Entwicklung eines dynamischen Schusses für humanoide Fußballroboter

von Stefan Otte

Gutachter:
Prof. Dr. Raúl Rojas

Betreuer:
Hamid Moballegh, MSc

28. November 2010
Zusammenfassung


Eidesstattliche Erklärung

Ich versichere, die Bachelorarbeit selbstständig und lediglich unter Benutzung der angegebenen Quellen und Hilfsmittel verfasst zu haben. Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Berlin, den 28. November 2010

Stefan Otte
Inhaltsverzeichnis

1 Einleitung.................................................................................................................. 3
  1.1 Motivation und RoboCup .................................................................................. 3
  1.2 Die KidSize-Liga und die FUmanoids ............................................................... 4
  1.3 Zielsetzung ......................................................................................................... 5
  1.4 Struktur der Arbeit .............................................................................................. 6

2 Grundlagen.................................................................................................................. 7
  2.1 Steuerungstechnik und Regelungstechnik .......................................................... 7
  2.2 Kinematik in der Robotik .................................................................................... 8
  2.2.1 Terminologie und Grundlagen .................................................................... 9
  2.2.2 Direkte Kinematik ....................................................................................... 10
    Denavit-Hartenberg-Konvention ........................................................................ 11
  2.2.3 Inverse Kinematik ....................................................................................... 12
    Analytische Ansätze ......................................................................................... 13
    Piepers Ansatz ................................................................................................. 14
    Numerische Lösungsansätze .......................................................................... 15
  2.3 Fortbewegung und Stabilität ............................................................................. 16

3 Die FUmanoid-Plattform 2009 ............................................................................... 19
  3.1 Hardware ........................................................................................................... 19
  3.2 Sensoren ............................................................................................................ 20

4 Bewegungen im FUmanoid-System ....................................................................... 23
  4.1 Statische Motions ............................................................................................. 23
  4.2 Dynamische Motions – der Walker .................................................................. 25
    4.2.1 Pseudo-Inverse Kinematik ...................................................................... 26
    4.2.2 Regelung .................................................................................................. 27

5 Entwicklung der Schüsse ......................................................................................... 29
  5.1 Zielsetzung für den dynamischen Schuss ............................................................ 29
  5.2 Schuss im Laufen – der Mini-Schuss ................................................................. 29
    5.2.1 Umsetzung ............................................................................................... 30
    5.2.2 Probleme und Fazit ................................................................................ 32
  5.3 Stabilisierender Schuss im Stehen – der Kicker .................................................. 33
    5.3.1 Regelung und Stabilisierung des Schusses ................................................. 34
    5.3.2 Einteilung in Phasen ............................................................................... 36
    5.3.3 Einfachheit der Benutzung ..................................................................... 37
    5.3.4 Probleme und Fazit ................................................................................. 38
  5.4 Entwurf eines Schusses mit inverser Kinematik .................................................. 39
    5.4.1 Entwicklung der direkten Kinematik .......................................................... 39
5.4.2 Entwicklung der inversen Kinematik ........................................ 41
5.4.3 Analyse und Ausblick ............................................................. 45

6 Ausblick ...................................................................................... 46
6.1 Bessere Einbindung des Schusses in das Verhalten ...................... 46
6.2 Verbesserungen des Schusses ..................................................... 46
6.3 Entwicklung in anderen Gebieten ............................................. 47

7 Zusammenfassung ...................................................................... 48

Abbildungsverzeichnis ................................................................. 49

Literaturverzeichnis ................................................................. 50
1 Einleitung

Der RoboCup ist eine Veranstaltung, bei der Universitäten aus der ganzen Welt ihre Forschungsergebnisse im Bereich Künstliche Intelligenz (kurz KI) und Robotik vorstellen und vergleichen. Dies geschieht in Form eines Wettkampfes, in dem die Universitäten ihre Teams aus Fußball spielenden Robotern gegeneinander antreten lassen. Im RoboCup existieren verschiedene Ligen, die jeweils unterschiedliche Forschungsschwerpunkte aufweisen. In der Liga für humanoide Roboter ist besonders der zweibeinige Gang und Bewegung im Allgemeinen von Interesse.


1.1 Motivation und RoboCup

Es war abzusehen, dass der Mensch dem Computer nicht mehr lange überlegen sein wird, zumindest im Schach. 1997 war es dann soweit: der amtierende Schachweltmeister, Garry Kasparov, wurde vom Schachcomputer Deep Blue besiegt. Ein grandioser Erfolg für die KI; oder doch die Einsicht, dass Schach nicht die optimale Messlatte für die KI ist.


Wenn man KI nach Elaine Rich als

\begin{quote}
the study of how to make computers do things at which, at the moment, people are better\[18\]
\end{quote}

Fußball ist ein abgegrenztes Problemfeld was, wie Schach, einfach zu vergleichen ist. Es findet jedoch, anders als Schach, in einer hochdynamischen Umgebung statt. Die Roboter müssen selbstständig Daten erfassen. Diese unvollständigen und fehlerhaften Daten müssen zu einem kohärenten Bild der Welt umgeformt werden. Darauf aufbauend müssen die Spielsituationen bewertet werden um entsprechend reaktiv und/oder planend im Team zu kooperieren, mit dem Ziel letztendlich Tore zu schießen oder zu verhindern und das Spiel zu gewinnen. Zudem findet
1.2 Die KidSize-Liga und die FUmanoids

Dies alles in der wirklichen Welt statt. Insofern stellen Bewegungen eine große Herausforderung dar.


1.2 Die KidSize-Liga und die FUmanoids

Im RoboCup existieren, wie schon erwähnt, verschiedene Ligen für die Fußballroboter\(^1\) Anfangs gab es nur Ligen mit zentral gesteuerten fahrenden Robotern. Mit dem Ziel, den FIFA-Weltmeister 2050 zu schlagen, wurden immer mehr Ligen eingeführt und bestehende Ligen erweitert. Mit der Zeit wurden die Spielfelder und die Teams größer, es wurde mehr Augenmerk auf die Autonomie der Robotergestellt, und die Farbkodierung der Umwelt wurde reduziert.

Die Humanoid-Liga wurde 2005 eingeführt. Derzeit existieren drei Klassen in dieser Liga:

**KidSize**: Roboter bis zu einer Größe von 60 cm.

**TeenSize**: Roboter bis zu einer Größe von 130 cm.

**AdultSize**: Roboter bis zu einer Größe von 160 cm.

\[\text{Abbildung 1: Feld in der KidSize-Liga}\]


<table>
<thead>
<tr>
<th>Liga</th>
<th>Unterliga</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation</td>
<td>2D</td>
</tr>
<tr>
<td></td>
<td>3D</td>
</tr>
<tr>
<td>Small Size League</td>
<td>KidSize</td>
</tr>
<tr>
<td>Middle Size League</td>
<td>TeenSize</td>
</tr>
<tr>
<td>Humanoid-League</td>
<td>AdultSize</td>
</tr>
<tr>
<td>Standard Platform League</td>
<td>Rescue Simulation League</td>
</tr>
<tr>
<td></td>
<td>Rescue Robot League</td>
</tr>
<tr>
<td>RoboCupRescue</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboCup@Home</td>
<td></td>
</tr>
</tbody>
</table>

Tabelle 1: Liste aller Senior-Ligen im RoboCup 2010

1.3 Zielsetzung

Im Verhältnis zu anderen Ligen stellt das Schießen in den Humanoiden-Ligen eine besonders große Herausforderung dar. In der SmallSize- und MiddleSize-Liga können die Roboter einfach

³In den Jahren zuvor waren die ganzen Tore eingefärbt. Es existierten mehr Orientierungspfosten und es gab mehr Marker auf dem Feld.

In der KidSize-Liga sind solche Vorrichtungen nicht erlaubt. Der Roboter muss mit dem Fuß schießen und dabei die gleichen Servos verwendet, die den Roboter bewegen. Diese sind nicht zwangsläufig auf hohe Geschwindigkeiten ausgelegt. Außerdem steht der Roboter beim Schuss auf einem Bein und muss sein Gleichgewicht halten. Dies stellt ein weiteres großes Problem dar, was es zu bewältigen gilt.

Für Schüsse in der KidSize-Liga kommt meistens die Key-Frame-Technik zum Einsatz (siehe Abschnitt 2.3). Diese Methode ist sehr simpel und wird auch bei den Fumanoids verwendet. Sie ist jedoch denkbar ungeeignet für empfindliche Bewegungen wie den Schuss.

In dieser Arbeit sollen Alternativen zu dem Schuss auf Key-Frame-Basis entwickelt werden. Eine genauere Zielsetzung erfolgt in Abschnitt 5.1, nachdem das Fumanoid-System und die insbesondere die unterschiedlichen Bewegungsarten, inklusive der Probleme, analysiert worden sind.

1.4 Struktur der Arbeit

In Abschnitt 2 werden die Grundlagen von Bewegung erarbeitet. Was ist Kinematik? Wie kann man Bewegung steuern? Was ist Stabilität?

In Abschnitt 3 wird die Fumanoid-Plattform von 2009 im Detail vorgestellt.

In Abschnitt 4 wird die Bewegungserzeugung in der Fumanoid-Plattform beschrieben. Hierbei wird zum Einen besonders auf die statischen Bewegungen, die die Basis für den derzeitigen Schuss bilden, und zum Anderen auf den Walker, der die Basis für die Entwicklung der neuen Schüsse bildet, eingegangen.

In Abschnitt 5 werden dann die eigentlichen Schüsse entwickelt. Dazu gehört der Schuss im Laufen, der direkt in den Walker integriert wird, und der selbststabilisierende Schuss im Stehen. Ihre Vor-/Nachteile und Limitierungen werden besprochen. Aus dieser Diskussion folgt die Entwicklung eines kinematischen Modells der Roboter, auf der aufbauend die künftigen Schüsse und Bewegungen entwickelt werden können.

