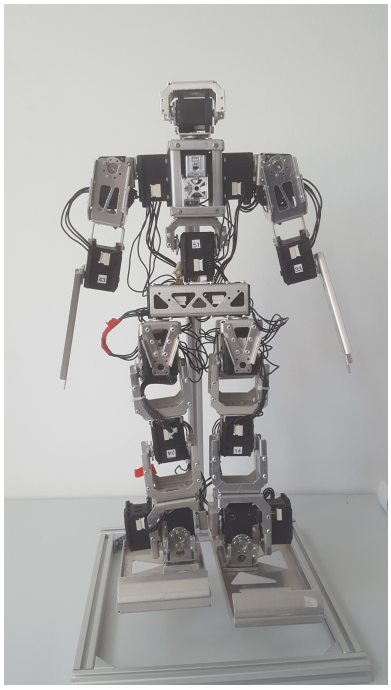
hochschule mannheim

HECTOR SEMINAR

Laufroboter



Abschlussbericht der Kooperationsphase 2016/17

Durchgeführt an der Hochschule Mannheim

Betreuer: Prof. Dr. Thomas Ihme

Finn Buschmann
In den Pfädelsäckern 14
69121 Heidelberg

Holger Hujjon
Wilhelm-Herz-Str. 27
68766 Hockenheim

Inhaltsverzeichnis

1 Abstract....3

2 Einleitung....3

2.1 Material....3

3 Theorie....3

4 Methoden....5

4.1 Winkeltabelle (ROS)....5

4.2 Praxistests mit Elvis....6

4.3 Simulation mit Matlab....7

4.3.1 2D-Grundlage....7

4.3.2 Erweiterung auf 3D....10

4.3.3 Neue Funktionen....13

4.3.4 3D-Simulation Laufen....15

5 Fehlerbetrachtung....18

5.1 Gewichte der Beine....18

5.2 Vereinfachung physikalischer Prozesse....18

5.3 Fuß nicht komplett hochklappbar....20

6 Fazit....22

6.1 Erkenntnisse....22

6.2 Ergebnisse....22

6.3 Ausblick....23

7 Danksagung....23

8 Quellen....23

9 Anhang....24

9.1 Quellcode Aufstehen_v7....24

9.2 Quellcode Laufen_v5....28

10 Selbstständigkeitserklärung....33

1 Abstract

Walking is a complex way of locomotion. In this project, it was tried to make a handmade robot walk. The used humanoid robot Elvis is 550 mm tall, has 15 engines and 12 joints. Because it provides good analysis and conserves the fragile robot, the focus was on computer simulation. The main progress that was made was to expand an existing 2D-simulation to a third dimension which enabled a better stand-up-algorithm and the possibility to simulate walking.

2 Einleitung

Die vermeintliche Krone der Schöpfung, der Mensch, bedient sich einer besonderen Fortbewegungsart: Dem Laufen. Können fachunkundige Personen einem von Hobbybastelnden gebauten Roboter diesen komplexen Bewegungsablauf näherbringen?

2.1 Material

Laufroboter Elvis: humanoid, 15 Motoren, 12 Gelenke, 550 mm hoch und ca. 2,6 kg schwer. Er wurde in Handarbeit in einer Auflage von einem Exemplar gefertigt.

3 Theorie

Die Theorie, die in dem Projekt verwendet wurde, ist die Mathematik hinter Koordinatensystemen, genauer gesagt Koordinatentransformationen. Diese benötigt man für die Umrechnung der Koordinaten verschiedener Systeme und Teilsysteme ineinander. Dazu gehören auch die verschiedenen Teile unseres Roboters. Jedes Gelenk ist nämlich das Zentrum seines eigenen Koordinatensystems. Für die Darstellung würde das bedeuten, dass sämtliche Gelenke im Nullpunkt dargestellt werden würden. Die Koordinaten müssen aber für die Darstellung des Roboters alle in ein gemeinsames Koordinatensystem umgerechnet

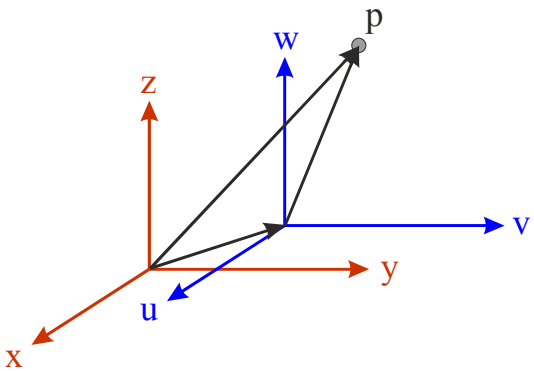


Illustration 1: Verschiebung von Koordinatensystemen

werden. Es gibt mehrere Möglichkeiten, wie Koordinatensysteme zu einander stehen können. Zum einen können sie mit gleicher Orientierung im Raum verschoben sein. Dann können sie um ihre Achsen gedreht werden und sie können verschiedene Skalen haben. Letzteres ist für uns nicht relevant, da die Skalen einheitlich festgelegt wurden. Außerdem könnte die Perspektive sich geändert haben, was für das Projekt irrelevant ist, da diese Berechnungen von Matlab durchgeführt werden. Bei der Translation (Verschiebung, Illustration 1), sind die gleichen Achsen der Koordinatensysteme parallel zu einander, nur der Ursprung wurde im Raum verschoben. Diese Verschiebungen lassen sich durch eine einfache Vektoraddition realisieren. Um den Ort eines Punktes in ein anderes, zu seinem System verschobenes, System umzurechnen, addiert man den Vektor, der vom Ursprung des ursprünglichen Koordinatensystems zum Ursprung des neuen Systems führt. Das Ergebnis dieser Addition ist dann der Ort des Punktes im neuen System.

Koordinatensysteme können auch zueinander gedreht sein. Die Rotation um die eigene Achse ist schon etwas komplizierter als die einfache Verschiebung. Erstmal müssen die Rotationen um die x, y beziehungsweise z-Achse unterschieden werden. Die Koordinate, die die Entfernung des Punktes vom Ursprung entlang der rotierenden Achse angibt, bleibt unverändert. Die anderen Koordinaten ändern sich, da der Punkt eine Kreisbahn um die Achse beschreibt. Letztendlich gibt es für die Rotation um eine bestimmte Achse um einen Winkel α je eine Matrix, mit der man den Ortsvektor des Punktes multiplizieren muss, um den Ortsvektor des Punktes im rotierten System zu erhalten. Aufgrund der Kreisbahn ist es nicht weiter verwunderlich, dass in diesen Matrizen sinus- und cosinus- Funktionen vorkommen:

$$R_{x,a}=\begin{pmatrix}1 & 0 & 0\\ 0 & \cos(\alpha) & -\sin(\alpha)\\ 0 & \sin(\alpha) & \cos(\alpha)\end{pmatrix}$$

Table 1: Drehung um die x-Achse mit dem Winkel α

$$R_{x,a}=\begin{pmatrix}\cos(\alpha) & -\sin(\alpha) & 0\\ \sin(\alpha) & \cos(\alpha) & 0\\ 0 & 0 & 1\end{pmatrix}$$

Table 2: Drehung um die z-Achse mit dem Winkel α

$$R_{x,a}=\begin{pmatrix}\cos(\alpha) & 0 & \sin(\alpha)\\ 0 & 1 & 0\\ -\sin(\alpha) & 0 & \cos(\alpha)\end{pmatrix}$$

Table 3: Drehung um die y-Achse mit dem Winkel α

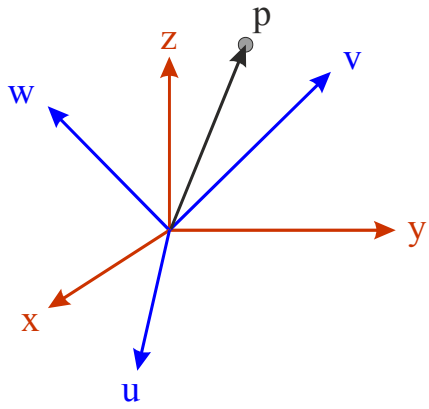


Illustration 2: Drehung um die x-Achse mit dem Winkel α

$$R = R_n \cdot R_{n-1} \cdot \ldots \cdot R_2 \cdot R_1$$

Table 4: Verkettung von Rotationen

Die Rotation um mehrere Achsen lässt sich durchführen, indem man die Matrizen der einzelnen Rotation um eine einzelne Achse miteinander multipliziert, und zwar von links nach rechts (Matrixmultiplikation ist nicht kommutativ) in der Reihenfolge, in der die Rotationen letztendlich auch durchgeführt werden. Dadurch entsteht eine Matrix, die dieRotation um diese Winkel beschreibt, und die man ebenso mit einem Vektor multiplizieren kann:

Es gibt auch noch eine Darstellung, in der man beliebige Rotationen und Verschiebungen gleichzeitig veranschaulichen kann, die homogene Transformationsmatrix. Dabei handelt es sich um folgendes 4x4 Feld:

$$R_{x,a} = \begin{pmatrix} R & T \\ P & S \end{pmatrix}$$

Table 5: Transformationsmatrix für homogene Transformation

In dieser Form werden Rotationen oben links (R - 3x3) und Translationen (Verschiebungen) rechts daneben (T - 3x1) eingetragen. Darunter findet sich noch Platz für Perspektivtransformationen (P - 1x3) und einen Skalierungsfaktor (S - 1x1), was für unser Projekt, wie schon erwähnt, keinerlei Rolle spielt. In dem Projekt wurden alle Skalen gleich eingestellt. Die Perspektivtransformationen in der Simulation wurden mittels des Matlab-Befehls `plot3` berechnet und nicht in unserer Simulation verarbeitet.

4 Methoden

Die Vorgehensweisen, mit denen der Laufroboter lauffähig gemacht werden sollte, werden im Folgenden vorgestellt.

4.1 Winkeltabelle (ROS)

Elvis wurde mithilfe sogenannter Winkeltabellen angesteuert: Ein ROS-Programm ordnete jeder Spalte einen Motor zu, jede Zeile entsprach einem Zeitabschnitt bzw. Simulationsschritt. Die Zelleninhalte gaben hierbei die absolute Rotation der Motoren im Gradmaß an. Die Winkeltabellen wurden mit LibreOffice Calc 4.2.8.2 bearbeitet und im CSV-Format gespeichert.[1]

1 Hierbei gilt es die Spracheinstellung zu beachten: Diese sollten auf „English“ eingestellt sein, da LibreOffice Calc andernfalls Punkt und Komma vertauscht und die Winkeltabelle dann nicht mehr durch das ROS-Programm eingelesen werden kann.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
8	0	0	0	17	80	0	0	0	0	-17	-80	0	60	30	-20	-60	-30	20	0	0	0
9	0	0	0	17	80	0	0	0	0	-17	-80	0	60	30	-30	-60	-30	30	0	0	0
10	0	0	0	17	80	0	0	0	0	-17	-80	0	60	30	-30	-60	-30	30	0	0	0
11	0	0	0	17	80	0	0	0	0	-17	-80	0	45	30	-30	-45	-30	30	0	0	0
12	0	0	0	0	83.11	0	0	0	0	0	-83.11	0	30	30	-30	-30	-30	30	0	0	0
13	0	0	0	0	78.048	0	0	0	0	0	-78.048	0	0	30	-30	0	-30	30	0	0	0
14	0	0	0	0	78.608	0	0	0	0	0	-78.608	0	-30	30	-30	30	-30	30	0	0	0
15	0	0	0	0	80.322	0	0	0	0	0	-80.322	0	-45	30	-30	45	-30	30	0	0	0
16	0	0	0	0	82.684	0	0	0	0	0	-82.684	0	-60	30	-30	60	-30	30	0	0	0
17	0	0	0	0	81.76	0	0	0	0	0	-81.76	0	-85	30	-5	85	-30	5	0	0	0
18	0	-7.3333	0	-4	80.46	0	0	-7.3333	0	4	-80.46	0	-85.467	30	-5	85.467	-30	5	0	0	0
19	0	-14.667	0	-8	79.144	0	0	-14.667	0	8	-79.144	0	-85.933	30	-5	85.933	-30	5	0	0	0
20	0	-22	0	-12	77.804	0	0	-22	0	12	-77.804	0	-86.4	30	-5	86.4	-30	5	0	0	0
21	0	-29.333	0	-16	76.433	0	0	-29.333	0	16	-76.433	0	-86.867	30	-5	86.867	-30	5	0	0	0
22	0	-36.667	0	-20	75.022	0	0	-36.667	0	20	-75.022	0	-87.333	30	-5	87.333	-30	5	0	0	0
23	0	-44	0	-24	73.563	0	0	-44	0	24	-73.563	0	-87.8	30	-5	87.8	-30	5	0	0	0
24	0	-51.333	0	-28	72.047	0	0	-51.333	0	28	-72.047	0	-88.267	30	-5	88.267	-30	5	0	0	0
25	0	-58.667	0	-32	70.462	0	0	-58.667	0	32	-70.462	0	-88.733	30	-5	88.733	-30	5	0	0	0
26	0	-66	0	-36	68.798	0	0	-66	0	36	-68.798	0	-89.2	30	-5	89.2	-30	5	0	0	0
27	0	-73.333	0	-40	67.043	0	0	-73.333	0	40	-67.043	0	-89.667	30	-5	89.667	-30	5	0	0	0
28	0	-80.667	0	-44	65.182	0	0	-80.667	0	44	-65.182	0	-90.133	30	-5	90.133	-30	5	0	0	0
29	0	-88	0	-48	63.199	0	0	-88	0	48	-63.199	0	-90.6	30	-5	90.6	-30	5	0	0	0
30	0	-95.333	0	-52	61.078	0	0	-95.333	0	52	-61.078	0	-91.067	30	-5	91.067	-30	5	0	0	0
31	0	-102.67	0	-56	58.8	0	0	-102.67	0	56	-58.8	0	-91.533	30	-5	91.533	-30	5	0	0	0
32	0	-110	0	-60	56.344	0	0	-110	0	60	-56.344	0	-92	30	-5	92	-30	5	0	0	0
33	0	-110	0	-60	56.355	0	0	-110	0	60	-56.355	0	-92	30	-5	92	-30	5	0	0	0
34	0	-110	0	-60	54.249	0	0	-110	0	60	-54.249	0	-88	30	-1.875	88	-30	1.875	0	0	0

