FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Bestimmung von Hominoidsegmenten aus 3D-Bodyscannerdaten

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

im Rahmen des Studiums

Medieninformatik

eingereicht von

Katharina Schiffl
Matrikelnummer 0409282

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: Mag. DI Dr. Margrit Gelautz, ao. Univ. Prof
Zweitbetreuer: Univ.-Prof. Dipl.-Ing. Dr. techn. Arnold Baca

Wien, 15.07.2011

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

„Hiermit erkläre ich,
dass ich diese Arbeit selbständig verfasst habe,
dass ich die verwendeten Quellen und
Hilfsmittel vollständig angegeben habe
und dass ich die Stellen der Arbeit –
einschließlich Tabellen, Karten und Abbildungen –,
die anderen Werken oder dem Internet
im Wortlaut oder dem Sinn nach entnommen sind,
auf jeden Fall unter Angabe der Quelle
als Entlehnung kenntlich gemacht habe."

Wien, 15. 07. 2011

Abstract

Models of the human body are used to analyze and simulate body movements in a variety of fields, including sports science and computer games. These models vary in detail and accuracy, depending on their field of application. In this work, we focus on the hominoid of Hatze, which was originally developed in the context of biomechanical applications.

This thesis investigates the utility of a 3D body scanner for the automatic segmentation of scan data into the 17 segments of Hatze's anthropomorphic model. We propose a scan procedure and a set of data processing algorithms that ease the process of measuring a body and speed up the calculation of the related segment parameters. These parameter values can be employed afterwards for the simulation of human movements in areas such as sports, orthopedics or medicine.

As part of our investigations, we assess the accuracy of the used scanner (Vitronic VITUS smart 3D Bodyscanner). Basic sources of error such as the distance between scan lines, scan holes and geometric distortions are discussed, and their influence on the body measurements is further explored. We finally demonstrate the performance and repeatability of our automatic measurements in experiments with several test persons and compare the results with manual reference measurements.

Kurzfassung

In verschiedenen Bereichen der Sportwissenschaft, der Forschung aber auch in Computerspielen werden zur Analyse und Simulation der menschlichen Bewegung Modelle des menschlichen Körpers eingesetzt. Diese Körpermodelle variieren je nach Einsatzzweck an Detailliertheit und Genauigkeit. Soll ein Modell eines realen Menschen erstellt werden, muss eine – meist händische – Datenaufnahme erfolgen, welche aufwändig ist und eine Reihe von Fehlerquellen enthalten kann. Eines dieser Modelle, das Hominoid von Hatze, steht im Mittelpunkt dieser Arbeit.

Diese Diplomarbeit untersucht die Verwendung eines 3D-Bodyscanners zur automatischen Unterteilung von Scandaten in die 17 Segmente des anthropomorphen Körpermodells von Hatze. Eine Scanvorschrift und eine Reihe von Algorithmen werden entwickelt, die die Aufnahme von Körpermaßen und somit die Berechnung der Segmentparameterwerte für Simulationen von menschlichen Bewegungsabläufen beschleunigen. Diese Parameterwerte können dann in Bereichen wie Sport, Orthopädie und Medizin eingesetzt werden.

Ein weiterer Teil beschäftigt sich mit der Genauigkeit des zur Verfügung stehenden Scanners (Vitronic VITUS smart 3D-Bodyscanner). Dazu werden grundsätzliche Fehlerquellen, wie Scanlinienbreite, Scanlöcher und Ausreißer diskutiert und deren Einfluss berechnet. Tests mit verschiedenen Personen und der Vergleich der Resultate mit händischen Messungen werden schlussendlich die Performance und Wiederholbarkeit zeigen.

Inhaltsverzeichnis

1 Einleitung

1.1 Ziele

Überall dort, wo man menschliche Bewegungsabläufe computergestützt simulieren und analysieren möchte (z.B. Sportwissenschaften, Medizin, Computerspiele) benötigt man sogenannte Körpermodelle. Diese Körpermodelle variieren je nach Einsatzzweck an Detailliertheit und Genauigkeit. Soll ein Modell eines realen Menschen erstellt werden, muss eine – meist händische – Datenaufnahme erfolgen, welche aufwändig ist und eine Reihe von Fehlerquellen enthalten kann.

Das Hominoid von Hatze [4] ist ein anthropomorphes Körpermodell, das den menschlichen Körper in 17 Segmente unterteilt. Üblicherweise werden die Parameterwerte dieser Segmente aus 242 Messdaten berechnet, die direkt am Körper des Probanden vermessen werden.

Ziel dieser Arbeit ist die Entwicklung und der Test einer Methode, mit der die vom Vitronic VITUS smart 3D-Bodyscanner gelieferten Scandaten automatisch in die 17 Segmente des Hominoids von Hatze aufgeteilt werden können. Für den Test soll die Methode in einem Computerprogramm implementiert werden. Das Programm soll die Aufnahme von Körpermaßen und die Berechnung der Segmentparameterwerte für Simulationen von menschlichen Bewegungsabläufen beschleunigen.

Der Test soll zeigen, ob die Unterteilung in die 17 Segmente mit Hilfe des Vitronic VITUS smart 3D-Bodyscanners automatisch durchgeführt werden kann, und ob die Genauigkeit des Scanners und die Reproduzierbarkeit seiner Daten für wissenschaftliche Untersuchungen ausreichend ist. Dazu werden grundsätzliche Fehlerquellen, wie Scanlinienbreite, Scanlöcher und Ausreißer untersucht und deren prozentuelle Abweichungen berechnet. Sollten diese Werte hinreichend konstant sein, könnten die berechneten Ergebnisse automatisch korrigiert werden.

1.2 Vorgaben

1.2.1 Segmentierungsmodell - Das Hominoid von Hatze

Das Hominoid von Hatze [4] soll hier verwendet werden, da es sich bereits in Untersuchungen auf dem österreichischen Institut für Sportmedizin[1] als eines der gebräuchlichsten erwies. Es wird im Kapitel 2.1 detailliert beschrieben.

1 ÖISM; Auf der Schmelz 6, A-1150 Wien

1.2.2 Hardware - Vitronic VITUS smart 3D-Bodyscanner

Der VITUS smart 3D-Bodyscanner [11] der Firma Vitronic[2] wird für die Erstellung der 3D-Scans verwendet. Er besteht aus einem Podest, für die Platzierung des Probanden, und vier Säulen, die im Winkel von 90 Grad zueinander montiert sind. Jede der vier Säulen besitzt einen vertikal beweglichen Schlitten mit zwei Kameras und einem Laser, die gemeinsam mittels Lichtschnittverfahren[3] die Körperoberflächenkoordinaten speichern. Bei diesem Verfahren wird eine Laserlinie horizontal auf das Objekt geworfen, deren Verlauf aus einem bestimmten Winkel von den Kameras aufgenommen wird. Somit kann durch eine Bewegung der Schlitten in vertikaler Richtung die Objektkontur Scheibe für Scheibe erfasst werden. Dies resultiert in einer Punktewolke bestehend aus x-, y- und z-Koordinaten.

Technische Daten:

- Anzahl der Kameras: 8
- Aufnahmeprinzip: Lichtschnittverfahren, augensicher
- Gescanntes Volumen:
- 1000mm x 800mm (elliptische Grundfläche) x 2040mm (Höhe)
- Auflösung in X: ca.5mm
- Auflösung in Y: ca.5mm
- Auflösung in Z: ca.4mm
- Scangeschwindigkeit: variabel, typisch 11 Sekunden
- Abmessungen des Gesamtsystems: 2100mm x 1900mm x 2820mm
- Abmessungen der Scannersäulen: 80mm x 160mm x 2700mm

Er wird von der Herstellungsfirma Vitronic als der 3D-Ganzkörperscanner für Anwendungen von der Maßkonfektion bis zur Film-Animation bezeichnet [15]. Die Frage inwiefern er für biomechanische Anwendungen ausreichend ist, bleibt somit offen.

1.2.3 Software

Zusammen mit dem Scanner wird die passende Vitronic Software *WinSmart* und *ViViewer* [16] zur Verfügung gestellt, welche auf einem mit dem Scanner verbundenen Computer installiert wird. Durch WinSmart kann der Scanner gestartet und mit ViViewer die aufgenommenen Daten in einem passenden Format abgespeichert und verarbeitet werden (Näheres in Kapitel 3.1.1). Ein eigens für die Berechnung von Körperumfängen ausgelegtes Programm namens *BodyScan* [11], welches im Rahmen eines Projektpraktikums entwickelt wurde, und die dafür aufgenommenen 3D-Scans dienen als Grundlage dieser Diplomarbeit. Zur Auffindung der Meßstellen in BodyScan werden die vom Vitronic VITUS smart 3D-

2 3D-Scanner-Hersteller-Homepage: http://www.vitronic.de/vitus-3d-bodyscanner/vitus-3d-scanner/ (10.03.2011)

3 Höhenprofil-Ermittelung mittels projiziertem Laserstrahl und Kameraaufnahme der Laserreflexion vom Objekt; http://www.solabcon.de/lichtschnittverfahren-3d-vermessung.html (03.11.2010)

Bodyscanner aufgenommenen Daten während des Berechnungsvorganges in verschiedene Körperteile gegliedert. Somit liegt eine grobe Segmentierung bereits vor (Näheres dazu in Kapitel 2.3.3).

Es war vor Beginn der Diplomarbeit durchaus zu erwarten, dass in Weiterführung des vorhergehenden Projektpraktikums ein Großteil der erforderlichen Segmente ermittelt werden kann. Es war dabei nicht auszuschließen, dass zusätzliche Hilfe, wie Marker oder händische Markierungen am Scan, nötig sind. Aber auch wenn manuelle Unterstützung erforderlich sein sollte, konnte damit gerechnet werden, dass der Zeitaufwand zur Unterteilung des untersuchten Körpers in die 17 Segmente drastisch sinken würde.

1.3 Struktur

Die nachfolgende Arbeit beinhaltet eine Erläuterung der wichtigsten Funktionalmodelle in Kapitel 2. Das dieser Arbeit zugrundeliegende Hominoid von Hatze wird in Kapitel 2.2 beschrieben. Kapitel 2.3 beschäftigt sich mit verwandten Beispielen aus der Literatur, die den derzeitigen Stand der Technik erörtern.

In Kapitel 3 wird auf Vorarbeiten mit der vorhandenen Hardware eingegangen. Dies schließt ebenso die manuelle Datenerfassung zu Vergleichszwecken der (Hardware-) Scannerausgabe ein, wie die Probleme, die beim Scannen auftreten. Weiterführend werden Experimente zur Optimierung des erfassten Scans beschrieben.

Kapitel 4 erörtert die Algorithmen, die für das Programm dieser Diplomarbeit entwickelt wurden. Es werden die grundsätzlichen Möglichkeiten der Testbarkeit des Programmes vorgestellt sowie die Gründe erläutert, welche Vergleichsmöglichkeit schlussendlich gewählt wurde und warum andere verworfen wurden.

Die Implementierung der Software wird in Kapitel 5 beschrieben. Hier wird auch die Bedienung erklärt. Die Ergebnisse dieser Arbeit werden in Kapitel 6 diskutiert und in Kapitel 7 zusammengefasst.

Kapitel 8 beschäftigt sich mit weiterführenden Arbeiten, da während der Erstellung dieser Studie eine Reihe von Punkten gefunden wurden, deren Untersuchung den Rahmen dieser Diplomarbeit überschreiten würden. Dies vereinfacht, neben den darauffolgenden Anhängen, eine Weiterentwicklung des erstellten Segmentierungsprogrammes.

Die Anhänge beschreiben detailliert die erstellte Software (Modulbeschreibung), die verwendeten Dateiformate, sowie die Programmausgaben.

2 Funktionalmodelle und Segmentierungstechniken

Funktionalmodelle des Gliedersystems, wie sie nachfolgend beschrieben werden, können Fachleuten wie Medizinern, Sportwissenschaftlern oder Orthopäden computergestützte Methoden biomechanischer Bewegungsanalyse zugänglich machen und somit ihre Arbeit vereinfachen bzw. beschleunigen. Diese Modelle können, durch die Aufnahme definierter Messungen und Berechnungen, Aufschluss über die motorischen Fähigkeiten geben oder pathologische Veränderungen aufzeigen [6]. Ein solches anthropomorphes, mathematisch-geometrisches Modell des segmentierten menschlichen Körpers wird als Hominoid bezeichnet. Sie können genauso zu Tier- oder Roboter-Modellen modifiziert werden.

Der Vorteil von Funktionalmodellen gegenüber traditionellen Vorgehensweisen aus Sicht der Biomechanik liegt darin, dass sich bestimmte Charakteristika (z.B. Bewegungsumfang des linken Arms) einer Analyse zuführen lassen, ohne invasive Methoden (wie beispielsweise Belastungsmethoden) anwenden zu müssen. Stellt man nun die Analyse eines idealen Bewegungsablaufes dem realen Ablauf eines Probanden gegenüber, lassen sich z.B. Aussagen über pathologische Veränderungen (z.B. Bewegungseinschränkungen auf Grund von Muskelzerrungen oder schlecht verheilter Knochenbrüche) machen. Geeignete Modelle erlauben auch die Inversion der Bewegungsanalyse, d.h. sie ermöglichen die Simulation von Bewegungen. Sie lassen sich z.B. für die effiziente Entwicklung von Prothesen oder dem Design von Arbeitsumgebungen einsetzen. Hier können ohne aufwändige Versuche bereits beim Design am Computer Simulationen durchgeführt werden, die spätere Probleme aufzeigen und damit vermeiden lassen.

Im Folgenden werden einige wichtige gebräuchliche Funktionalmodelle und ihre Anwendungsgebiete beschrieben.

2.1 Wichtige Funktionalmodelle

Für Ganganalysen und deren Anwendung auf zweibeinige Roboter entwickelten Vubrokatovic und Juricic 1969 ein vier-segmentiges Modell [18]. Da jedoch ein Modell mit starren Beinen für diese Analysen sehr bald an seine Grenzen stößt, wurden mehr-segmentige Modelle entwickelt. Eines davon ist das fünf-segmentige Modell von Hemami und Farnsworth, welches beide Oberschenkel bzw. Unterschenkel mit Füßen und den Rumpf beinhaltet [7]. Für die effiziente Analyse komplexerer Bewegungsabläufe wurde bereits 1964 ein Modell mit 15 Segmenten als unterste Grenze angesehen [3]. Dieses Hanavan-Modell besteht aus geometrischen Körpern wie Kegelstümpfen oder Zylindern mit homogener Dichte. Jensen verbesserte dieses Modell und erweiterte es auf 16 Segmente aus elliptischen Scheiben, in welchem er Hals und Kopf ebenfalls separierte [8].

Die angeführten Modelle unterstellen einen symmetrischen Aufbau der Segmente und modellieren sie mit gleichförmiger Dichte. Außerdem berücksichtigt keines die Schultern als eigenständige Segmente. Hatze vermied diese Nachteile und

entwickelte ein 17-segmentiges Modell [6], wie im folgenden Kapitel 2.2 näher erläutert. Es erwies sich laut Untersuchungen [6] der hier angeführten Modelle als eines der genauesten Funktionalmodelle.

2.2 Das Hominoid von Hatze

Das Modell besteht aus 17 Segmenten, welche wie folgt nummeriert und bezeichnet sind:

- 1 Abdomino-thorakales Segment
- 2 Kopf-Nacken-Segment
- 3 Linkes Schultersegment
- 4 Linker Oberarm
- 5 Linker Unterarm
- 6 Linke Hand
- 7 Rechtes Schultersegment
- 8 Rechter Oberarm
- 9 Rechter Unterarm
- 10 Rechte Hand
- 11 Abdomino-pelvisches Segment
- 12 Linker Oberschenkel
- 13 Linker Unterschenkel
- 14 Linker Fuß
- 15 Rechter Oberschenkel
- 16 Rechter Unterschenkel
- 17 Rechter Fuß

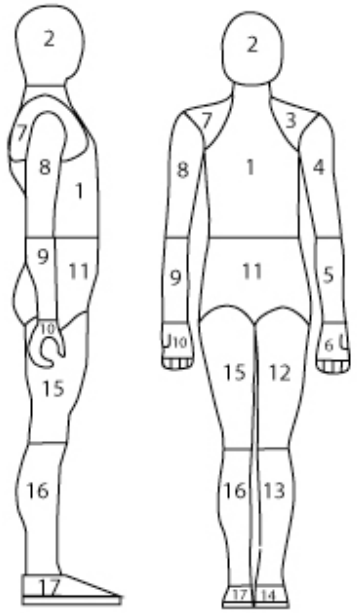


Abbildung 1 17 Segmente Hominoid [2]

Vorteile des Hatze Modells gegenüber anderen Modellen:

- Die Schultern werden als eigene Segmente modelliert.
- Jedes Segment wird in kleinere geometrische Strukturen zerlegt, sodass die Form und die Dichteverteilungen innerhalb des Segmentes detaillierter modelliert werden können.
- Es gibt keine Annahmen bezüglich Symmetrie.
- Das Modell differenziert zwischen Männern und Frauen.
- Die Dichte bestimmter Segmente wird mit Hilfe eines Fett-Parameters angepasst.
- Schwangerschaft und Dickleibigkeit werden speziell berücksichtigt.

Hatze [4] stellt ein Modell (Abb. 2) zur Berechnung von Parameterwerten anthropomorphischer Segmente vor. Die dazu notwendigen 242 Messwerte sind in Abb. 3 aufgelistet. Die Abweichung von den realen Werten ist im Allgemeinen kleiner als 3% und maximal 5%.

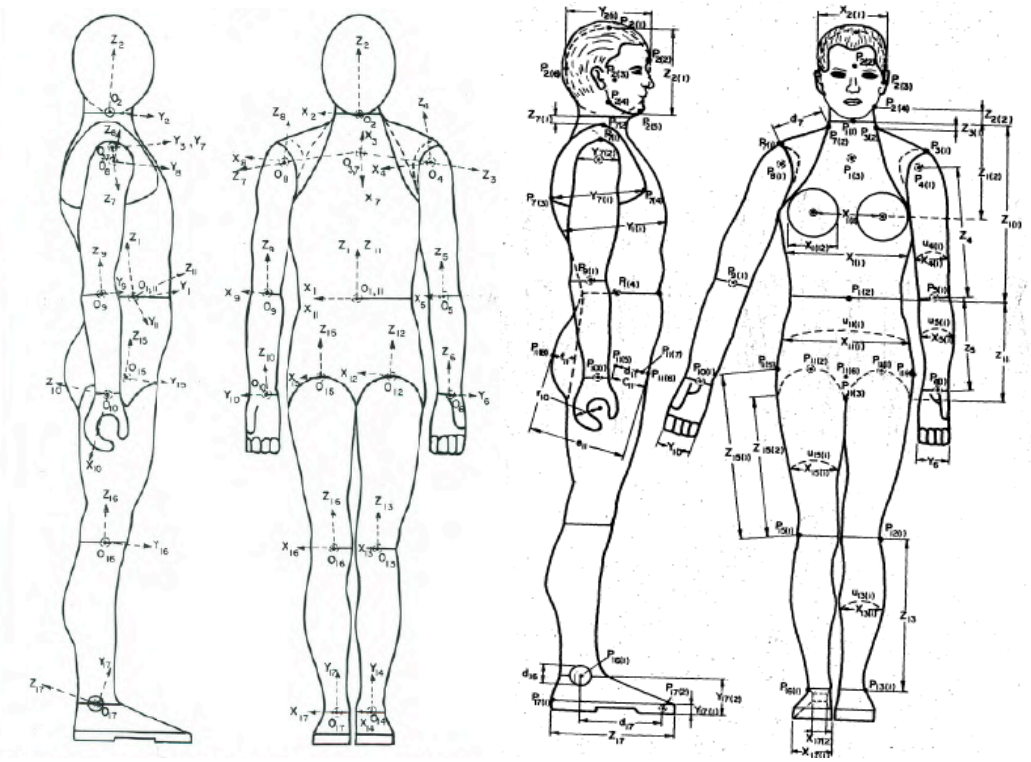


Abbildung 2 Lateral- und Frontalansicht des Hominoids von Hatze
rechts: mit den Konfigurationskoordinaten für die Analyse dreidimensionaler Bewegungen [4]
links: mit den 242 Messwerten [4]

Die Abfolge der Segmente im Aufnahmezettel (Abb. 3), die von oben nach unten gemessen werden, entspricht nicht der Reihenfolge der Nummerierungen nach Hatze (Abb. 1).

Die einzelnen Daten werden in X-Werte, Y-Werte, U-Werte, D-Werte sowie in Z-Werte gegliedert.

X-Werte beschreiben die Breitenabstände des Probanden, die Y-Werte die Tiefenabstände, die U-Werte die Umfänge der einzelnen Segmente, die D-Werte die Höhe (zum Beispiel des Knöchels) und die Z-Werte die Länge der Segmente.

Dafür sind die Segmente in zehn Messhöhen unterteilt. Bei den Händen und im Bauch- und Beckenbereich müssen andere bzw. weitere Werte berücksichtigt werden. Darum wurden noch die Wertebezeichnungen R, c11, d11, e11, f11 und XH11 eingeführt.

Die R-Werte geben jeweils den Radius des Kreises an, den Daumen und Zeigefinger bilden, wenn sich deren Fingerspitzen berühren.

Der Abstand XH11 wird mit dem Tasterzirkel gemessen, welcher mit den Enden an den Oberschenkelknochen angesetzt wird.

c11 ist der Abstand vom Punkt P11(6) zum Hüftgelenkspunkt P11(5). d11 ist der Abstand von P11(7) zu P11(5) (Abb. 2). e11 und f11 geben Auskunft über den Hüftumfang [2].