Abschnitt 6 diskutiert mögliche Erweiterungen des Schusses und Abschnitt 7 fasst die Arbeit zusammen.
2 Grundlagen

2.1 Steuerungstechnik und Regelungstechnik


- das System zu stabilisieren,
- Auswirkungen von Störungen zu beseitigen und
- das Verhalten des Systems zu optimieren.

In einem geschlossenen Regelkreis wird die *Regelgröße* \(y(t)\), der Istwert (die Istgeschwindigkeit des Autos), mit der *Führungsgröße* \(w(t)\), dem Sollwert (die Sollgeschwindigkeit des Autos), verglichen um die *Regelabweichung* \(e(t) = w(t) - y(t)\) (die Geschwindigkeitsdifferenz), zu berechnen. Die Regelabweichung \(e(t)\) wird in den Regler gegeben, der die *Stellgröße* \(u(t)\) in Abhängigkeit von der Regelabweichung \(e(t)\) berechnet, um das System entsprechend anzupassen. Die *Regelstrecke* entspricht dem System, welches zwischen Stellgröße \(u(t)\) und Regelgröße \(y(t)\) liegt (das System Auto mit allen Eigenschaften wie Gewicht, Beschleunigung, etc.). Zu der Regelstrecke gehören das *Stellglied* und *Messglied*. Das Stellglied wandelt das Regelsignal in ein Signal um, welches die Regelstrecke konkret beeinflusst (Gas geben mittels Gaspedal). Das *Messglied* misst die Regelgröße.

![Abbildung 2: Standardregelkreis mit Störungen – Closed-Loop-System](image)

In der Realität treten jedoch noch *Störungen* auf, die sowohl auf die Stellgröße, das Innere der Strecke, den Streckenausgang oder die Messung selbst einwirken. Störungen im Messglied \(m(t)\) müssen separat behandelt werden, da Messglieder oft eine eigene Dynamik aufweisen. Die restlichen Störungen lassen sich mit \(z(t)\) zusammenfassen.

Klassische Regler die in weit über 90% der Fälle eingesetzt werden sind die linearen *PID-Regler*. Diese behandeln den Proportional-, Integral-, und Differenzialteil der Regelabweichung. Als
Beispiel sei hier die Formel für einen PID-Regler gegeben:

\[ u(t) = K_P e(t) + K_I \int_0^t (e(\tau)) d\tau + K_D \frac{d(e(t))}{dt} \]

wobei \( K_P, K_I \) und \( K_D \) die Parameter für die einzelnen Regleranteile sind. Diese müssen so gewählt werden, dass die Regelungsaufgabe erfüllt wird.

Im Gegensatz zu der Regelungstechnik arbeitet die Steuerungstechnik komplett ohne Rückkopplung, weist also einen linearen Wirkungskreis auf. Im Englischen werden sie open loop systems genannt. Eventuelle Differenzen von Steuergröße \( y(t) \) und Führungsgröße \( w(t) \) werden nicht berechnet und nicht zur Korrektur genutzt. Solche Systeme können immer dann eingesetzt werden, wenn die Relation zwischen Eingabegröße und Ausgabegröße feststeht und nicht mit Störungen zu rechnen ist, oder diese tolerierbar sind.

![Abbildung 3: Eine einfache Steuerkette – open loop system](Abbildung3.jpg)

Der Tempomat als Open-Loop-System realisiert, würde nur mangelhaft funktionieren. Angenommen Soll- und Istgeschwindigkeit wären identisch, so würde eine kleine Steigung die Steuerung unwirksam machen. Der Vorteil von Open-Loop-Systemen ist, dass sie sehr einfach zu implementieren sein können und für bestimmte Aufgabenbereiche völlig ausreichende Ergebnisse erzielen.

2.2 Kinematik in der Robotik

Fast jeder Roboter manipuliert auf irgendeine Weise seine Umwelt. Dies kann unter anderem mit Greifern geschehen, oder mit einem Bein um den Ball zu schießen. Wie aber müssen die einzelnen Gelenke eines Roboterbeines stehen um den Fuß an eine bestimmte Stelle zu bewegen; oder wo steht der Greifer, wenn die einzelnen Gelenke des Greifers einen bestimmten Winkel haben? Der Mensch löst solche Probleme ständig und unbewusst. Für die Robotik stellen sie jedoch große Herausforderungen dar.

Die **Kinematik** als Teilgebiet der klassischen Mechanik ist die Lehre der Bewegung von Objekten im Raum. Für die Robotik lässt sich Kinematik präziser als die Lehre von der Relation der unabhängigen Variablen der Gelenke einer **kinematischen Kette** (siehe Abschnitt 2.2.2) und der kartesischen Position im Raum beschreiben. Die Objekte in der Kinematik sind starre Körper und Systeme von starren Körpern. Bei starren Körpern handelt es sich um idealisierte Körper, die nicht verformbar sind. Folglich haben zwei Punkte des Körpers immer den gleichen Abstand.
2.2 Kinematik in der Robotik

Die für diese Arbeit interessanten Probleme der Kinematik lassen sich in zwei Felder aufteilen: die direkte Kinematik (siehe Abschnitt 2.2.2) und die inverse Kinematik (siehe Abschnitt 2.2.3). Zuvor wird jedoch die Terminologie eingeführt, die zur Diskussion nötig ist.

2.2.1 Terminologie und Grundlagen


Abbildung 4: Roll-Nick-Gier [20].

In der Kinematik kann jede Bewegung eines Objektes im Raum durch eine Transformation (Translationen und Rotationen) beschrieben werden. Dabei kommen homogene Koordinaten zum Einsatz. Homogene Koordinaten in \(\mathbb{R}^3\) werden durch vier Koordinaten \((x, y, z, w)\) mit \(w \neq 0\)

\[\text{Es existieren auch alternative Repräsentationen, die sich für bestimmte Aufgaben besser eignen. Diese werden hier jedoch vernachlässigt.}\]
dargestellt. \( w \) wird in den meisten Fällen gleich 1 gesetzt und entspricht nur einer Skalierung.
Der Punkt \((x, y, z, w)\) entspricht dabei dem Punkt \((\frac{x}{w}, \frac{y}{w}, \frac{z}{w})\). Der Vorteil homogener Koordinaten ist, dass sich jede affine Transformation als einfache Matrixmultiplikation durchführen lässt.

Eine Rotation um die \(x\)-Achse um den Winkel \(\alpha\) wird durch die Rotationsmatrix
\[
R_x(\alpha) = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \alpha & -\sin \alpha & 0 \\
0 & \sin \alpha & \cos \alpha & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
realisiert. Eine Rotation um die \(y\)-Achse um den Winkel \(\beta\) wird durch die Rotationsmatrix
\[
R_y(\beta) = \begin{pmatrix}
\cos \beta & 0 & \sin \beta & 0 \\
0 & 1 & 0 & 0 \\
-\sin \beta & 0 & \cos \beta & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
realisiert. Eine Rotation um die \(z\)-Achse um den Winkel \(\gamma\) wird durch die Rotationsmatrix
\[
R_z(\gamma) = \begin{pmatrix}
\cos \gamma & -\sin \gamma & 0 & 0 \\
\sin \gamma & \cos \gamma & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
realisiert. Folgende Rotationsmatrix rotiert um alle Achsen um beliebige Winkel \(\alpha, \beta, \gamma\):
\[
R_{xyz}(\alpha, \beta, \gamma) = R_x(\alpha) \times R_y(\beta) \times R_z(\gamma)
\]
Eine Translation um \((d_x, d_y, d_z)^T\) mit homogenen Koordinaten lässt sich mit folgender Matrix darstellen:
\[
\text{Translation}(d_x, d_y, d_z) = \begin{pmatrix}
1 & 0 & 0 & d_x \\
0 & 1 & 0 & d_y \\
0 & 0 & 1 & d_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

### 2.2.2 Direkte Kinematik

2.2 Kinematik in der Robotik

Etwas formaler: die Pose des Endeffektors $x, y, z, \alpha, \beta, \gamma$ soll als Funktion von Gelenkparametern $q_1, \ldots, q_n$ ausgedrückt werden:

$$f_k(q_1, q_2, \ldots, q_n) \rightarrow (x, y, z, \alpha, \beta, \gamma)$$

Das FK-Problem entspricht dabei dem Problem, das Koordinatensystem des Endeffektors in das Koordinatensystem des Bezugsrahmens zu überführen. Zwei unterschiedliche Koordinatensysteme können mittels einer Transformation ineinander überführt werden. Jedes Koordinatensystem eines Gelenks wird dazu sukzessiv mittels einer Transformation in das Koordinatensystem des Bezugsrahmens überführt:

$$^{0}_nT = \prod_{i=1}^{n} i^{-1}T_i(\delta_i) \quad (1)$$

wobei $i^{-1}T_i$ die Transformation des Ortskoordinatensystems von Glied $i$ in das Ortskoordinatensystem von Glied $i - 1$ ist. $\delta_i$ sind die Parameter der Transformationsmatrix, also Verschiebungen und Rotationen. Es gibt viele Wege die Transformationsmatrizen zu erzeugen. Die Denavit-Hartenberg-Konvention hat sich dabei als Standardmethode etabliert.