Illustration 3: Ausschnitt aus einer Winkeltabelle

4.2 Praxistests mit Elvis

Da von vorherigen Arbeitsgruppen bereits Winkeltabellen zum Aufstehen vorhanden waren, konnten Praxistests durchgeführt werden. Dabei zeigte sich, dass die vorhandenen Winkeltabellen zwar grundsätzlich geeignet waren, allerdings sehr unzuverlässig. Es war ersichtlich, wie sie funktionieren sollten, aber Elvis fiel stets hin.

Aufgrund der Bauweise von Elvis lösten sich bei praktischen Versuchen immer wieder einige Schrauben und Kabel klemmten sich zwischen Gelenke oder lösten sich aus ihrer Verankerung. Deshalb waren Praxistests stets aufwändig und konnten nicht in großer Zahl durchgeführt werden.

Auch gestaltete sich die Auswertung der Praxistests als schwierig, da oft nicht ersichtlich war, ob Elvis hinfiel weil es ein Hardware-Problem gab, oder ob es ein Hardware-Problem gab weil Elvis hingefallen war. Auch die Kombination aus hoher Anzahl veränderbarer Parameter (Welcher Motor hätte sich wann und wie anders verhalten müssen, damit Elvis nicht umgekippt wäre?) und geringer Anzahl an möglichen Versuchen erschwerte den Erkenntnisgewinn.

Aufgrund dessen wurden im weiteren Verlauf Praxistests hauptsächlich zur Verifizierung und Verbesserung der in der Simulation bereits optimierten Winkeltabellen verwendet.

4.3 Simulation mit Matlab

Da Praxistests sich als einerseits aufwändig und andererseits als wenig ergiebig herausstellten, wurde als weitere Methode die Simulation mit Matlab verwendet.

4.3.1 2D-Grundlage

Es stand bereits eine 2D-Simulation zur Verfügung, die das Ausführen einer Winkeltabelle visualisieren und den Schwerpunkt (und ob dieser über dem Fuß liegt oder nicht) anzeigen konnte (Illustration 6, Illustration 4).

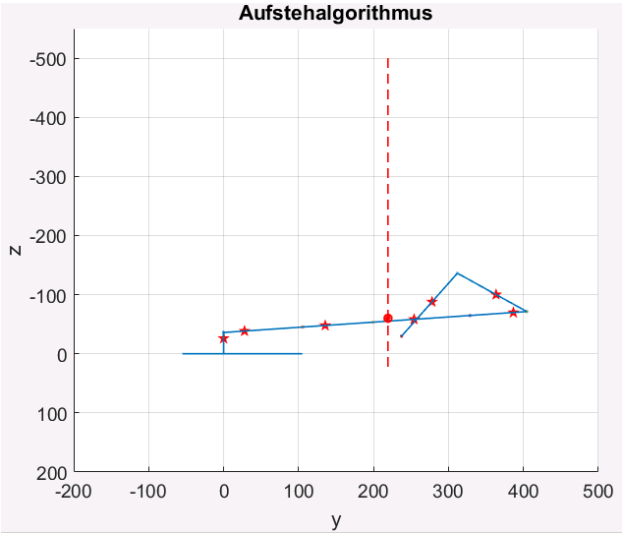


Illustration 4: Simulation des Aufstehalgorithmus.

Illustration 5: Elvis von oben, in Bauchlage auf einer Tischplatte

Das dafür verantwortlich Matlab-Programm funktioniert wie folgt: Zuerst erhält das Programm allerlei Konstanten, nämlich die Abstände der verwendeten Gelenke, die Schwerpunkte und die Gewichte der Segmente (Text 1).

Hierbei wurde der Roboter, der über 15 Motoren und 12 Gelenke verfügt, auf 7 Segmente und 6 Gelenke reduziert (Illustration 7), da die Anzahl der am Aufstehalgorithmus beteiligten Motoren nur 11 beträgt und diese Zahl durch die Symmetrie des Aufstehalgorithmus (linkes und rechtes Bein, linker und rechter Arm, machen gleichzeitig die gleiche Bewegung) weiter reduziert werden kann (Praxistests zeigten allerdings, dass diese Reduktion die Simulation zu ungenau machte).

Auch wurde der Fußpunkt mittels Nullvektor fest an einem Punkt fixiert (Text 2), denn beim Aufstehen verändert sich die Position des Fußes nicht. Damit dient der Fußpunkt als Referenzkoordinatensystem.

Die zu verwendenden Winkel werden in jedem Simulationsschritt eingelesen, die eigentliche Simulation übernimmt das ausgelagerte Programm `simulation.m` (Text 3).

Nachdem jetzt alle erforderlichen Werte in das Programm eingespeist worden sind, kann die Simulation beginnen: `mi` ist der aktuelle, neue Punkt, `mLast` der vorherige Punkt; der Körper des Roboters entsteht, indem eine Linie von `mLast` zu `mi` gezeichnet wird, dann `mi` = `mLast` gesetzt wird, das nächste `mi` berechnet wird, die beiden wieder verbunden werden usw.

Die Berechnung von `mi` funktioniert nach den Prinzipien der Koordinatentransformation wie folgt:

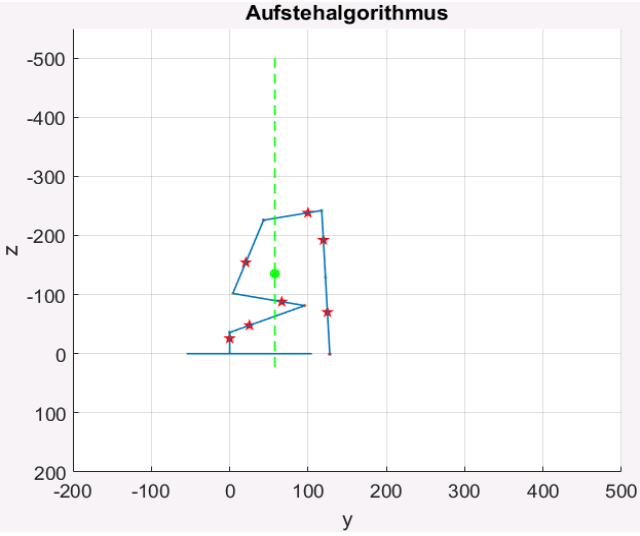


Illustration 6: Die Färbung der gestrichelten Linie und des Sterns zeigen, ob der Schwerpunkt über den Füßen liegt oder nicht.

```
% Verschiebungsparameter (Abstaende der Motoren)
t15 = -36;
t14 = -106;
t12 = -94;
t51 = -130;
t31 = -76;
t33 = 113;
thand = 130;
% 1 x 7 Matrix fuer die Verschiebungsparameter
tparam = [t15 t14 t12 t51 t31 t33 thand];

%Schwerpunkte der einzelnen Gelenke
s00 = -26;
s15 = -28;
s14 = -30;
s12 = -55;
s51 = -58; %stimmt noch nicht
s31 = 50;
s33 = 59;
% 1 x 7 Matrix fuer die Verschiebungsparameter
sparam = [s00 s15 s14 s12 s51 s31 s33];

%Gewichte der Segmente (benannt nach unterem Punkt)
g00 = 314;
g15 = 270.4;
g14 = 206.4;
g12 = 463.4;
g51 = 394.6;
g31 = 371.6;
g33 = 37.6;
% 1 x 7 Matrix fuer die Gewichte
gparam = [g00 g15 g14 g12 g51 g31 g33];
```

Text 1: Definieren von Konstanten

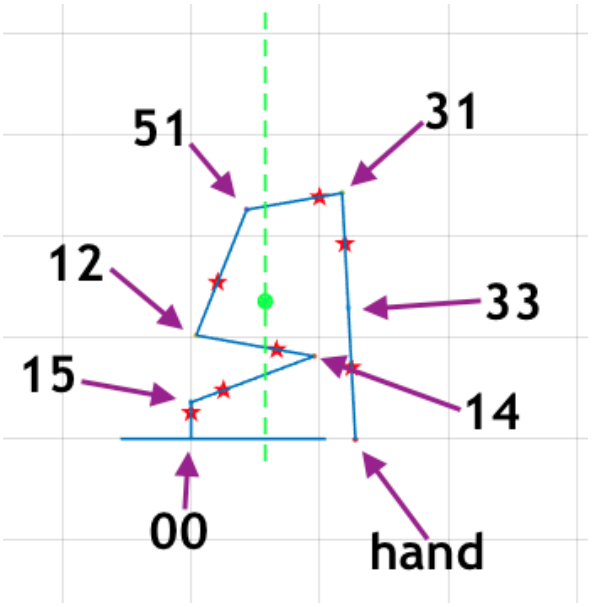


Illustration 7: Benennung der Punkte, die Anfang und Ende der Segmente bestimmen.

Das neue `mi` ist das alte `mi` plus den Verschiebungsvektor `t`. Zur Bestimmung von `t` wird `t` zunächst der Wert des Koordinatenursprungs (der Fußpunkt) zugewiesen, dann wird ihm mithilfe des Verschiebungsparameters die Länge des Segments zugewiesen. Als nächstes erhält der Vektor seine Rotation (aus der Winkeltabelle), denn bislang stand er parallel zu z-Achse. Dafür wird `gesamtWinkel` verwendet, damit das nächste Segment in Bezug auf die Rotation relativ zum letzten steht. Die Linie, die den Fuß darstellt, wird zu Beginn des Programms einmal gezeichnet, da die Position des Fußes sich nicht verändert.