SUBJECT: P.K. SEX: M AGE: 21 MASS (kg): 77.3 HEIGHT (mm): 1860 DATE: 10 February 2011
REMARKS (Somatotype, etc): Rower

SEGMENT

Abdomino-thoractic	X1	X5	X6	X7	X8	X9	X10	X11	X12	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10	Z1	Z2
	127	290	298	291	273	263	285	0	0	113	137	200	221	236	226	229	231	212	199	469	0
Kopf-Nacken	X1	Y1	Z1	Z2																	
	130	196	207	44																	
linke Schulter	D3	Y1	Y2	Z1																	
	139	228	129	36																	
rechte Schulter	D7	Y1	Y2	Z1																	
	138	228	127	37																	
linker Oberarm	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	Z4
	110	106	100	98	93	83	80	82	82	91	410	369	343	329	315	298	282	274	263	252	322
rechter Oberarm	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	Z8
	111	109	103	94	86	78	79	81	84	94	425	380	345	321	307	295	282	278	257	251	322
linker Unterarm	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	Z5
	107	104	100	97	89	78	70	66	65	64	275	283	287	280	264	235	207	190	175	173	293
rechter Unterarm	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	Z9
	103	105	104	100	92	80	72	70	65	65	275	283	287	280	264	235	207	190	175	173	293
linke Hand	R6	Y6																			
	54	90																			
rechte Hand	R10	Y10																			
	54	90																			
Abdomino-pelvic	X1	X2	X3	X4	X5	X6	X7	X8	XM11	Z11	U1	U2	U3	U4	U5	U6	U7	C11	D11	E11	F11
	296	295	309	330	341	346	347	355	271	252	782	788	822	857	911	940	950	98	86	208	46
linker Oberschenkel	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	Z1	Z2
	172	177	171	163	151	141	129	120	112	113	550	537	536	528	500	470	435	412	390	384	475	383
rechter Oberschenkel	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	Z1	Z2
	175	178	174	167	156	147	135	127	120	116	561	550	543	530	513	481	443	413	378	380	475	380
linker Unterschenkel	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	Z13	D13
	106	99	110	125	119	101	88	73	60	71	343	344	383	405	380	328	287	248	237	260	448	30
rechter Unterschenkel	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	Z16	D16
	106	106	116	128	120	103	88	76	62	70	343	346	381	402	382	331	292	261	235	252	448	31
linker Fuß	X1	X2	Y2	Y2	Z14	D14															
	101	63	31	71	240	165															
rechter Fuß	X1	X2	Y2	Y2	Z17	D17															
	101	63	31	71	240	165															

Abbildung 3 Beispiel-Aufnahmezettel der 242 Messwerte [2]

Tabelle 1 Segmentparameter [6]

Segmentvolumen

Segmentmasse

3 Hauptträgheitsmomente bzgl. der Hauptträgheitsachsen durch den Segmentschwerpunkt

3 Segmentschwerpunktskoordinaten

Neigungswinkel der z-Hauptträgheitsachse relativ zur segmentären z-Achse

3 Ursprungskoordinaten relativ zum lokalen Koordinatensystem des vorhergehenden Segments

Um nun eine personalisierte Bewegungsanalyse mit Hilfe des Hominoids erstellen zu können, werden für die Bewegungsgleichungen die Werte bestimmter Segmentparameter[4] gebraucht [6]. Diese sind in Tabelle 1 angeführt.

In Hatzes Arbeit aus dem Jahre 1979 [4] wird beschrieben, wofür das Modell gebraucht wird. Es geht dabei hauptsächlich um Computersimulationen menschlicher Bewegungsabläufe. Die Modellierung der Segmente (Arme, Beine etc.) in einfache geometrische Formen (Ellipsoide, Zylinder etc.) wird diskutiert und die Formeln der geometrischen Körper, aus denen das Segment zusammengesetzt ist und wie es berechnet wird, erläutert. Die Funktionalmodelle, wie unter 2.1. beschrieben, werden vorgestellt und es wird darauf hingewiesen, dass in keinem dieser Modelle die Schultern separat modelliert werden.

Aus Vereinfachungen in den diversen Modellen ergeben sich Fehler, wenn es um die Simulation von Bewegungsabläufen geht. Diese Fehler und Ungenauigkeiten sind aber schwer nachzuweisen, weil die Messungen in der Realität sehr schwer durchzuführen sind. Weitere Fehler ergeben sich durch Symmetrieannahmen oder durch fehlende Geschlechtsunterscheidung. Auch im Hominoid von Hatze können Fehler durch Vereinfachungen nicht zur Gänze vermieden werden. Eine Vereinfachung ist z.B. die Annahme von rigiden Segmenten. D.h. die Verteilung von Fett, Muskeln, Blut etc. ändert sich nicht. Vergleiche zwischen den vom Modell berechneten und den (real) gemessenen Werten zeigen Abweichungen von maximal 5% [4]. Diese Fehler werden aber akzeptiert, weil das Modell dadurch einfacher wird. Eine weitere Vereinfachung ist die Annahme fixer und scharfer Grenzen der Segmente. Es wird besprochen, wie die Abgrenzung stattfindet - nämlich einfach durch die sichtbare Form der Segmente (d.h. die Grenzen bilden sichtbare Parameter wie kleinster Umfang etc.). Diese Annahme kann jedoch auch zu Fehleinschätzungen führen. In Hatzes Arbeit aus dem Jahre 1980 [5] führt er aus, um wie viel Prozent sein Modell falsch rechnet und unter welchen Umständen diese maximalen Fehler auftreten. Er misst dafür vier, von der Statur und dem Alter unterschiedliche, Personen und vergleicht die Abweichungen:

- der einzelnen Segment-Volumen
- der Verhältnisse der Segmentlängen zu der Höhe des Körperschwerpunktes
- der Trägheitsmomente

Die maximalen absoluten Abweichungen sind laut seiner tabellarischen Auflistung (Abb. 4):

- 5,17% beim Volumen (v) des abdomio-pelvischen Segments einer Testperson, die eine Badehose getragen hat
- 10,9% beim Körpermittelpunkt-zu-Segment-Verhältnis (r) beim rechten Oberschenkel einer männlichen Testperson, welche bei dem gegebenen Modell (Abb. 2) bei Männern immer wiederkehrt
- 5,03% beim Trägheitsmoment (i) des linken Oberschenkels, welche sich durch die Unfähigkeit des Entspannens des linken Hüftmuskels bei der Bewegung erklärt.

4 In dieser Diplomarbeit wird für die Namen der Segmentparameter einfach nur der Begriff „Segmentparameter" gebraucht, für deren Ausprägung der Begriff „Parameterwert".

Die Umstände der Fehleinschätzungen liegen somit immer im Bereich der Datenaufnahme, welche durch vorhergehende genaue Definitionen der Erfassung teilweise kontrolliert werden können. Das Fazit aus [6] ist, dass auf Grund seiner Genauigkeit, Vielseitigkeit und leichten Umsetzung das Modell ein geeignetes Mittel bietet, die Parameterwerte der Segmente zu berechnen, welche sonst nur experimentell durch relativ arbeitsintensive Methoden gewonnen werden können. Beispielsweise können die Volumina nur mittels Eintauchen des Körpers in Wasser und Messen des dadurch verdrängten Wasservolumens bestimmt werden.

Table 1. Comparison of computed and experimentally determined segmental parameter values (in the m · kg · s system, volumes in litre)

	Subject F.B. (male, 23)						Subject C.P. (male, 26)					
Segment	V	v	R	r	$\bar{I}_x^p$	i_x	V	v	R	r	$\bar{I}_x^p$	i_x
Abdomino-thoracic	19.111	−0.85	0.439	—	0.3117	—	19.803	2.83	0.444	—	0.3302	—
Head-neck	4.475	4.99	0.517	8.75	0.0303	—	4.537	4.08	0.516	8.91	0.0337	—
Left shoulder	1.438	—	0.727	—	0.0047	—	2.042	—	0.706	—	0.0080	—
Right shoulder	1.890	—	0.711	—	0.0071	—	2.121	—	0.699	—	0.0084	—
Left arm	2.110	−3.49	0.432	0.84	0.0196	—	2.123	−0.14	0.437	−0.30	0.0203	—
Right arm	2.021	−2.07	0.437	−0.14	0.0168	—	2.340	−0.86	0.428	1.81	0.0229	—
Left forearm	1.023	3.49	0.417	2.96	0.0067	—	1.223	4.45	0.413	4.05	0.0086	—
Right forearm	1.190	2.46	0.404	6.05	0.0079	—	1.313	2.74	0.412	4.10	0.0093	—
Left hand	0.453	−5.15	0.515	−1.7	0.0011	—	0.416	0.95	0.533	−5.37	0.0010	—
Right hand	0.446	0.89	0.531	−4.96	0.0011	—	0.417	0.71	0.524	−3.65	0.0010	—
Abdomino-pelvic	8.543	3.03	0.368	—	0.0399	—	9.614	5.17	0.395	—	0.0541	—
Left thigh	8.258	2.04	0.479	−10.6	0.1475	3.14	8.744	4.88	0.473	−9.23	0.1653	5.03
Right thigh	8.278	4.52	0.480	−10.9	0.1415	−4.81	8.729	3.26	0.466	−7.78	0.1702	1.44
Left leg	3.628	−1.06	0.412	4.87	0.0615	0.43	3.856	−4.19	0.420	3.03	0.0798	2.98
Right leg	3.686	−4.72	0.417	9.63	0.0663	1.22	3.798	−0.98	0.417	3.69	0.0747	2.63
Left foot	0.887	3.59	—	2.24	0.0041	—	1.032	0.67	—	4.77	0.0051	—
Right foot	0.923	3.85	—	6.98	0.0042	—	1.055	−2.53	—	0.55	0.0051	—

Table 2. Comparison of computed and experimentally determined segmental parameter values (in the m · kg · s system, volumes in litre)

	Subject R.M. (female, 31)						Subject G. R. (male, 12)					
Segment	V	v	R	r	$\bar{I}_x^p$	i_x	V	v	R	r	$\bar{I}_x^p$	i_x
Abdomino-thoracic	14.537	−1.39	0.429	—	0.2064	—	8.384	2.05	0.422	—	0.0832	—
Head-neck	3.595	3.87	0.542	5.58	0.0209	—	2.774	4.41	0.536	5.52	0.0141	—
Left shoulder	1.110	—	0.722	—	0.0027	—	0.778	—	0.696	—	0.0016	—
Right shoulder	1.146	—	0.709	—	0.0029	—	0.782	—	0.706	—	0.0016	—
Left arm	1.616	1.71	0.445	2.06	0.0137	—	0.748	−2.00	0.442	−1.49	0.0039	—
Right arm	1.505	0.46	0.443	1.61	0.0119	—	0.757	0.37	0.438	0.54	0.0038	—
Left forearm	0.835	4.33	0.429	0.23	0.0046	—	0.511	3.53	0.425	1.06	0.0024	—
Right forearm	0.809	−1.61	0.436	−1.39	0.0048	—	0.516	2.94	0.426	0.93	0.0023	—
Left hand	0.285	−0.11	0.498	1.58	0.0005	—	0.209	−3.48	0.525	−3.86	0.0003	—
Right hand	0.288	0.43	0.499	1.38	0.0005	—	0.204	−2.54	0.526	−3.94	0.0003	—
Abdomino-pelvic	11.208	4.08	0.438	—	0.0689	—	4.183	3.12	0.415	—	0.0131	—
Left thigh	9.166	2.32	0.441	−1.85	0.1563	4.18	3.260	3.59	0.469	−8.32	0.0335	2.87
Right thigh	8.955	3.09	0.436	−0.69	0.1492	4.03	3.338	4.18	0.461	−6.49	0.0361	−3.14
Left leg	3.310	0.12	0.429	0.92	0.0453	1.18	1.641	−0.45	0.430	0.74	0.0180	−0.88
Right leg	3.487	−1.04	0.437	−0.92	0.0505	2.46	1.634	0.33	0.433	0.00	0.0173	−1.01
Left foot	0.842	1.13	—	0.41	0.0036	—	0.566	1.43	—	1.04	0.0019	—
Right foot	0.887	2.22	—	4.01	0.0027	—	0.544	−0.87	—	2.08	0.0018	—

Abbildung 4 Vergleich von berechneten und experimentellen Segmentparameterermittlungen [5]

2.3 Verwandte Arbeiten

Die Grundlage der Berechnung der in Tabelle 1 angeführten Parameterwerte sind, wie oben beschrieben, die 242 händisch aufgenommenen Messwerte. Da die Prozedur der Messwertaufnahme zeitaufwändig und fehleranfällig ist, soll in dieser Diplomarbeit ein Weg gefunden werden, aus einem 3D-Körper-Scan die beschriebenen Parameter direkt und schneller zu berechnen. Dazu wurde recherchiert, welche Methoden vor dieser Diplomarbeit in der Literatur beschrieben werden. Aus den gefundenen Arbeiten werden hier jene näher vorgestellt, die die höchste Relevanz für diese Diplomarbeit haben. Gesucht wurden Arbeiten, die einerseits die Segmentierung von [4] anstreben (Kap. 2.3.1) und sich andererseits mit jeglicher Segmentierung von 3D-Scans und der Erkennung von Merkmalen am Körper ohne Zuhilfenahme von Markern beschäftigen.

2.3.1 Automatische Segmentierung nach Hatze aus Videobildern mit Hilfe von Markern

Eine mögliche Herangehensweise zur automatischen Segmentierung wird in „Precise Determination of Anthropometric Dimensions" von Baca [1] vorgestellt. Hier wird die Segmentierung eines Körpers auf Grund von Videobildern mit Hilfe von Markierungsbändern beschrieben. Eine Person nimmt vor einer Kamera vier genau definierte Positionen (aufrecht, links, rechts und mit vorgebeugtem Oberkörper) ein. Zusätzlich wird noch ein Bild eines Referenzrahmens gemacht, wenn sich die Aufnahmebedingungen (Abstand zur Kamera, Aufnahmewinkel etc.) ändern. Mit Hilfe dieser Referenz lassen sich die Abmessungen des Körpers auf Grund der Pixelanzahl und durch Subpixel-Interpolation sehr genau bestimmen.

Als Unterstützung zur Segmenterkennung werden bestimmte Grenzstellen (Ober-/Unterarm, Schultern etc.) mit schwarzen Bändern markiert. Die Ergebnisse (berechnet im eigens dafür programmierten Programm VIDANT) werden dann zur Berechnung der Parameterwerte in das Programm VISEPA übergeleitet.

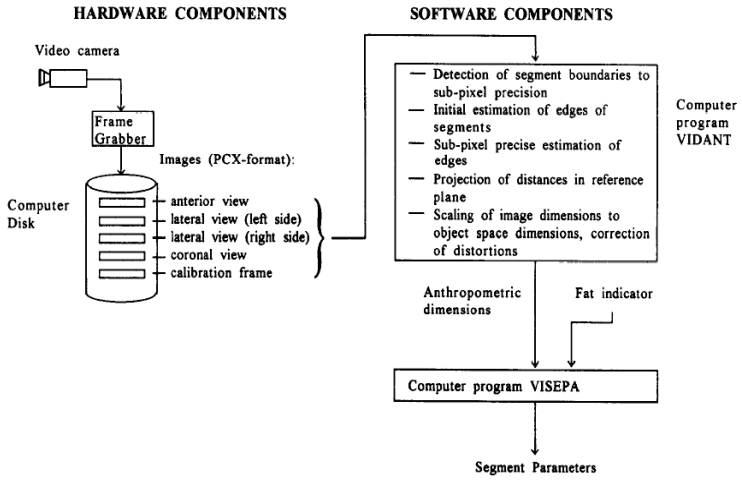


Abbildung 5 Systemübersicht für Parameterwerte-Berechnung am österreichischen Institut für Sportmedizin entwickelt [1]

VISEPA ist, wie das von Hatze erstellte Berechnungsprogramm ANSEPA [4], eine für die vorliegende Segmentierung aus Videobildern [1] modifizierte Software zur Berechnung der Werte der Segmentparameter (Tab. 1). Diese benötigt jedoch nur 220 der 242 anthropometrischen Daten. Die restlichen 22 Daten, die nicht direkt aus den Videobildern berechnet werden können, werden mit Regressionsalgorithmen abgeleitet.

Bei der Berechnung der Parameterwerte mit Hilfe dieser Programme zeigen sich im Bereich der Trägheitsmomentsberechnungen (1) die größten Abweichungen gegenüber der Berechnung händisch vermessener Körperparameter (Abb. 6). Denn bei kleinen Segmenten wirkt sich das Trägheitsmoment (1) sehr sensitiv aus, da die Abstände quadratisch in die Berechnung mit eingehen. Das Trägheitsmoment J ergibt sich aus (1):

$$J = \sum_i m_i r^2_i \qquad (1)$$

wobei m_i der Masse und r_i dem senkrechten Abstand des *i*-ten Pixelpunktes auf der Drehachse entspricht. Minimale Abweichungen haben somit relativ starken Einfluss. Bei den Berechnungen der Länge, des maximalen Durchmessers, der Masse, des Volumens und der z-Koordinate des Masseschwerpunkts liegt der Fehlerprozentsatz im mittleren um die 5 % (Abb. 6).

Segment	Length (%)	Maximum diameter (%)	Mass (%)	Volume (%)	Z-coordinate of mass centroid (mm)	Principal moments of inertia (%)		
Abdomino-thoracic	1.9	2.4	4.8	4.9	5.0	3.1	4.0	8.9
Head	2.2	4.4	3.2	3.0	8.0	6.6	8.2	7.7
Left thigh	2.4	3.9	3.4	3.5	6.0	2.4	5.6	9.8
Left leg	2.0	4.4	6.5	6.7	6.0	7.8	7.6	11.4
Left forearm	1.9	4.6	7.9	7.5	2.0	7.1	7.3	13.7

Abbildung 6 Mittlere Abweichungen video-basierter Segmentberechnung nach [1]

2.3.2 Manuelle Segmentierung aus einem 3D-Scan ohne Marker

Siebert und Ju gehen in ihrer Arbeit über Animationen in Zusammenhang mit Computerspielen [9] auch auf das Problem der Segmentunterteilung ein. Die Arbeit zeigt einen Weg auf, wie Scandaten, die von dem 3D-Scanner Cyberware[5] geliefert werden, bearbeitet und auf ein Modell in der Software Poser4[6] abgebildet werden können. Sowohl die Scandaten als auch das Modell sind sehr detailliert und präzise. Die Kombination der Outputdaten von Cyberware und Poser4 liefert sehr realistische computeranimierte Bewegungen.

Der Schwerpunkt dieser Arbeit liegt hier auf Simulationen und Spielen und weniger auf technisch-mathematischen Berechnungen. Es werden mehrere Konzepte zur Modellierung eines Körpers vorgestellt (polygonal surface model, joint-dependent local deformation operators etc.). Es stellt sich auch hier das Problem der Segmentierung und Zuordnung der Scandaten zu den Gliedmaßen im Poser4-Modell.

Die Autoren segmentieren die Scandaten mit Hilfe eines selbst geschriebenen Java[7]-Programms. Dazu benötigen sie aber interaktiven Eingriff. Die Software ordnet die Segmente also nicht automatisch zu. Es wird allerdings eine zukünftige Möglichkeit diskutiert, auf Grund bestimmter Orientierungspunkte die Segmentierung automatisch durchzuführen.

5 http://www.cyberware.com/products/scanners/wbx.html (01.03.2011)
6 http://poser.smithmicro.com/ (01.03.2011)
7 objektorientierte Programiersprache; http://www.java.com/en/ (01.03.2011)

2.3.3 Automatische Umfangsberechnung aus einem 3D-Scan ohne Marker

Dem österreichischen Institut für Sportmedizin wurde von der Firma Vitronic ein VITUS 3D-Bodyscanner zur Verfügung gestellt. Dieser besitzt vier Laser, einen an jeder Ecke, die waagrecht in den Raum strahlen. Vom Körper wird eine Linie zurückgeworfen, die mit insgesamt 8 Kameras aufgenommen wird. Aus dem Verlauf dieser Linie kann die Vitronic Software den Körper rekonstruieren.

„BodyScan" [11] nennt sich die Software, die für die Vergleichsmöglichkeit der Bodyscanner-Daten gegenüber händischen Messungen vor der Diplomarbeit als Projektarbeit entworfen wurde. Sie errechnet die Körpergröße, den Taillen-, Hüft-, Bein-, Arm- und Bauchumfang aus den 3D-Daten. Die 3D-Werte werden nach der Positionierungsvorschrift von Lohman, Roche und Martorell [12] aufgenommen und die Umfänge nach deren Messvorschrift berechnet. Diese Vorschriften können zur Segmentierung der gefragten 17 Bereiche nur bedingt verwendet werden. Für die Berechnung der Umfänge muss BodyScan jedoch eine grobe Segmentierung vornehmen, auf der diese Diplomarbeit aufbauen konnte. Ein wesentlicher Unterschied zu der Teilung des Hominoids von Hatze ist z. B. die Trennung des Oberarmsegmentes an der Schulter (Abb. 7).

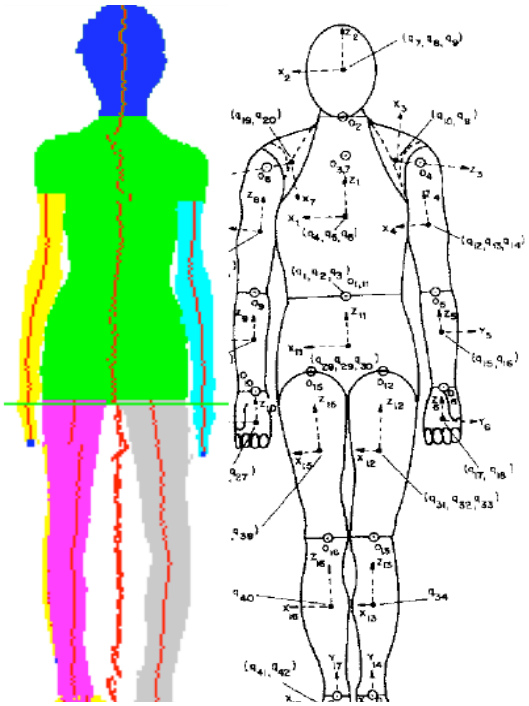


Abbildung 7 Segmentierung nach BodyScan [11] links gegenüber der des Hominoids von Hatze [4] rechts

Es liegt somit als Output der vorhandenen Software die grobe Abschätzung von sechs der 17 Segmente vor: der Rumpf, der Kopf, die zwei Arme und die zwei Beine. Diese Diplomarbeit soll darauf aufbauend die Segmentierung nach Hatze vornehmen. Die bereits erkannten Segmente müssen weiter unterteilt bzw. im Becken- und Schulterbereich verbessert werden.