**Denavit-Hartenberg-Konvention:** Der Vorteil der DH-Konvention ist, dass eine minimale Darstellung verwendet wird – vier Parameter beschreiben jedes Glied-Gelenk-Paar. Diese Parameter können einfach erlangt werden. Dazu nutzt die DH-Konvention homogene Matrizen und ein spezielles Schema zur Festlegung der Ortskoordinatensysteine für jedes Glied/Gelenk. Jede Transformation $i^{-1}T_i$ wird dabei als das Produkt von vier Basis-Transformationen beschrieben:

1. die Rotation $\theta_i$ um den Gelenkwinkel

   $$Rot_{z, \theta_i} = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. eine Translation $d_i$ um den Gelenkabstand

   $$Trans_{z, d_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. eine Translation $a_i$ um die Armelementlänge

   $$Trans_{x, a_i} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
4. eine Rotation $\alpha_i$ um die Verwindung

$$
Rot_{x,\alpha_i} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \alpha_i & -\sin \alpha_i & 0 \\
0 & \sin \alpha_i & \cos \alpha_i & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
$$

$i^{-1}T_i$ ergibt sich wie folgt:

$$
i^{-1}T_i = Rot_{x,\theta_i} \times Trans_{z,d_i} \times Trans_{x,a_i} \times Rot_{x,\alpha_i}
$$

$a, \alpha$ sind konstante Werte und durch den Aufbau der kinematischen Kette des Roboters gegeben. $\theta, d$ sind die Gelenkvariablen. Handelt es sich um ein Rotationsgelenk, so ist $\theta$ variabel und $d$ konstant; handelt es sich um ein Schubgelenk, so ist $d$ variabel und $\theta$ konstant. $\theta_i, a_i, d_i, \alpha_i$ sind die Parameter die Glied/Gelenk $i$ bestimmen. Man erhält also eine Tabelle von Parametern, die die kinematische Kette beschreiben.

Mit den so erhaltenen Transformationsmatrizen und Formel (1) kann man sukzessiv die Koordinatensysteme ineinander überführen und so die FK lösen.

### 2.2.3 Inverse Kinematik

Die inverse Kinematik ist das Gegenstück zur direkten Kinematik. Sie beantwortet die Frage, welche Werte die Gelenkparameter bei gegebener Pose des Endeffektors haben. Etwas formaler ausgedrückt, wollen wir die Gelenkparameter $q_1, q_2, \ldots, q_n$ als Funktion der Pose des Endeffektors $x, y, z, \alpha, \beta, \gamma$ ausdrücken:

$$
i_k(x, y, z, \alpha, \beta, \gamma) \rightarrow (q_1, q_2, \ldots, q_n)
$$

Das IK-Problem ist ungleich schwerer als das der FK, da es das Lösen von nichtlinearen Gleichungen beinhaltet. Nichtlineare Gleichungen müssen nicht zwingend eine Lösung haben, und falls Lösungen existieren, müssen diese nicht eindeutig sein.

Übertragen auf die Mechanik von kinematischen Keten lassen sich einige der Probleme der IK sehr schön verdeutlichen. Es stellen sich die folgenden Fragen:

- Ist die Zielposition in Reichweite der kinematischen Kette?
- Verfügt die kinematische Kette über genügend Freiheitsgrade und sind diese richtig angeordnet um eine gewisse Pose einzunehmen?

---

*Pieper hat gezeigt, dass mit einem 6-DoF-Greifer, der die letzten drei Gelenke als ein spherisches Gelenk (die drei Gelenke schneiden sich in einem Punkt) realisiert, die Gleichungen immer lösbar sind, also jede beliebige Pose in Reichweite eingenommen werden kann [17].
Falls die Pose auf mehrere Arten erreichbar ist (redundante Gelenke), welche Konfiguration ist die ideale?

Falls die Pose nicht erreichbar ist, wie soll die kinematische Kette sich verhalten?

Ferner müssen auch die Fälle von unzulässigen Konfigurationen berücksichtigt werden, bei denen es z.B. Kollisionen der kinematischen Kette selbst oder mit der Umwelt geben würde. Dies wird aber nur aus Gründen der Vollständigkeit erwähnt und ist nicht Teil dieser Arbeit.

Abbildung 5 stellt einige der Probleme anhand eines einfachen plenaren Greifers dar.

Abbildung 5: IK: mögliche Lösungen und Probleme bei einer einfachen kinematischen Kette in 2D. Der kleine Punkt ist die Zielposition der kinematischen Kette.

Aufgrund dieser Probleme gibt es keine Standardlösung für die IK. Da oft Echtzeitanforderungen an Roboterarme und somit an die IK gestellt werden, wird die Lösung der IK oft auf den Roboter angepasst oder der Roboter wird so gebaut, dass die Lösung der IK einfach ist.

Je nach kinematischer Kette lässt sich die IK mit Methoden der geschlossenen Form (analytische Ansätze) oder Methoden der nicht geschlossenen Form (numerische Ansätze) lösen.

**Analytische Ansätze:** Bei simplen kinematischen Ketten lässt sich die IK sehr effizient mittels Trigonometrie lösen. Der Endeffektor einer kinematischen Kette mit nur einem Rotationsgelenk
bewegt sich z.B. auf dem Kreis mit dem Radius der Länge des Gliedes der kinematischen Kette. Solche kinematischen Ketten sind allerdings eher uninteressant.


Angenommen wir haben eine kinematische Kette mit sechs Freiheitsgraden bei der sich die letzten drei Gelenke in einem Punkt schneiden (siehe Abbildung 6). Das Hüftgelenk sei der Endeffektor an der Fuß der Bezugsrahmen.

![](image)

2.2 Kinematik in der Robotik

In solch einer kinematischen Kette bestimmen die ersten drei Glieder die Position des Endeffektors, die letzten drei bestimmen die Orientierung. Erst wird die inverse Positions kinematik für die ersten drei Glieder mittels eines geometrischen Ansatzes gelöst. Die erhaltenen Parameter für die ersten drei Gelenke bestimmen so das Ortskoordinatensystem des sphärischen Gelenks, der Hüfte. Anschließend werden bei der inversen Rotationskinematik die restlichen Parameter für die Rotation des sphärischen Gelenks in Abhängigkeit zu dem Ortskoordinatensystems der Hüfte berechnet.

Piepers Ansatz stellt einen einfachen und sehr effizienten Weg dar, die IK zu lösen. Die meisten Industrieroboter weisen einen Aufbau auf, der das Anwenden der Piepers Solution ermöglicht.

**Numerische Lösungsansätze:** Numerische Ansätze kommen immer zum Einsatz, wenn es keine einfachen geometrischen oder algebraischen Lösungen gibt. Bei numerischen Ansätzen handelt es sich um Verfahren, die sich iterativ der Lösung annähern.

Das IK-Problem wird hierbei als Optimierungsproblem formuliert. Es soll eine Menge von Parametern gefunden werden (die Gelenkparameter), die einen Fehler (den Abstand des Endeffektors zum Zielpunkt) minimieren.

Die klassische Lösung für diese Art von Problemen ist der Gradientenabstieg. Der Gradient kann mit den üblichen Methoden bestimmt werden (Ableitungen, Random-Search, Particle-Swarm-Optimization, etc.). Je nach kinematischer Kette sind unterschiedliche Methoden unterschiedlich erfolgreich. Es kann jedoch zu den üblichen Problemen beim Gradientenabstieg kommen: mehrere Minima/Maxima, überspringen von Minima/Maxima, zu langsamer Gradientenabstieg, etc.


\[ J(\theta) = \left( \frac{\partial s_i}{\partial \theta_j} \right)_{i,j} \]

Generell wird die Jacobi-Matrix zur Approximation von mehrdimensionalen Funktionen verwendet. Im Fall der IK stellt die Jacobi-Matrix die lineare Approximation der direkten Kinematik \( f_k \) dar und approximiert wie sich die Pose \( x, y, z, \alpha, \beta, \gamma \) verändert, wenn man die Gelenkparameter \( q \) anpasst. Abbildung verdeutlicht den Zusammenhang. Mittels der invertierten oder der transponierten Jacobi-Matrix kann man so die Bewegung abschätzen und iterativ eine Lösung suchen.

2.3 Fortbewegung und Stabilität

Es existieren viele Ansätze, um den zweibeinigen Gang zu realisieren. Einige sind auf genaue Modelle angewiesen, andere versuchen mit einfachen Methoden den Gang zu stabilisieren und wieder andere machen sich die Dynamik des Systems zu eigen. Im Folgenden werden die verbreitetsten Ansätze kurz vorgestellt.

Ein simpler Ansatz einen Gang zu implementieren basiert auf der Key-Frame-Technik\(^5\). Dabei stellen Key-Frames Posen des Roboters dar\(^6\). Ein Key-Frame ist durch eine Liste der Servopositionen gegeben. Diese Key Frames werden nacheinander abgespielt.


\(^5\)Die Key-Frame-Methode kann natürlich auch für andere Bewegungen genutzt werden (siehe Abschnitt 4.1).
\(^6\)Die Posen des Roboters sind nicht mit den Posen des Endeffektors zu verwechseln.
2.3 Fortbewegung und Stabilität

Abbildung 8: Beispielhaftes CPG von Team Nimbro aus [1].

CPGs sind von der Natur inspiriert und kommen als neuronale Netze vor. Diese erzeugen ebenfalls Schwingungen die so Bewegungsmuster für Teile des Körpers, wie Herzschlag oder Kaubewegung, erzeugen. CPGs spielen auch eine Rolle beim Gang von Wirbeltieren und dem Menschen [16, 6].


Bei der dynamischen Stabilität werden dynamische Effekte der Bewegung mit einbezogen und es können so höhere Geschwindigkeiten erreicht werden. Der sogenannte Zero Moment Point
(kurz ZMP) ist das dynamische Stabilitätskriterium [22, 21]. Dabei werden alle Kräfte und Momente berücksichtigt, die auf das Fußgelenk des stehenden Fusses des Roboters einwirken (siehe Abbildung 9). Der ZMP beschreibt dabei den Punkt, an dem sich alle Kräfte und Momente aufheben. Sofern dieser ZMP in der konvexen Hülle aller Kontaktpunkte des Bodens liegt, ist der Roboter dynamisch stabil.