Alle Variablen, die mit `s` beginnen, dienen der Schwerpunktberechnung. Hierbei wird im gleichen Schritt, in dem ein Segment berechnet und gezeichnet wird, der Schwerpunkt des Segments berechnet, eingezeichnet (mit einem roten Stern, s. o.) und gespeichert. Die Position des Schwerpunktes wird bestimmt, indem vom vorherigen `mi` die in `sparam` gespeicherte Distanz entlang `t` gegangen wird. Um den Gesamtschwerpunkt zu bestimmen werden die Einzelschwerpunkte gemäß der Schwerpunktgleichung gewichtet addiert. Abschließend wird überprüft ob dieser über dem Fuß liegt oder nicht und der Gesamtschwerpunkt und die durch ihn und parallel zur z-Achse verlaufenden Linie in der entsprechenden Farbe eingezeichnet.

```
% Nullvektor = Fusspunkt des Roboters
m0 = [0; 0; 0; 1];
```

Text 2: Fixierung des Fußpunktes

```
%Winkel importieren
winkeltabelle = importdata('winkeltabelle_8.csv',',', 1);

anzZeilen = 137; %Anzahl der Zeilen der Winkeltabelle

for i = 1 : 1 : anzZeilen

    % Winkel
    alpha15 = winkeltabelle.data(i, 5);
    alpha14 = winkeltabelle.data(i, 4);
    alpha12 = -winkeltabelle.data(i, 2);
    alpha51 = winkeltabelle.data(i, 19);
    alpha31 = winkeltabelle.data(i, 13);
    alpha33 = winkeltabelle.data(i, 15)-90;
    % 1 x 6 Matrix fuer die Winkel
    winkel = [alpha15 alpha14 alpha12 alpha51 alpha31 alpha33];


    %eigentliche Transformation und Simulation - siehe simulation.m
    if(simulation(winkel, tparam, m0, sparam, gparam) == false)
        disp('Break')
        disp(i)
    end

end
```

Text 3: Einlesen der Winkel

Durch Aneinanderreihung der so entstehenden Einzelbilder entsteht die Simulation (Illustration 8, Illustration 9, Illustration 10, Illustration 11).

4.3.2 Erweiterung auf 3D

Die erste vorgenommene Erweiterung ergänzte die Anzeige der Gelenke, wodurch die Simulation anschaulicher wurde (Illustration 12).

Nach einer neuen Messreihe, bei der Abstände von Motoren, Schwerpunkte und Gewichte von Segmenten bestimmt worden waren - diese waren für die 2D Simulation nur in der reduzierten Form vorhanden - konnte die Erweiterung der Simulation auf 3 Dimensionen unter Einbeziehung fast aller Segmente umgesetzt werden. Der erste Schritt war wiederum die Fütterung des Programms mit `tparam`, `sparam`, `gparam` und `winkel`, analog zur 2D Version, nur mit mehr Daten (Text 5).

In der 3D-Version konnten die Punkte nicht mehr am Stück, wie bei einer Zeichnung ohne Absetzen des Stiftes, gezeichnet werden, deswegen wurde der in Illustration 14 abgebildete Zeichenpfad implementiert.

`simulation.m` wurde um die x-Koordinate erweitert, was an den meisten Stellen keine größeren Änderungen erforderte. Der neue Zeichenpfad wurde durch einen zweiten festen Fußpunkt und das Speichern des Punktes 61 in der bekannten for-Schleife realisiert (Text 6).

Auch musste das Programm nun unterscheiden, welche Motoren in x-Richtung und welche in y-Richtung rotieren (Text 7).

```
function weiter = simulation(winkel, tparam, m0, sparam, gparam)
    weiter = true;
    mLast = m0;
    mi = m0;
    gesamtWinkel = 0;
    %Fuss
    line([0, 0], [-55, 105], [0, 0])
    sy = [0 0 0 0 0 0 0];
    sz = [0 0 0 0 0 0 0];

    for(i = 1 : 1 : 7)
        t = m0; %Koordinatenursprung setzen, ausgehen vom Nullpunkt
        t = transl(0, 0, tparam(i)) * t; %Verschiebung des Koordinatenursprungs t um den
Verschiebungsparameter in z-Richtung

        if (i>=2) gesamtWinkel = gesamtWinkel + winkel(i-1); end %Das Fussgelenk wird nie gedreht, daher i>1
        t = trotx(gesamtWinkel, 'deg') * t; %Koordinatentransformation: Drehung um x-Achse um die Summe der
einzelnen Winkel


        s = t/tparam(i) * sparam(i); %t als Einheitsvektor fuer Schwerpunkte
        si = [mi(1)+s(1) ; mi(2)+s(2) ; mi(3)+s(3); 1]; %Vektoraddition: transformierte Vektoren werden zum
vorherigen Vektor mx addiert
        plot3(si(1), si(2), si(3), 'pr') %Anzeige der Schwerpunkte

        %Abspeichern der Schwerpunktkoordinaten zur spaeteren Berechnung des
        %Ges.Schwerpunktes
        sy(i)= si(2);
        sz(i)= si(3);

        mi = [mi(1)+t(1) ; mi(2)+t(2) ; mi(3)+t(3); 1]; %Vektoraddition: transformierte Vektoren werden zum
vorherigen Vektor mx addiert
        plot3(mi(1), mi(2), mi(3), '.') %Anzeige der Punkte
        line([0, 0], [mLast(2), mi(2)], [mLast(3), mi(3)]); %Linie zwischen dem letzten Punkt und mx
        mLast = mi;

        if (mi(3) > 45) weiter = false; end

    end
    %Gesamtschwerpunkt berechnen mit Schwerpunktgleichung
    gesamtsy = 0;
    gesamtsz = 0;
    gesamtmasse = 0;
    for (i = 1 : 1 : 7)
       gesamtsy = gesamtsy + sy(i) * gparam(i);
       gesamtsz = gesamtsz + sz(i) * gparam(i);
       gesamtmasse = gesamtmasse + gparam(i);
    end

    gesamtsy = gesamtsy / gesamtmasse;
    gesamtsz = gesamtsz / gesamtmasse;

    disp(gesamtsz);

     %Ueberpruefung, ob ueber dem Fuss
    if (gesamtsy < -55 || gesamtsy > 105)
        plot3(0, gesamtsy, gesamtsz, '*r') %Anzeige des Punktes
        plot3(0, gesamtsy, gesamtsz, 'or') %Anzeige des Punktes
        plot3([0,0], [gesamtsy, gesamtsy], [-500,30], '--r')%Linie zur vertikalen Projektion des
Gesschwerpunktes
    else
        plot3(0, gesamtsy, gesamtsz, '*g') %Anzeige des Punktes
        plot3(0, gesamtsy, gesamtsz, 'og') %Anzeige des Punktes
        plot3([0,0], [gesamtsy, gesamtsy], [-500,30], '--g')%Linie zur vertikalen Projektion des
Gesschwerpunktes
    end

end
```

Text 4: `simulation.m`

Auch die Schwerpunktberechnung wurde erweitert (Text 8), hier wurden 12 (in der 2D-Version waren es 7) Segmente berücksichtigt. Durch den neue Zeichenpfad mit Sprüngen

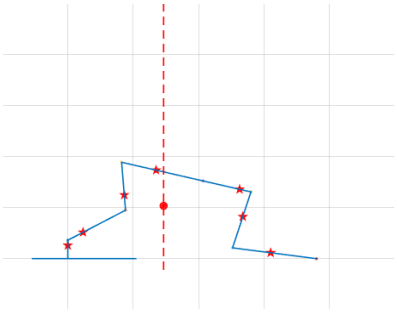


Illustration 8: 2D-Aufsteh-algorithmus: Zeile 40

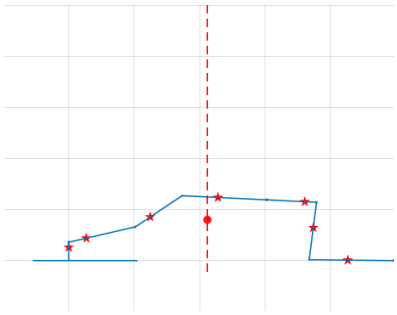


Illustration 9: 2D-Aufsteh-algorithmus: Zeile 30

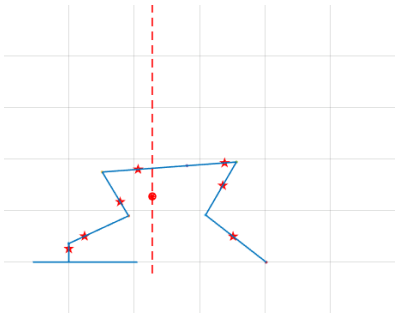


Illustration 10: 2D-Aufsteh-algorithmus: Zeile 50

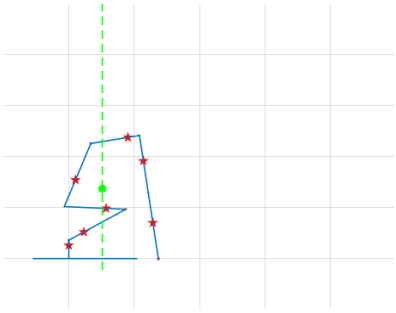


Illustration 11: 2D-Aufsteh-algorithmus: Zeile 60

und zur Reduktion der Komplexität in der Schwerpunktberechnung wird nicht wie im Grundprogramm bei jedem `i` ein Einzelschwerpunkt berechnet. Damit die Schwerpunktberechnung dennoch die bereits berechneten Verschiebungsvektoren `t` direkt nutzen kann, also in der gleichen for-Schleife untergebracht werden kann, beinhaltet `sparam` einige Platzhalter (hier mit dem Wert 999); damit das richtige Gewicht verwendet wird, wird der Schwerpunkt-Counter `sc` verwendet.

Der Gesamtschwerpunkt wird weiterhin mit der Schwerpunktgleichung berechnet (gewichtete Addition der Einzelschwerpunkte).

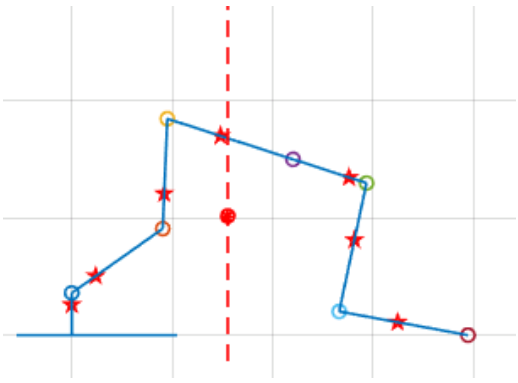


Illustration 12: Anzeige der Gelenke

```
% Verschiebungsparameter (Abstaende der Motoren)
t00 = 0; %Fusspunkt
t15 = -36;
t14 = -106;
t12 = -94;
t11 = -70;

t70 = -42.5;

t71 = -0.00000000000001; %Fusspunkt 2
t25 = -36;
t24 = -106;
t22 = -94;
t21 = -70;

t51 = -60;
t61_1 = -70;
t61_2 = 0;

t31 = 40;
t32 = 41;
t33 = 110;
thandl = 133;

t41 = 40;
t42 = 41;
t43 = 110;
thandr = 133;

% 1 x 24 Matrix fuer die Verschiebungsparameter
tparam = [t00 t15 t14 t12 t11 t70 t71 t25 t24 t22 t21 t70 t51 t61_1 t31 t32 t33 thandl t61_2 t41 t42 t43 thandr 0];
```

Text 5: `tparam` für die 3D-Simulation

Das so erweiterte Programm kann den Aufstehalgorithmus inklusive Einzel- und Gesamtschwerpunkten dreidimensional visualisieren (Illustration 13, Illustration 15, Illustration 16, Illustration 17).

Theoretisch sollte damit auch die Simulation asymmetrischer Aufstehalgorithmen möglich sein, dies wurde aber nicht getestet.

4.3.3 Neue Funktionen

Um die Simulation besser nutzen zu können, wurden zwei weitere Verbesserungen am Programm vorgenommen.