2.3.4 Automatische Segmentierung aus einem 3D-Scan mit Hilfe vorhandener Daten

Lu und Wang [10] haben im Jahre 2000 eine Normierungstabelle erstellt, die die experimentelle Messwertfindung von Testergebnissen unterstützen kann. In ihrer Arbeit wird ein Programm beschrieben, das mit Hilfe dieser Normierungstabelle auf Grund eines 3D-Scans Orientierungspunkte eines Körpers automatisch erkennt. Mit Hilfe verschiedener Algorithmen (Silhouettenanalyse, Umfangsmessung, Grauwerterkennung und Konturverfolgung) werden 12 Punkte und 3 Ebenen (Abb. 8) erkannt.

Es wird versucht, die Segmentierung eines Körpers ohne Marker am Körper befestigen zu müssen durchzuführen. Denn die Setzung von Markern per Hand lässt auch bei Vorliegen genauer Markierungspositionen zuviel Willkür und würde somit eine zusätzliche Fehlerquelle bedeuten.

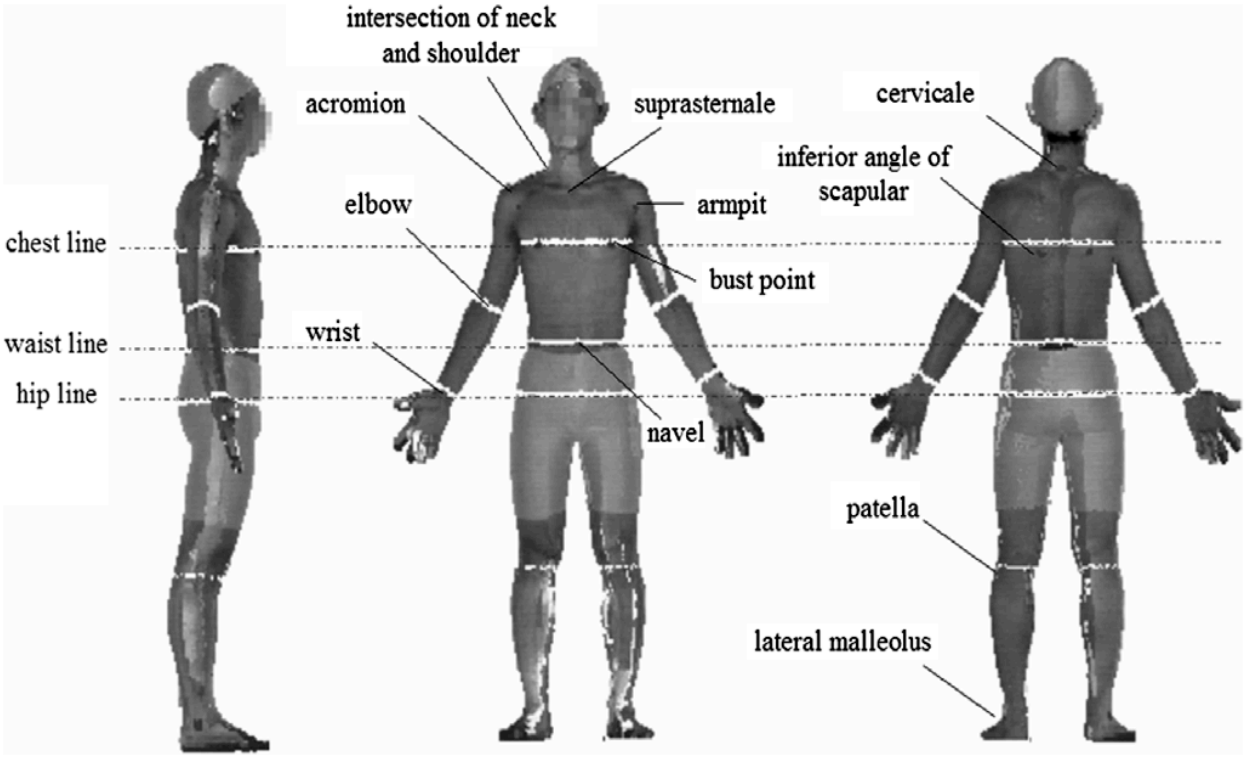


Abbildung 8 Die 12 Orientierungspunkte und drei Bezugslinien [10]

Der wesentliche Unterschied zu anderen vergleichbaren Programmen: für die Erkennung der Merkmale wird eine Datenbank mit statistischen Werten der (asiatischen) Bevölkerung herangezogen. Diese Datenbank, die von Lu und Wang im Jahre 2000 erstellt wurde, enthält Angaben, wo sich bestimmte Punkte befinden. Ein Beispiel: der Nabel in Relation zur Größe einer Person befindet sich zwischen 58,40 und 58,81% der Körpergröße. Damit lässt sich die Suche nach einem Merkmal auf ein kleines Gebiet eingrenzen und die Verarbeitungsgeschwindigkeit erhöhen.

Das System wurde von den Autoren getestet, indem 189 Personen im Alter zwischen 18 und 30 Jahren fünf Mal per Hand vermessen und fünf Mal gescannt

wurden. Es zeigt sich, dass alle Merkmale zuverlässig erkannt wurden und die Reproduzierbarkeit sehr hoch ist. Allerdings wird von den Autoren die Genauigkeit des Scanners kritisiert und es wird in Frage gestellt, ob der von ihnen benutzte Vitronic VITUS 3-D 1600 Scanner für anthropometrische Messungen geeignet ist.

2.3.5 Schlussfolgerungen aus den untersuchten Arbeiten

- Das Studium der verwandten Arbetien zeigte, dass bis heute kein Programm bekannt ist, welches die Segmentierung nach Hatze [4] aus aufgenommenen Daten automatisch und ohne Marker durchführt. Es gibt zwar Ansätze [1] [9] [10] [11], die in die Nähe kommen, dennoch fehlt die durchgehende automatische Verarbeitung, welche in der vorliegenden Arbeit angestrebt wird.

- Die Untersuchungen von [10] deuten darauf hin, dass die Genauigkeit eines Vitronic VITUS 3D-Bodyscanners wissenschaftlichen Ansprüchen nicht genügt. Bei wissenschaftlichen Auswertungen müssen die Ergebnisse innerhalb allgemein akzeptierter Toleranzen liegen, die laut [4] [1] Abweichungen von 3% von den realen Werten erlauben. Die Genauigkeit alleine ist jedoch nicht das einzige Kriterium. Die Scans und die Ergebnisse müssen auch reproduzierbar sein. Wenn ein Körper mehrere Male gescannt wird, müssen die Ergebnisse natürlich jedes Mal gleich sein, d.h. jedes Mal innerhalb der o.a. Toleranz liegen.

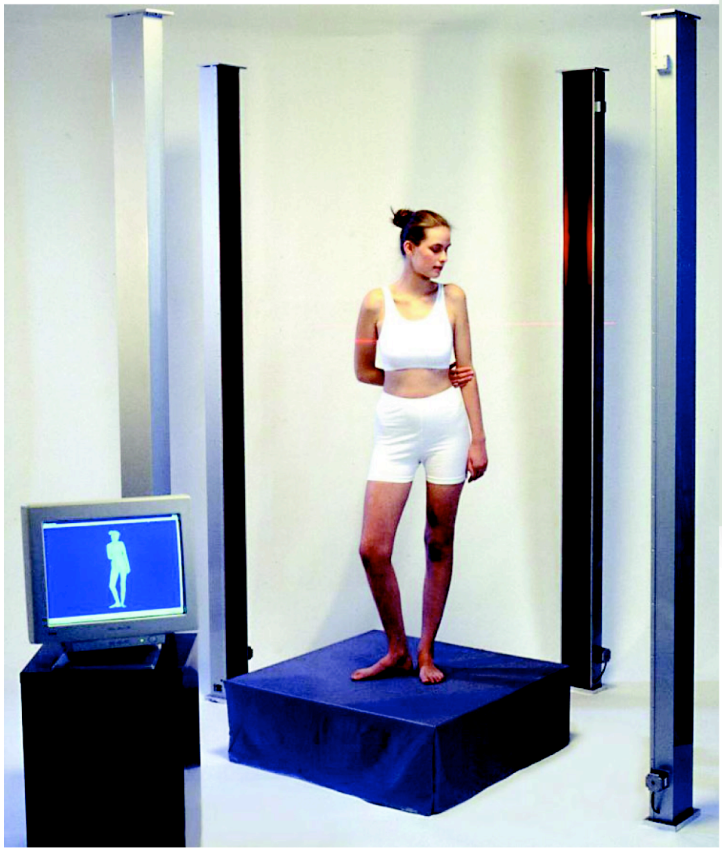
Es gilt daher eine Methode zu entwickeln, die

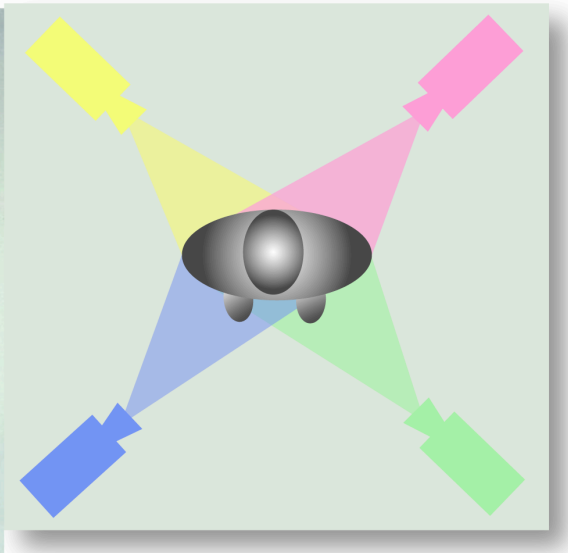
– einen 3D-Körperscan mit dem zur Verfügung stehenden Vitronic VITUS smart 3D-Bodyscanner automatisch und ohne Marker in die 17 Segmente unterteilen kann.

– zumindest einen mechanischen Parameterwert oder einen Maßwert jedes Segmentes mit einem Fehler kleiner 3% berechnen kann, damit eine Verifikation mit anderweitig errechneten Werten durchgeführt werden kann.

3 Vorarbeiten

3.1 Hardware





Scanprinzip

Vitus smart

Abbildung 9 Vitronic VITUS smart 3D-Bodyscanner [15]

Der bereits anfänglich (technische Daten siehe Kap. 1.2.2) beschriebene VITUS smart 3D-Bodyscanner der Firma Vitronic (Abb. 9) besitzt eine Auflösung von etwa 4mm in vertikaler und etwa 5mm in horizontaler Richtung [15].

3.1.1 Scanvorgang

Die zu scannende Person stellt sich in die Mitte des 3D-Scanners. Sie muss sich in starrer Haltung nach der Vorschrift von Lohman, Roche und Martorell [12] positionieren, die wie folgt definiert ist: Der Proband soll aufrecht stehen (Kopf hoch, Schultern zurück, Brust heraus – keine Buckelhaltung), ohne die aufrechte Haltung zu übertreiben bzw. sich nicht künstlich strecken. Die Arme sollen seitlich locker herunterhängen, Handflächen gestreckt, Finger zusammen, Handflächen zum Körper zeigend. Zwischen Körper und Armen soll ein Abstand von mehr als 1 cm bestehen - ohne dass er erzwungen wird. Die Füße soll 10 cm voneinander entfernt

sein, die Beine parallel. Idealerweise gibt es im Schritt auch einen Abstand von zumindest 1 cm (was normalerweise durch Badehose oder Bikini automatisch erzeugt wird). Die Person soll eine Badehaube sowie Badehose, Bikini oder ähnliches tragen. Diese Kleidungsstücke sollen aus nicht-reflektierendem, hellem Material bestehen und eng am Körper anliegen (weite Boxershorts ergeben z.B. nicht auswertbare Scans). Wenn keine Badehaube zur Verfügung steht, müssen längere Haare unbedingt so am Hinterkopf zusammengebunden werden, dass der Halsbereich frei bleibt und gut sichtbar ist.

Der Scanner wird über den angeschlossenen Computer mit der dazugehörigen Vitronic Software WinSmart [16] gestartet. Die an den vier Ecksäulen angebrachten Laser (Abb. 9) und Kameras werden nun von oben nach unten bewegt. Bei diesem Verfahren wird pro Ecksäule jeweils eine Laserlinie horizontal auf das Objekt geworfen, deren Verlauf aus einem bestimmten Winkel von den Kameras aufgenommen wird. Somit kann durch die Laserbewegung in z-Richtung die Objektkontur Scheibe für Scheibe erfasst werden. Dies resultiert in einer Punktwolke bestehend aus X-, Y- und Z-Koordinaten. Innerhalb weniger Sekunden haben die Laser die Person abgetastet, fahren wieder nach oben in die Ruheposition und der ganze Vorgang ist abgeschlossen. Die erfassten Daten werden dann am angeschlossenen PC von der Software ViViewer [16] grafisch dargestellt. Bereits im Steuerungsprogramm für den Scanner können Datenbereinigung, Messungen etc. durchgeführt und die Daten dann exportiert werden.

Die ViViewer-Datenbereinigung wurde bei der Erstellung der Körperumfangsberechnungs-Software BodyScan [11] genauer betrachtet. Der Benutzer kann durch Auswahl einer Maskengröße die 3D-Daten glätten und unabhängig voneinander mit Abstandsschwellenangaben bereinigen. So einfach die Erfassung eines Scans und die anschließende Datenverarbeitung bzw. -bereinigung auch sind, werden doch im praktischen Umgang eine Reihe von Fragen aufgeworfen, auf die später noch genauer eingegangen wird (siehe Kapitel 3.1.2).

Für diese Diplomarbeit wurde von vorhandenem Datenmaterial ausgegangen. Wegen der Vergleichbarkeit und abschließenden Beurteilung wurden konsistente Datensätze aus Scan und händischer Vermessung benötigt. Die Datensätze fielen im Zuge der Arbeit von Peter Piniel und Elisabeth Bredl [13], die während der Entwicklung von BodyScan [11] entstand, an. In der Vermessung der Körperumfänge in BodyScan musste eine vorgegebene Haltung eingenommen werden, die in der Arbeit von Lohman, Roche und Martorell [12] beschrieben wird. Auch wenn diese Position für die Segmentierung nicht zwingend notwendig ist, wurde sie zunächst beibehalten, da die Abfolge der Datenanalyse in dem zu dieser Diplomarbeit entwickelten Programm von einer bestimmten Haltung ausgeht. Einer Abänderung und Experimenten mit anderen Scanpositionen steht jedoch nichts im Wege.

3.1.2 Probleme der gescannten Daten

Eine genauere Analyse der gescannten Daten zeigt eine Reihe von Problemen auf:

- Es gibt immer wieder Ausreißer, d.h. Punkte, die ganz offensichtlich nicht zur gescannten Person gehören. Sie liegen weit abseits des Hauptobjektes.

- Zwiebelschalen: Die gescannten Punkte jedes einzelnen Lasers werden in das Datenfile übernommen, egal ob sie denselben Teil eines Objektes abdecken, den auch ein anderer Laser erfasst. Diesen Schluss lässt jedenfalls eine Analyse der obj-Files[8] zu. Dadurch vergrößert sich die Anzahl der Punkte und damit auch die zu untersuchende Datenmenge. Da die Punkte verschiedener Laser zwar sehr nahe beieinander, aber eben nicht exakt übereinander liegen, ist eine Konturverfolgung oft sehr mühsam oder gar unmöglich.

  Dieser Zwiebelschaleneffekt lässt sich durch die Kalibrierung reduzieren. Bei einem unkalibrierten Scan liegen diese Schalen weiter auseinander und laufen teilweise auch ineinander.

- Besonders am Kopfanfang gibt es immer wieder Ebenen, die sehr wenige Punkte beinhalten. Sie werden wahrscheinlich durch einzelne Haare erzeugt und ergeben für die weiteren Berechnungen keine sinnvollen Daten.

- Nicht zentrierter Scan: Nicht immer liegen die Scandaten so vor, dass ihr Mittelpunkt mit dem Zentrum des Scanner-Koordinatensystems übereinstimmt.

- Es hat sich gezeigt, dass es entlang der Oberfläche immer wieder einzelne Punkte gibt, die etwas weiter vom Mittelpunkt entfernt liegen als ihre Nachbarn. Diese Punkte werden mit einem einfachen Glättungsalgorithmus korrigiert.

- Scanlöcher sind das Hauptproblem an einem Scan. Sie entstehen, wenn der Laser einen Bereich nicht einsehen kann. Gute Beispiele dafür sind der Achsel- und der Schrittbereich. Die Problematik dabei ist, dass die Software diese Bereiche künstlich auffüllen muss. Egal wie gut der Interpolationsalgorithmus ist, er führt Fehler ein, die das Ergebnis unter Umständen derartig stark verfälschen, dass eine wissenschaftliche Aussage nicht mehr getroffen werden kann.

- Die Scanpunkte der Ebenen liegen nicht in einer Linie untereinander, dadurch sind Interpolationen zwischen Scanebenen schwer möglich (vgl. Abb. 10).

8 Der genaue Aufbau eines Datenfiles wird noch erläutert. Im Wesentlichen handelt es sich um ein Textfile, das mit jedem Texteditor eingelesen und bearbeitet werden kann. Es zeigte sich, dass dieses Datenfile aus 8 Sektionen aufgebaut ist. Löscht man gezielt Sektionen und lädt das Datenfile in MeshLab, kann man den schalenartigen Aufbau und die sich überlappenden Sektionen (die vermutlich den einzelnen Kameras entsprechen) sehr gut beobachten.

- Digitalisierungsfehler: Durch die Abtastung an diskreten Punkten (Auflösung) geht der genaue Verlauf zwischen diesen Punkten verloren. Werden z.B. Extrempunkte (z.B. anteriore oder posteriore Punkte) am Umriss gesucht, ist nicht sichergestellt, dass diese mit den abgetasteten Punkten übereinstimmen.

  Lineare Interpolationen zwischen den abgetasteten Punkten führen hier nicht zum Ziel, wie die folgende Abbildung verdeutlicht. Ganz links sind die erfassten (abgetasteten) Punkte von 3 Ebenen zu sehen. In der Mitte sind zwei mögliche Kurvenverläufe dargestellt, wobei der gesuchte Extrempunkt einmal über, einmal unter Ebene z+1 liegt.

  Rechts wird eine Möglichkeit ausgeführt, durch Interpolation mit Hilfe von höheren (quadratischen, kubischen etc.) Funktionen über mehrere Ebenen den tatsächlichen Extremwert festzustellen. Diese Betrachtungen gelten nicht nur für die Interpolation von zusätzlichen Ebenen, sondern auch für Punkte innerhalb einer Ebene (man stelle sich Abb. 10 als die Punkte einer Ebene vor und nicht als Punkte verschiedener Ebenen).

  Im Rahmen dieser Diplomarbeit wurden die notwendigen 3D-Interpolationsalgorithmen nicht entwickelt bzw. eingebaut. Dies wäre z.B. Gegenstand einer weiterführenden Arbeit.

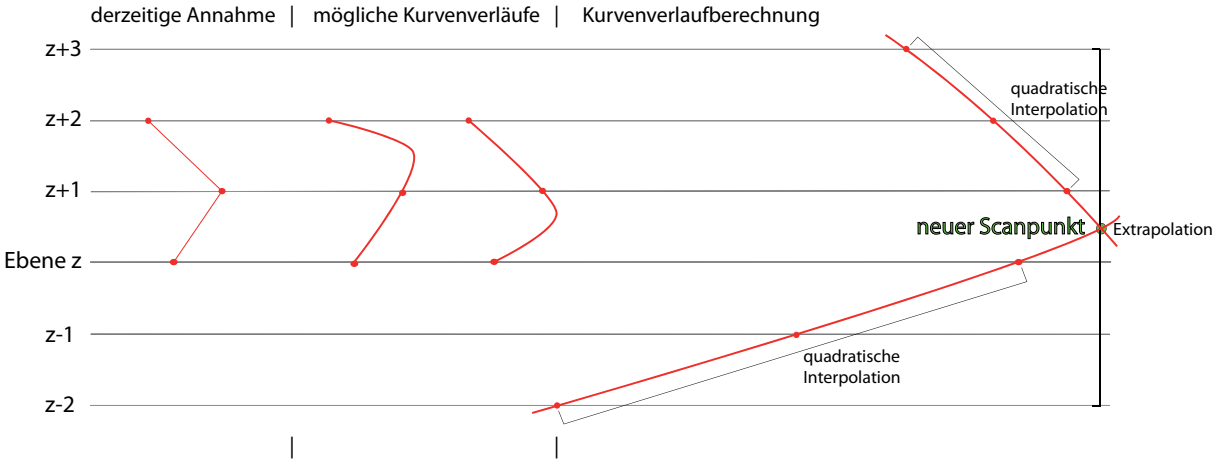


Abbildung 10 Körperkonturpunktfindung mittels Interpolation der Scanebenenpunkte

3.1.3 Manuelle Erfassung von Vergleichsdaten

Um die Ausgaben des Segmentierungsprogramms auf ihre Genauigkeit beurteilen zu können, werden sie mit händisch erfassten Daten verglichen. Die bereits besprochenen 242 Messwerte wurden händisch an Probanden von Peter Piniel und Elisabeth Bredl im Zuge ihres Forschungspraktikums für Biomechanik [13] aufgenommen und aus diesen Werten mit dem Programm ANSEPA die

Parameterwerte berechnet. Die ANSEPA-Berechnungen (vgl. Anhang 2), die bei [13] durchgeführt wurden, enthalten Daten über die Koordinaten, das Volumen, die Masse und das Trägheitsmoment der einzelnen Segmente, sowie des gesamten Körpers. Die Segmentlänge, die schlussendlich für den Vergleich herangezogen wurde, findet sich im Aufnahmeblatt. Dieselben Probanden von [13] wurden auch mit dem Vitronic VITUS smart 3D-Bodyscanner gescannt. So entstanden konsistente und damit vergleichbare Datensätze aus Parameterwerten und Scans. Auch wenn nicht damit zu rechnen ist, dass sich Segmentlängen innerhalb weniger Wochen ändern, kann das z.B. beim Volumen nicht mehr ganz ausgeschlossen werden. Daher ist wichtig, dass Scan und händische Vermessung zum gleichen Zeitpunkt durchgeführt werden, wie es bei den vorhandenen Vergleichsdaten gemacht wurde.