Abbildung 9: ZMP aus [22].


3 Die FUmanoid-Plattform 2009


3.1 Hardware

Die Hardware wurde im Rahmen einer Diplomarbeit 2009 von Mariusz Kukulski von Grund auf neu entworfen. Die Roboter wiegen ca. 4,3 kg bei einer Höhe von 59 cm.


Dynamixel-Servos haben einen 8-bit Mikrocontroller der sich um die Ansteuerung der Servos kümmert. Befehle können bequem mittels eines von Robotis spezifizierten Protokolls über einen seriellen Halb-Duplex-Bus (RS-485) an die Servos geschickt werden. Zu den unterstützten

⁷In früheren Versionen der 2009-Modelle gab es einen zusätzlichen Servo für die Kopfneigung.
3.2 Sensoren

Befehlen gehören unter anderem das Setzen der Motorposition und der Drehkraft, das Abfragen der Motorposition, der Belastung, der Temperatur, und viele mehr. Die Servos verfügen nur über einen P-Regler (siehe Abschnitt 2.1). Dies kann dazu führen, dass sie unter Last ihre Positionen nicht anfahren können.

<table>
<thead>
<tr>
<th></th>
<th>RX-28</th>
<th>RX-64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gewicht</td>
<td>67 g</td>
<td>125 g</td>
</tr>
<tr>
<td>Masse</td>
<td>36 x 51 x 36 mm</td>
<td>40 x 61 x 41 mm</td>
</tr>
<tr>
<td>Betriebsspannung</td>
<td>12-18,5 V</td>
<td>12-18,5 V</td>
</tr>
<tr>
<td>Haltemoment</td>
<td>37 kgf-cm</td>
<td>52 kgf-cmA</td>
</tr>
<tr>
<td>Geschwindigkeit</td>
<td>67 RPM</td>
<td>64 RPM</td>
</tr>
<tr>
<td>Verbindung</td>
<td>RS-485</td>
<td>RS-485</td>
</tr>
<tr>
<td>Positionssensor</td>
<td>Potentiometer</td>
<td>Potentiometer</td>
</tr>
<tr>
<td>Auflösung</td>
<td>1024/300°</td>
<td>1024/300°</td>
</tr>
</tbody>
</table>

Tabelle 2: Eigenschaften der Dynamixel-Servos.


Als Hauptrecheneinheit kommt ein 80 x 20 mm großer verdex pro der Firma Gumstix Inc. zum Einsatz. Der verdex pro verfügt über einen ARM-Prozessor auf PXA270-Basis mit 600 MHz und 128 MB RAM. Er ist die einzige Recheneinheit im Roboter und ist so für alle anfallenden Berechnungen zuständig. Dazu gehören die Bildverarbeitung, die Selbstlokalisierung, die Weltmodellierung, das Verhalten, die Bewegungen, die Netzwerk- und die Motorkommunikation.

3.2 Sensoren

Das Regelwerk des RoboCups beschränkt die Sensorik der Roboter auf passive, menschenähnliche Sensoren. Das bedeutet, dass keine Laserscanner, Ultraschallensoren, etc. benutzt werden dürfen.

3.2 Sensoren

DIE FUMANOID-PLATTFORM 2009

(a) Kinematisches Modell und interne Benennung der Motoren inklusive der Motoren-ID

(b) Roboter Tim beim RoboCup 2009 in Graz

Abbildung 11: Roboter 2009

Abbildung 12: *verdex pro* der Firma Gumstix Inc.
Öffnungswinkel von 175°. Die ermöglicht es dem Roboter die Füße und gleichzeitig das gegnerische Tor zu sehen, wenn der Roboter im eigenen Tor steht. Deswegen ist kein Servo für die Kopfnieigung nötig.


Ferner verfügen die Roboter über je vier Schalter unter jedem Fuß. Sie zeigen an, ob der Fuß den Boden berührt oder nicht. So kann festgestellt werden, ob der Roboter noch steht, bzw. wann ein neuer Schritt beim Laufen initialisiert werden soll.


\*Die Rotation des Roboters, die für die Selbstlokalisierung sehr hilfreich wäre, kann aufgrund der fehlenden sechsten Achse leider nicht berechnet werden.
4 Bewegungen im FUmanoid-System

Im FUmanoid-System existieren zwei Arten von Motions: statische Motions und dynamische Motions.

4.1 Statische Motions


<table>
<thead>
<tr>
<th>Motor 1</th>
<th>Motor 2</th>
<th>Motor 3</th>
<th>Motor 4</th>
<th>Motor 5</th>
<th>Motor 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,508</td>
<td>0,507</td>
<td>0,507</td>
<td>0,507</td>
<td>0,507</td>
<td>0,507</td>
</tr>
<tr>
<td>0,520</td>
<td>0,519</td>
<td>0,519</td>
<td>0,519</td>
<td>0,519</td>
<td>0,519</td>
</tr>
<tr>
<td>0,852</td>
<td>0,851</td>
<td>0,851</td>
<td>0,851</td>
<td>0,851</td>
<td>0,851</td>
</tr>
<tr>
<td>1,82</td>
<td>1,82</td>
<td>1,82</td>
<td>1,82</td>
<td>1,82</td>
<td>1,82</td>
</tr>
<tr>
<td>2,40</td>
<td>2,40</td>
<td>2,40</td>
<td>2,40</td>
<td>2,40</td>
<td>2,40</td>
</tr>
<tr>
<td>0,20</td>
<td>0,20</td>
<td>0,20</td>
<td>0,20</td>
<td>0,20</td>
<td>0,20</td>
</tr>
<tr>
<td>0,508</td>
<td>0,507</td>
<td>0,507</td>
<td>0,507</td>
<td>0,507</td>
<td>0,507</td>
</tr>
<tr>
<td>0,520</td>
<td>0,519</td>
<td>0,519</td>
<td>0,519</td>
<td>0,519</td>
<td>0,519</td>
</tr>
<tr>
<td>0,852</td>
<td>0,851</td>
<td>0,851</td>
<td>0,851</td>
<td>0,851</td>
<td>0,851</td>
</tr>
<tr>
<td>1,82</td>
<td>1,82</td>
<td>1,82</td>
<td>1,82</td>
<td>1,82</td>
<td>1,82</td>
</tr>
<tr>
<td>2,40</td>
<td>2,40</td>
<td>2,40</td>
<td>2,40</td>
<td>2,40</td>
<td>2,40</td>
</tr>
<tr>
<td>0,20</td>
<td>0,20</td>
<td>0,20</td>
<td>0,20</td>
<td>0,20</td>
<td>0,20</td>
</tr>
</tbody>
</table>


Generell handelt es sich bei den statischen Motions um ein Open-Loop-System. Einmal gestartet, wird eine statische Motion bis zum Ende abgespielt. Statische Motions können zwar auch unterbrochen werden – beispielsweise, wenn die Füße keinen Bodenkontakt mehr haben und der Roboter fällt – sonst fließt aber kein Feedback in die Bewegung ein.

Statische Motions sind typischerweise Aufstehbewegungen, Unterhaltungsbewegungen (z.B. Winken, Klatschen und Jubeln), der Einwurf eines Balles, aber auch der Schuss. Für einfache Bewegungen, wie die Wink-Bewegung, eignen sich statische Motion bestens, da die Relation zwischen verändertem Motorwert und resultierender Bewegung bekannt ist und die Auswirkung auf das System minimal sind.

Für störanfällige Bewegungen, wie z.B. den Schuss, eignen sich statische Motions allerdings weniger gut. Balanciert der Roboter beim Schuss auf einem Bein, ist es nicht mehr klar, was passiert, wenn der Arm um einige Grad gehoben wird. Die Bewegung kann sich stabilisierend auswirken oder den Roboter zu Fall bringen. Kleine Unterschiede in der Startposition bei dem

* Aus Kompatibilitätsgründen zu alten Robotern beinhaltet die Liste 22 und nicht 21 Servos, da anfänglich ein Servo für die Kopfneigung vorhanden war. Der letzte Wert ist die Zeitangabe.
4.1 Statische Motions

Abbildung 13: FUremote mit einigen Plugins aktiviert.

Abbildung 14: MotionEditor in FUremote mit einem linken Schuss in Bearbeitung
4.2 Dynamische Motions – der Walker

Schuss können darüber entscheiden, ob es ein stabiler Schuss wird, oder ob der Roboter fällt. Ferner kann der Roboter beim Schuss auch einen anderen Roboter berühren oder die Motoren können durch das Spiel heiß geworden sein und andere Eigenschaften als beim Aufnehmen des Schusses aufweisen.

Statische Motions werden mit dem MotionEditor-Plugin (siehe Abbildung 14) von FUremote (siehe Abbildung 13) aufgenommen. FUremote ist das Administrationsprogramm für die FUmanoids-Software auf Eclipse/RCP-Basis.

Das Aufnehmen einer Motion läuft wie folgt ab:

- Jede Bewegung beginnt mit der Idle-Pose, dem aufrechten Stehen.
- Die Servos, deren Position geändert werden sollen, werden ausgestellt.
- Selbige Servos werden per Hand in die entsprechende Zielposition gebracht.
- Die Servos werden wieder angestellt.
- Die Position der Servos wird gelesen.
- Die neue Pose wird mit einer Zeit versehen, die die Geschwindigkeit für den Posenübergang bestimmt.
- Diese Schritte werden solange wiederholt, bis die Bewegung komplett ist.
- Anschließend wird die Motion per Copy&Paste in den C++-Quellcode eingefügt und im System verfügbar gemacht.