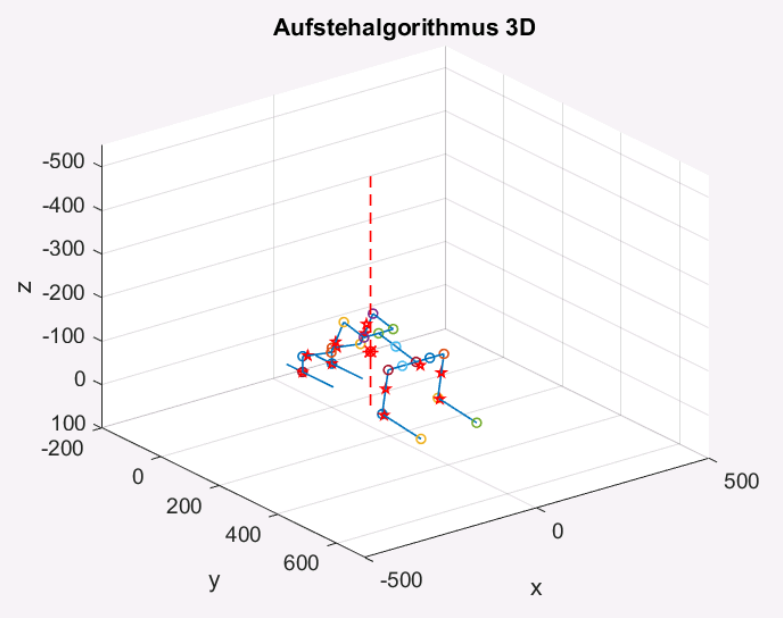


Illustration 13: Visualisierung des Aufstehalgorithmus

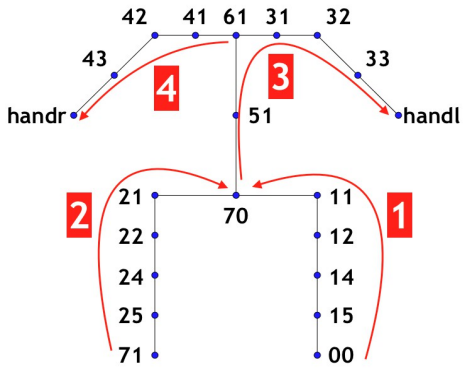


Illustration 14: Zeichenpfad der 3D-Simulation

```
%zum Punkt 71 (2. Fuss / Fuss r) springen
if(i==7), gesamtWinkelx = 0; gesamtWinkely = 0; mi = m02; mLast = m02; end

%Punkt 61 (Kopf) speichern
if(i==15)
    m_61 = mi;
    gesamtWinkelx_61 = gesamtWinkelx;
    gesamtWinkely_61 = gesamtWinkely;
end

%zum Punkt 61 (Kopf) zurueckspringen
if(i == 20), gesamtWinkelx = gesamtWinkelx_61; gesamtWinkely = gesamtWinkely_61; mi = m_61; mLast = m_61; end
```

Text 6: Neuer Zeichenpfad: 2. Fußpunkt & Speichern und Zurückspringen zu Punkt 61

```
%wann um x, wann um y rotieren
if(not(i==6||i==12||i==15||i==16||i==20||i==21))
    %x
    t = transl(0, 0, tparam(i)) * t; %Verschiebung des Koordinatenursprungs t um den
Verschiebungsparameter in z-Richtung

    if (i>=2), gesamtWinkelx = gesamtWinkelx + winkel(i-1); end %Das Fussgelenk wird nie gedreht, daher
i>1
    t = trotx(gesamtWinkelx, 'deg') * t; %Koordinatentransformation: Drehung um x-Achse um die Summe
der einzelnen Winkel

else
    %y
    t = transl(0, 0, tparam(i)) * t; %Verschiebung des Koordinatenursprungs t um den
Verschiebungsparameter in x-Richtung

    if (i>=2), gesamtWinkely = gesamtWinkely + winkel(i-1); end %Das Fussgelenk wird nie gedreht, daher
i>1
    t = troty(gesamtWinkely, 'deg') * t; %Koordinatentransformation: Drehung um y-Achse um die Summe
der einzelnen Winkel

end
```

Text 7: Unterscheidung zwischen Motoren die in x- und die in y-Richtung rotieren

```
% 1 x 24 Matrix fuer die Schwerpunkt-Verschiebungsparameter
sparam = [999 s00 s15 s14 s12 s11 999 s71 s25 s24 s22 s21 s70 s51 s61 s31 s32 s33 shandl s61 s41 s42
s43 shandr];

% 1 x 12 Matrix fuer die Gewichte
gparam = [g00 g15 g14 g71 g25 g24 g70 g51 g32 g33 g42 g43];


sx = [0 0 0 0 0 0 0 0 0 0 0 0];
sy = [0 0 0 0 0 0 0 0 0 0 0 0];
sz = [0 0 0 0 0 0 0 0 0 0 0 0];

sc = 1; %Counter fuer Schwerpunkte

%Einzelschwerpunktberechnung
if (not(sparam(i)==999))

    s = t/tparam(i) * sparam(i); %t als Einheitsvektor fuer Schwerpunkte
    si = [mi(1)+s(1) ; mi(2)+s(2) ; mi(3)+s(3); 1]; %Vektoraddition: transformierte Vektoren werden
zum vorherigen Vektor mx addiert

    plot3(si(1), si(2), si(3), 'pr') %Anzeige der Schwerpunkte

    %Abspeichern der Schwerpunktkoordinaten zur spaeteren Berechnung des
    %Ges.Schwerpunktes
    sx(sc)= si(1);
    sy(sc)= si(2);
    sz(sc)= si(3);

    sc = sc + 1;
end
```

Text 8: Schwerpunktberechnung in der 3D-Simulation

Als erstes wurde der sogenannte Debug-Mode implementiert. Dieser erlaubt es, die Simulation im Gegensatz zum bisherigen Programm, bei dem eine Unterbrechung des

Abspielens der Simulation nicht möglich war, Schritt für Schritt abzuspielen und ermöglicht somit eine bessere Analyse (Text 9).

Die zweite Verbesserung war ein Bestandteil des Programms, der zwecks Optimierung der Winkeltabellen überprüfte, ob und wenn ja wie weit ein Teil des Roboters in der Realität im Boden wäre (Text 10[2], Illustration 18). Dafür können bereits vorhandene Werte genutzt werden.

4.3.4 3D-Simulation Laufen

Mithilfe des so entstandenen Programms konnte zwar der Aufstehalgorithmus wesentlich verbessert werden, um ein Laufen zu simulieren ist es allerdings ungeeignet.

Bislang waren die Koordinaten der beiden Fußpunkte als konstant definiert, was beim Laufen nicht mehr möglich ist. Im neuen Programm wird immer ein Fuß als Fixpunkt genommen, relativ zu diesem wird dann der restliche Roboter konstruiert (Illustration 20, Illustration 19). Handstand oder Hüpfen können so nicht simuliert werden. Die Position des anderen Fußes wird nach jedem Bewegungsschritt gespeichert. Durch ein abwechselndes Verwenden der Füße als Fixpunkte kann der Roboter sich im virtuellen Raum bewegen.

Welcher Fuß derzeit Fixpunkt ist, gibt Spalte 21 der Winkeltabelle an. 0 heißt beide, 1 nur der linke Fuß, 2 nur der rechte Fuß (Text 11).

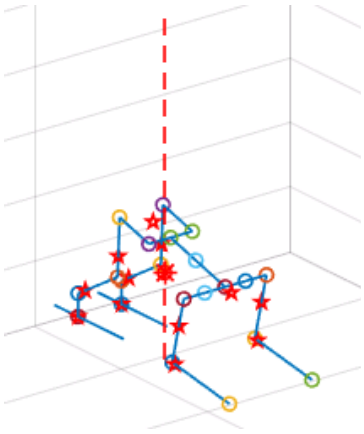


Illustration 15: Visualisierung des Aufstehalgorithmus (Ausschnitt)

2 Bei `i` = 19 und `i` = 24 ist keine Überprüfung notwendig: bei 19 ist der Sprung zurück zum Kopf und 24 ist eine Art Platzhalter.

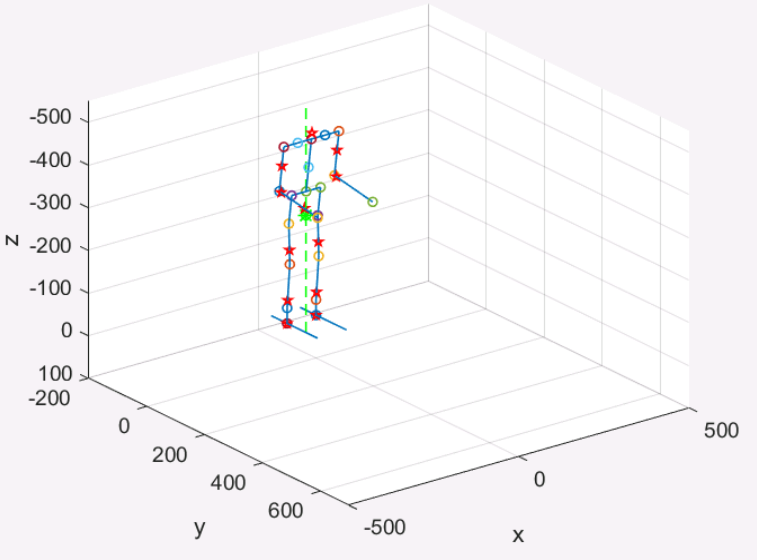


Illustration 16: Visualisierung des Aufstehalgorithmus (Ausschnitt)

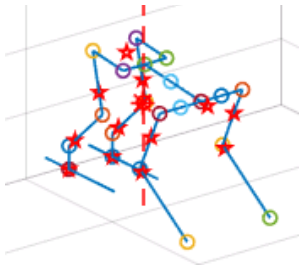


Illustration 17: Visualisierung des Aufstehalgorithmus (Ausschnitt)

```
debug = 0;
disp '***';
disp 'Debug-Mode aktiviert: mit Maustaste Zeile fuer Zeile; Leertaste um Debug-Mode zu beenden';
disp '***';

for i = 1 : 1 : anzZeilen

    if debug == 0
        debug = waitforbuttonpress;
    end
```

Text 9: Implementierung des Debug-Mode

Auch die Schwerpunktberechnung wurde entsprechend angepasst: Nun wird überprüft, ob der Schwerpunkt über dem Fuß liegt, der gerade Bodenkontakt hat (Text 12).

Haben beide Füße Bodenkontakt, überprüft das Programm die Lage des Schwerpunktes wie bisher. Wenn beide Füße Bodenkontakt haben und versetzt stehen, funktioniert die Schwerpunktberechnung so nicht, denn hierfür müsste das Program überprüfen ob der Schwerpunkt über der konvexen Hülle, die beide Füße beinhaltet, liegen, was es derzeit noch nicht tut.

Zur besseren Visualisierung werden die Füße nun auch als Rechteck und nicht mehr nur als Strecke wie bisher dargestellt (Illustration 21).

```
izupunkt={'00', '15', '14', '12', '11', '70', '71', '25', '24', '22', '21', '70', '51', '61', '31', '32',
'33', 'handl', '61', '41', '42', '43', 'handr'};

%Gibt es Punkte, die in der Realitaet im Boden waeren?
if(mi(3)>0 && i~= 19 && i~=24)
    disp(['Punkt' , izupunkt(i),'ist', num2str(mi(3)),'im Boden!']);
end
```

Text 10: Anzeigen, wenn und wie weit Teile des Roboters im Boden sind

```
Command Window
Zeile: 8
Zeile: 9
Zeile: 10
    'Punkt'    'handl'    'ist'    '18.0571'    'im Boden!'

    'Punkt'    'handr'    'ist'    '18.0571'    'im Boden!'

Zeile: 11
    'Punkt'    'handl'    'ist'    '4.8793'    'im Boden!'

    'Punkt'    'handr'    'ist'    '4.8793'    'im Boden!'

Zeile: 12
    'Punkt'    'handl'    'ist'    '2.8561'    'im Boden!'
```

Illustration 18: Ausgabebeispiel, wenn Teile des Roboters im Boden sind

Auch implementiert wurden die Motoren, die für die seitlichen Rotationen (um die y-Achse) verantwortlich sind. Beim Aufstehalgorithmus wurden sie nicht gebraucht, aber beim Laufen sind sie für die Gewichtsverlagerung wichtig (Illustration 22).

Da mehrere Motoren an den Gelenken 15, 25, 12, 22, 32, 42 sitzen, werden die zusätzlichen Rotationen als Bonuswinkel hinzugefügt (Illustration 23, Text 13).

Die Realitätsnähe des so erweiterten Simulationsprogramms konnte aufgrund des weiter unten beschriebenen Defekts nicht mehr getestet werden. Allerdings wurde eine Winkeltabelle erstellt, die zumindest bestätigt, dass die Simulation für sich funktioniert (Illustration 24, Illustration 25, Illustration 26, Illustration 27). Der darin beschriebene Bewegungsablauf zeigt, dass das Ansteuern der bislang ungenutzten Motoren und die Schwerpunktberechnung beim Stehen auf einem Bein in der Simulation funktionieren, wodurch die theoretische Möglichkeit des Simulieren eines Laufalgorithmus' gegeben ist.