Abbildung 11 Händische Datenerfassung der rechten Schultersegmentbreite Y

Die folgende Tabelle (Tab. 2) listet die dabei entstandenen Scan(.obj)-Dateien auf, die für die Segmentierungssoftware die passende Positionierung des Probanden beinhalten. Es ist jedoch nicht bei allen der aufgelisteten Scans klar, wie oft geglättet, bereinigt und ob vorher kalibriert wurde. Darum wird mit Hilfe des 3D-Programms MeshLab[9] versucht, aus der Genauigkeit übereinander liegender Schichten der Außenhülle auf die Vorkalibrierung zu schließen. Außerdem wird die erforderliche Haltung untersucht, was zu dem Schluss führt, dass die Kalibrierung bei der vorgeschriebenen Probandenhaltung nur einmal vorliegt (Tab. 2).

9 http://meshlab.sourceforge.net (15.02.2011)

Die Scans mit der Bemerkung *ja*[1] in der Spalte *kalibriert** machen Probleme bei der Entscheidung, ob sie kalibriert wurden oder nicht. Es scheint so, als ob die Kalibrierung bzw. die Genauigkeit der 3D-Bodyscanner-Ausgabe mit der Zeit nachlässt, d.h. die Objekte in den Scandateien besitzen Stellen an denen der Umriss aus mehreren überlappenden Ebenen besteht (Zwiebelschaleneffekt), welcher allerdings nicht so stark ausgeprägt ist wie bei wirklich unkalibrierten Scans.

Für die Segmentierung sollte die Kalibrierung nicht stören - sie wirkt sich nur auf die Genauigkeit der Berechnungen aus. Die Längen in der z-Richtung bzw. die Anzahl der Ebenen werden davon nicht betroffen. In x- und y-Richtung könnten Berechnungen (Umfänge, Breitenmessungen etc.) allerdings ungenau werden.

Die Segmentierungssoftware glättet und vereint („verschmilzt") nahe liegende Punkte miteinander, um den Zwiebelschaleneffekt der unkalibrierten Scandateien zu eliminieren. Dabei werden alle Punkte, die näher als 3mm zusammenliegen durch einen Punkt ersetzt, dessen Koordinaten dem Mittelpunkt der gelöschten Punkte entsprechen (Kap. 5.3.2). Die dabei interpolierten Punkte wandern somit an eine andere Stelle und führen im mm-Bereich Ungenauigkeiten ein. Wobei natürlich erwähnt werden muss, dass die Scanpunkte durch die fehlende Kalibrierung vermutlich bereits an einer falschen Stelle liegen. Der Wert von 3mm wurde wegen der Auflösung des Scanners gewählt. Er soll bei der Hälfte der nominellen Auflösung liegen.

Pseudocode 1 Punkteverschmelzung

```
Prozedur: Verschmelze alle Punkte

Zweck:    Ruft solange Verschmelze Punkte auf, bis alle möglichen Punkte verschmolzen sind.

Setze verschmolzenePunkte := 1

Solange verschmolzenePunkte <> 0

  verschmolzenePunkte := VerschmelzePunkte


Prozedur: Verschmelze Punkte

Zweck:    Alle Punkte, die unter einem Mindestabstand  beeinander liegen, werden durch einen
          Punkt ersetzt

Setze idx1 := 0

Setze entferntePunkte := 0

Solange idx1 < Länge der Vertexliste

  Setze Mittelwertberechnung zurück

  Hole das Element idx1

  Merke die Koordinaten dieses Elements

  Für alle Elemente zwischen idx1+1 und Länge der Vertexliste

    Berechne Abstand zu Element idx1

    Falls Abstand < Grenzwert

    dann

      entferne dieses Element aus der Liste

      merke die Koordinaten des entfernten Elements zur

        Mittelwertberechnung

      setze entferntePunkte := entferntePunkte + 1

    ansonsten

      untersuche nächstes Element

  Berechne Mittelwert aus allen gemerkten Koordinaten

  Ersetze die Koordinaten des Elements idx1 durch die berechneten Mittelwerte

Gib die Anzahl der entfernten Punkte zurück
```

Tabelle 2 Vorliegende Scandateien – Kalibrierung - passende Haltung - Prüfung

Scan	kalibriert*	Haltung	Sonstiges
01_Proband	nein	ja	Unterhose erlaubt keine Becken/Oberschenkel-Segmentierung, Füße zusammen - Trennung schwierig
02_Proband	nein	ja	Beckensegmentierung schwierig, Füße unbrauchbar, viele Scanlöcher
03_Proband	nein	ja	—
04_Proband	nein	ja	Füße problematisch
05_Proband	nein	nein	Scanlöcher an Beine und linker Hand
06_Proband	nein	ja	rechter Fuß problematisch
07_Proband	nein	ja	Füße problematisch
08_Proband	nein	nein	Füße problematisch, Unterhose behindert Beckensegmentierung
09_NadineS1	nein	ja	Füße problematisch
10_Proband	nein	ja	Kalibration fraglich, Störungen weniger als bei anderen Scans, Füße problematisch, Knoten im Nacken könnte stören
11_Proband	nein	nein	Haare stören, Füße problematisch, Störungen bei den Händen
12_Proband	nein	nein	Füße problematisch
13_Proband	nein	nein	Füße unmöglich
14_Proband	ja	ja	Füße nur leicht problematisch
15_Proband	nein	nein	rechter Fuß problematisch
16_Proband	ja	nein	linker Fuß problematisch
18_Proband	ja	nein	Füße, Becken vorne problematisch
19_Proband	ja	nein	Füße leicht problematisch
20_Proband	ja	nein	rechter Fuß, Knoten im Nacken problematisch
21_Proband	ja	nein	Störungen bei den Füßen - Ergebnis fraglich
22_Proband	ja[1]	nein	Kalibration dürfte nicht 100%ig sein - leichte Störungen - Bänder im Nacken, Füße (rechter) problematisch
23_Proband	ja[1]	nein	Füße leicht problematisch
24_Proband	ja[1]	nein	Füße problematisch, vor allem rechts
25_Proband	ja	nein	Füße problematisch, vor allem rechts
26_Proband	ja[1]	nein	Beine/Füße haben Scanlöcher, Füße problematisch, Marker
29_Proband	ja[1]	nein	Füße problematisch, vor allem rechts, Unterhose könnte Beckensegmentierung behindern
31_Proband	ja	nein	Füße, Unterhose leicht problematisch
32_Proband	ja	nein	Füße leichte Störungen
33_Proband	ja[1]	nein	Füße leichte Störungen, Knoten im Nacken problematisch
34_Proband	ja	nein	Füße problematisch
35_Proband	ja[1]	nein	Füße mittlere Störungen, Knoten im Nacken problematisch
36_Proband	ja	nein	Füße leichte Störungen
37_Proband	ja[1]	nein	Füße leichte Störungen

3.1.4 Theoretische Fehlerabschätzung

Eine Gegenüberstellung von Segmentlängen zur Anzahl an Scanebenen ergibt folgende Tabelle (Tab. 3). Die Segmentlängen wurden händisch vermessen. Die

Grenzen (Ebenen) der Segmente werden während der Segmentierung intern berechnet. Eine spezielle Version der Segmentierungssoftware zur Erstellung der nachfolgenden Tabelle (Tab. 3) gab die Segmentlängen nicht nur in Millimetern sondern auch in Scanebenen aus.

Tabelle 3 Gegenüberstellung der Segmentlängen (mm) - Ebenenanzahl

Nummer	Segmentname	Länge [mm]	Scanebenen
1	Abdomino-thorakales Segment	422.2	118
2	Kopf-Nacken	252.7	71
3	Linke Schulter	173.5	49
4	Linker Oberarm	332.2	93
5	Linker Unterarm	256.4	72
6	Linke Hand	173.3	49
7	Rechte Schulter	169.7	48
8	Rechter Oberarm	350.2	98
9	Rechter Unterarm	241.9	68
10	Rechte Hand	177,0	50
11	Abdomino-pelvisches Segment	238.2	67
12	Linker Oberschenkel	332.2	93
13	Linker Unterschenkel	408,0	114
14	Linker Fuß*	49.2	15
15	Rechter Oberschenkel	335.7	94
16	Rechter Unterschenkel	404.5	113
17	Rechter Fuß*	49.2	15

* Hier gibt die Tabelle die Höhe des Segmentes an, nicht die Länge.

Diese Gegenüberstellung zeigt, dass (mit Ausnahme der Füße) jedes Segment aus etwa 50-100 Scanebenen besteht. Am Beispiel Oberarm, der aus etwa 100 Ebenen besteht, sieht man, dass jede Ebene mit etwa 1% Anteil in die Länge dieses Segmentes eingeht. Bedingt durch die Digitalisierung sind Fehler im Bereich +/- 1 Ebene durchaus zu erwarten. Im schlimmsten Fall ist mit 2 Ebenen (einer am Anfang, einer am Ende des Segmentes) Fehler zu rechnen. D.h.: alleine durch systembedingte Fehler kann bereits eine Abweichung von 2% entstehen. Besteht ein Segment, wie die Hände, aus nur 50 Ebenen, beträgt der Fehleranteil einer Ebene bereits 2%.

Die Auflösung in horizontaler Richtung ist etwa 5mm [15]. Die Füße werden waagrecht gemessen und haben eine Länge von etwa 250mm (vgl. Tabelle 3). Eine Abweichung von 5mm bedeutet einen Fehler von 2%. Gleiches gilt auch für die Berechnung von Durchmessern und den darauf aufbauenden Volumsberechnungen. Bei einen Segmentdurchmesser von 10cm (für Arme oder Beine) ergibt eine Abweichung um einen Scanpunkt einen Fehler von 5%. Jede

Berechnung basierend auf diesen Werten führt damit bereits einen Fehler ein, der über der Toleranzgrenze von 3% liegt. Die Auflösung des Scanners müsste also deutlich unter 1mm fallen (um das Vierfache steigen) um die max. Abweichung von 3% nicht zu übersteigen. Ein Zusammenhang muss noch erwähnt werden: Da der Abstand der Scanebenen konstant ist, geht das Gewicht jeder Ebene bei kleineren Personen tendenziell stärker in den Fehler ein.

3.2 Experimente

3.2.1 Verbesserung der gescannten Daten durch eine ideale Scanposition

Auch wenn Scanlöcher erkannt und aufgefüllt werden können, ist es doch besser, sie gleich von vorne herein zu vermeiden. Das Ziel der folgenden Experimente war also, eine Position zu finden, in der die Scanschatten möglichst minimal sind. Sie lassen sich nicht komplett vermeiden, aber soweit minimieren, dass der Interpolationsalgorithmus der Software eine recht gute Approximation des tatsächlichen Oberflächenverlaufs liefert.

Um die Ungenauigkeiten im Schrittbereich und den Füßen zu reduzieren, wurde die zu scannende Person um 45 Grad nach rechts gedreht. Die Laser des Scanners befinden sich in den Ecken. Wenn die Person gerade nach vorne steht, verdecken einander die Oberschenkel. Eine Drehung um einen fixen Winkel lässt sich zudem in der Software leicht korrigieren. Diese Drehung ermöglicht dem Scanner den Bereich zwischen den Oberschenkeln genauer abzutasten und so können dort die Scanschatten weiter reduziert werden. Allerdings wurde beobachtet, dass die Arme nun Schatten auf den Körper werfen. In weiteren Experimenten konnte herausgefunden werden, dass abgespreizte Arme diese Schatten wieder reduzieren. Weitere Reduktionen konnten erreicht werden, indem der Oberkörper wieder zurückgedreht wurde.

Es ergibt sich somit eine Idealposition um Scanschatten zu minimieren. Allerdings ist diese Position für die zu scannende Person nicht besonders bequem. Es muss auch darauf geachtet werden, dass die Arme nun nicht aus dem Scanbereich herausfallen und somit abgeschnitten werden. Abbildung 12 zeigt diese Idealposition, Das Foto (Abb. 12 links) wurde bei den Experimenten aufgenommen, mit Unterstützung von Martin K. als Proband. Sein gescannter Datensatz wurde im Programm MeshLab visualisiert (Abb. 12 rechts).

Bezüglich der Genauigkeit im Fußbereich fiel auf, dass in den letzten Ebenen der Scan normalerweise sehr unpräzise wird. Versuchsweise wurde der Proband gebeten, sich auf zugeschnittene Holzklötze bekannter Größe zu stellen. Diese Klötze können (da die Abmessungen bekannt sind) aus dem Scan herausgerechnet werden und könnten außerdem (bei Bedarf) zur Kalibrierung herangezogen werden. Interessanterweise konnten nun die Füße sehr gut aufgelöst werden. Es sind sogar

die Zehen zu erkennen, welche bis jetzt immer verschwommen waren. Trotzdem sind noch Artefakte vorhanden; ein seltsamer Strahlenkranz an der Kante Fuß/Holzklotz zum Beispiel.

Während des Scannens fiel dem Probanden auf, dass die Oberseite des Podestes stark reflektierte. Vielleicht erzeugt das Streulicht auf dem Podest diese Fehler. Nachdem kein schwarzes, nicht reflektierendes Tuch zur Abdeckung zur Verfügung stand, wurde probeweise eine schwarze Tasche unter die Klötze gelegt. Es scheint, als ob die Artefakte dadurch reduziert werden (obwohl natürlich durch die Tasche eine Reihe anderer Ungenauigkeiten eingeführt wird).

Später durchgeführte Überprüfungen und Recherchen konnten diese Beobachtung bestätigen. Ein schwarzes Tuch unter oder über den Klötzen führte zwar zu keiner signifikanten Veränderung dieser Strahlenbrechung, jedoch nur auf Grund von leichten Unebenheiten im Tuch; darum würden sich ein dunkles Podest und Klötze am besten als Untergrund eignen. Ebenso wurde bei den Versuchen festgestellt, dass weiße Kleidung (Unterwäsche) zu besseren Scanergebnissen als schwarze führt.

Ungewollte Reflexionen scheinen das Hauptproblem für einen sauberen Scan zu sein. Nicht nur Umgebungslicht ist störend, sondern auch das Streulicht, das durch die Person selber bzw. durch das Podest erzeugt wird. Wird dieser Gedanke weitergeführt, sollte vielleicht der Körper der Person durch Puder abgedämpft werden. Damit stellt sich aber auch die Frage, welcher Aufwand für einen sauberen Scan noch gerechtfertigt ist.

3.2.2 Verbesserung der gescannten Daten durch die Kalibrierung

In einem ersten Versuch wurde eine Kalibrierbox (Abb. 14), mit den Abmessungen 40 x 25 x 15 cm gebaut. Diese Box wurde möglichst genau rechtwinkelig zum Podest ausgerichtet und gescannt. Es zeigte sich, dass dieser Versuch kein geeignetes Mittel ist, die Genauigkeit des Scanners zu überprüfen. Das Problem liegt darin, dass die Box – egal wie genau man vorgeht – immer minimal zum Scanner verdreht ist.

Die einfache Größenberechnung mit dem Programm *ScanBox* (das nur den kleinsten und größten x bzw. y-Wert sucht und voneinander subtrahiert) ist dadurch nicht möglich. Es müsste die Box gesucht werden, die Kanten interpoliert werden, da der Scanner keine absolute Gerade liefert, und schlussendlich die Drehung korrigiert werden. All diese Schritte führen vermutlich zu viele Fehler ein um eine verlässliche

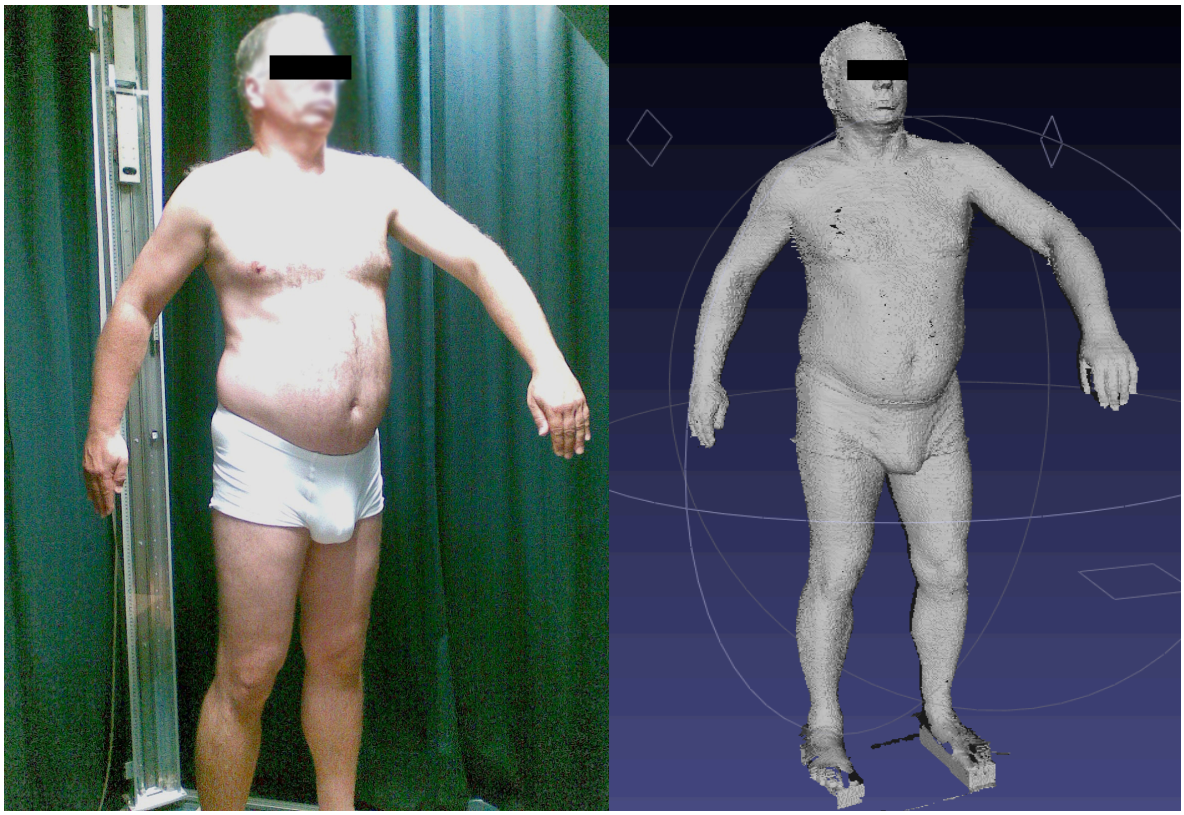


Abbildung 12 Ideale Position für wenig Scanlöcher

Aussage zur Genauigkeit des Scanners treffen zu können. So wurde das zur Verfügung stehende Kalibrierrohr (Abb. 13), das einen Durchmesser von 11cm aufweist, gescannt und mit dem Programm ScanBox analysiert. Hier steht man nicht vor dem oben erwähnten Verdrehungsproblem. Nach einer Kalibration des Scanners wurde die Abmessung in x und y-Richtung korrekterweise mit 11,03cm und 11,01cm angegeben. Diese Abweichungen liegen im Zehntel-Prozent-Bereich. Allerdings muss bemerkt werden, dass ein Durchmesser von 110mm ein ganzzahliges Vielfaches der erwähnten horizontalen Scannerauflösung von 5mm ist.

Die wiederholte Kalibrierung des Scanners erwies sich als äußerst wichtig, Scanpausen von drei Wochen führten zu einem sehr ungenauen Scan. Es empfiehlt sich also, vor einer Scanreihe das Kalibrierrohr zu scannen und – wie in der Kalibrieranleitung von Vitronic [17] beschrieben – zu untersuchen. Es muss sich im Querschnitt ein annähernd sauberer Kreis ergeben. Sind sehr deutlich zwiebelschalenartige Kreissegmente zu erkennen, muss der Scanner neu kalibriert werden.

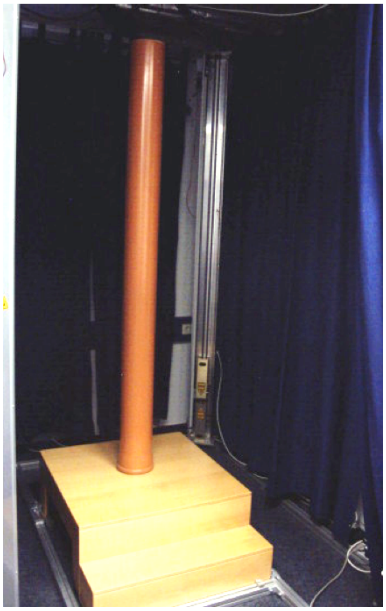


Abbildung 13 Kalibrierrohr

Abbildung 14 Kalibrierbox

3.3 Vorhandene Software (BodyScan)

Aus der vorhandenen Software BodyScan wurde in aller erster Linie die User Interface Umgebung übernommen. Weiters die grundsätzlichen Datenstrukturen zur Speicherung eines Scans, die Ein-/Ausgaberoutinen zum Einlesen eines Scans und zur Ausgabe der bearbeiteten Daten, sowie die notwendigen Unterstützungsroutinen zur Datenverarbeitung. Die Bereinigung der eingelesenen Daten (siehe Kapitel 5.3.2) geschieht nach demselben Prinzip wie bei BodyScan, da sich am Ursprung der Rohdaten nichts geändert hat.

BodyScan unterteilt die Körpersegmente auf Grund der Abstände zwischen Armen, Beinen und Körper. Die Verarbeitungsschritte, die zu dieser Segmentierung führen, wurden übernommen und sind in Kapitel 5.3.3 beschrieben.