Auch wenn bei den statischen Motions darauf geachtet wird, dass jede Pose stabil ist, so ist aufgrund der entstehenden Momente trotzdem nicht garantiert, dass die Bewegung als Ganzes ausgeführt, stabil ist.

4.2 Dynamische Motions – der Walker

Bestimmte Arten von Bewegungen, wie z.B. der zweibeinige Gang, lassen sich nur zu einem bestimmten Maß als Open-Loop-System realisieren. Im FUmanoid-System stellen die dynamischen Motions die Implementierung von Closed-Loop-Systemen dar.

Der Walker, bis jetzt die einzige dynamische Motion, ist für die omnidirektionalen zweibeinigen Gang zuständig. Dabei wird durch ein Feedback (aus IMU, Positions messungen der Servos und Fußschalter) die laufende Bewegung angepasst und stabilisiert. Dies geschieht z.B. durch Anpassung der Schrittweite, oder Zurücksetzen der Schrittphase sobald der schwingende Fuß den Boden berührt.

Dynamische Motions werden im Gegensatz zu statischen Motions nicht Schritt für Schritt aufgenommen. Sie stellen vielmehr ein System dar, welches über Parameter konfiguriert werden kann. Der Walker beschreibt zum Beispiel die Schrittweite, Zeit für einen Schritt, Seitwärts schwingung, etc. und passt diese entsprechend dem Feedback an. Um die Bewegung im Raum
zu beschreiben wird die sogenannte *Pseudo Inverse Kinematik* genutzt (siehe Abschnitt 4.2.1). Auf dieser aufbauend wird der Gang geregelt (siehe Abschnitt 4.2.2).

### 4.2.1 Pseudo-Inverse Kinematik


[Abbildung 15: PIK bei den Modellen 2010. Verkürzung des Beines.](image)

Um das Prinzip der PIK zu verdeutlichen, seien hier ein paar Bewegungen beschrieben. Das Bein des Roboters lässt sich zum Beispiel verkürzen, indem HIP_PITCH, KNEE_TOP, KNEE_BOTTOM und FOOT_PITCH bewegt werden (siehe Abbildung 15). Der Roboter kann sich zur Seite lehnen, indem der Servo FOOT_ROLL bewegt wird. Um den Oberkörper bei geneigtem Fuß senkrecht zu halten, muss der Servo HIP_ROLL entsprechend in die andere Richtung bewegt werden.
Ferner wird direkt mit Motorpositionen (1024 Positionen für den Öffnungswinkel von 300 deg) und nicht mit Winkeln gerechnet um die Servos anzusteuern. So lassen sich teure Berechnungen einsparen.

### 4.2.2 Regelung


\[
E_P = mgh \quad (2)
\]

\[
E_K = \frac{1}{2}mv^2 \quad (3)
\]

\[
E_{total} = E_P + E_K = mgh + \frac{1}{2}mv^2 \quad (4)
\]

Während der Schwingphase, die als invertiertes Pendulum modelliert werden kann, ist die Energie des Walkers konstant.


Durch die Analyse des Simplest Walkers wurden Parameter identifiziert, die die Energie des Walkers kontrollieren und den Gang steuern können. Die PIK wurde so aufgebaut, dass der Parameterraum zum Energetisieren und Steuern des Gangs direkt in den Gelenkraum der Servos

\(^{10}\)Andere Methoden zum Energetisieren wurden ebenfalls getestet: Änderung von FOOT_PITCH oder des Oberkörpers.
abgebildet wird. Hier ist ein Ausschnitt der PIK:

\[
\text{PIK} = \begin{pmatrix}
\text{StanceFoot} \\
\text{FlyingLegContraction} \\
\text{StanceLegContraction} \\
\text{COMShiftFrontal} \\
\text{COMShiftLateral} \\
\text{StepLengthFrontal} \\
\text{StepLengthLateral} \\
\text{HipRotation} \\
\text{TorsoAngle}
\end{pmatrix}
\rightarrow
\begin{pmatrix}
\text{LEFT\_HIP\_YAW} \\
\text{RIGHT\_HIP\_YAW} \\
\text{LEFT\_HIP\_ROLL} \\
\text{RIGHT\_HIP\_ROLL} \\
\text{LEFT\_HIP\_PITCH} \\
\text{RIGHT\_HIP\_PITCH} \\
\text{LEFT\_KNEE\_TOP} \\
\text{RIGHT\_KNEE\_TOP} \\
\text{LEFT\_KNEE\_BOTTOM} \\
\text{RIGHT\_KNEE\_BOTTOM} \\
\text{LEFT\_FOOT\_PITCH} \\
\text{RIGHT\_FOOT\_PITCH} \\
\text{LEFT\_FOOT\_ROLL} \\
\text{RIGHT\_FOOT\_ROLL}
\end{pmatrix}
\]
5 Entwicklung der Schüsse

5.1 Zielsetzung für den dynamischen Schuss

Hier seien die Probleme des statischen Schusses zusammengefasst. Aus diesen Problemen ergeben sich die genauen Zielsetzungen für die neuen Schüsse.


Der Schuss als statische Motion kann nicht auf Störungen reagieren und sich nicht stabilisieren. Oft wird der Roboter während der Schussbewegung von einem gegnerischen Roboter berührt und zu Fall gebracht. Auch wenn die Roboter einer Familie alle gleich aufgebaut sind, gibt es minimale Unterschiede durch Bau und Abnutzung. Diese Unterschiede werden zwar mit individuellen Anpassungen der Motorpositionen (Offsets) minimiert, trotzdem kann es vorkommen, dass ein statischer Schuss auf den einen Roboter tadellos funktioniert und auf einem anderen Roboter zu einem Fall führt. Statische Schüsse sind insgesamt zu instabil.

Aus diesen Problemen und Einschränkungen ergeben sich die Ziele für die neuen Schüsse. Die statischen Schüsse, die auf der Key-Frame-Technik und einem Open-Loop-System basieren, sollen durch Schüsse mit einer flexibleren Methode und einem Closed-Loop-System ersetzt werden. Mittels der Rückkopplung soll der Schuss stabilisiert werden.

Ferner soll mit intuitiveren Werten als der Tabelle mit den Servopositionen gearbeitet werden. Die intuitiveren Werte sorgen dafür sorgen, dass der Schuss an sich einfacher zu verändern ist. Auch sollen die Schüsse einfacher an die einzelnen Roboter anzupassen sein.

Als erster Versuch wird ein kleiner Schuss direkt in den Walker integriert (siehe Abschnitt 5.2). Der Walker kümmert sich um die Stabilisierung. Dann wird ein komplette eigenständiger Schuss im Stehen entwickelt (siehe Abschnitt 5.3). Dieser Schuss verfügt über eine eigenständige Steuerung und Stabilisierung. Über einige wenige Parameter lassen sich die Schüsseleigenschaften bequem per Konfigurationsdatei einstellen.

Die Probleme der beiden neu entwickelten Schüsse berücksichtigen, wird eine FK und eine IK der Roboter entwickelt. Auf der FK und der IK aufbauend wird ein Grundgerüst für zukünftige Schüsse (und Bewegungen allgemein) entworfen.

5.2 Schuss im Laufen – der Mini-Schuss

Die Idee für den Schuss im Laufen basiert darauf, dass ein normaler Schuss inklusive Vorbereitung zu viel Zeit in Anspruch nimmt. Der Roboter muss sich genau vor dem Ball positionieren, stehen bleiben, warten bis er stabil steht, sich gegebenenfalls repositionieren, den Schuss ausführen und warten bis der Roboter in einer stabilen Position ist um den Walker wieder zu starten.

Die Überlegung zum Mini-Schuss basiert darauf, dass es relativ einfach ist, den Ball zu dribbeln. Der Roboter läuft gegen den Ball und führt ihn so in die Zielrichtung. Dabei ist der Ball zwangsläufig vor den Füßen. Bei einer guten Schussposition wird der Mini-Schuss aktiviert ohne den Walker zu unterbrechen. Das schwingende Bein schnellt weiter nach vorne als beim normalen Schritt, schießt den Ball und schnellt wieder zurück. Der Walker arbeitet ganz normal weiter, so dass der Mini-Schuss auf das Laufen im besten Fall keinen Einfluss hat. Für den Walker stellt das Verlassen der idealen Trajektorie des schwingenden Beins eine Störung dar, die aber ausgeglichen werden kann.

5.2.1 Umsetzung

Ein Vorteil des Mini-Schusses ist, dass er direkt in den Walker integriert werden kann. Ein Großteil der Infrastruktur kann so wiederverwendet werden.

Der Ablauf des Schusses ist wie folgt. Das Verhalten des Roboters bekommt die Ballinformationen über das Weltmodell und löst den Mini-Schuss in dem richtigen Moment aus. Beim Walker wird ein Flag für den Mini-Schuss und die Schussseite (Links oder Rechts) gesetzt. Sofern das richtige Bein dabei ist das schwingende zu werden, wird der Mini-Kick ausgeführt und das schwingende Bein schnellt nach vorn. Liegt der Ball nicht mehr an einer schießbaren Position werden die Flags vom Verhalten zurückgesetzt und kein Schuss wird ausgeführt.

Für den eigentlichen Schuss gibt es zu berücksichtigen, wie die Bewegung ausgeführt wird (Erweiterung der PIK), wie die Bewegung gesteuert wird und wie lang die Schwingphase ist.


5.2 Schuss im Laufen – der Mini-Schuss

(a) Normale Bewegung beim Gehen.

(b) Schussbewegung bei gesetztem KICKOFFSET.

Abbildung 16: Seitenansicht des Mini-Schusses.