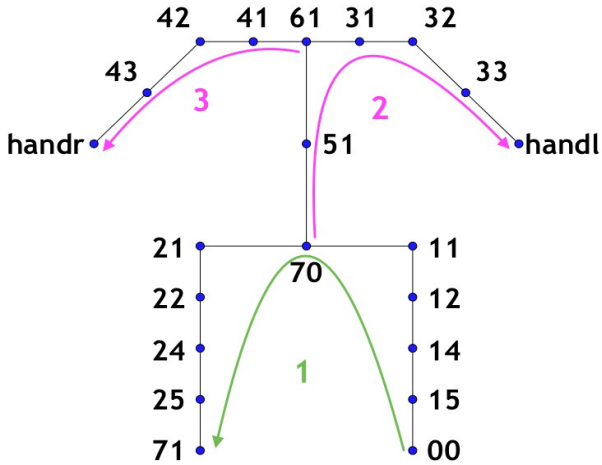


Illustration 20: Neue Konstruktionsvorschrift. Start linker Fuß.

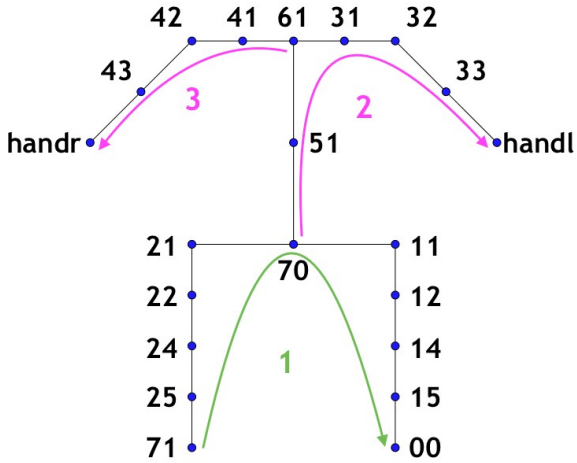


Illustration 19: Start rechter Fuß.

```
startfuss = winkeltabelle.data(i, 21);

if(startfuss <= 1)
    m0 = m0_1;
else
    m0 = m0_2;
end


%Speichern der Fusspunkte (damit der Roboter weiss wo er ist)
if(i==11)
    if(startfuss <= 1), m0_2 = mi; end
    if(startfuss == 2), m0_1 = mi; end
end
```

Text 11: Welcher Fuß wird als Fixpunkt verwendet?

```
if(startfuss == 1)
    if (gesamtsx > m0_1(1)+67 || gesamtsx < m0_1(1)-25 || gesamtsy < m0_1(2)-57 || gesamtsy >
m0_1(2)+106)
        color = 'r';
    else
        color = 'g';
    end
end

if(startfuss == 2)
    if (gesamtsx < m0_2(1)-67 || gesamtsx > m0_2(1)+25 || gesamtsy < m0_2(2)-57 || gesamtsy >
m0_2(2)+106)
        color = 'r';
    else
        color = 'g';
    end
end
```

Text 12: Überprüfen, ob Schwerpunkt über dem Fuß liegt, der auf dem Boden steht

5 Fehlerbetrachtung

5.1 Gewichte der Beine

Beim Auseinderbauen von Elvis fiel auf, dass die beiden Beine unterschiedlich viel wiegen. Allerdings war der Unterschied so gering, dass es für den Roboter keine Bedeutung hatte. Das Problem erhielt keine weitere Beachtung.

5.2 Vereinfachung physikalischer Prozesse

Auch in der verbesserten Simulation werden viele physikalischen Prozesse weiterhin nicht berücksichtigt. Dazu gehören die Trägheit des Roboters, Reibung zwischen dem Roboter und der Oberfläche und Umweltparameter wie Temperatur und Luftdruck. Die Programmierung dieser Umstände wäre zu kompliziert gewesen, die Trägheit kann bedingt

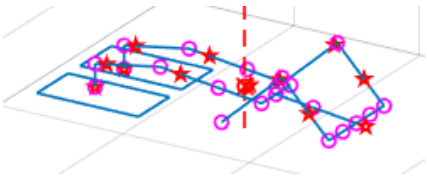


Illustration 21: Füße, die Bodenkontakt haben, werden als Rechteck dargestellt

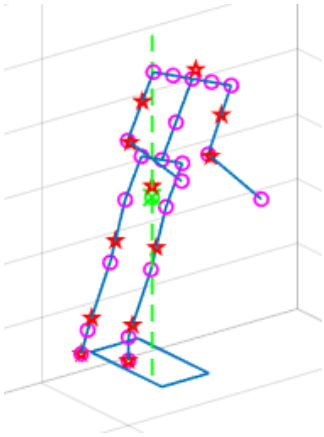


Illustration 22: Beispiel für seitliche Rotation aus dem Fuß

durch die kleinschrittigen Bewegungen des Roboters wohl vernachlässigt werden, die Umweltparameter sind durch ihren geringen Einfluss vernachlässigbar.

Reibungskräfte allerdings scheinen unseren Beobachtungen zufolge in der Praxis relevant zu sein, diesen sollte in zukünftigen Projekten vielleicht mehr Aufmerksamkeit gewidmet werden.

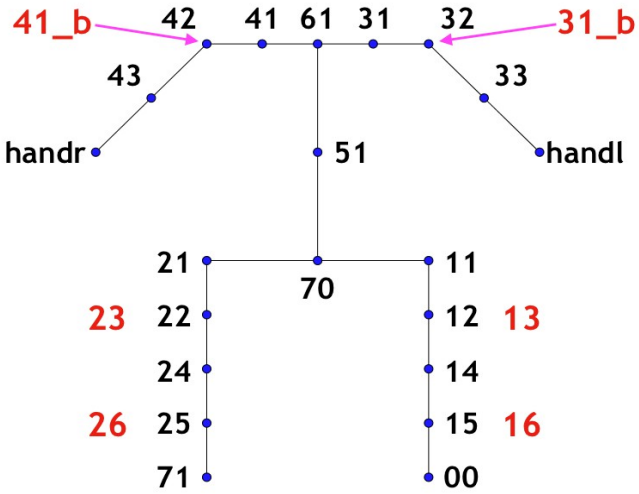


Illustration 23: Gelenken, an denen 2 Motoren sitzen, werden 2 Winkel zugewiesen

```
%Bonuswinkel fuer Laufen
w16 = winkeltabelle.data(i, 6);
w26 = winkeltabelle.data(i, 12);
w13 = winkeltabelle.data(i, 3);
w23 = winkeltabelle.data(i, 9);

w31_b = winkeltabelle.data(i, 1);
if(w32<0), w31_b = -w31_b - 90; else, w31_b = w31_b - 90; end
w41_b = winkeltabelle.data(i, 7);
if(w42<0), w41_b = w41_b + 90; else, w41_b = w41_b + 90; end


if(i==3||i==5||i==9||i==11||i==16||i==20) %Motoren fuer seitliche Rotation: 16,13,23,26; 31_b, 41_b
    gesamtWinkely = gesamtWinkely + bonuswinkel(bw);
    bw = bw + 1;
end
```

Text 13: Gelenke, an denen 2 Motoren sitzen, müssen auch um 2 Winkel rotieren

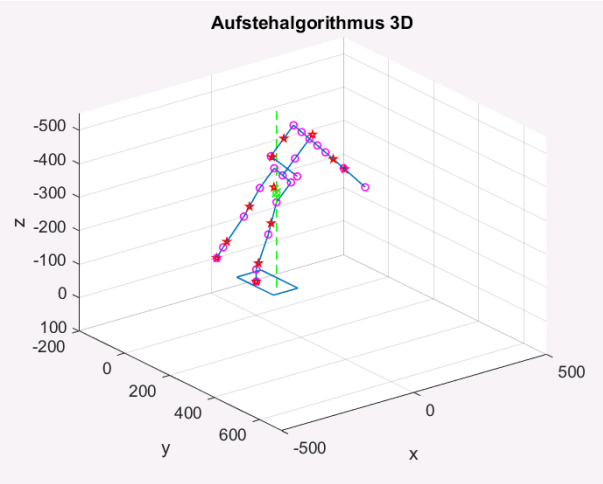


Illustration 24: Um auf einem Bein stehen zu können, muss das Ungleichgewicht durch Strecken des gegenüberliegenden Armes ausgeglichen werden.

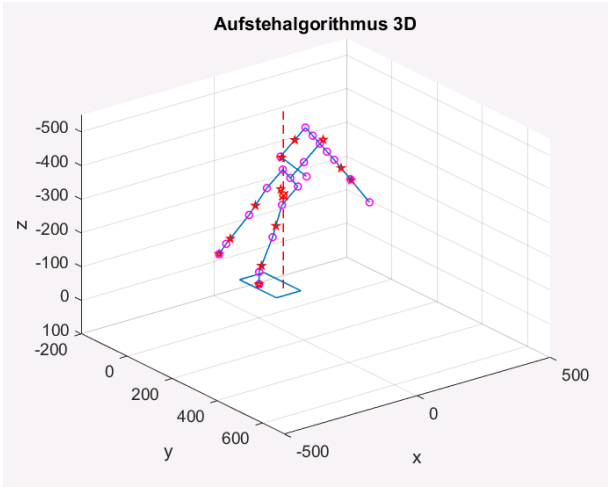


Illustration 25: Wenn die Neigung zu einer Seite hin zu groß wird, entsteht ein Ungleichgewicht.

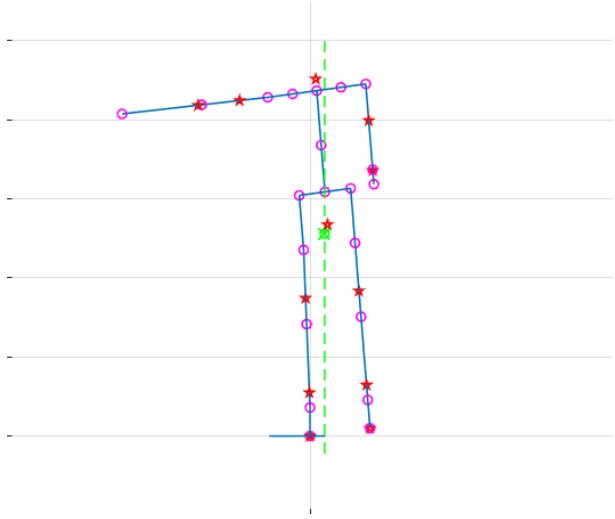


Illustration 26: Durch Ausstrecken des Arms bleibt der Roboter im Gleichgewicht.

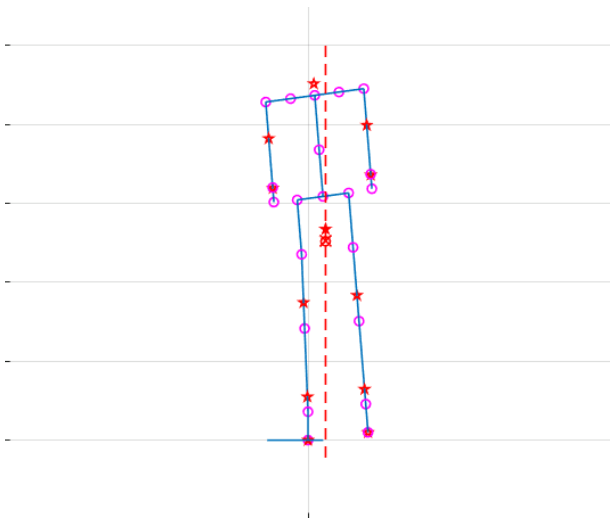


Illustration 27: Bei gleicher Neigung, aber ohne Ausstrecken des Arms, liegt der Schwerpunkt nicht mehr über dem den Boden berührenden Fuß.

5.3 Fuß nicht komplett hochklappbar

Der Aufbau von Elvis lässt es nicht zu, dass die Füße komplett an die Beine angewinkelt sind. In der Starthaltung des Aufstehalgorithmus aus der Bauchlage der Vorgängergruppe war dies allerdings vorausgesetzt. Da diese Haltung durch den Motor nicht zu erreichen war, kam es zu Problemen, die dafür sorgten, dass der Roboter beim Aufstehen meist hinfiel. Das Problem wurde durch eine Veränderung der Winkeltabelle behoben, die die Startposition des Roboters in eine realistische Position veränderte und die ersten Bewegungen anpasste, damit der Roboter in die für den weiteren Algorithmus benötigte Position kam (Illustration 29, Illustration 28).

Dadurch konnte Elvis nun ohne Fehler aus der Bauchlage aufstehen.