Es muss an dieser Stelle darauf hingewiesen werden, dass manche Funktionen zwar in BodyScan und der Segmentierungssoftware denselben Namen haben, aber verschiedene Aufgaben erledigen. Die Methode analyze() z.B. berechnet in BodyScan die gewünschten Umfänge, in dem hier entwickelten Programm, genannt *HatSe* (Hatze Segmentierung), wird jedoch die Segmentierung durchgeführt. Ähnliches gilt für Objekte. Das Result-Objekt heißt zwar gleich, enthält aber die unterschiedlichen Resultate der beiden Programme. Weitere Unterschiede werden im Kapitel 5.2 besprochen.

4 Algorithmen

Zur Auffindung bestimmter, markanter Körperstellen wird eine Reihe selbst entwickelter Basis-Algorithmen eingesetzt. Diese lassen sich wie folgt einteilen:

4.1 Berechnung des Umfanges

Diese vom Projekt BodyScan [11] übernommene Funktion geht von folgender Beobachtung aus: Die Messung des Umfanges eines Körperteils geschieht mit Hilfe eines Maßbandes. Im Normalfall wird dieses Maßband über Körpervertiefungen (z.B. den Nabel) gespannt. Um dieses Verhalten zu simulieren, geht dieser Algorithmus davon aus, dass im Querschnitt gesehen jedes Körpersegment (Arme, Beine, Körper, Kopf etc.) ein geschlossenes Polygon ist, das im Wesentlichen konvex verläuft. In manchen Bereichen ist diese Konvexität nicht gegeben (z. B.: Nabel). Diese Bereiche müssen gesondert berücksichtigt werden. Der Umfang wird durch eine konvexe Hüllkurve bestimmt, die um dieses Polygon gelegt wird („virtuelles Maßband", Bzgl. Abb. 15). Bestimmte Landmarks können nun durch den größten oder kleinsten Umfang in einem abgegrenzten Bereich bestimmt werden.

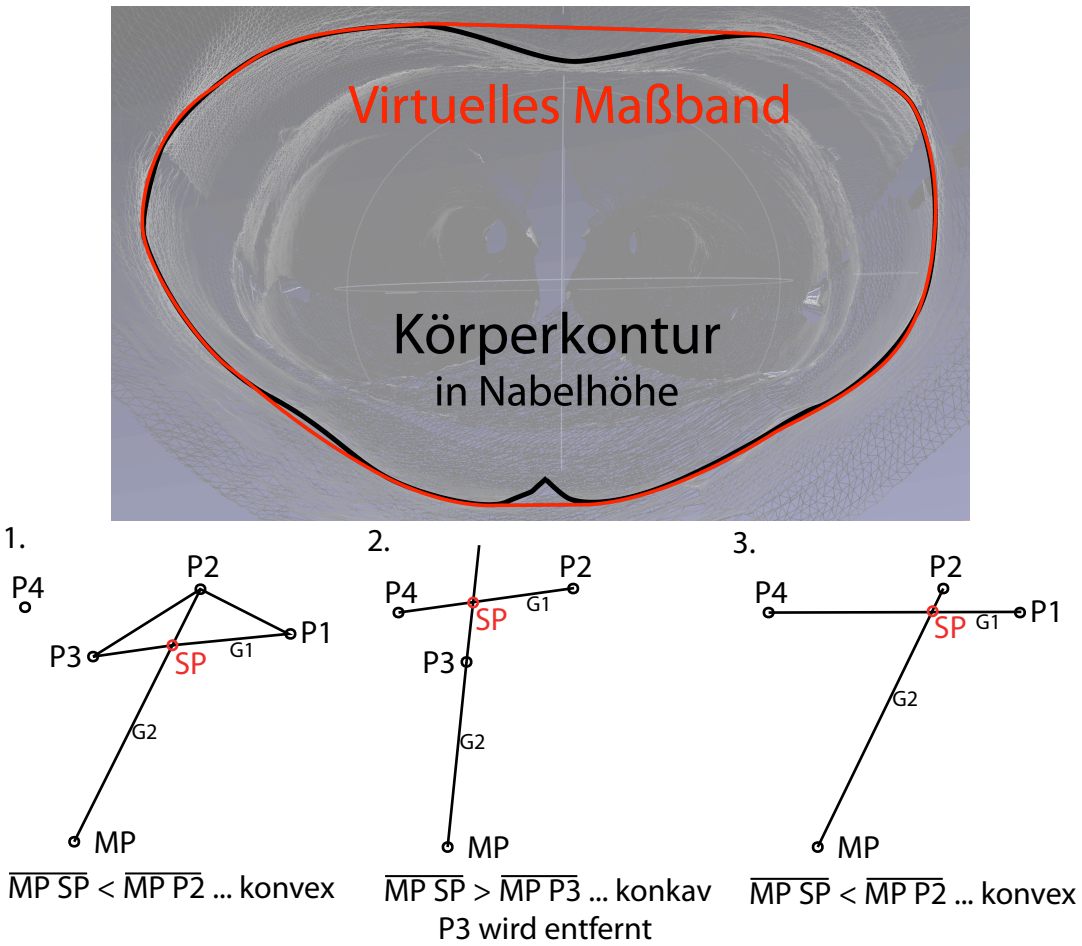


Abbildung 15 Visualisierung der Körperumfangsberechnung [11] mittels virtuellen Maßbands

Pseudocode 2 Virtuelles Maßband

Prozedur: Erstelle Hüllkurve

Zweck: Simulierung eines Maßbandes um eine Körperkontur

Berechne den Mittelpunkt der Punkteliste

Nimm die ersten drei Punkte der Punkteliste

Solange Punkte in der Liste vorhanden sind

Falls diese Kurve konvex verläuft

dann

gehe um einen Punkt weiter

ansonsten

entferne den mittleren Punkt aus der Liste

gehe um einen Punkt zurück

Prozedur: IstKonvex

Zweck: Stelle fest, ob eine Kurve durch drei Punkte konvex zu einem gegebenen Mittelpunkt verläuft

Berechne eine Gerade G1 durch die Punkte P1 und P3

Berechne eine Gerade G2 durch die Punkte P2 und MP

Berechne den Schnittpunkt SP zwischen G1 und G2

Berechne Abstand A1 zwischen MP und SP

Berechne Abstand A2 zwischen MP und P2

Falls A1 < A2

dann

verläuft die Kurve konvex

ansonsten

verläuft die Kurve konkav

4.2 Breitenmessung

Die Messung der Breite in X- bzw. Y-Richtung liefert auch Daten zur Auffindung der Orientierungspunkte. Die Punkte können in der jeweiligen Achse sortiert werden. Die Differenz der jeweils untersuchten Koordinate zwischen erstem und letztem Punkt liefert die gesuchte Information (Abb. 16).

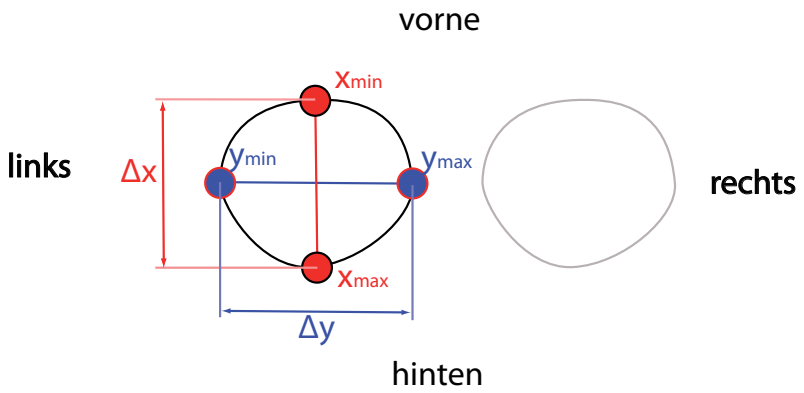


Abbildung 16 Visualisierung der Breitenberechnung

4.3 Umrissanalyse

Wird der Scan in der Frontalansicht oder Lateralansicht (Seitenansicht) betrachtet, können Landmarks durch Erfüllung bestimmter Bedingungen gefunden werden. Das kann z. B. die Über- oder Unterschreitung eines bestimmten Winkels der Tangente an den Umriss (z.B. Hals-Schulter-Übergang bei der Schultersegmentauffindung, vgl. Abb. 18) sein oder anteriore[10] bzw. posteriore[11] Punkte (z.B. beim Gesäß, vgl. Abb. 17). Die äußersten Punkte in einer Ansicht lassen sich dadurch finden, dass die Scandaten nach X- bzw. Y-Koordinate sortiert werden. Der erste bzw. letzte Eintrag in der sortierten Liste gibt dann den jeweils äußersten Punkt an.

Eine weitere Form der Analyse ist die Berechnung von Änderungen über eine bestimmte Anzahl an Scanebenen (Z-Richtung). Da die Scanebenen in kleinen Bereichen schwanken und somit keine eindeutige Tendenz von einer Ebene zur nächsten feststellbar ist, muss die Änderung über eine größere Anzahl untersucht werden. Im Übergang Kopf zu Hals etwa findet eine sehr signifikante Änderung des Mittelpunktes in X-Richtung (hinten-vorne) statt, wenn man den Mittelpunkt im Abstand von mehreren Ebenen betrachtet. (Auch wenn der Mittelpunkt kein Merkmal des Umrisses ist, finden doch dieselben Algorithmen Anwendung.) Andere Analysen beobachten den Abstand der Kontur zu einer beliebigen festen Linie, z. B. einem festen X/Y-Koordinatenwert in jeder Ebene oder einer gedachten Linie zwischen zwei Punkten des Körpers.

10 vorne liegend
11 hinten liegend

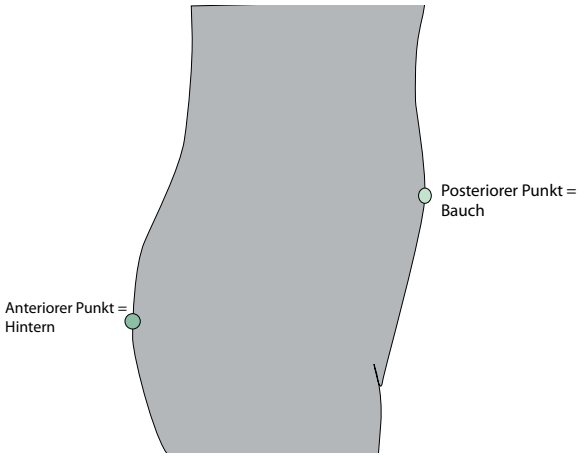


Abbildung 17 Anteriorer/posteriorer Punkt

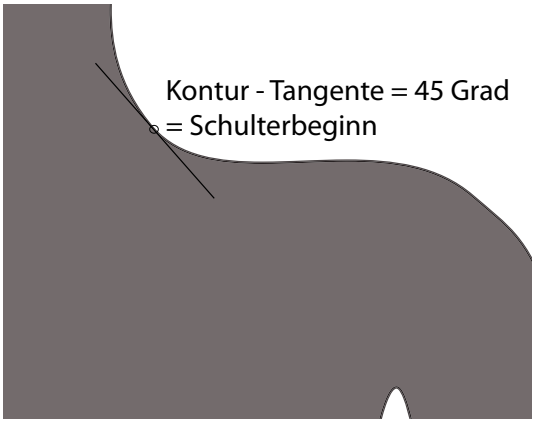


Abbildung 18 Schulterbeginnfindung

Pseudocode 3 Kopferkennung

Prozedur: Kopfsuche zwischen Start- und Endebene

Zweck: Bestimme das Kopfende

Setze aktuelle Ebene = startEbene

Setze nächste Ebene = aktuelle Ebene + Abstand

Solange aktuelle Ebene < Endebene - Abstand ist

Berechne Differenz der X-Koordinate der Mittelpunkte der aktuellen und der nächsten Ebene

Falls die Differenz > Grenzwert

dann

Kopf gefunden, beende Prozedur

ansonsten

untersuche nächste Ebene

4.4 Konturverfolgung

Konturverfolgungen versuchen bestimmten Merkmalen im 3-dimensionalen Raum zu folgen. Eine Hautfalte kann z. B. an einem bestimmten, markanten Punkt aufgefunden werden. Danach wird ihr Verlauf solange untersucht, solange das signifikante Merkmal (Tiefe der Falte) vorhanden ist (Abb. 19). Hier ist das signifikante Merkmal die Tiefe der Falte (Δx_i), wobei die Falte selber durch den roten Punkt markiert wird (Stelle des kleinsten X).

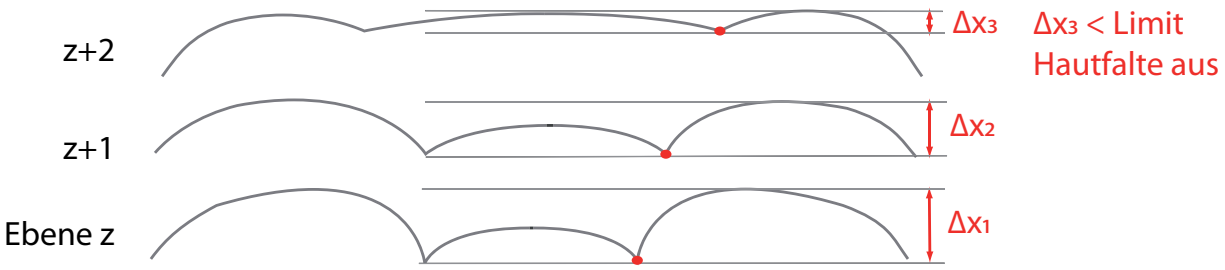


Abbildung 19 Hautfaltenberechnung über mehrere Z(Höhen)-Ebenen

In all diesen Analysen werden nicht nur die kartesischen Koordinaten (X/Y) berücksichtigt, sondern auch Polarkoordinaten. Die Berechnung wird in Anhang 1.1.2 bis 1.1.5 beschrieben. Hier sei nur erwähnt, dass mit ihrer Hilfe z. B. lokale Minima gefunden werden können, indem einfach der Radius beobachtet wird (z.B.: Nabel).

4.5 Segmentlängen

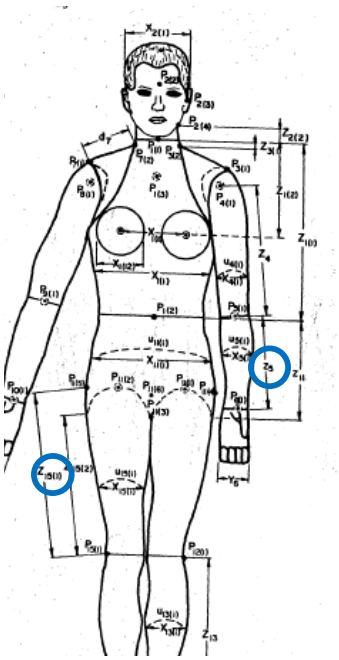


Abbildung 20 Beispiele berechneter Segmentlängen

Bis auf die Füße werden die Segmentlängen durch ihre Ausdehnung in Z-Richtung (von oben nach unten), die Segmentlänge der Füße in X-Richtung (von hinten nach vorne) definiert. Da alle Grenzen im Laufe der Segmenterkennung festgelegt werden, ist die Bestimmung der Segmentlänge (vgl. blau gekennzeichnete Z-Parameterwerte in Abb. 20) eine einfache Subtraktion der Z-Positionen der

Grenzebenen. Eine alternative Berechnungsweise ist, die Achse des Segmentes zu finden. Dazu wird der Mittelpunkt der Startebene mit dem Mittelpunkt der Endebene verbunden und diese Länge gemessen. Beide Programmvarianten sind nach wie vor im Code enthalten und können mittels der Konstanten `USE_AXIS_FOR_LENGTH` (siehe Anhang 1 - `Constants.java`) ausgewählt werden. Es hat sich jedoch gezeigt, dass die Ergebnisse nur marginal voneinander abweichen, da in der gewählten Scanposition die Segmentlängen bereits parallel zur z-Achse ausgerichtet sind (Abb. 21).

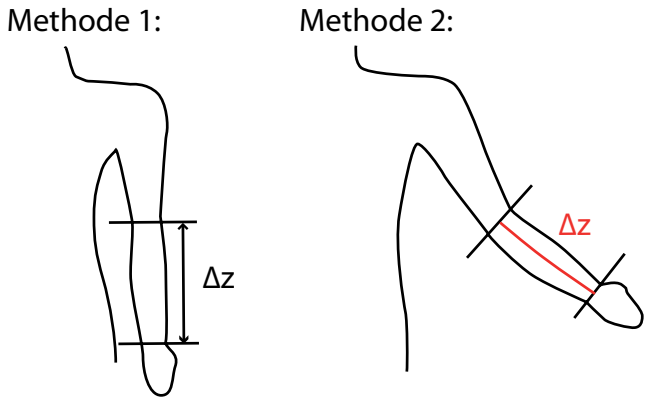


Abbildung 21 Berechnungsmethoden für die Segmentlängen

4.6 Überlegungen zur Testbarkeit

Sehr früh in der Programmentwicklung wurden Überlegungen angestellt, wie Genauigkeit und Korrektheit der entwickelten Programmsegmentierung überprüft werden können. Das Ziel der Arbeit war, ein Programm zu schaffen, das einen 3D-Scan möglichst ohne manuelle Eingriffe in die 17 Segmente nach Hatze unterteilen kann. Das Ergebnis sind also Punktwolken aus Raumpunkten (sog. *Vertices*), die den einzelnen Segmenten zugeordnet werden können.

Abgesehen von einer visuellen Überprüfbarkeit mittels des Programms MeshLab (indem die Daten in einer Art Explosionsdarstellung, wie es von manchen Handbüchern technischer Geräte bekannt ist, ausgegeben werden), wäre es sinnvoll, eine Gegenüberstellung zu bekannten Messwerten durchzuführen. Obwohl eine Berechnung der Segmentparameter nicht primäres Ziel dieser Arbeit ist, können doch einzelne Parameter recht einfach berechnet werden:

- *Segmentlängen:* Sofern Segmente parallel zu den Hauptachsen der gescannten Person liegen, lassen sich Segmentlängen sehr einfach berechnen. Diese Daten können dann mit den händisch vermessenen Werten (z) (Abb. 3) verglichen werden. Wenn die Segmente anders liegen, muss versucht werden, die Achsen zu finden und diese entsprechend zu transformieren.

- *Volumen*: Im Zuge der Segmentierung werden die Koordinaten der Vertices eines Segmentes auch in Polarkoordinaten bzgl. des Mittelpunktes der Ebene, zu der sie gehören, umgerechnet. Es ist trivial, diese Daten dazu zu verwenden, die Fläche jeder Ebene und aus den Flächen zweier benachbarter Ebenen das Volumen dieser Scheibe zu berechnen. Durch Addition dieser Einzelvolumina kann zumindest eine erste Näherung des Segmentvolumens berechnet werden. Dieses Ergebnis kann dann mit den Ergebnissen des Programms ANSEPA aus [13] verglichen werden.

- *Masse*: Aus dem Volumen und der durchschnittlichen Dichte des menschlichen Körpers kann die Masse eines Segmentes berechnet werden. Das Hatze-Modell unterteilt ein Segment jedoch in weitere Subsegmente mit unterschiedlichen Dichten. Es sind daher Abweichungen zu den mit ANSEPA berechneten Massen zu erwarten.

- *Schwerpunkte*: Obwohl es einfach ist, den Mittelpunkt eines Segmentes zu berechnen, entspricht er nicht dem Schwerpunkt. Es wird sich zeigen, wie schwierig die Schwerpunktberechnung ist und ob dies zu Ergebnissen führt, die als Vergleichskriterium zu der ANSEPA-Ausgabe herangezogen werden können.

Im Rahmen der Diplomarbeit wurde zur Evaluierung der Ergebnisse die Segmentlänge herangezogen, weil sie am genauesten zu berechnen war. Eine Volumsberechnung findet sich zwar im Programm, sie wird aber wegen einiger Unzulänglichkeiten (die noch beschrieben werden) nicht verwendet.

5 Implementierung

5.1 Programmiersprache und -umgebung

Wie schon bei der Erstellung der Software zur Berechnung von Körperumfängen BodyScan [11] stand auch bei dem zu entwickelnden Programm die Forderung nach Portabilität im Vordergrund. Das Programm sollte ohne großen Aufwand auf verschiedenen Hard- und Softwareplattformen (PC, Mac, Windows, OS/X, Linux...) ausführbar sein und dadurch eine gewisse Zukunftssicherheit haben. Modularität, Dokumentierbarkeit und Wartbarkeit waren weitere Anforderungen, sowie – nicht zuletzt – die Kostenfrage. Teure Softwarepakete und Entwicklungsumgebungen schieden für die Erstellung der Diplomarbeit von vornherein aus. Aus diesen Vorgaben ergab sich die Entscheidung, Java zu benützen. Obwohl diese Programmiersprache nicht zu den effizientesten gehört, scheint sie doch für diese Aufgabe ideal zu sein. Zudem sind das SDK (Software Development Kit) und das RTE (RunTime Environment) frei auf der Java-Website[12] verfügbar.

Für die grafische Darstellung wurde Swing und AWT (Abstract Window Toolkit) verwendet. Beides sind Teile der Java Foundation Class und erlauben die Erstellung eines plattformunabhängigen GUIs (Graphical User Interface). AWT ist nur eine dünne Abstraktionsschicht zum darunter liegenden Betriebssystem. D.h. die Darstellung z.B. einer Auswahlliste hängt von der Plattform ab, auf der das Programm schließlich läuft und sieht unter Windows anders aus als unter Mac OS/X. Darauf baut Swing auf, welches eine plattformunabhängige grafische Benutzeroberfläche zur Verfügung stellt.

Als Entwicklungsumgebung bot sich Eclipse[13] an, eine ebenfalls frei verfügbare und in Java geschriebene Entwicklungsumgebung (IDE, Integrated Development Environment). Programmerstellung und Fehlersuche gehen mit diesem mächtigen Werkzeug sehr schnell und flüssig voran. Außerdem unterstützt die Umgebung Javadoc, womit Beschreibungen direkt im Quelltext durchgeführt werden können. Diese werden nach dem richtigen Setzen der vordefinierten Javadoc-Tags zu HTML-Dokumentationsdateien generiert. Das finale Java-Programm kann dann zusammen mit allen Bibliotheken als JAR (Java Archive) exportiert werden, welches auf jedem Rechner ausführbar ist, auf dem das Java-RTE installiert ist.