Schwingung) gering ist, oder wenn der Roboter durch externe Einflüsse einen Stoß bekommt und es so zu einem vorzeitigen Bodenkontakt kommt. Die 340 ms für einen Schritt werden in der Realität selten erreicht, weswegen sich nur ein sehr begrenztes Zeitfenster für den eigentlichen Schuss bietet.

```
1 . . .
2 GOAL[MOTOR_RIGHT_KNEE_TOP] += + KICKOFFSET;
3 GOAL[MOTOR_RIGHT_KNEE_BOTTOM] += + KICKOFFSET;
4 GOAL[MOTOR_RIGHT_FOOT_PITCH] += - 2 * KICKOFFSET;
5 GOAL[MOTOR_RIGHT_ARM_PITCH] += - 3 * KICKOFFSET;
6 . . .
```

Listing 2: Ausschnitt aus der erweiterten PIK mit KICKOFFSET für rechten Mini-Schuss.

Für die Form von KICKOFFSET gibt es mehrere Möglichkeiten. KICKOFFSET kann als eine lineare Funktion implementiert werden. Dabei müssen zwei Phasen eingebaut werden: eine zum Vorwärts-, und eine zum Zurückschwingen.

Sinuskurven-ähnliche Bewegungen sind eher weich und führen bei geeigneter Wahl des Intervalls automatisch zu einem Zurückschwingen des Beins. Es ist also nur eine Phase für den Schuss notwendig. Solche weichen Bewegungen beeinträchtigen den Walker weniger, erreichen gegebenenfalls aber nicht die nötige Geschwindigkeit.


Die Geschwindigkeit des schwingenden Beines bestimmt maßgeblich die Reichweite des Schusses. Durch eine Verlängerung der Schussphase kann man entsprechend stärkere Bewegungen
realisieren. Allerdings ist der Einfluss auf die Stabilität des Walkers entsprechend groß.

Maximalwerte für KICKOFFSET, Art der Funktion für KICKOFFSET, Zeitdauer der Schussphase und Erweiterung der PIK wurden experimentell ermittelt. Beste Ergebnisse sind mit einem sehr kleinen Zeitfenster (die Schussphase dauert 30% der Schritthase und startet 30 ms nachdem der neue Schritt angefangen hat) und einem hohen konstanten KICKOFFSET erreicht worden. Dies hat ein Übersteuern der Servos zur Folge, was eine kurze aber sehr schnelle Bewegung realisiert. In der Schussphase bewegen sich KNEE_TOP, KNEE_BOTTOM und FOOT_PITCH¹¹. Auch wird der Arm auf Schussseite zusätzlich nach hinten geschwungen um den, durch den Schuss entstehenden, Momenten entgegenzuwirken (siehe Listing 3).

Schon beim experimentieren mit den Parametern hat sich gezeigt, dass es sehr schwer ist, den Ball richtig zu treffen.

### Listing 3: Pseudo-Code der Implementierung des Mini-Schusses.

```python
def walker():
    init()
    while walking:
        get_sensor_data()
        cale_walking_motion()
        cale_walking_error()
        if kick_flag and correct_leg_swinging and in_kick_phase:
            cale_kick_motion()
            pik_kicker()
        else:
            pik_walker()
        set_servos()
```

5.2.2 Probleme und Fazit

Die Ergebnisse des Schusses sind durchwachsen. Bei optimalen Bedingungen ist der Schuss schnell durchgeführt und erreicht eine Weite von bis zu 2, 5 m. Dabei muss der laufende Roboter den Ball gut treffen, die Ballposition muss also genau stimmen. Je schneller der Roboter läuft, desto weiter rollt der Ball.

Das größte Problem ist, den Ball richtig zu treffen. Der Ball kann nur geschossen werden, wenn er vor dem schwingenden Bein liegt. Dies ist aber nur bei jedem zweiten Schritt gegeben. Ferner muss er auch an einer bestimmten Stelle liegen, um eine gute Weite zu erreichen. Die Ballposition aus dem Weltmodell schwankt aber zu stark um verlässliche Vorhersagen zu treffen. Der Schuss wird oft zu früh oder zu spät ausgelöst. Eine aktive Schrittplanung und Ausführung würde es vereinfachen, den Roboter genauer zu positionieren. Der Walker akzeptiert derzeit nur Geschwindigkeiten (ohne Einheiten) und erlaubt keine aktive Schrittplanung, was auch an

¹¹Andere Kombinationen wurden ebenfalls getestet.
der fehlenden IK liegt. Ferner ist die Regelung des Walkers für solche Szenarien nicht ausgelegt, da die Schrittweite als Mittel zum Stabilisieren des Roboters benutzt wird und nicht direkt beeinflussbar ist.

Den Mini-Schuss als einzigen Schuss – und somit als Torschuss – einzusetzen, hat sich als nicht praktikabel erwiesen. Dies liegt vor allem daran, dass der Ball zu selten gut getroffen wird. Als Torschuss eignet sich der Schuss auf Grund der, im Verhältnis zum statischen Schuss, geringen Geschwindigkeit und Reichweite nur, wenn das gegnerische Team keinen guten Torwart aufweist.

Sofern der Schuss aber besser in das Verhalten integriert werden würde, könnte er sich als wertvoll erweisen. Der Schuss könnte im Zweikampf (zwei Roboter stehen voreinander und kämpfen um den Ball) eingesetzt werden um den Ball schnell zu bewegen und so einen Vorteil zu erhalten. Eine aktive Schrittplanung würde das Hauptproblem, die geringe Trefferquote beim Schießen, beseitigen und den Schuss viel effektiver machen.

### 5.3 Stabilisierender Schuss im Stehen – der Kicker

Da sich der Mini-Schuss insgesamt als zu schwach und unzuverlässig herausgestellt hat und der statische Schuss zu instabil ist, wurde ein weiterer Schuss entwickelt. Es soll sich dabei um einen Schuss mit hoher Reichweite handeln, der sich selbst stabilisiert. Ferner soll er einfacher anpassbar sein und die Parameter sollen intuitiver sein, als die Tabelle bei dem statischen Schuss. In Anlehnung an den Walker wird dieser Schuss im Folgenden Kicker genannt.

Der Ablauf des Kickers ist wie folgt:

- das Verhalten steuert den Walker und positioniert den Roboter vor dem Ball,
- das Verhalten löst den Kicker aus,
- der Walker wird beendet,
- der Kicker wird gestartet,
- der Schuss wird ausgeführt,
- der Walker wird wieder gestartet.

Es hat sich in [12, 23, 15] bewährt, Schüsse in Phasen zu unterteilen, die für unterschiedliche Aufgaben zuständig sind: das Heben, das Ausholen, das Schwingen des Beines, etc.. Dieser Ansatz wird auch beim Kicker verfolgt (siehe Abschnitt 5.3.3).

Der Kicker wird, wie der Walker, als dynamische Motion implementiert. Er ist somit gleichberechtigt zum Walker und anders als der Mini-Kick nicht Teil des Walkers. Um Schussbewegungen zu ermöglichen wird die PIK des Walkers ebenfalls wie in Abschnitt 5.2 erweitert.

Für den Kicker lässt sich nicht die Regelung des Walkers aus Abschnitt 4.2.2 verwenden. Folglich muss eine eigene Regelung entwickelt werden (siehe Abschnitt 5.3.1). Idealerweise würde der Kicker den Schwerpunkt (oder sogar die Momente) zu jedem Zeitpunkt der Bewegung berechnen.
5.3 Stabilisierender Schuss im Stehen – der Kicker


5.3.1 Regelung und Stabilisierung des Schusses

Das Problem bei der Regelung ist, dass nicht viel über das Modell, den Roboter, bekannt ist: weder ein genaues kinematisches Modell, noch die Massenverteilung. Das macht Analyse und Vorhersagen über das System und die Auswahl eines passenden Reglers schwierig.


Bekannt ist, dass der statische Schuss scheitert, wenn die Zielposition des FOOT_ROLL-Servos nicht eingenommen wird. Dies kann zum Einen daran liegen, dass die Servos nur über einen P-Regler verfügen und die P-Regler für bestimmte Regelstrecken eine bleibende Regelabweichung aufweisen [9]. Um die Regelung der Servos um einen I- und/oder D-Teil zu erweitern, müsste die Firmware der Servos angepasst werden. Robotis Inc. verbietet dies jedoch. Zum Anderen kann es daran liegen, dass der Servo beim Anfahren der Idealposition überlastet. Hat der FOOT_ROLL -Servo seine Idealposition eingenommen, wirkt auf ihn quasi keine Belastung ein, da sich der Schwerpunkt über dem Standfuß befindet (siehe Abbildung 17(b)). Um den Servo zu entlasten, werden die Arme zur Gewichtsverlagerung benutzt.
Im Kicker wird versucht die Regelabweichung von der Idealposition zu minimieren. Die Regelabweichung \( e(t) \) ist die Differenz der Führungsgröße (die Idealposition von FOOTROLL) und der Regelgröße (der Ist-Position von FOOTROLL).

Um die Regelabweichung zu minimieren, kommt ein Integrator zum Einsatz, der über \( e(t) \) integriert. Die so berechnete Stellgröße \( u(t) \), bestimmt wie sehr die Arme zum Gewichtsverlagern benutzt werden. I-Regler sind generell eher langsam, schwingen unter Umständen, aber verhindern bleibende Regelabweichungen. Ein großes \( K_I \) sorgt dafür, dass der Integrator schneller greift, aber auch übersteuert. Zur Stabilisierung des Roboters wird \( K_I \) relativ groß gewählt. Bei Erreichen der Zielposition, \( e(t) = 0 \), wird der Integrator jedoch zurückgesetzt. So greift der Integrator schneller, übersteuert jedoch nicht beziehungsweise weniger.

