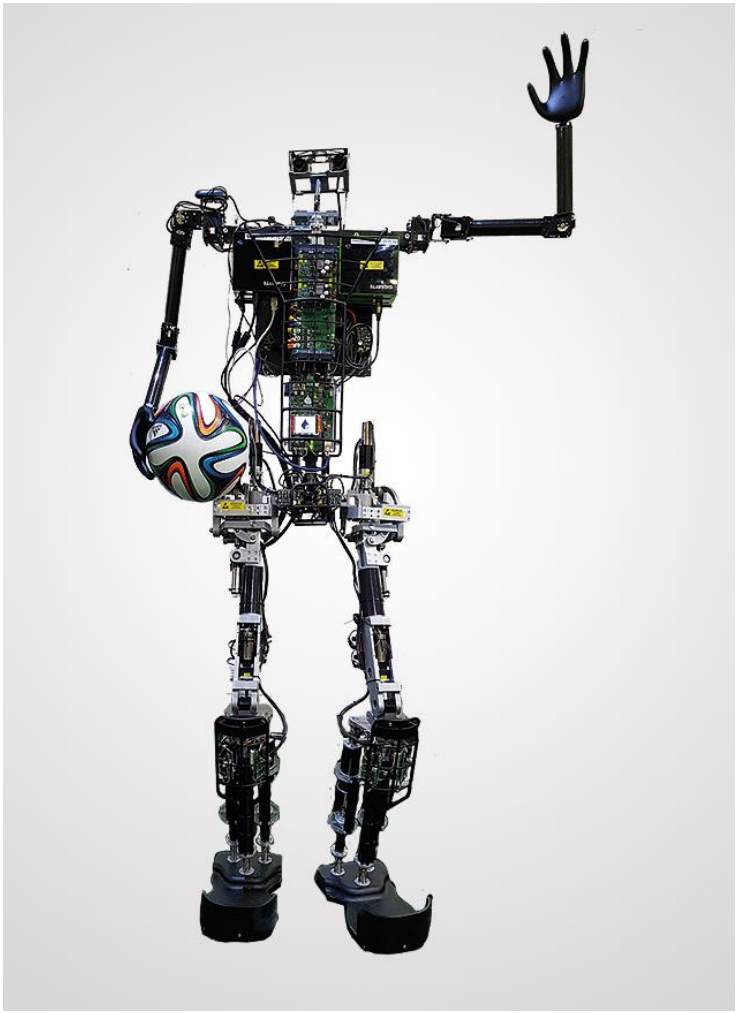
Laufen im Fußball

- Zum Ball rennen hat derzeit vier Ebenen
  - Ebene 4
    - Wohin soll ich rennen?
    - In welche Richtung will ich zum Ball stehen?
  - Ebene 3
    - Hindernisvermeidung
  - Ebene 2
    - Umrechnung ins lokale Koordinatensystem
    - Teilweise Anpassung an Ziel
  - Ebene 1
    - Laufen auf zwei Beinen
  - Höhere Ebenen instrumentieren niedrigere
  - Ebenen können übersprungen werden



Sweaty

- Zweibeiniger Roboter der Hochschule Offenburg
  - 22 Freiheitsgrade
  - Kamera
  - Gyro
  - Accelerometer
  - Drucksensoren
- Roboterbau
  - Prof. Hochberg
- Bildverarbeitung
  - Prof. Wülker
- Entscheidungssoftware
  - Prof. Dorer



Zusammenfassung

- Roboter Architekturen definieren, wie anhand von Sensordaten (und Zielen) Verhalten auszuführen sind
- Vorwärtskinematik erlaubt die Berechnung der Pose eines Körperteils
- Rückwärtskinematik erlaubt die Berechnung der Gelenkstellungen, um eine Pose zu erreichen
- Laufen auf zwei Beinen… ist schwierig

- Fürs Leben
  - Wer die Welt wahrnimmt ist klar im Vorteil