Die verwendeten Programmversionen sind:

Java 1.6.0 update 10 und
Eclipse 3.6.0

12 http://www.java.com/en/ (27.12.2010)
13 Programmiersoftware und integrierte Entwicklungsumgebung; http://www.eclipse.org/

5.2 Modellerstellung und Datenstrukturen

Ausgehend von den Erfahrungen des Projektes BodyScan [11], welches aus den 3D-Daten des Vitronic Scanners Körperumfänge berechnet, wurden die Datenstrukturen entworfen und erweitert. Basierend auf den grundlegenden Objekten (`VertexObject`, `FaceObject`) und Listen daraus (`VertexList`, `FaceList`) wurden hierarchisch übergeordnete Objekte geschaffen (z.B.: `zObj` für die Daten einer Ebene). Ganz oben in der Hierarchie steht das `MeshObject`, das alle Informationen zu einem eingelesenen Scan enthält (vgl. Anhang 1). Das Segmentierungsprogramm ist in der Lage, mehrere Scans gleichzeitig einzulesen, darzustellen und auszuwerten. Jeder Scan wird in einem eigenen Panel dargestellt, d.h. jedem Panel ist genau ein `MeshObject` zugeordnet.

Unterstützend wurde eine Reihe weiterer Objekte eingeführt, z.B. das `MeanObj` zur Berechnung von Mittelwerten, ein `Utils-Objekt`, das logging-Funktionen zur Verfügung stellt, oder das `InterpolObj`, das lineare oder quadratische Interpolationen zwischen Punkten durchführt. `Results` wird wiederum die Ergebnisse beinhalten. `InputFile` und `OutputFile` dienen zum Einlesen bzw. zur Ausgabe von Daten und erlauben es, das Programm modular anzupassen bzw. an diesen Stellen PlugIns zu schaffen. `Constants` enthält alle Konstanten, die im Programm verwendet werden. Viele Aspekte können damit an spätere Erkenntnisse angepasst werden. Es wäre grundsätzlich auch möglich, diese Konstanten durch Eingabefelder im Programm veränderbar zu machen.

5.3 Funktionsweise

Mit dem Programm HatSe (Hatze Segmentierung) können 3D-Scans im OBJ oder ASC Format (sog. *Meshes*) analysiert werden. Da das OBJ Format neben den eigentlichen Scanpunkten (*Vertices*) auch Informationen darüber enthält, wie diese Vertices zusammenhängen (Flächen / *Faces*), ist es grundsätzlich vorzuziehen. Das ASC Format enthält nur die Vertices. Manche Informationen können aber einfacher aus dem OBJ-File gewonnen werden. Das Programm versucht diese Informationen auch aus dem ASC-Format abzuleiten, allerdings kann das zu Fehlern bei der Bestimmung von Ausreißern führen.

Das Programm geht zur Gewinnung der Messdaten nach folgenden Schritten vor:

- Einlesen der Daten
- Bereinigung
- Erkennung der Segmente (grob/fein)
- Berechnung der Kontrolldaten (Segmentlängen, Volumina etc.)

5.3.1 Einlesen der Daten

Nach dem Start von HatSe wird das Hauptfenster geöffnet. Mit dem Button „Datei laden" wird der Dateidialog geöffnet, mit dem eine Scandatei ausgewählt werden kann.

Definitionen

- Die z-Achse verläuft von oben (Kopf) nach unten (Füße). Die Werte steigen von oben nach unten an.
- Die x-Achse verläuft von hinten (Rücken) nach vorne (Brust). Die Werte steigen von hinten nach vorne an.
- Die y-Achse verläuft von links nach rechts (aus der Sicht der gescannten Person). Die Werte steigen von links nach rechts an.
- Als "Ebene" werden Vertices mit gleichem Z-Wert bezeichnet.

Diese Definitionen sind notwendig, da die X- und Y-Achsen des Scans im Gegensatz zur Definition des Hatze-Modells vertauscht sind.

Dateiaufbau

In einer OBJ-Datei beginnt jede Zeile mit einem String. Dieser besteht in den verwendeten Dateien aus einem oder zwei Buchstaben. Danach folgen weitere Daten. Die verwendeten Einträge sind:

v (Vertex) – ein Punkt, nachfolgend die x/y/z-Koordinaten

f (Face) – eine Fläche, gefolgt von den Vertex und VertexNormal-Indices der beteiligten Punkte (3 Paare der Form v/vt/vn). vt (VertexTexture) wird von der Vitronic Software nicht verwendet.

Der Eintrag vn (VertexNormale) ist in der OBJ-Datei der Vitronic Software zwar vorhanden, wird aber in dieser Arbeit nicht verwendet.

Der Zusammenhang zwischen den Daten wird durch den Index bestimmt. Ein Eintrag der Form

```
f 27//32 58//66 123//130
```

bedeutet, dass diese Fläche vom 27., 58. und 123. Vertex mit dem 32., 66. und 130. Normalvektor bestimmt wird.

Der Index wird dabei vom Auftreten der Daten in der Objektdatei bestimmt. Alle erwähnten Elemente können gemischt auftreten, der Index gibt die Nummer des erwähnten Elementes an (nicht zu verwechseln mit der Zeilennummer). Im

Segmentierungsprogramm werden die Zeilen der Reihe nach in entsprechende Listen eingelesen und miteinander verlinkt.

In einer ASC-Datei gibt jede Zeile einen Punkt der Punktwolke an. Jede Zeile enthält die x, y und z-Koordinate eines Vertex (in dieser Reihenfolge), gefolgt von einem optionalen Farbwert. Es gibt hier keinerlei weitere Informationen.

5.3.2 Bereinigung der Daten

- Die Scans weisen Scanlöcher auf. Das sind Stellen, an denen keine Abtastung erfolgen konnte (vgl. Kapitel 3.1).

- Die Scans sind nicht immer zentriert, d.h. der Mittelpunkt des Meshes liegt nicht immer bei (x/y/z) = (0/0/0).

- Die Scans weisen mehrere, übereinanderliegende, teilweise ineinanderfließende Ebenen (Zwiebelschalen) auf.

- Es gibt isolierte Punkte (stray points), d.h. einzelne Punkte innerhalb und außerhalb des Meshs, die nicht zum eigentlichen Körper gehören.

- Die Oberfläche des Scans ist nicht glatt.

Vor der weiteren Verarbeitung muss daher eine Bereinigung der Daten durchgeführt werden:

- Nach dem Einlesen des Scans werden im ersten Schritt alle Ebenen gelöscht, die weniger als eine bestimmte Anzahl an Vertices aufweisen. Damit werden Fehlscans im Kopfbereich (Anfang) und darüber eliminiert.

- Danach werden isolierte Punkte gelöscht. Das sind Punkte, die keinem Face angehören (OBJ Files) oder deren Abstand zum nächstliegenden Punkt über einem bestimmten Schwellwert liegt (ASC Files).

- Anschließend werden alle Punkte, die näher als 3mm zusammenliegen, durch einen Punkt ersetzt, dessen Koordinaten dem Mittelpunkt der gelöschten Punkte entsprechen (Verschmelzen von Punkten).

  Diese Maßnahme eliminiert die Zwiebelschalen. Allerdings ist nach dieser Operation das Modell nicht mehr konsistent (die Face-Informationen stimmen nicht mehr), da eben einige Punkte gelöscht werden. In der weiteren Verarbeitung wird die Face-Information nicht mehr benötigt, daher wird dieses Verhalten derzeit nicht korrigiert.

- Aus den so bereinigten Punkten wird jeweils der maximale und minimale Wert jeder Achse ermittelt (Boundary Box), ein Offset berechnet und alle Punkte entsprechend korrigiert. Diese Berechnung findet jedoch nicht als isolierter

Schritt statt, sondern wurde teilweise in die vorherigen Schritte integriert um die Verarbeitungsgeschwindigkeit zu steigern. Nach diesem Schritt liegt der Mittelpunkt des Meshes annähernd bei (0/0/0). Damit sind einfache Unterscheidungen wie links/rechts oder vorne/hinten sehr schnell möglich.

- An dieser Stelle wird der Scan grob in die Körperteile nach BodyScan [11] unterteilt (Kopf, Torso, Arme, Beine). Im Querschnitt bildet jeder dieser Körperteile eine geschlossene Kurve um einen Mittelpunkt. Das erlaubt die Konvertierung in Polarkoordinaten und eine einfache Glättung der Oberflächen. Diese Glättung geschieht durch Mittelwertbildung der Radien zweier benachbarter Punkte.

- Es wird wiederum eine Verschmelzung durchgeführt, da durch die Glättung eventuell wieder Punkte unter die 3mm-Grenze zusammenrücken.

- Nach der Glättung werden Scanlöcher gesucht und aufgefüllt. Betrachtet werden die Punkte einer Scanebene, die zu einem Körperteil gehören. Aus der Winkeldifferenz der Polarkoordinaten zweier benachbarter Punkte wird eine mittlere Differenz berechnet. Überschreitet die Winkeldifferenz zwischen zwei benachbarten Punkten die zweifache mittlere Differenz, wird ein Scanloch angenommen.

  Aus der Winkeldifferenz und der mittleren Differenz wird eine Anzahl an Punkten berechnet, die eingefügt werden sollen. Diese Anzahl wird auf den nächsten ganzzahligen Wert aufgerundet und aus der Winkeldifferenz und der Anzahl der Punkte ein Winkelwert errechnet. Das Scanloch wird nun dadurch aufgefüllt, dass Punkte mit diesem Winkelwert eingefügt werden. Der Radius wird linear zwischen den Radien der beiden Eckpunkte interpoliert.

  Durch das Einfügen der Punkte kann es passieren, dass sich die interpolierten Kurven zweier Körperteile überlappen (z.B. bei den Oberarmen und dem Torso). Diese Überlappung wird dadurch korrigiert, dass im Überlappungsbereich die Y-Koordinaten der Punkte durch den Y-Mittelwert beider Punkte ersetzt werden.

5.3.3 Erkennung der Segmente

- Kopferkennnung: Es hat sich herausgestellt, dass der Adamsapfel in einem Scan nicht aufzufinden ist. Daher wird in einer ersten Näherung die Kopferkennung nach BodyScan durchgeführt (signifikante Änderung der X-Koordinate des Körpermittelpunktes). Dies ist eine Variante der Umrissanalyse, bei der der Umriss durch die X-Koordinaten der Körpermittelpunkte der betrachteten Ebenen ersetzt wird. In einem späteren Schritt wird, wenn der Bauch erkannt wurde, der posteriore Punkt an der Vorderseite zwischen dieser ersten Näherung und dem Bauch gesucht. Als eine bessere Näherung des Kopfendes (O2) wird nun die halbe Distanz zwischen dem bisherigen Kopfende und dem posterioren Punkt angenommen.

- Groberkennung Arme und Beine: Dieser von BodyScan [11] übernommene Schritt untersucht den Punktabstand in Y-Richtung Ebene für Ebene. Liegt der Abstand über einem bestimmten Wert, wird eine Lücke erkannt. Je nach Lage dieser Lücke (sehr nahe am 0-Punkt der Y-Achse oder weiter entfernt) wird auf eine Arm- oder Beinlücke geschlossen. Die Punkte jeder Ebene werden nun den einzelnen Segmenten auf Grund ihrer Y-Koordinate zugeordnet.

- Schultererkennung: Beginnend von der ersten Näherung des Kopfendes wird die Silhouette des Körpers in der Y-Ebene verfolgt. Der Schulterbeginn (Übergang Hals-Schulter – Punkt P7(2) für die rechte Schulter) befindet sich dort, wo die Tangente an die Silhouette den 45 Grad Winkel über- bzw. unterschreitet.

  Nach der Armerkennung wird das rechte bzw. linke Acromion gesucht (Punkt P7(1) für die rechte Schulter). Wieder wird das Über- bzw. Unterschreiten des 45 Grad Winkels an die Silhouette als erste Näherung angenommen. Beide Punkte P7(1) und P7(2) werden mit Hilfe der Umrissanalyse bestimmt.

  Der Achselpunkt des Schultersegments wird durch die Armerkennung festgelegt.

  Die Kurve zwischen Achsel und P7(2) ist aus dem Scan nicht ableitbar. Sie soll an der Rückseite durch Punkt P7(3), dem posterioren Punkt des Scapula, laufen. Die mathematische Formulierung dieser Kurve zur Trennung des Schultersegmentes vom Torso fällt jedoch schwer. Punkt P7(3) liegt zudem relativ weit unten (manchmal sogar unter der Achsel) und relativ weit medial. Eine übliche quadratische Interpolation durch diese drei Punkte führt zu keinem praktisch verwertbaren Ergebnis.

  Aus diesem Grund werden vorläufig Faktoren verwendet, mit denen der Punkt P7(3) in der Y (links/rechts) und Z (oben/unten)-Achse verschoben werden kann. Die quadratische Kurve durch die drei Punkte P7(2), P7(3) und Achsel liefert dann brauchbare Ergebnisse. Die Trennung zwischen Oberarm und Schulter wird mittels Spiegelung der Außenlinie des Armes zwischen Achsel und Acromion um die Achse Acromion – Mittelpunkt des Oberarmes auf Achselhöhe durchgeführt.

- Armerkennung: Im Querschnitt befindet sich zwischen Torso und Armen eine Lücke. Diese Lücke wird bei der groben Unterteilung gesucht und im Bereich der Achsel nach oben interpoliert. Diese Interpolation ist notwendig, weil bei einzelnen Ebenen keine Lücke erkannt wird. Diese Lücken liegen an der Vorder- und Rückseite nicht unbedingt an derselben Stelle (Y-Koordinate ist ungleich). Daher werden zwischen dem vorläufigen Armbeginn und der Schulter die Lücken separat an der Vorder- und Rückseite gesucht, dazwischen wieder interpoliert und so der Armbeginn in Richtung Schulter endgültig festgelegt.

  Der Übergang Ober-/Unterarm befindet sich beim Ellbogen (P9(1) für den rechten Ellbogen). Es hat sich gezeigt, dass dieser Punkt durch die Körperhaltung

im Scan sehr einfach zu finden ist: Er wird mit Hilfe der Umrissanalyse durch den posterioren Punkt an der Vorderseite des Armes bestimmt.

- Handerkennung: Ausgehend vom Ellbogen wird mittels Umfangberechnung die Stelle des kleinsten Umfanges gesucht. Hier muss die Erkennung rechtzeitig gestoppt werden, da im Fingerbereich die Umfänge wesentlich kleiner sind. Für die Fingerlänge wurde demnach ein Wert aus den vorliegenden Scans empirisch ermittelt.

- In ähnlicher Weise werden die Beine weiter unterteilt. Nach der ersten Erkennung der Beine (in der Mitte des Körpers befindet sich eine Lücke in Y-Richtung und der Interpolation im Schrittbereich) werden Ober- und Unterschenkel gesucht.

  Es gibt auch hier eine signifikante Änderung im Kurvenverlauf. Dazu wird auf der posterioren Seite eine Verbindungslinie zwischen dem obersten und untersten Punkt des Beines gezogen. Von dieser virtuellen Verbindungslinie wird der Abstand zum tatsächlichen Beginn des Beines gemessen. Der anteriore Punkt ergibt sich nun in Höhe des Punktes O16 für das rechte Bein und analog dazu für das linke Bein. Hier wird der Umriss in Bezug auf eine feste, gedachte Linie analysiert.

  Interessant ist die Beobachtung des Abstandes der Beine an der Vorder- und der Rückseite des Scans und das Ende der Lücke in z-Richtung, Hört diese Lücke im Beckenbereich an der Vorderseite signifikant weiter unten auf als an der Rückseite, kann darauf geschlossen werden, dass der Scan von einem männlichen Probanden stammt (Geschlechtserkennung).

- Füße: Die Trennung zwischen Unterschenkel und Füßen wird auf Höhe des Knöchels durchgeführt (Punkte P13(1) bzw. P16(1)). Sie werden mit Hilfe der Breitenmessung gesucht und festgelegt.

- Becken: Der Punkt P11(3) (Oberschenkelbeginn im Schritt) wurde bereits bestimmt. Von diesem Punkt aus wird die Hautfalte zwischen Becken und Oberschenkel mittels Konturverfolgung untersucht, solange eine signifikante Tiefe vorhanden ist. Ist die Falte nicht mehr tief genug, wird abgebrochen. Diese Trennkurve wird geglättet und dient zur Abtrennung von Oberschenkel und Becken.

  An der oberen Stelle der Trennkurve gibt es keine Hautfalte mehr. Hier dient der letzte erkannte Punkt zusammen mit den beiden Punkten an den Seiten (Außen-, Innenseite) des Oberschenkels als drei Stützpunkte, zwischen denen quadratisch interpoliert wird.

  Auf der Rückseite wird mittels Konturverfolgung des Beckens die Hautfalte des Gesäßes bestimmt. Auch hier kann sie nur bis zu einem bestimmten Punkt verfolgt werden. Von dort werden die fehlenden Punkte wieder quadratisch interpoliert.

- Als vorletzter Schritt wird nun der Nabel gesucht. Zwischen Arm- und Beinbeginn wird der anteriore Punkt an der Körpervorderseite gesucht. In einem festgelegten Bereich über bzw. unter diesem Punkt wird nun ein lokales Minimum gesucht. Hier kommt wieder die Umwandlung in Polarkoordinaten zugute. Der Nabel bestimmt die Trennung zwischen dem abdomino-thorakalem und dem abdomino-pelvischem Segment. Für diesen Schritt wird eine Kombination aus Umrissanalyse und Konturverfolgung verwendet.

- Nun wird noch das Kopfende wie oben bereits erwähnt korrigiert.

5.3.4 Probleme bei der Segmenterkennung

Bestimmte Punkte am Körper lassen sich zwar recht genau ertasten, sind aber im Scan schwer bzw. gar nicht zu finden. Einer dieser Punkte ist der Adamsapfel. Es gibt zwar Scans, in denen dieser Punkt in der Silhouette sehr prominent zu sehen ist, aber das sind eher Ausnahmen. Interessanterweise konnte eine recht gute Annäherung (wie unter 4.3.3 erläutert) gefunden werden, die für die verwendeten Scans zutraf. Hier müssen jedoch weitere Untersuchungen zeigen, ob diese Annahme gerechtfertigt ist oder ob an dieser Stelle doch ein Marker verwendet werden muss.

Eine weitere Problematik ist die Definition des Schultersegmentes. Wie bereits erwähnt, ist der genaue mathematische Verlauf der Trennkurve Schulter/Oberkörper schwer bestimmbar. Als vorläufige Hilfe werden die Koordinaten des posterioren Punktes des Scapula (P7(3)) mit einem Faktor in Constants.java verändert. Die Differenz der Y- und Z-Koordinaten zwischen P7(3) und dem Acromion P7(1) (bzw. P3(3) und P3(1) für die linke Schulter) werden mit diesem Faktor multipliziert. Dadurch können die Punkte P7(3) und P3(3) Richtung Acromion verschoben werden und die Trennlinie einfach quadratisch interpoliert werden (vgl. Abb. 27 in Anhang 1 - `calcLeftShoulder()/calcRightShoulder()`). Bei Vorliegen genauerer mathematischer Beziehungen muss dieser Teil entsprechend angepasst werden.

Ein grundsätzliches Problem ist die Qualität des Scans an sich. Das wurde bereits ausgiebig diskutiert. Besonders zeigen sich diese Probleme im Fußbereich. Die Knöchelerkennung und damit die Fußsegmentierung ist hier eher als Zufall zu bezeichnen. Werte, die auf Grund dieser Segmentierung berechnet werden, sind als unzuverlässig einzustufen.

5.3.5 Berechnung der Kontrolldaten

Die Berechnung der Segmentlängen wurde bereits im Kapitel 4.5 beschrieben. Experimentell wurden Methoden zur Volumsberechnung implementiert. Es hat sich jedoch herausgestellt, dass die Volumsberechnung für einen Vergleich der Segmentierungssoftware mit den realen Daten nicht geeignet ist. Der Grund sind

die sehr hohen Abweichungen, die auf den Digitalisierungsfehler zurückzuführen sind. Da aber die Algorithmen im Programm vorhanden sind und in weiterführenden Arbeiten verwendet werden könnten, werden sie hier dennoch beschrieben:

- Nachdem die Segmente einmal bestimmt worden sind, kann das Volumen berechnet werden. Alle Vertices sind in Ebenen aufgeteilt, und innerhalb einer Ebene sind die Vertices nach Winkeln, ausgehend vom Ursprung an der Vorderseite, sortiert. Dadurch liegen immer zwei benachbarte Vertices auch in der Sortierung nebeneinander.

- Der Mittelpunkt jeder Ebene kann berechnet werden.

- Zwei benachbarte Punkte spannen mit dem Mittelpunkt ein Dreieck auf. Die Fläche dieses Dreiecks ist sehr einfach mit Hilfe der Heronschen Flächenformel[14] zu berechnen. Alle derart berechneten Flächen einer Ebene werden zur Gesamtfläche aufaddiert. Aus dem Mittelwert zweier benachbarter Flächen und dem Abstand wird ein Zylindervolumen berechnet und die derartig ermittelten Teilvolumina zu einem Gesamtvolumen aufaddiert.

- Dies funktioniert in bestimmten Bereichen (z.B. Becken) nicht. Bedingt durch den Kurvenverlauf des Querschnittes kann die Reihenfolge der Punkte in den Polarkoordinaten nicht eindeutig bestimmt werden (vgl. Abb. 22 – die richtige Reihenfolge ist 1-2-3, die Reihenfolge, die durch die Sortierung nach Winkeln erkannt wird ergibt sich jedoch als 1-3-2). Hier wird ein anderer Ansatz verwendet:

- Der Querschnitt wird in einen vorderen und einen hinteren Teil unterteilt. Dazu werden die am weitesten links und rechts liegenden Punkte ermittelt. Die x-Koordinate dieser beiden Punkte wird interpoliert und ergibt die Unterscheidung in Vorder- und Rückseite.