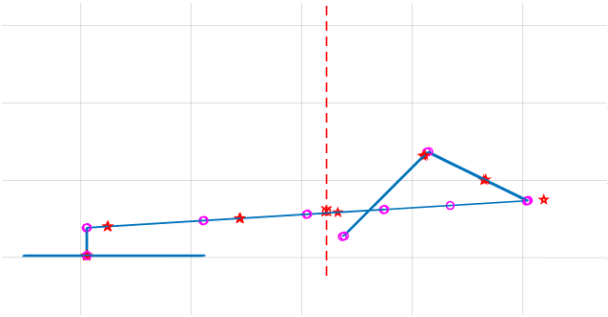


Illustration 29: Alte Position

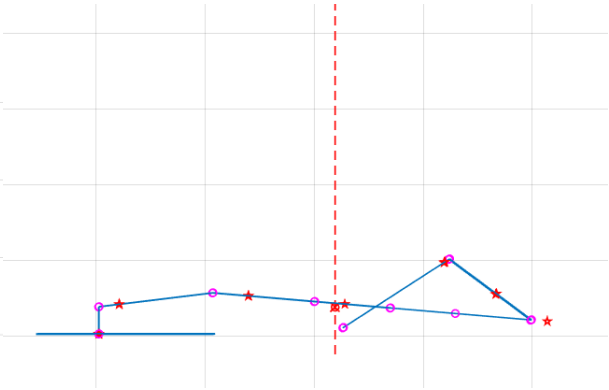


Illustration 28: Neue Position

6 Fazit

6.1 Erkenntnisse

Elvis ist fragil, bei unseren wenigen Praxistests lösten sich Schrauben und Kabel, deshalb können Praxistests nicht in großer Zahl durchgeführt werden. Auch deswegen und wegen der besseren Analysemöglichkeiten ist die Simulation bei der Entwicklung und Optimierung von Bewegungsabläufen von entscheidender Bedeutung. Beim Arbeiten mit der Simulation dürfen die Vereinfachungen aber nicht vergessen werden, sonst führen diese zu Komplikationen - wie beispielsweise in "6.1.3 Fuß nicht komplett hochklappbar" beschrieben.

6.2 Ergebnisse

Der verbesserte Aufstehalgorithmus, der den Roboter aus der Bauchlage heraus mit hoher Zuverlässigkeit in eine stabile Standposition bringt, bei der beide Füße nebeneinander stehen, ist eine wichtige Grundlage für alle weiteren Bewegungsabläufe aus dem Stehen heraus, zu denen auch das Laufen zählt. Dieser konnte dank der 3D-Simulation entwickelt werden, die in ihrer Grundvariante nicht nur einen besseren Überblick über das Geschehen durch 3D-Ansicht und Rotationsmöglichkeiten bietet, sondern auch zusätzliche neue Funktionen beinhaltet, wie den Debug-Mode und das Anzeigen von Körperteilen, die sich in der Simulation im Boden befinden. Auch ist damit die Simulation asymmetrischer Aufstehalgorithmen erstmals möglich. Schlussendlich ist als Ergebnis noch die Erweiterung der Simulation zu nennen, die eine Bewegung im virtuellen Raum durch variable Fußpunkte, das Integrieren der Motoren für seitliche Rotation in das Programm und die Schwerpunktanzeige, die berücksichtigt, ob der Schwerpunkt über dem Fuß liegt, der gerade am Boden ist, wenn nur ein Fuß am Boden ist. Durch diese Erweiterungen sollte das Simulieren eines Laufalgorithmus möglich sein (siehe hierzu auch 6.4 Ausblick).

Auch wenn Elvis von bipeder Fortbewegung noch viele Versuche entfernt sein dürfte, wurden auf dem Weg dort hin beachtliche Fortschritte - insbesondere im virtuellen Bereich - erzielt.

Des Weiteren erweiterten die an der Kooperation Teilnehmenden ihre Kenntnisse und Erfahrungen in Robotik und Informatik, insbesondere in den Bereichen Koordinatentransformation und Kinematik in Theorie und Praxis - und hatten Freude an der Sache.

6.3 Ausblick

Während er von einer anderen Gruppe benutzt wurde, traten an einigen elektronischen Komponenten von Elvis Schäden auf. Zum aktuellen Zeitpunkt wissen wir nicht, ob er je wieder benutzbar sein wird.

Allerdings ist die Simulation auch auf andere Laufroboter ähnlicher Bauweise anwendbar, dafür müssten lediglich einige Parameter angepasst werden. Im Abschnitt "Methoden" ist die Programmstruktur auch deswegen detailliert erklärt, damit das Simulationsprogramm weiter verwendet werden kann.

In Folgeprojekten sollte den Reibungskräften und der damit verbundenen Betrachtung der Untergrund- und Schuhoberflächen mehr Aufmerksamkeit gewidmet werden, als dies bei diesem Projekt der Fall war.

Bei einem Gespräch mit einem Mediziner fiel auf, dass den am Projekt Beteiligten die biologischen Hintergründe bipeder Fortbewegung unbekannt waren. Eine genauere Betrachtung dieser könnte auch mit einem Erkenntnisgewinn bezüglich Laufrobotern verbunden sein.

7 Danksagung

An dieser Stelle möchten wir mehreren Personen danken. An erster Stelle der Hector Stiftung und ganz besonders Herr und Frau Hector, die die Arbeit des Hector Seminars und damit auch dieses Kursprojekt überhaupt erst ermöglicht haben. Wir bedanken uns bei all den Kursleiter*innen, die uns bis zur Projektphase Wissen, Fertigkeiten und Erfahrungen vermittelt haben, sowie denen, die uns während der Projektphase begleiteten und an der Nachbereitung des Projekts beteiligt waren; hierzu zählen vorallem Herr Raque, Herr Gölz, Herr Rudolph, Herr Taulien und Frau Krämer. Wir danken der Hochschule Mannheim für die bereitgestellte Infrastruktur. In besonderem Maße danken wir Herrn Prof. Dr. Thomas Ihme, der mit uns viele Vor- und Nachmittage, verbracht hat, uns mit Theorie und Praxis unterstützt hat und stets für Fragen zur Verfügung stand. Zu guter Letzt danken wir allen, die hier nicht erwähnt werden, weil wir ihre Bedeutung nicht bemerkt oder sie leider vergessen haben.

8 Quellen

Informationsquellen

https://services.informatik.hs-mannheim.de/~ihme/lectures/AMR_Files/03_Transformationen.pdf, 2017-09-14
https://services.informatik.hs-mannheim.de/~ihme/lectures/AMR_Files/04_Kinematik.pdf, 2017-09-14

Bildquellen

Alle Bilder im Bereich "3 Theorie" https://services.informatik.hs-mannheim.de/~ihme/lectures/AMR_Files/03_Transformationen.pdf, 2017-09-14

Alle anderen Bilder: eigene Fotografien / Screenshots

9 Anhang

9.1 Quellcode Aufstehen_v7

```
%Winkel importieren
winkeltabelle = importdata('winkeltabelle_8.csv',',', 1);

anzZeilen = 137; %hier sollte die Anzahl der Zeilen der Winkeltabelle stehen

debug = 0;
disp '***';
disp 'Debugmode aktiviert: mit Maustaste Zeile fuer Zeile; Leertaste um Debugmode zu beenden';
disp '***';

commandwindow;

% Verschiebungsparameter (Abstaende der Motoren)
t00 = 0; %Fusspunkt
t15 = -36;
t14 = -106;
t12 = -94;
t11 = -70;

t70 = -42.5;

t71 = -0.00000000000001; %Fusspunkt 2
t25 = -36;
t24 = -106;
t22 = -94;
t21 = -70;

%t70 = -42.5;

t51 = -60;
t61_1 = -70;
t61_2 = 0;

t31 = 40;
t32 = 41;
t33 = 110;
thandl = 133;

t41 = 40;
t42 = 41;
t43 = 110;
thandr = 133;

% 1 x 24 Matrix fuer die Verschiebungsparameter
%tparam = [t15 t14 t12 t51 t31 t33 thandl];
tparam = [t00 t15 t14 t12 t11 t70 t71 t25 t24 t22 t21 t70 t51 t61_1 t31 t32 t33 thandl t61_2 t41 t42 t43 thandr 0];

%Schwerpunkte der einzelnen Gelenke, alle mit 999 werden nicht gebraucht
s00 = -0;
s15 = -19;
s14 = -33;
s12 = 999;
s11 = 999;
s70 = 42;

s71 = 0;
s25 = -19;
s24 = -33;
s22 = 999;
s21 = 999;

s51 = -85;

s61 = 999;
s31 = 999;
s32 = 47;
s33 = 6; %todo wert ueberpruefen!
```

```
shandl = 999;

s41 = 999;
s42 = 47;
s43 = 6;
shandr = 999;

% 1 x 24 Matrix fuer die Schwerpunkt-Verschiebungsparameter
sparam = [999 s00 s15 s14 s12 s11 999 s71 s25 s24 s22 s21 s70 s51 s61 s31 s32 s33 shandl s61 s41 s42 s43 shandr];

%Gewichte der Segmente (benannt nach unterem Punkt), alle mit 0 werden
%nicht gebraucht
g00 = 115.6;
g15 = 238;
g14 = 125.6;
g12 = 0;
g11 = 0;

g70 = 746.4;

g71 = 115.6;
g25 = 238;
g24 = 125.6;
g22 = 0;
g21 = 0;

g51 = 492.2;
g61 = 0;

g31 = 0;
g32 = 202;
g33 = 18.4;
ghandl = 0;

g41 = 0;
g42 = 202;
g43 = 18.4;
ghandr = 0;
% 1 x 12 Matrix fuer die Gewichte
gparam = [g00 g15 g14 g71 g25 g24 g70 g51 g32 g33 g42 g43];

% Nullvektor = Fusspunkt des Roboters
m0 = [0; 0; 0; 1];

% Fusspunkt 2
m02 = [-85; 0; 0; 1];

%set(h,'Position',[200 200 720 576]);

for i = 1 : 1 : anzZeilen

    if debug == 0
        debug = waitforbuttonpress;
    end

    disp(['Zeile: ',num2str(i)])

    clf

    %Einrichten des Fensters
    grid on
    set(gca,'FontSize',14);
    set(gca,'DefaultLineLineWidth',1.2)
    xlabel('x')
    ylabel('y')
    zlabel('z')
    title ('Aufstehalgorithmus 3D')
    axis([-500 500 -200 700 -550 100]) %todo: x-Bereich anpassen
    set(gca,'zdir', 'reverse') %Richtung der z-Achse umdrehen
    set(gca,'ydir', 'reverse') %Richtung der z-Achse umdrehen

    %view fuer 2d-model: view(-90,0)
    %view(-90,0)
    view(3) %todo: view optimieren
    %view(-180,0);
    h = figure(1);
    set(h,'Units','inches','Position',[1 1 7.19 5.76]);
    %set(h,'Units','inches','Position',[1 1 3.6 2.88]);
    hold on


    % Winkel

    alpha51 = winkeltabelle.data(i, 19);

    alpha15 = winkeltabelle.data(i, 5);
```

```
    alpha14 = winkeltabelle.data(i, 4);
    alpha12 = winkeltabelle.data(i, 2)*(-1);
    alpha32 = winkeltabelle.data(i, 13); %Achtung: Gelenk 32 f¸r sim, Motor 31 in Praxis
    alpha33 = winkeltabelle.data(i, 15)-90;

    %neue Winkel
    %symmetrische
    alpha25 = winkeltabelle.data(i, 11)*(-1);
    alpha24 = winkeltabelle.data(i, 10)*(-1);
    alpha22 = winkeltabelle.data(i, 8)*(-1);
    alpha42 = winkeltabelle.data(i, 16)*(-1);
    alpha43 = (winkeltabelle.data(i, 18)+90)*(-1);

    %neue Winkel, nicht in Winkeltabelle enthalten
    alpha00 = 0;
    alpha11 = 90;
    alpha70 = 0;
    alpha71 = 0;
    alpha21 = -90;
    alpha61_1 = -180;
    alpha61_2 = 0;
    alpha31 = 0;
    alphahandl = 180;
    alpha41 = 0;
    alphahandr = 0;

    % 1 x 23 Matrix fuer die Winkel
    winkel = [alpha00 alpha15 alpha14 alpha12 alpha11 alpha70 alpha71...
        alpha25 alpha24 alpha22 alpha21 alpha70 alpha51 alpha61_1 alpha31...
        alpha32 alpha33 alphahandl alpha61_2 alpha41 alpha42 alpha43 alphahandr];

    %eigentliche Transformation und Simulation - siehe simulation.m
    if(simulation(winkel, tparam, m0, m02, sparam, gparam) == false)
        disp('Break')
        %disp(i)
        %disp('Break2')
    end

    %Matrix fuer Simulation
    M(i) = getframe;

end
```