Die Position von FOOTROLL und der Wert von \( K_I \) wurden experimentell ermittelt. Die seitliche Stabilisierung funktioniert in der Nähe des Arbeitspunktes sehr gut und hat sich in der Praxis bewährt.

```python
def kicker():
    while kick_in_progress:
        get_sensor_data()
        get_error()

        if inBalancePhase():
            calc_sideramp()
        else if inLiftLegPhase():
            calc_leglength()
        else if inSwingBackPhase():
            calc_swingramp()
        else if inKickPhase():
            calc_swingramp()
        else if inLegReturnPhase():
            calc_swingramp()
        else if inLoweringLegPhase():
            calc_leglength()

        if inKickAdditionPhase():
            calc_kickaddition()
        if inStandingPhase():
            reset_everything()
        return

pik_dyn_kicker()
set_motors()
```

Listing 4: Pseudo-Code des Kickers
5.3 Stabilisierender Schuss im Stehen – der Kicker

5.3.2 Einteilung in Phasen:

Der Schuss ist in folgende acht Bewegungsphasen aufgeteilt:

**BalancePhase:** Das Gewicht des Roboters wird auf das Nicht-Schuss-Bein verlagert (siehe Abbildung 18(b)). Der Parameter SIDERAMP bestimmt, wie sehr sich der Roboter zur Seite lehnt. SIDERAMP ist der Parameter, der für die seitliche Stabilität verantwortlich ist und im vorherigen Abschnitt besprochen wurde. In den folgenden Phasen wird diese Position gehalten.

**LiftLegPhase:** Das Schussbein wird über den Parameter LEGCONTRACTION verkürzt (siehe Abbildung 18(c)).

**LegSwingBackPhase:** Das Schussbein holt nach hinten aus (siehe Abbildung 18(d)). Der Parameter SWINGRAMP bestimmt, wie weit das Bein ausholt.

**KickPhase:** Das Schussbein schnellst nach vorne und schießt den Ball (siehe Abbildung 18(e)). Für diese Schwingung ist ebenfalls der Parameter SWINGRAMP zuständig. Ausholen und Schießen stehen dabei in einem festen Verhältnis.

**LegReturnPhase:** Das Schussbein schwingt zurück (siehe Abbildung 18(g)). SWINGRAMP bestimmt auch hier, wie weit das Bein zurückgesetzt wird.

**LoweringLegPhase:** Das Schussbein wird wieder auf volle Länge verlängert und der Roboter verlagert sein Gewicht auf beide Beine (siehe Abbildung 18(h)).
5.3 Stabilisierender Schuss im Stehen – der Kicker

KickAdditionPhase: Die Phase kann parallel zu den anderen Phasen ablaufen (siehe Abbildung 18(f)). Für den Fall, dass die Schwingung nicht ausreicht, kann über den Parameter KICKOFFSET der Unterschenkel zusätzlich beschleunigt werden.

StandingPhase: Hierbei handelt es sich um keine wirkliche Phase. Alle Parameter werden auf 0 gesetzt und der Kicker wird beendet.


Die Variablen SIDERAMP, LEGCONTRACTION, SWINGRAMP, und KICKOFFSET bestimmen, wie die Bewegung aussieht. SIDERAMP ist durch die Hardware der Roboter gegeben. LEGCONTRACTION muss zum Einen groß genug sein, um beim Schuss nicht in den Boden zu treten, und zum Anderen klein genug sein, um nicht über den Ball zu treten. SWINGRAMP bestimmt, wie weit ausgeholt und geschossen wird. Soll ein kurzer Schuss entwickelt werden, wird dieser Parameter entsprechend klein gewählt. Für KICKOFFSET wurde die PIK des Walkers erweitert. KICKOFFSET wird auf die beiden Knie-Servos und den Fuß-Servo gegeben um eine stärkere Schwingung des Unterschenkels zu erzielen. Dies ist nur für starke Schüsse nötig.


Um die Bewegungen zu beschreiben werden in den meisten Phasen weiche stetige Funktionen verwendet. Weiche Funktionen verursachen weniger ruckartige Bewegungen.

5.3.3 Einfachheit der Benutzung

Die Parameter können bequem per Konfigurationsdatei gesetzt werden. Dies macht ein Anpassen des Kickers sehr einfach. Der kritische Parameter SIDERAMP, kann so gegebenenfalls für jeden Roboter individuell angepasst werden.
Tabelle 3: Ergebnisse des Kickers verglichen mit dem statischen Schuss.

<table>
<thead>
<tr>
<th></th>
<th>Reichweite</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Durchschnitt</td>
</tr>
<tr>
<td>Statischer Schuss</td>
<td>591 cm</td>
</tr>
<tr>
<td>Kicker</td>
<td>569 cm</td>
</tr>
</tbody>
</table>

Durch setzen von ein paar Parametern lässt sich so ein neuer Schuss erzeugen. Um einen weiten Schuss zu erstellen ist allerdings mehr Aufwand nötig, da die Servos dabei am Limit arbeiten und die Stabilisierung gegebenenfalls nicht mehr greift.

5.3.4 Probleme und Fazit

Für ein vergleichendes Experiment wurden die jeweils schusskräftigsten Varianten des statischen Schusses und des Kickers getestet. Dabei wurden je zehn Schüsse durchgeführt. Die zurückgelegte Distanz wurde gemessen. Die maximale gemessene Distanz beträgt aufgrund der beschränkten Räumlichkeiten 600 cm (was einer Feldlänge entspricht). Die durchschnittliche Reichweite bei beiden Schüssen wäre insofern größer. Ferner wurde gezählt, wie viele der Schüsse stabil waren, d.h. nicht zu einem Fall geführt haben. Die Ergebnisse stehen in Tabelle 3.

Der Kicker erreicht fast die Weite des statischen Schusses, ist aber wesentlich stabiler. Auch wenn die Reichweite des Kickers etwas geringer ist als die des statischen Schusses, so ist sie doch höher als die Reichweite der Schüsse der meisten anderen Teams, die im RoboCup teilnehmen.

Die Schussparameter sind wesentlich intuitiver als die Tabelle mit Servopositionen eines statischen Schusses. Neue Schüsse sind innerhalb weniger Sekunden zu erstellen (sofern sie nicht auf hohe Reichweite ausgelegt sein sollen). Es kann aber vorkommen, dass bei bestimmten Parameterkombinationen das Schussbein in den Boden tritt.

Dadurch, dass die Parameter bequem über eine Konfigurationsdatei gesetzt werden können, kann der Kicker schnell auf die einzelnen Roboter angepasst werden. Der SIDERAMP-Parameter hat sich dabei als essentiell für die Stabilität des Schusses herausgestellt und kann ebenfalls über die Konfigurationsdatei geändert werden.

Insgesamt funktioniert die seitliche Stabilisierung über die Arme sehr gut. Allerdings können andere Roboter gegen die Arme laufen und den Schuss so destabilisieren.

5.4 Entwurf eines Schusses mit inverser Kinematik

Das größte Problem bei den in Abschnitt 5.2 und 5.3 beschriebenen Schüssen ist, dass kein gutes Modell des Roboters existiert. Es lassen sich keine genauen Vorhersagen über das Verhalten und die Bewegung des Roboters machen. Insofern müssen viele Experimente durchgeführt werden, um einen verlässlichen Schuss zu erhalten.

In diesem Abschnitt wird ein erstes kinematisches Modell des Roboters entwickelt. Auf diesem aufbauend, kann ein Schuss analog zu dem in Abschnitt 5.3 beschriebenen Schuss entwickelt werden. Auch bildet das kinematische Modell die Grundlage für erweiterte Modelle, die auch die Massenverteilung des Roboters miteinbeziehen.

Um die IK des Roboters am Computer testen zu können, wird die FK für den Roboter entwickelt (siehe Abschnitt 5.4.1). Mit der FK lässt sich die Pose des Roboters visualisieren. Auch lässt sich so überprüfen, dass das Ergebnis der IK korrekt ist.


5.4.1 Entwicklung der direkten Kinematik


Um die FK zu berechnen wird die Denavit-Hardenberg-Konvention genutzt (siehe Abschnitt 2.2.2). Die Glieder des Roboters werden dazu vermessen. Die Ergebnisse sind in Abbildung 19 zu sehen.

Die hier entwickelte FK berechnet für jedes Bein die Endeffektorpose, des Hüftgelenks, ausgehend von einem Bezugsrahmen, dem Fußgelenk. Die Entfernung vom Boden zum ersten Servo (35 mm) wird vernachlässigt. Sofern der Fuß immer parallel zum Boden ist, was bei jeder Bewegung der Fall sein soll, handelt es sich um eine Konstante, die mit dem Höhenunterschied zwischen Fuß und Hüftgelenk verrechnet werden kann.
125

Abbildung 19: Maße des Beines

<table>
<thead>
<tr>
<th>Tabelle 4: Denavit-Hartenberg Parameter für die Beine der Roboter 2009.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>HIP_PITCH</td>
</tr>
<tr>
<td>HIPROLL</td>
</tr>
<tr>
<td>HIP_YAW</td>
</tr>
<tr>
<td>KNEE_TOP</td>
</tr>
<tr>
<td>KNEE_BOTTOM</td>
</tr>
<tr>
<td>FOOT_PITCH</td>
</tr>
<tr>
<td>FOOT_RROLL</td>
</tr>
</tbody>
</table>


Das Programm zur Visualisierung der FK, der Kinematik-Simulator KiSi, ist in Abbildung 20 zu sehen. Über die Regler lassen sich die einzelnen Gelenke ansteuern und die Anzeige wird entsprechend aktualisiert. So lässt sich kontrollieren, ob die FK dem erwarteten Verhalten des realen Roboters entspricht.
5.4 Entwurf eines Schusses mit inverser Kinematik

(a) Bein in KiSi. Die vier Punkte stellen die Ortskoordinatensysteme dar, in denen sich alle Achsen treffen.