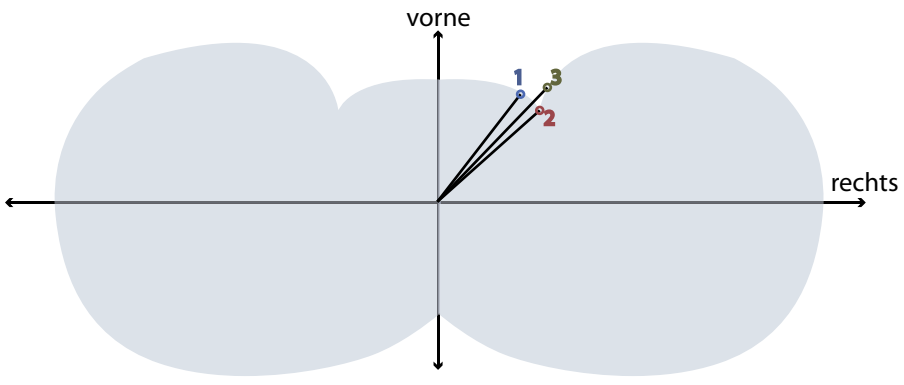


Abbildung 22 Kurvenverlauf des Beckenbodenquerschnittes

14 Heron'sche Flächenformel; Wikipedia: http://de.wikipedia.org/wiki/Dreieck (18.10.2010)

- Die Vertexliste stellt einen Polygonzug dar, dessen Fläche, bezogen auf eine der Achsen (x oder y), mit Hilfe der Gauß'schen Trapezformel[15] berechnet werden kann.

Ein Vergleich mit der Dreiecks-Methode zeigte jedoch nur marginale Unterschiede. Daher wurde dieser Ansatz nicht weiter verfolgt.

5.4 Benutzerinterface

Das Segmentierungsprogramm erlaubt das Einlesen einer ASC- bzw. OBJ-Datei, die als Ergebnis des 3D-Scans vorliegt. Nachdem das OBJ Format (siehe Kap. 5.3.1 - Dateiaufbau) neben den einzelnen Scanpunkten (Vertices) auch Informationen darüber enthält, wie diese Vertices zusammenhängen (Flächen / Faces), ist es grundsätzlich vorzuziehen. Das ASC Format enthält nur die Scanpunkte, alle anderen Informationen müssen daher zusätzlich errechnet werden. Die Vitronic-Software erlaubt den Export des Scans in beiden Formaten. Wie schon erwähnt, sollte das OBJ-Format vorgezogen werden.

Nach dem Einlesen der Daten werden diese bereinigt (siehe Kapitel 5.3.2). Scanlöcher, Ausreißer und andere Artefakte werden eliminiert. Anschließend erfolgt eine erste Unterteilung in die einfach zu erkennenden Gliedmaßen (Arme, Beine, Körper und Kopf). In weiteren Schritten werden diese Segmente nach den Vorschriften von Hatze weiter zerlegt (Schultersegmente, Ober-, Unterarme, Hände, Ober-, Unterschenkel, Füße sowie die beiden Abdomino-Segmente) (siehe Kapitel 5.3.3). Danach werden die Kennwerte (Segmentlängen) berechnet, um eine Überprüfung mit experimentell gemessenen Werten zu ermöglichen.

Beim Starten des Programms öffnet sich zunächst ein leeres Fenster (Abb. 23).

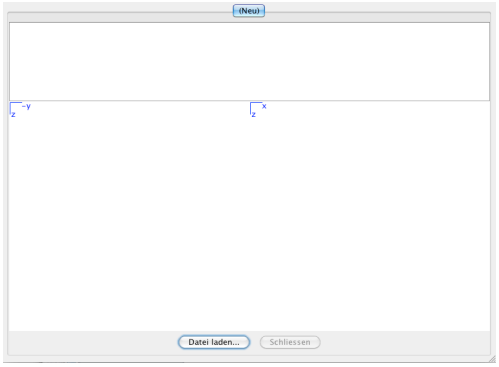


Abbildung 23 Programmstart Fenster

Ein Klick auf „Datei laden" zeigt den Öffnen-Dialog. Hier kann die zu untersuchende Scandatei ausgewählt werden. Es werden die Formate „obj" und „asc" unterstützt (Abb. 24).

15 Gauß'sche Trapezformel; Wikipedia: http://de.wikipedia.org/wiki/Gaußsche_Trapezformel (28.05.2010)

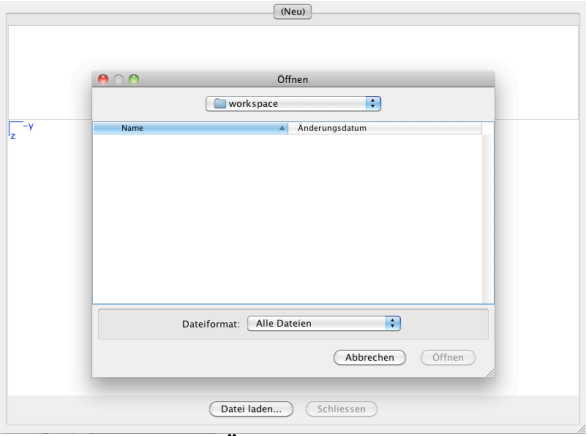


Abbildung 24 Datei-Öffnen Dialog

Nachdem eine gültige Datei ausgewählt wurde, wird die Segmentierung (siehe Kap. 5.3.2/3) durchgeführt. Das kann je nach Leistung des eingesetzten Rechners 10 Sekunden[16] und länger dauern. Während dieser Zeit werden nur Berechnungsinformationen in der Java-Konsole von Eclipse ausgegeben, im Userinterface ist keine Aktivität sichtbar. Anschließend wird eine Liste der Segmentlängen ausgegeben sowie die Segmentierung grafisch dargestellt (Abb. 25).

Das Hauptfenster (Abb. 25) besteht aus folgenden Teilen (von oben nach unten):

- Übersicht aller geöffneten Dateien
- Liste der Messergebnisse
- Grafische Visualisierung
- Werkzeugleiste

Jede geöffnete Datei wird in einem eigenen Panel dargestellt. Der Name jeder Datei wird im Titel des Panels eingetragen. Das letzte Panel trägt immer den Titel „(Neu)" (vgl. Abb. 25). Durch Anklicken dieses Panels wird ein neues, leeres Panel geöffnet und eine neue Datei kann zur Analyse geladen werden. Im Bereich direkt darunter werden die berechneten Segmentlängen aufgelistet. Im Mittelbereich des Fensters wird der gescannte Körper dargestellt. Die 17 Segmente werden (wie in Abb. 25) farblich hervorgehoben. Bedingt durch die geringe Auflösung des Bildschirms gegenüber der auf 2D umgerechneten 3D-Daten, kann die Darstellung fehlerhaft erscheinen (Näheres dazu in Kapitel 6.1). Dies ist lediglich eine Auswirkung der grafischen Ausgabe und hat keinen Einfluss auf die tatsächliche Berechnung.

Die Werkzeugleiste im unteren Bereich des Fensters enthält zwei Schaltflächen:

- „Datei laden" zeigt den Öffnen-Dialog, mit dem eine neue Datei zur

16 Betriebssystem: Mac OSX 10.6.5; verwendete Hardware: Apple iMac 10,1; Prozessortyp: Intel Core 2 Duo; Prozessorgeschwindigkeit: 3,06 GHz; Speicher: 8GB

Berechnung geladen werden kann. Dieser Button ist jedoch nur in einer leeren Ansicht verfügbar, d.h. solange keine Datei geöffnet wurde.

- „Schließen" beendet die aktuelle Ansicht und das Panel verschwindet aus der Auflistung im oberen Bereich des Fensters. Dieser Button ist nur verfügbar, falls eine Datei geladen wurde.

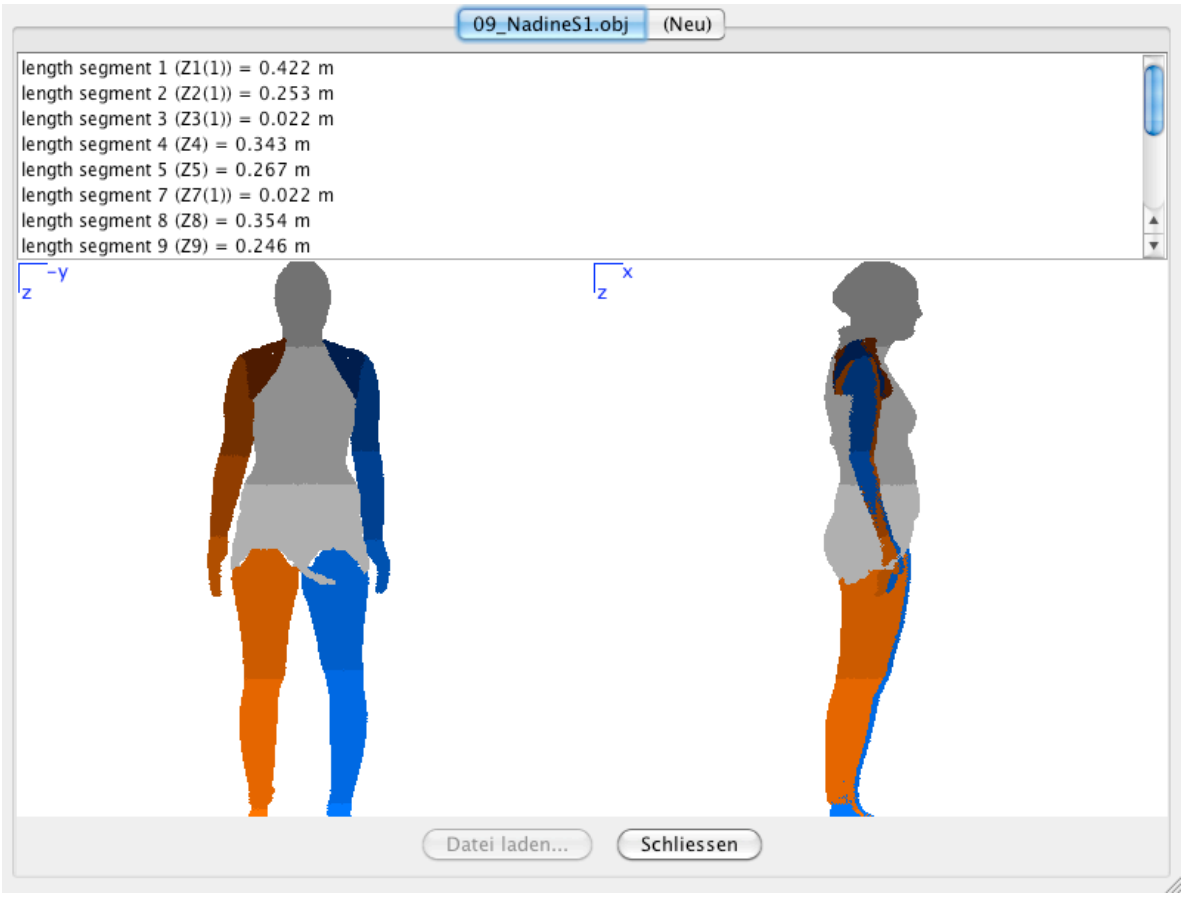


Abbildung 25 Programm: Darstellung der Segmentierung

6 Ergebnisse

6.1 Segmentierung des 3D-Scans

Das Segmentierungsprogramm visualisiert die Segmentierung auf seiner Benutzeroberfläche durch Einfärben der einzelnen Scanpunkte, je nach Segmentzugehörigkeit. Abbildung 26 zeigt diese Ausgabe nach Einlesen einer Scandatei (09_NadineS1.obj), hier mit der experimentellen Angabe der 17 Segmentvolumina, die später durch die Segmentlängen ersetzt wurden.

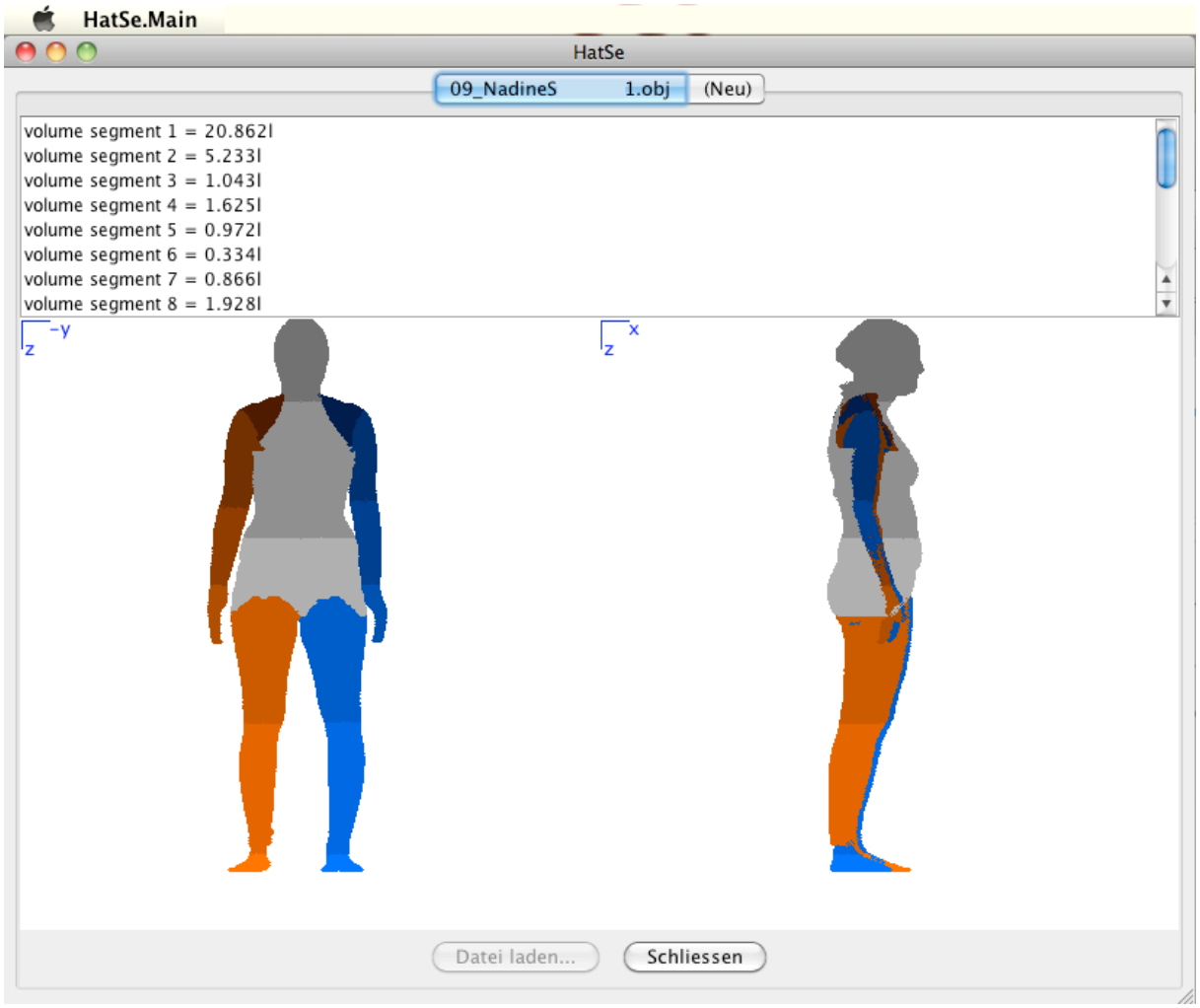


Abbildung 26 Ergebnissegmentierung der Segmentierungssoftware von Nadine S. Scandatei

Das Programm zeigt einen groben Überblick, die echte visuelle Kontrolle erfolgt mit der 3D-Visualisierungssoftware MeshLab. Zu diesem Zweck generiert das Programm eine explodierte Ansicht der Segmentierung des Körpers in einer ASC-Datei (Name_exploded.asc), die mit MeshLab angezeigt werden kann (Abb. 27).

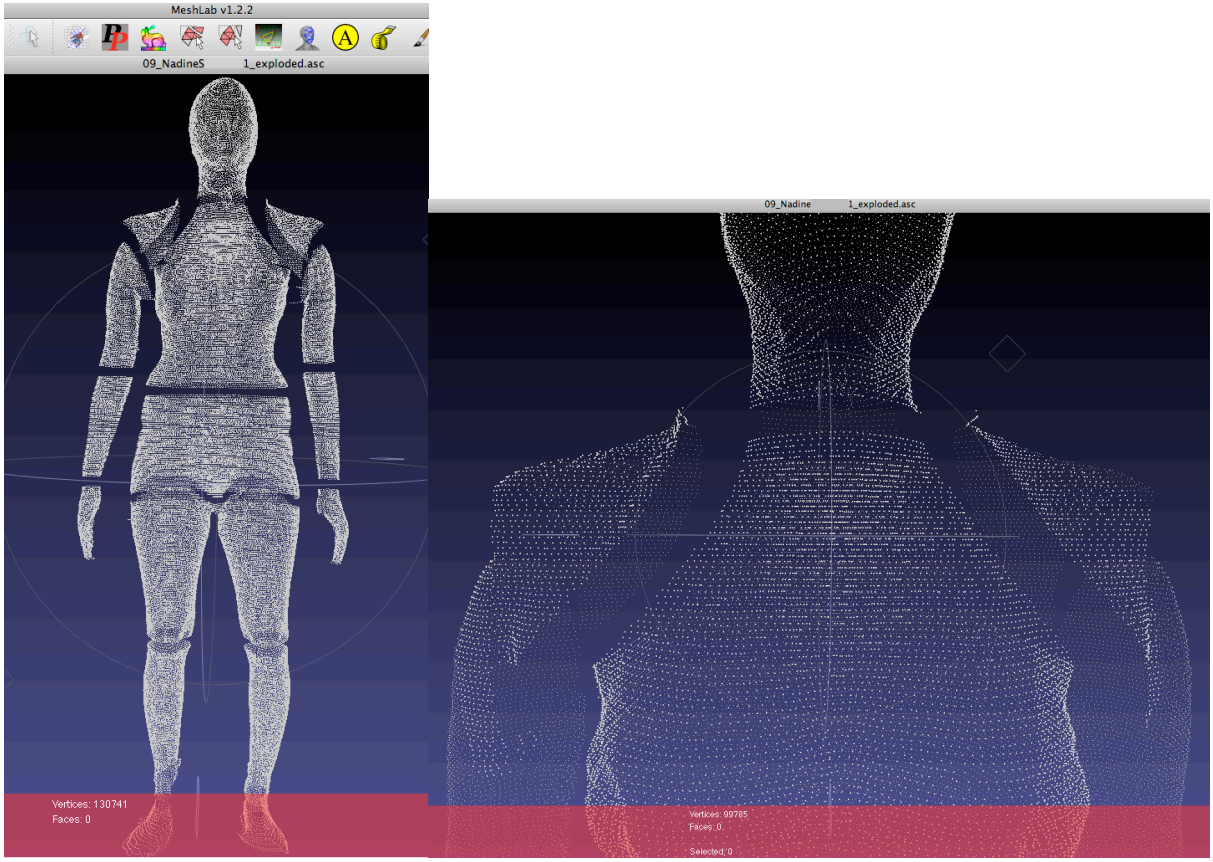


Abbildung 27 Explodierte Ansicht der Segmentierung in MeshLab

Bedingt durch die geringe Auflösung des Bildschirms und das Von-links-nach-rechts-Zeichnen der Visualisierung (Abb. 26) kann es passieren, dass verdeckt liegende Punkte fälschlicherweise dargestellt werden. Das Programm muss die Punktkoordinaten runden, um sie auf der geringeren Bildschirmfläche darstellen zu können, wodurch ein gering nach rechts verschobener hinterer (z.B. am Rücken liegender) Punkt, der normalerweise von der Vorderseite verdeckt wird, minimal nach links gerundet wird und somit den an der Vorderseite liegenden, vorher gezeichneten Punkt überschreibt. Daraus resultieren die Ausreißer an der rechten Schulter von Nadine S1's Segmentierungs-Visualisierung (Abb. 26). Nachdem der Schwerpunkt der Segmentierungssoftware aber auf der Erkennung der Segmentgrenzen lag, wurde die Implementierung einer korrekten 3D-Darstellung nicht in Angriff genommen. Beim näheren Betrachten der von MeshLab erzeugten Darstellung (Abb. 27) sieht man, dass die in der Ansicht des Segmentierungsprogramms gezeichneten Ausreißer (Abb. 27 – z.B. rechte Schulter) in den richtigen Segmenten liegen. Diese explodierte dreidimensionale Ansicht dient somit zur exakten Visualisierung des Programmergebnisses.

6.2 Auswertung der Vergleichsdaten

Um die Genauigkeit von Berechnungen auf Basis des 3D-Scans im Vergleich mit real gemessenen Werten beurteilen zu können, werden im Segmentierungsprogramm die Längen der Körpersegmente berechnet. Tabelle 6 stellt diese berechneten den gemessenen Werten an einem Beispiel gegenüber. Sie werden im Programm im Hauptfenster ausgegeben (vgl. Abb. 28).

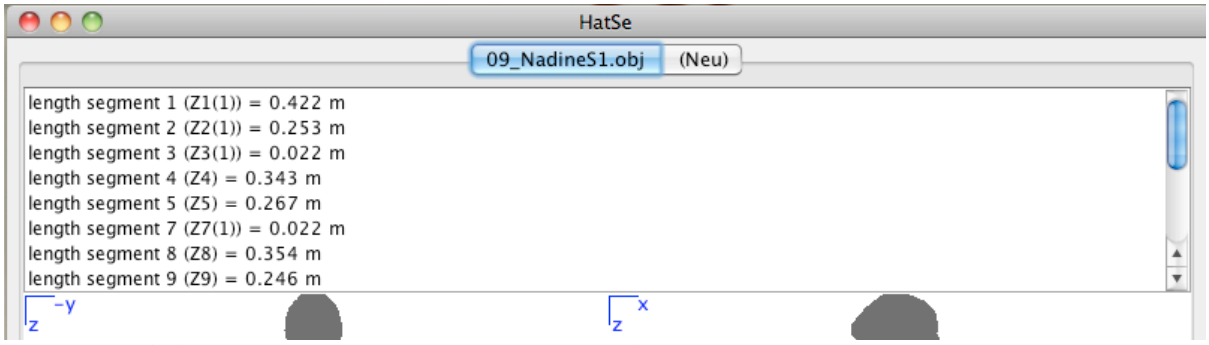


Abbildung 28 Segmentierungsprogramm-Anzeige der Segmentlängen

Die Segmente sind bei der Programm-Anzeige nach der Nummerierung laut Hatze (Abb. 1) sortiert. In Tabelle 4 werden die Segmente wie am Aufnahmezettel der 242 Messwerte (Abb. 3) gereiht um auf den Unterschied aufmerksam zu machen.