[simulation.m]

```
function weiter = simulation(winkel, tparam, m0, m02, sparam, gparam)
    weiter = true;
    mLast = m0;
    mi = m0;
    gesamtWinkelx = 0;
    gesamtWinkely = 0;
    %Fuss
    line([0, 0], [-55, 105], [0, 0])
    sx = [0 0 0 0 0 0 0 0 0 0 0 0 0];
    sy = [0 0 0 0 0 0 0 0 0 0 0 0 0];
    sz = [0 0 0 0 0 0 0 0 0 0 0 0 0];

    sc = 1; %Counter fuer Schwerpunkte

    %Fuss 2
    line([-85, -85], [-55, 105], [0, 0])

    izupunkt={'00', '15', '14', '12', '11', '70', '71', '25', '24', '22', '21', '70', '51', '61', '31', '32', '33',
'handl', '61', '41', '42', '43', 'handr'};

    for i = 1 : 1 : 24

        %Koordinatenursprung setzen, ausgehen vom Nullpunkt
        t = m0;

        %zum Punkt 71 (2. Fuss / Fuss r) springen
        if(i==7), gesamtWinkelx = 0; gesamtWinkely = 0; mi = m02; mLast = m02; end

        %Punkt 61 (Kopf) speichern
        if(i==15)
            m_61 = mi;
            gesamtWinkelx_61 = gesamtWinkelx;
            gesamtWinkely_61 = gesamtWinkely;
        end

        %zum Punkt 61 (Kopf) zurueckspringen
        if(i == 20), gesamtWinkelx = gesamtWinkelx_61; gesamtWinkely = gesamtWinkely_61; mi = m_61; mLast = m_61; end

        %wann um x, wann um y rotieren
```

```
        if(not(i==6||i==12||i==15||i==16||i==20||i==21))
            %x
            t = transl(0, 0, tparam(i)) * t; %Verschiebung des Koordinatenursprungs t um den Verschiebungsparameter
in z-Richtung

            if (i>=2), gesamtWinkelx = gesamtWinkelx + winkel(i-1); end %Das Fussgelenk wird nie gedreht, daher i>1
            t = trotx(gesamtWinkelx, 'deg') * t; %Koordinatentransformation: Drehung um x-Achse um die Summe der
einzelnen Winkel

        else
            %y
            t = transl(0, 0, tparam(i)) * t; %Verschiebung des Koordinatenursprungs t um den Verschiebungsparameter
in x-Richtung

            if (i>=2), gesamtWinkely = gesamtWinkely + winkel(i-1); end %Das Fussgelenk wird nie gedreht, daher i>1
            t = troty(gesamtWinkely, 'deg') * t; %Koordinatentransformation: Drehung um y-Achse um die Summe der
einzelnen Winkel

        end

        %Einzelschwerpunktberechnung; moeglicherweise noch fehlerhaft ;)
        if (not(sparam(i)==999))

            s = t/tparam(i) * sparam(i); %t als Einheitsvektor fuer Schwerpunkte
            si = [mi(1)+s(1) ; mi(2)+s(2) ; mi(3)+s(3); 1]; %Vektoraddition: transformierte Vektoren werden zum
vorherigen Vektor mx addiert

            plot3(si(1), si(2), si(3), 'pr') %Anzeige der Schwerpunkte

            %Abspeichern der Schwerpunktkoordinaten zur spaeteren Berechnung des
            %Ges.Schwerpunktes
            sx(sc)= si(1);
            sy(sc)= si(2);
            sz(sc)= si(3);

            sc = sc + 1;
        end

        mi = [mi(1)+t(1) ; mi(2)+t(2) ; mi(3)+t(3); 1]; %Vektoraddition: transformierte Vektoren werden zum
vorherigen Vektor mx addiert
        plot3(mi(1), mi(2), mi(3), 'o') %Anzeige der Punkte
        line([mLast(1), mi(1)], [mLast(2), mi(2)], [mLast(3), mi(3)]); %Linie zwischen dem letzten Punkt und mx
        mLast = mi;

        %Gibt es Punkte, die in der Realitaet im Boden waeren?
        if(mi(3)>0 && i~= 19 && i~=24)
            disp(['Punkt' , izupunkt(i),'ist', num2str(mi(3)),'im Boden!']);
        end

        %if (mi(3) > 45), weiter = false; end

    end
    %Gesamtschwerpunkt berechnen mit Schwerpunktgleichung
    gesamtsx = 0;
    gesamtsy = 0;
    gesamtsz = 0;
    gesamtmasse = 0;

    for (i = 1 : 1 : 12)
        gesamtsx = gesamtsx + sx(i) * gparam(i);
        gesamtsy = gesamtsy + sy(i) * gparam(i);
        gesamtsz = gesamtsz + sz(i) * gparam(i);
        gesamtmasse = gesamtmasse + gparam(i);
    end

    gesamtsx = gesamtsx / gesamtmasse;
    gesamtsy = gesamtsy / gesamtmasse;
    gesamtsz = gesamtsz / gesamtmasse;

    %disp(gesamtsx);

    %Ueberpruefung, ob ueber dem Fuss fuer vertikale Linie
    if (gesamtsy < -55 || gesamtsy > 105)
        color = 'r';
    else
        color = 'g';
    end

    plot3(gesamtsx, gesamtsy, gesamtsz, ['*' color], 'MarkerSize', 10); %Anzeige des Punktes
    plot3(gesamtsx, gesamtsy, gesamtsz, ['o' color]); %Anzeige des Punktes
    plot3([gesamtsx,gesamtsx], [gesamtsy, gesamtsy], [-500,30], ['--' color]); %Linie zur vertikalen Projektion d.
Gesamtschwerpunkte

end
```

9.2 Quellcode Laufen_v5

```
%{

2017-09-13
Finn Buschmann, Holger Hujion

*Funktioniert, ausser der Schwerpunktberechnung wenn beide Fuesse auf dem Boden
sind und nicht nebeneinander stehen (ist fuer unsere Demo-Simulation nicht
notwendig)

*Verwendet stets einen Fuss als Fixpunkt (genannt startfuss; muss auf Boden stehen), dieser wird
durch die Spalte 21(entspricht U) der Winkeltabelle angegeben:
0 = beide Fuesse (linker Fuss wird dann verwendet); 1 = linker Fuss; 2 = rechter Fuss
%}

% Reine Simulation

%Winkel importieren
winkeltabelle = importdata('winkeltabelle_x.csv',',', 1);

anzZeilen = 34;  %hier sollte die Anzahl der Zeilen der Winkeltabelle stehen

debug = 1; %0 = deaktiviert, 1 = aktiviert
if(debug)
    disp '***';
    disp 'Debugmode aktiviert: mit Leertaste Zeile fuer Zeile; Mausklick um Debugmode zu beenden';
    disp '***';
end

commandwindow;

% Verschiebungsparameter (Abstaende der Motoren)
t00 = 0; %Fusspunkt
t15 = -36;
t14 = -106;
t12 = -94;
t11 = -70;

t70 = -42.5;

t71 = -0.00000000000001; %Fusspunkt 2
t25 = -36;
t24 = -106;
t22 = -94;
t21 = -70;

t51 = -60;
t61 = -70;

t31 = 40;
t32 = 41;
t33 = 110;
thandl = 133;

t41 = 40;
t42 = 41;
t43 = 110;
thandr = 133;

% 1 x 21 Matrix fuer die Verschiebungsparameter
%tparam = [t00 t15 t14 t12 t11 t70 t71 t25 t24 t22 t21 t70 t51 t61_1 t31 t32 t33 thandl t61_2 t41 t42 t43 thandr 0];
tparam1 = [t00 t15 t14 t12 t11 t70 t70 t21 t22 t24 t25 t51 t61 t31 t32 t33 thandl t41 t42 t43 thandr];
tparam2 = [t71 t25 t24 t22 t21 t70 t70 t11 t12 t14 t15 t51 t61 t31 t32 t33 thandl t41 t42 t43 thandr];

%Schwerpunkte der einzelnen Gelenke, alle mit 999 werden nicht gebraucht
s00 = -0;
s15 = -19;
s14 = -33;
s12 = 999;
s11 = 999;
s70 = 42;

s71 = 0;
s25 = -19;
s24 = -33;
s22 = 999;
s21 = 999;

s51 = -85;

s61 = 999;
s31 = 999;
s32 = 47;
s33 = 6; %todo wert ueberpruefen!
```

```
shandl = 999;

s41 = 999;
s42 = 47;
s43 = 6;
shandr = 999;

% 1 x 20 Matrix fuer die Schwerpunkt-Verschiebungsparameter
%sparam = [999 s00 s15 s14 s12 s11 999 s71 s25 s24 s22 s21 s70 s51 s61 s31 s32 s33 shandl s61 s41 s42 s43 shandr];
sparam1 = [999 s00 s15 s14 s12 s11 s21 s22 t22-s24 t24-s25 t25-s71 s70 s51 s61 s31 s32 s33 s61 s41 s42 s43];
sparam2 = [999 s71 s25 s24 s22 s21 s11 s12 t12-s14 t14-s15 t15-s00 s70 s51 s61 s31 s32 s33 s61 s41 s42 s43];

%Gewichte der Segmente (benannt nach unterem Punkt), alle mit 0 werden nicht gebraucht
g00 = 115.6;
g15 = 238;
g14 = 125.6;

g70 = 746.4;

g71 = 115.6;
g25 = 238;
g24 = 125.6;

g51 = 492.2;

g32 = 202;
g33 = 18.4;

g42 = 202;
g43 = 18.4;
% 1 x 12 Matrix fuer die Gewichte
gparam = [g00 g15 g14 g71 g25 g24 g70 g51 g32 g33 g42 g43];

global m0_1; %damit alle Funktionen das gleiche m0_1, m0_2 verwenden
global m0_2;

m0_1 = [0; 0; 0; 1]; % Fusspunkt 1 (links)
m0_2 = [-85; 0; 0; 1]; % Fusspunkt 2 (rechts)

%set(h,'Position',[200 200 720 576]);

for i = 1 : 1 : anzZeilen
    if debug == 1
        debug = waitforbuttonpress;
    end

    disp(['Zeile: ',num2str(i)])

    clf

    %Einrichten des Fensters
    grid on; set(gca,'FontSize',14); set(gca,'DefaultLineLineWidth',1.2);
    xlabel('x'); ylabel('y'); zlabel('z'); title ('Aufstehalgorithmus 3D');
    axis([-500 500 -200 700 -550 100]) %todo: x-Bereich anpassen
    set(gca,'zdir', 'reverse') %Richtung der z-Achse umdrehen
    set(gca,'ydir', 'reverse') %Richtung der z-Achse umdrehen

    %view fuer 2d-model: view(-90,0)
    %view(-90,0)
    view(3) %todo: view optimieren
    %view(-180,0);
    h = figure(1);
    set(h,'Units','inches','Position',[1 1 7.19 5.76]);
    %set(h,'Units','inches','Position',[1 1 3.6 2.88]);
    hold on


    % Winkel

    w51 = winkeltabelle.data(i, 19);

    w15 = winkeltabelle.data(i, 5);
    w14 = winkeltabelle.data(i, 4);
    w12 = winkeltabelle.data(i, 2)*(-1);
    w32 = winkeltabelle.data(i, 13); %Achtung: Gelenk 32 f¸r sim, Motor 31 in Praxis
    w33 = winkeltabelle.data(i, 15)-90;

    %neue Winkel (von 2D zu 3D)
    %symmetrische
    w25 = winkeltabelle.data(i, 11)*(-1);
    w24 = winkeltabelle.data(i, 10)*(-1);
    w22 = winkeltabelle.data(i, 8)*(-1);
    w42 = winkeltabelle.data(i, 16)*(-1);
    w43 = (winkeltabelle.data(i, 18)+90)*(-1);

    %neue Winkel, nicht in Winkeltabelle enthalten
```

```
    w00 = 0;
    w11 = 90;
    w70_1 = 0;
    w70_2 = 0; %todo: hier passenden Wert einsetzen
    w71 = 0;
    w21 = -90;
    w61_1 = 90;
    w61_2 = -90;
    whandl = 180;
    whandr = 0;
    w31 = 0;
    w41 = 0;

    %Bonuswinkel fuer Laufen
    w16 = winkeltabelle.data(i, 6);
    w26 = winkeltabelle.data(i, 12);
    w13 = winkeltabelle.data(i, 3);
    w23 = winkeltabelle.data(i, 9);

    w31_b = winkeltabelle.data(i, 1);
    if(w32<0), w31_b = -w31_b - 90; else, w31_b = w31_b - 90; end
    w41_b = winkeltabelle.data(i, 7);
    if(w42<0), w41_b = w41_b + 90; else, w41_b = w41_b + 90; end

    %tparam = [t00 t15 t14 t12 t11 t70 t71 t25 t24 t22 t21 t70 t51 t61_1 t31 t32 t33 thandl t61_2 t41 t42 t43 thandr
0];

    % 1 x 20 Matrix fuer die Winkel
    %winkel = [w00 w15 w14 w12 w11 w70 w71 w25 w24 w22 w21 w70 w51 w61_1 w31 w32 w33 whandl w61_2 w41 w42 w43
whandr];
    winkel1 = [w00 w15 w14 w12 w11 w70_1 w21-90 -w22 -w24 -w25 w70_2 w51 w61_1 w31 w32 w33 w61_2 w41 w42 w43];
    winkel2 = [w71 w25 w24 w22 w21 w70_1 w11+90 -w12 -w14 -w15 w70_2 w51 w61_1 w31 w32 w33 w61_2 w41 w42 w43];

    % 1 x 6 Matrix fuer die Bonuswinkel, werden benoetigt, da 2 Motoren am gleichen Gelenk sitzen (12 & 13, 22 & 23).
    % 13 und 23 sind dabei fuer die seitliche Rotation (links/rechts) verantwortlich, 12 & 22 fuer vorne/hinten;
    % Gleiches gilt an den Gelenken 25 & 26, 15 & 16: hier ist 15 und 25 fuer
    % vorne/hinten verantwortlich, 26 & 16 fuer die seitliche Rotation
    % Genauso bei der Schulter
    bonuswinkel1 = [w16 w13 w23 w26 w31_b w41_b];
    bonuswinkel2 = [w26 w23 w13-90 w16 w31_b w41_b];

    startfuss = winkeltabelle.data(i, 21);

    if(startfuss <= 1)
        tparam = tparam1;
        sparam = sparam1;
        winkel = winkel1;
        bonuswinkel = bonuswinkel1;
    else
        tparam = tparam2;
        sparam = sparam2;
        winkel = winkel2;
        bonuswinkel = bonuswinkel2;
    end

    zeile = i;

    %eigentliche Transformation und Simulation - siehe simulation.m
    %if(simulation(winkel, bonuswinkel, tparam, m0_start, sparam, gparam, startfuss, zeile) == false)
        %disp('Break')
        %disp(i)
        %disp('Break2')
    %end

    simulation(winkel, bonuswinkel, tparam, sparam, gparam, startfuss);

    %Matrix fuer Simulation
    M(i) = getframe;

end
```