(b) Regler für die Motorwerte in KiSi.

Abbildung 20: Kinematik Simulator KiSi.

5.4.2 Entwicklung der inversen Kinematik

Aufgrund der begrenzten Rechenleistung wird ein analytischer Ansatz zur Lösung der IK gewählt\(^\text{[21]}\). Dabei gelten folgende Bedingungen und Annahmen:

- Als Referenzpunkt wird die Hüfte des Roboters gewählt.
- Der Endeffektor ist der Fuß (wieder unter Vernachlässigung des Abstandes vom Boden zur ersten Servo-Achse).
- Die Knie-Servos sollen sich immer zu gleichen Teilen bewegen, um die Belastung zu minimieren. Gleichzeitig wird so das Problem der redundanten Kinematik beseitigt.
- Der Fuß des Roboters soll immer parallel zum Referenzkoordinatensystem stehen. Somit wird nur Yaw für die Orientierung des Endeffektors zugelassen. Die IK wird zu einer

\(^{12}\)Parallele Versuche die IK für die Modelle 2010, die eine komplexere Kinematik aufweisen, numerisch mittels Jacobi-Inverse, Jacobi-Transpase, und Gradientenabstieg zu lösen, haben sich als zu rechenintensiv herausgestellt. Für die Modelle 2010 wurde die IK letztendlich mit Piepers Ansatz gelöst.
5.4 Entwurf eines Schusses mit inverser Kinematik

Funktion von

\[
\begin{pmatrix}
  x \\
  y \\
  z \\
  \text{yaw}
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  \text{HIP_PITCH} \\
  \text{HIP_ROLL} \\
  \text{HIP_YAW} \\
  \text{KNEE_TOP} \\
  \text{KNEE_BOTTOM} \\
  \text{FOOT_PITCH} \\
  \text{FOOT_ROLL}
\end{pmatrix}
\]

Die Berechnung der IK erfolgt in drei Schritten:

1. Schritt:

- \text{HIP_YAW} ist der einzige Servo, der um die Yaw-Achse drehbar ist. Deswegen wird \text{HIP_YAW} der Wert \text{yaw} direkt zugewiesen.

  \[
  \text{HIP_YAW} = \text{yaw}
  \]

- Die Koordinaten \(x\), \(y\) und \(z\) werden entsprechend rotiert.

  \[
  x_2 = x \cdot \cos(yaw) + y \cdot \sin(yaw)
  \]

  \[
  y_2 = x \cdot \sin(yaw) + y \cdot \cos(yaw)
  \]

  \[
  z_2 = z
  \]

Abbildung 21: Schritt 1: Draufsicht.

- \text{HIP_YAW} ist der einzige Servo, der um die Yaw-Achse drehbar ist. Deswegen wird \text{HIP_YAW} der Wert \text{yaw} direkt zugewiesen.

- Die Koordinaten \(x\), \(y\) und \(z\) werden entsprechend rotiert.
2. Schritt:


- HIP_ROLL wird aus $y_2$ und $z_2$ berechnet.

\[
\begin{align*}
    x_3 &= x_2 \\
    y_3 &= y_2 \\
    z_3 &= \sqrt{y_2^2 + z_2^2} \\
    \text{HIP_ROLL} &= \text{atan2}(y_2, z_2)
\end{align*}
\]

- Um den Fuß parallel zu halten, wird FOOT_ROLL = −HIP_ROLL gesetzt.

\[
\text{FOOT_ROLL} = -\text{HIP_ROLL}
\]
5.4 Entwurf eines Schusses mit inverser Kinematik

3. Schritt:

Die effektive Beinlänge $d_f$ wird aus $x_3$ und $z_3$ berechnet.

$$d_f = \sqrt{z_3^2 + x_3^2} \quad (14)$$

- Oberschenkel $leg_u$ und Unterschenkel $leg_l$ sind gleich lang: $leg_u = leg_l$. Das Knie ist immer parallel zur Strecke der effektiven Länge des Beines $d_f$. Somit lässt sich $\beta$ einfach berechnen. $\beta$ bestimmt KNEE_TOP und KNEE_BOTTOM.

$$\beta = \arccos \left( \frac{d_f - knee}{2}, leg_u \right) \quad (15)$$

KNEE_TOP = $-\beta$ \quad (16)

KNEE_BOTTOM = $-\beta$ \quad (17)
5.4 Entwurf eines Schusses mit inverser Kinematik

- $\alpha$ und $\beta$ bestimmen HIP\_PITCH.

$$\alpha = \text{atan2}(x_3, z_3)$$  \hspace{1cm} (18)

$$\text{HIP\_PITCH} = \alpha + \beta$$  \hspace{1cm} (19)

- FOOT\_PITCH wird so gesetzt, dass der Fuß parallel steht.

$$\text{FOOT\_PITCH} = -\alpha + \beta = \gamma$$  \hspace{1cm} (20)


5.4.3 Analyse und Ausblick

6 Ausblick

In den jeweiligen Abschnitten wurden potentielle Erweiterungen schon angesprochen. Hier werden sie wieder aufgegriffen und weiter ausgeführt.

6.1 Bessere Einbindung des Schusses in das Verhalten

In dieser Arbeit wurde sich primär auf die Entwicklung des Schusses konzentriert. Die Einbindung in das Verhalten wurde größtenteils vernachlässigt.


6.2 Verbesserungen des Schusses


6.3 Entwicklung in anderen Gebieten

Der folgende Schritt sollte die Berechnung der entstehenden Momente sein, um auch die dynamischen Effekte der Bewegung zu berücksichtigen. Dies sollte einen wesentlich stärkeren und stabileren Schuss ermöglichen.


6.3 Entwicklung in anderen Gebieten

Die hier entwickelten Methoden und Erkenntnisse sind nicht nur für den Schuss einsetzbar, sondern auf viele Bereiche der FUmanoids übertragbar.

Die FK kann zur Visualisierung aller Bewegungen im MotionEditor verwendet werden. Statische Motions würden so wesentlich anschaulicher werden.


Die IK könnte dazu genutzt werden, statische Motions durch Bewegungen mit IK zu ergänzen oder zu ersetzen.

Die IK könnte in den MotionEditor eingebaut werden. Alle statischen Motions könnten mit der IK beschrieben werden. Die FK könnte ebenfalls zur Visualisierung dieser Bewegungen eingesetzt werden.

Die wohl größte Potential hat die IK bei der Entwicklung besserer Laufalgorithmen. Die IK lässt eine genauere Beschreibung der Bewegungen zu, was wiederum eine genauere Regelung zulässt. Sofern das kinematische Modell des Roboters noch um Massen ergänzt wird, könnten auch alternative Gangmechanismen, wie ZMP, getestet werden.
7 Zusammenfassung

In dieser Arbeit wurden verschiedene Schüsse für humanoide Fußballroboter in der KidSize-Liga entwickelt.

Dazu wurde erst das nötige Handwerkszeug erarbeitet: wie werden Systeme geregelt; wie lässt sich Bewegung beschreiben und wie funktioniert der zweibeinige Gang bei Robotern?

Die Fumanoids-Plattform wurde unter besonderer Berücksichtigung der Bewegungen der Robotenvorgestellt.


Abschließend wurde ein Ausblick gegeben, wie der Schuss (und auch andere Bereiche) im Fumanoid-System verbessert werden könnten.
<table>
<thead>
<tr>
<th>Nr.</th>
<th>Titel</th>
<th>Seite</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Feld in der KidSize-Liga [4]</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Standardregelkreis mit Störungen – Closed-Loop-System</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Eine einfache Steuerkette – open loop system</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Roll-Nick-Gier [20]</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>IK: mögliche Lösungen und Probleme bei einer einfachen kinematischen Kette in 2D. Der kleine Punkt ist die Zielposition der kinematischen Kette</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Piepers Ansatz: Prinzip der kinematischen Entkopplung</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>Approximation der Bewegung eines Greifers durch Jacobi-Matrix</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>Beispielhaftes CPG von Team Nimbro aus [1]</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>ZMP aus [22]</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>Roboter 2009</td>
<td>21</td>
</tr>
<tr>
<td>12</td>
<td>verdex pro der Firma Gumstix Inc</td>
<td>21</td>
</tr>
<tr>
<td>13</td>
<td>FUremote mit einigen Plugins aktiviert</td>
<td>24</td>
</tr>
<tr>
<td>14</td>
<td>MotionEditor in FUremote mit einem linken Schuss in Bearbeitung</td>
<td>24</td>
</tr>
<tr>
<td>15</td>
<td>PIK bei den Modellen 2010. Verkürzung des Beines</td>
<td>26</td>
</tr>
<tr>
<td>16</td>
<td>Seitenansicht des Mini-Schusses</td>
<td>31</td>
</tr>
<tr>
<td>17</td>
<td>Verschiedene Szenarien beim Stabilisieren des Schusses. Schwarze Kreise stellen den Schwerpunkt dar.</td>
<td>34</td>
</tr>
<tr>
<td>18</td>
<td>Phasen des Kickers</td>
<td>36</td>
</tr>
<tr>
<td>19</td>
<td>Maße des Beines</td>
<td>40</td>
</tr>
<tr>
<td>20</td>
<td>Kinematik Simulator KiSi</td>
<td>41</td>
</tr>
<tr>
<td>21</td>
<td>Schritt 1: Draufsicht</td>
<td>42</td>
</tr>
<tr>
<td>22</td>
<td>Schritt 2: Frontansicht</td>
<td>43</td>
</tr>
<tr>
<td>23</td>
<td>Schritt 3: Seitenansicht</td>
<td>44</td>
</tr>
</tbody>
</table>
Literatur