[simulation.m]

```
function weiter = simulation(winkel, bonuswinkel, tparam, sparam, gparam, startfuss)
    weiter = true;

    global m0_1;
    global m0_2;

    if(startfuss <= 1)
        m0 = m0_1;
    else
        m0 = m0_2;
    end
```

```
    mLast = m0;
    mi = m0;
    gesamtWinkelx = 0;
    gesamtWinkely = 0;

    sx = [0 0 0 0 0 0 0 0 0 0 0 0];
    sy = [0 0 0 0 0 0 0 0 0 0 0 0];
    sz = [0 0 0 0 0 0 0 0 0 0 0 0];

    sc = 1; %Counter fuer Schwerpunkte
    bw = 1; %Counter fuer Bonuswinkel

    %Fuss 1
    %line([0, 0], [-55, 105], [0, 0])
    %Fuss 2
    %line([-85, -85], [-55, 105], [0, 0])

    %Startfuss / Startfuesse malen
    if(startfuss == 1 || startfuss == 0)
        line([m0_1(1)+67, m0_1(1)+67], [m0_1(2)-57, m0_1(2)+106], [m0_1(3), m0_1(3)])
        line([m0_1(1)-25, m0_1(1)-25], [m0_1(2)-57, m0_1(2)+106], [m0_1(3), m0_1(3)])
        line([m0_1(1)+67, m0_1(1)-25], [m0_1(2)-57, m0_1(2)-57], [m0_1(3), m0_1(3)])
        line([m0_1(1)+67, m0_1(1)-25], [m0_1(2)+106, m0_1(2)+106], [m0_1(3), m0_1(3)])
    end
    if(startfuss == 2 || startfuss == 0)
        line([m0_2(1)-67, m0_2(1)-67], [m0_2(2)-57, m0_2(2)+106], [m0_2(3), m0_2(3)])
        line([m0_2(1)+25, m0_2(1)+25], [m0_2(2)-57, m0_2(2)+106], [m0_2(3), m0_2(3)])
        line([m0_2(1)-67, m0_2(1)+25], [m0_2(2)-57, m0_2(2)-57], [m0_2(3), m0_2(3)])
        line([m0_2(1)-67, m0_2(1)+25], [m0_2(2)+106, m0_2(2)+106], [m0_2(3), m0_2(3)])
    end

    izupunkt={'00', '15', '14', '12', '11', '70', '71', '25', '24', '22', '21', '70', '51', '61', '31', '32', '33',
'handl', '61', '41', '42', '43', 'handr'};

    for i = 1 : 1 : 21

        %Zurueckspringen
        %zum Punkt 70 (Beckenmitte)
        if(i == 12), gesamtWinkelx = gesamtWinkelx_70; gesamtWinkely = gesamtWinkely_70; mi = m_70; mLast = m_70; end
        %zum Punkt 61 (Punkt der in der Mitte zw. beiden Schultern liegt)
        if(i == 18), gesamtWinkelx = gesamtWinkelx_61; gesamtWinkely = gesamtWinkely_61; mi = m_61; mLast = m_61; end

        %Nullvektor setzen
        t = [0;0;0;1];

        %wann um x, wann um y rotieren
        if(not(i==6||i==7||i==14||i==15||i==18||i==19))
            %x
            t = transl(0, 0, tparam(i)) * t; %Verschiebung des Koordinatenursprungs t um den Verschiebungsparameter
in z-Richtung

            if (i>=2), gesamtWinkelx = gesamtWinkelx + winkel(i-1); end %Das Fussgelenk wird nie gedreht, daher i>1
            t = trotx(gesamtWinkelx, 'deg') * t; %Koordinatentransformation: Drehung um x-Achse um die Summe der
einzelnen Winkel


            if(i==3||i==5||i==9||i==11||i==16||i==20) %Motoren fuer seitliche Rotation: 16,13,23,26; 31_b, 41_b
                gesamtWinkely = gesamtWinkely + bonuswinkel(bw);
                bw = bw + 1;
            end

            if(i==8), gesamtWinkely = gesamtWinkely-90; end
            if(i==12), gesamtWinkely = gesamtWinkely-90; end

            t = troty(gesamtWinkely, 'deg') * t;

        else
            t = trotx(gesamtWinkelx, 'deg') * t;

            %z
            t = transl(0, 0, tparam(i)) * t; %Verschiebung des Koordinatenursprungs t um den Verschiebungsparameter
in z-Richtung

            if (i>=2), gesamtWinkely = gesamtWinkely + winkel(i-1); end %Das Fussgelenk wird nie gedreht, daher i>1
            t = troty(gesamtWinkely, 'deg') * t; %Koordinatentransformation: Drehung um y-Achse um die Summe der
einzelnen Winkel

        end

        %disp(gesamtWinkelx);
        %disp(gesamtWinkely);

        %Einzelschwerpunktberechnung; moeglicherweise noch fehlerhaft
        if (not(sparam(i)==999))
```

```
            s = t/tparam(i) * sparam(i); %t als Einheitsvektor fuer Schwerpunkte
            si = [mi(1)+s(1) ; mi(2)+s(2) ; mi(3)+s(3); 1]; %Vektoraddition: transformierte Vektoren werden zum
vorherigen Vektor mx addiert

            plot3(si(1), si(2), si(3), 'pr') %Anzeige der Schwerpunkte

            %Abspeichern der Schwerpunktkoordinaten zur spaeteren Berechnung des
            %Ges.Schwerpunktes
            sx(sc)= si(1);
            sy(sc)= si(2);
            sz(sc)= si(3);

            sc = sc + 1;
        end


        mi = [mi(1)+t(1) ; mi(2)+t(2) ; mi(3)+t(3); 1]; %Vektoraddition: transformierte Vektoren werden zum
vorherigen Vektor mx addiert
        plot3(mi(1), mi(2), mi(3), 'om') %Anzeige der Punkte
        line([mLast(1), mi(1)], [mLast(2), mi(2)], [mLast(3), mi(3)]); %Linie zwischen dem letzten Punkt und mx
        mLast = mi;

        %Gibt es Punkte, die in der Realitaet im Boden waeren?
        if(mi(3)>0 && i~= 19 && i~=24)
            disp(['Punkt' , izupunkt(i),'ist', num2str(mi(3)),'im Boden!']);
        end

        %Zwischenspeichern von Punkten fuer Zurueckspringen
        %Punkt 70 (Beckenmitte) speichern
        if(i==6)
            m_70 = mi;
            gesamtWinkelx_70 = gesamtWinkelx;
            gesamtWinkely_70 = gesamtWinkely;
        end

        %Punkt 61 (Punkt der in der Mitte zw. beiden Schultern liegt) speichern
        if(i==13)
            m_61 = mi;
            gesamtWinkelx_61 = gesamtWinkelx;
            gesamtWinkely_61 = gesamtWinkely;
        end

        %Speichern der Fusspunkte (damit der Roboter weiss wo er ist)
        if(i==11)
            if(startfuss <= 1), m0_2 = mi; end
            if(startfuss == 2), m0_1 = mi; end
        end

    end

    %Gesamtschwerpunkt berechnen mit Schwerpunktgleichung
    gesamtsx = 0;
    gesamtsy = 0;
    gesamtsz = 0;
    gesamtmasse = 0;

    for (i = 1 : 1 : 12)
        gesamtsx = gesamtsx + sx(i) * gparam(i);
        gesamtsy = gesamtsy + sy(i) * gparam(i);
        gesamtsz = gesamtsz + sz(i) * gparam(i);
        gesamtmasse = gesamtmasse + gparam(i);
    end

    gesamtsx = gesamtsx / gesamtmasse;
    gesamtsy = gesamtsy / gesamtmasse;
    gesamtsz = gesamtsz / gesamtmasse;

    %Ueberpruefung, ob ueber dem Fuss / den Fuessen fuer vertikale Linie
    %(funktioniert bei ueberkreuzten Beinen nicht)
    if(startfuss == 0)
        if (gesamtsx > m0_1(1)+67 || gesamtsx < m0_2(1)-76 || (m0_1(2) <= m0_2(2) && (gesamtsy < m0_1(2)-57 ||
gesamtsy > m0_2(2)+106)) ||(m0_2(2) <= m0_1(2) && (gesamtsy < m0_2(2)-57 || gesamtsy > m0_1(2)+106)))
            color = 'r';
        else
            color = 'g';
        end
    end

    if(startfuss == 1)
        if (gesamtsx > m0_1(1)+67 || gesamtsx < m0_1(1)-25 || gesamtsy < m0_1(2)-57 || gesamtsy > m0_1(2)+106)
            color = 'r';
        else
            color = 'g';
        end
    end
```

```
    if(startfuss == 2)
        if (gesamtsx < m0_2(1)-67 || gesamtsx > m0_2(1)+25 || gesamtsy < m0_2(2)-57 || gesamtsy > m0_2(2)+106)
            color = 'r';
        else
            color = 'g';
        end
    end

    plot3(gesamtsx, gesamtsy, gesamtsz, ['x' color], 'MarkerSize', 10); %Anzeige des Punktes
    plot3(gesamtsx, gesamtsy, gesamtsz, ['o' color]); %Anzeige des Punktes
    plot3([gesamtsx,gesamtsx], [gesamtsy, gesamtsy], [-500,30], ['--' color]); %Linie zur vertikalen Projektion d.
Gesamtschwerpunkte

    m0 = [0; 0; 0; 1];

end
```

10 Selbstständigkeitserklärung

Hiermit versichere ich, dass ich diese Arbeit unter der Beratung durch Prof. Dr. Thomas Ihme selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden, sowie Zitate kenntlich gemacht habe.

Ort Datum Unterschrift