Tabelle 4 zeigt eine Gegenüberstellung der händisch vermessenen zu den automatisch berechneten Segmentlängen. Die letzte Spalte gibt die Abweichung in Prozent an. Gute Ergebnisse ergeben sich für die Unterarme, Ober- und Unterschenkel. Hier bewegen sich die Abweichungen im Bereich weniger Prozent (4,03 – 6,37%, und einem Ausreißer mit 1,19%). Dies entspricht Abweichungen bis zu 22 mm.

Diese Werte liegen nicht weit von der akzeptablen Toleranzgrenze von 3% entfernt. Eine erste Erklärung ist der Einfluss des Digitalisierungsfehlers (siehe auch Kapitel 3.1.2). Bei jeder Digitalisierung ist mit einem Fehler von +/-1 Wert zu rechnen. Experimentell wurden deshalb die berechneten Segmentlängen um +/- 1 Scanebene (3,6 mm)[17] verändert und die Fehler in Prozent errechnet. Es zeigt sich, dass in einigen Fällen die Abweichungen unter die geforderten 3% fallen, mit einer maximalen Abweichung von 5,09% (Tabelle 5). Wird zum Beispiel der berechnete Wert des rechten Unterarms (der mit 0,246m um -4,47% vom gemessenen Wert abweicht um 1 Scanebene (3,6 mm) erhöht, sinkt die Abweichung auf -2,96% und damit unter die tolerierte Fehlergrenze. Dies zeigt den Einfluss der systembedingten Auflösung.

17 Die Auflösung von 3,6mm wurde innerhalb des Programmes errechnet, indem die z-Werte zweier benachbarter Ebenen voneinander subtrahiert wurden.

Tabelle 4 Segmentlängenvergleich anhand Probandin Nadine S.

SegmentNr	Segmentname	MesswertNr (vgl. Abb.3)	gemessen [m]	HatSe [m]	Abweichung [%]
1	ABDOMINO-THORACTIC	Z1(1)	0,431	0,422	**-2,13**
2	HEAD-NECK SEGMENT	Z2(1)	0,208	0,253	**17,79**
3	LEFT SHOULDER	Z3(1)	0,060	0,022	**-172,73**
7	RIGHT SHOULDER	Z7(1)	0,060	0,022	**-172,73**
4	LEFT ARM	Z8	0,286	0,343	**16,62**
8	RIGHT ARM	Z4	0,285	0,354	**19,49**
5	LEFT FOREARM	Z5	0,250	0,267	**6,37**
9	RIGHT FOREARM	Z9	0,257	0,246	**-4,47**
6	LEFT HAND				
10	RIGHT HAND				
11	ABDOMINO-PELVIC	Z11	0,258	0,253	**-1,98**
12	LEFT THIGH	Z12(2)	0,329	0,311	**-5,79**
15	RIGHT THIGH	Z15(2)	0,332	0,336	**1,19**
13	LEFT LEG	Z13	0,413	0,397	**-4,03**
16	RIGHT LEG	Z16	0,416	0,394	**-5,58**
14	LEFT FOOT	Z14	0,261	0,184	**-41,85**
17	RIGHT FOOT	Z17	0,255	0,201	**-26,87**

In Tabelle 4 sind die folgenden Werte nicht aussagekräftig:

- linker und rechter Oberarm: Die Längenberechnung erfolgt vom Schultergelenk bis zum Ellbogen. Im Programm wird die Position des Schultergelenks jedoch nicht ermittelt (weil es für die Segmentierung nicht notwendig ist) – daher ist diese Angabe eher als eine experimentelle Ausgabe anzusehen.

- Füße: In diesem Bereich liefert der Scanner extreme Ungenauigkeiten. Auch hier ist die Ausgabe als experimentell anzusehen (für den Fall, dass doch einmal ein genauerer Scan eingelesen wird).

- Schultersegmente, Kopf/Nackensegment: Das Ende des Kopf/Nacken-Segmentes wird durch die Position des Adamsapfels bestimmt. Dieser ist jedoch im Scan nicht exakt aufzufinden. Die angrenzenden Segmente sind daher unter diesem Aspekt zu sehen.

Tabelle 5 Segmentlängenvergleich +/-1 Scanebene

SegmentNr	Segmentname (MesswertNr)	gemessen [m]	HatSe [m]	Abweichung [%]	+ 1 Scanebene + 0,04 [m]	Abweichung [%]	– 1 Scanebene - 0,04 [m]	Abweichung [%]
5	LEFT FOREARM (Z5)	0,250	0,267	**6,37**	0,271	**7,61**	0,263	**5,09**
9	RIGHT FOREARM (Z9)	0,257	0,246	**-4,47**	0,250	**-2,96**	0,242	**-6,02**
12	LEFT THIGH (Z12(2))	0,329	0,311	**-5,79**	0,315	**-4,58**	0,307	**-7,03**
15	RIGHT THIGH (Z15(2))	0,332	0,336	**1,19**	0,340	**2,24**	0,332	**0,12**
13	LEFT LEG (Z13)	0,413	0,397	**-4,03**	0,401	**-3,10**	0,393	**-4,98**
16	RIGHT LEG (Z16)	0,416	0,394	**-5,58**	0,398	**-4,63**	0,390	**-6,56**

Experimentell wurden auch die Volumina der einzelnen Segmente mit jenen verglichen, die sich aus dem händischen Vermessen der selben Person (09_NadineS1) und nachfolgendem Berechnen mit dem Programm ANSEPA ergeben (Tab. 6):

Tabelle 6 Vergleich: berechnete Segmentvolumen nach manueller Aufnahme und Berechnung in ANSEPA gegenüber Volumsberechnung mit Segmentierungsprogramm HatSe

SegmentNr	Segmentname	ANSEPA Volumen (l)	HatSe Volumen (l)	Abweichung (%)
1	Abdomino-thorakales S.	17,031	20,862	22,49427515
2	Kopf-Nacken	5,784	5,233	-9,526279391
3	linke Schulter	1,062	1,043	-1,789077213
4	linker Oberarm	1,953	1,625	-16,79467486
5	linker Unterarm	0,805	0,972	20,74534161
6	linke Hand	0,236	0,334	41,52542373
7	rechte Schulter	0,978	0,866	-11,45194274
8	rechter Oberarm	1,930	1,928	-0,103626943
9	rechter Unterarm	0,886	0,885	-0,112866817
10	rechte Hand	0,242	0,271	11,98347107
11	Abdomino-pelvisches S.	13,027	17,555	34,75857834
12	linker Oberschenkel	7,890	6,664	-15,53865653
13	linker Unterschenkel	2,766	2,858	3,326102675
14	linker Fuß	0,906	0,455	-49,77924945
15	rechter Oberschenkel	8,220	6,836	-16,83698297
16	rechter Unterschenkel	2,865	2,787	-2,722513089
17	rechter Fuß	0,919	0,285	-68,98803047
		67,5	71,459	5,865185185

Wie schon oben ausgeführt, sind nur einige der aufgelisteten Werte vergleichbar. Dies sind die Unterarme sowie die Beinsegmente (Ober- und Unterschenkel). Diese Werte liegen innerhalb oder sehr nahe der akzeptablen Fehlergrenze. Die vergleichsweise große Abweichung von 20% des Volumens des linken Unterarms

wurde nicht weiter untersucht, ist aber vermutlich auf die Interpolation der Segmentlöcher (Kap. 5.3.2) und Digitalisierfehler (Kap. 3.1.2) zurückzuführen.

7 Zusammenfassung

Die Berechnung der Parameterwerte (Volumen, Masse, Trägheitsmomente usw.) der 17 Segmente des Hominoids von Hatze aus einem realen Körper erfordert die händische Aufnahme von 242 Messwerten. Dies ist zeitaufwändig und fehleranfällig. Mit Hilfe eines modernen 3D-Scanners lassen sich jedoch einfach und schnell (wenige Sekunden für einen Scan) recht genaue Abbildungen des menschlichen Körpers erstellen.

Zentrales Ergebnis der vorliegenden Arbeit ist eine Vorschrift und eine Software, mit deren Hilfe die automatische bzw. computergestützte Unterteilung dieser Scandaten in die 17 Segmente ohne Setzen von zusätzlichen Markierungen möglich ist. Bei Beachtung der in Kapitel 3.1.1 beschriebenen Vorgehensweise beim Scannen, können die Scandaten mit minimalem Fehler erfasst werden. Die entwickelten Algorithmen (Bereinigung des Scans, Umfangsberechnung, Konturanalyse, Breitenmessung, Konturverfolgung) ermöglichen die Auffindung der gesuchten Merkmale, an Hand derer schließlich die Segmentierung durchgeführt wird. Im Gegensatz zu bisherigen Methoden der Datenaufnahme, ist es somit nicht mehr notwendig, Markierungen mit einem Stift auf den Körper zu zeichnen. Ein weiterer Vorteil für den zu untersuchenden Probanden ist es außerdem, dass vielleicht ihm unangenehme Berührungen oder Messvorgänge wegfallen („Eingriff in die Intimsphäre").

Für die Analyse der Güte der hier entwickelten Segmentierung wurden in dieser Arbeit optische Beurteilungen des segmentierten Scans im Programm MeshLab sowie Vergleiche von berechneten Parameterwerten (Segmentlänge) mit händischen Vermessungen durchgeführt. Diese Analysen ergaben, dass die Segmenterkennung an sich mit guten Ergebnissen durchgeführt werden kann. Sie wird lediglich durch die Auflösung des Scanners begrenzt.

Die Arbeit zeigt, dass die Genauigkeit des Scanners nicht optimal ist. Theoretische Überlegungen und praktische Untersuchungen (siehe Kap. 3) haben gezeigt, dass die Abtastgenauigkeit des Scanners selber gerade an der Grenze dessen liegt, was für eine wissenschaftliche Auswertung im Bereich der Biomechanik notwendig ist. Die Grenze der geforderten Abweichung von 3% wird in zu vielen Fällen überschritten.

Es stellt sich heraus, dass Fehler in der Segmenterkennung eingeführt werden durch:

- Scanneraufl̈osung von 4-5 mm. Bei kleinen Segmenten fällt diese Auflösung sehr stark ins Gewicht.
- Gerade Scanebenen messen fehlerhaft, wenn Segmentebenen nicht in der Scanebene verlaufen.
- Segmentgrenzen werden nach mathematischen Funktionen berechnet, was bei einem menschlichen Körper nicht zwingend perfekte Ergebnisse liefert (z.B. Schultererkennungsfehler und Oberschenkelerkennungsfehler).

- Segmenterkennung kann eventuell verbessert werden (Genauigkeit und Verarbeitungsgeschwindigkeit), wenn eine standardisierte Körperteil-Verhältnistabelle eingesetzt werden kann.

Es ist jedoch zu erwarten, dass mit dem technischen Fortschritt und einer damit verbundenen Erhöhung der Genauigkeit des Scanners diese systemimmanenten Fehler unter die geforderte Grenze fallen. Gehen wir von einer Verdoppelung oder Vervierfachung der derzeitigen Auflösung aus, könnte ein 3D-Scanner durchaus eine verwendbare und vielversprechende Alternative zur herkömmlichen, händischen Vermessungsmethode werden. Auch wenn sich auf Grund der durchgeführten Versuche die Körperhaltung der zu scannenden Person ändern würde, wären die Algorithmen grundsätzlich noch gültig. Das Programm muss lediglich so angepasst werden, dass es die richtigen Gliedmaßen im richtigen Scanbereich sucht.

8 Weiterführende Arbeiten

Der Schwerpunkt dieser Diplomarbeit lag in der Entwicklung einer Methode zur automatischen Segmentierung von Daten eines 3D-Scanners und der Erstellung eines unterstützenden Programms. Im Folgenden werden Probleme und Erkenntnisse zusammengefasst, die während der Diplomarbeit aufgetreten sind und die die Grundlage weiterführender Arbeiten bilden könnten:

Die verwendeten Algorithmen können verbessert, verfeinert und genauer kalibriert werden. Besonderes Augenmerk sollte auf die Entwicklung von Verfahren wie 3D-Interpolationen zur Auffüllung von Scanlöchern oder Glättungsalgorithmen zur Eliminierung von Ausreißern gelegt werden. Mit ihrer Hilfe könnten die Daten besser ausgewertet und genauere Ergebnisse erzielt werden.

Daten von einem unkalibrierten Scanner sind ungeeignet für eine zuverlässige Auswertung (Scanlöcher, Ausreißer, Zwiebelschalen wie in Kapitel 3.1.2 erläutert). Hier wäre zu untersuchen, welche Methoden (Glättung, Interpolation etc.) die Datenaufbereitung - ohne neue Fehler einzuführen - verbessern können. Auch eine automatische Kalibrierung oder Warnung vor unkalibrierten Scans wäre denkbar, wenn der Scanner einige Wochen nicht kalibriert wurde. Das Vitus Manual [17] empfiehlt eine Kalibration einmal im Monat.

Im Kapitel 3.2.1 wurden Untersuchungen zur Auffindung einer idealen Scanposition zur Vermeidung von Scanlöchern beschrieben. Da im Zuge dieser Arbeit jedoch keine Scans mit dieser Position vorlagen, wurde auf eine Anpassung der Software verzichtet. Im selben Kapitel wurde auch die Möglichkeit beschrieben, den Probanden auf Klötze zu stellen, um die Abtastung im Fußbereich zu verbessern. Diese Maßnahme sollte in einer zukünftigen Version des Programms, genauso wie die veränderte Haltung, entsprechend berücksichtigt werden. Da die Abmessungen der Klötze bekannt sind, könnten sie auch den Ausgangspunkt für eine automatische Kalibrierung darstellen oder die Grundlage zur Beurteilung liefern, ob der Scanner kalibriert ist oder nicht.

Bei der Segmentierung hat sich die Erkennung des Adamsapfels als Problem erwiesen. Auch wenn dieser Punkt bei manuellen Vermessungen sehr gut ertastet werden kann, ist die automatische Auffindung fast unmöglich. Nur bei wenigen Probanden ist dieses Merkmal in den 3D-Daten hinreichend ausgeprägt. Fragen, die sich stellen, sind:

- Kann auf Grund anderer Merkmale die Position des Adamsapfels gefunden werden?
- Gibt es andere Kriterien, die das Ende des Kopf-Nacken-Segmentes beschreiben?
- Oder ist es unumgänglich, an dieser Stelle einen Marker zu verwenden?

Ebenso stellte sich die Segmentierung der Schultern für die Automatik als problematisch heraus. Der Verlauf der Trennkurven zwischen den Schultersegmenten und abdomino-thorakalem Segment wurde in dieser Arbeit wegen der einfacheren Verarbeitung als quadratisch angenommen. Kurven höherer Ordnung, mehrere Kurvensegmente oder Splines mögen vielleicht bessere Ergebnisse liefern. Das gilt sinngemäß auch für den Beckenbereich. Auch hier wurden teilweise Kurvenverläufe als quadratisch angenommen um die Verarbeitung zu vereinfachen.

Sowohl die entwickelten Algorithmen als auch die Überlegungen zur Genauigkeit basieren auf den Rohdaten des Scanners. Ziel einer weiterführenden Arbeit könnte die Untersuchung sein, ob Interpolationen zwischen den Scanpunkten die Genauigkeit der Auswertung verbessern können. Der Ansatzpunkt ist, dass einfache Mittelwertbildungen nicht zielführend sind. Es müsste vielmehr die Körperkontur über mehrere Scanpunkte verfolgt und der Schnittpunkt dieser Konturverläufe zwischen zwei Scanpunkten gefunden werden. Damit könnte die Genauigkeit der Position von wichtigen Orientierungspunkten erhöht werden. Dies gilt nicht nur für die Punkte einer Ebene sondern auch für die Scanebenen selber (vgl. Abb. 10).

Nachdem die Segmentierung einmal automatisch durchgeführt wurde, liegt es natürlich nahe, die Parameterwerte (siehe Tabelle 1) jedes Segmentes zu berechnen. Es wäre zu untersuchen, welche Voraussetzungen, welche Eingangsparameter etc. dafür notwendig sind. Mit Hilfe geeigneter Annahmen und mathematischer Mittel lassen sich die Segmente sicher in ihre Teilsegmente zerlegen und daraus die geforderten Parameterwerte ermitteln.

In [10] wird eine automatische Datenerfassung beschrieben, die zur Eingrenzung der Lage der Orientierungspunkte eine Tabelle mit Körperproportionen verwendet. Diese Tabelle basiert auf Vermessungen von Probanden im asiatischen Raum. Eine ähnliche Tabelle wurde im Rahmen des Projekts CAESAR [14] für die nordamerikanische und europäische Bevölkerung erstellt. Die hohen Kosten der Datenbank (US$ 10.000,-) verhinderten die Einbindung in diese Diplomarbeit. Eine ähnliche, kostengünstigere Version stand nicht zur Verfügung. Ein interessantes Projekt wäre entweder die Erstellung einer Proportionendatenbank oder die Untersuchung, ob solche Datenbanken für den Forschungsbereich (kostengünstig) zur Verfügung stehen und wie sie zur Auswertung in diese Arbeit eingebunden werden können.

Anhang 1 Modulbeschreibung

Nachfolgend eine Auflistung aller Module und deren Funktionsweise. Siehe auch die mitgelieferte JavaDoc.

Generell gilt: `get/set...()` sind getter- und setter-Funktionen für die jeweiligen privaten Werte eines Objektes. Sie werden daher nicht näher beschrieben.

`Constants.java`

Dieses File enthält Konfigurationskonstanten, mit deren Hilfe die Segmenterkennung in gewissen Grenzen manipuliert werden kann. Zum Beispiel geschieht die Erkennung der Kopfgröße dadurch, dass sich der X-Wert des Torso-Mittelpunktes innerhalb einer bestimmten Anzahl an Ebenen um einen bestimmten Mindestwert ändert. Beide Werte wurden empirisch aus den vorliegenden Testscans ermittelt. Es ist zu erwarten, dass weitere Scans eine Anpassung dieser Werte erfordern.

`START_ANGLE`	Dieser Winkelwert gibt die erlaubte Abweichung zum Nullpunkt an um ihn noch als diesen zu identifizieren.
`MIN_POINTS`	Wie viele Vertices in einer Liste vorhanden sein müssen, damit der Mittelpunkt errechnet wird.
`MIN_LEVEL_POINTS`	Wie viele Vertices in einem ScanLevel vorhanden sein müssen, damit er als gültig erkannt wird.
`MAX_DISTANCE`	Punkte, deren Abstand diesen Schwellwert unterschreitet, werden zu einem Punkt verschmolzen.
`SINGLE_DISTANCE`	In einem ASC-File ist keine Face-Information vorhanden. Einzelne Punkte können daher nur erkannt werden, wenn der kleinste Abstand zum nächsten Punkt größer als diese Distanz ist.

Die Kopferkennung geschieht dadurch, dass die Änderung der X-Koordinate des Körpermittelpunktes innerhalb einer bestimmten Anzahl an Scanebenen einen bestimmten Wert überschreitet.

Die nachfolgenden Konstanten beschreiben diese Änderung:

`HEAD_LEVELS`	Die Anzahl der Ebenen.
`HEAD_DX`	Die Änderung der x-Koordinate in mm
`HEAD_START`	Die Kopferkennung wird ab dieser Ebene begonnen…
`HEAD_END`	… und an dieser Ebene gestoppt.
`FINGER_LENGTH`	Die Suche nach dem Handgelenk wird diese Anzahl an Ebenen vor dem Armende gestoppt.
`FEET_LEVELS`	Dieser Wert wird für die Ober-/Unterschenkel-Trennung als Höhe der Füße angenommen. Die tatsächliche Fußhöhe wird erst nachher berechnet.

`ARM_OFFSET_START`	Wie viele Ebenen unter dem erkannten Schulterbeginn begonnen wird, die Arme zu suchen.
`ARM_OFFSET_END`	Wie viele Ebenen vor dem Scan-Ende aufgehört wird, die Arme zu suchen.
`ARM_DIST`	Die Distanz zwischen Arm und Torso.
`LEG_Y`	In diesem Bereich um den 0-Wert der y-Achse werden die Beine erkannt.
`ARM_STOP_Y`	Die verfeinerte Armsuche wird so viele mm vor der Körpermitte gestoppt.
`UMBI_LEVELS_UP`	Bestimmt die Anzahl Ebenen über/unter dem anterioren Punkt an der Vorderseite des Körpers (Bauch) innerhalb denen der Nabel gesucht wird.
`UMBI_DELTA_IN`	Die Tiefe des Nabels zum anteriorem Punkt an der Vorderseite.
`UMBI_DELTA_OUT`	Die Tiefe des Nabels zur angrenzenden Körperumgebung.
`PELVIC_MIN_DX`	Die Tiefe der Hautfalte im Schritt zum Oberschenkel.
`PELVIC_MIN_LEV`	Die Mindestanzahl an Ebenen, die die Hautfalte verfolgt wird.
`USE_AXIS_FOR_LENGTH`	Die Länge der Segmente wird durch die Differenz der z-Wert der Start- und Endebene berechnet. Wird diese Konstante auf „true" gesetzt, wird der Abstand der Mittelpunkt der Ebenen gemessen. Das ist etwas aufwändiger, führt aber zu keinen genaueren Ergebnissen.
Die Konstanten `SEG_..._X/Y/Z`	werden für die Explosionsansicht der Ausgabe verwendet. Sie bestimmen die Lage jedes Segmentes zur ursprünglichen Lage und können nach Bedarf angepasst werden.

Alle nicht beschriebenen Konstanten (ab dem Kommentar „SYSTEM CONSTANTS" beschreiben Indizes und Array-Größen und dürfen nur verändert werden, wenn das Programm an sich verändert werden muss (z.B. bei einer Erweiterung des Modells auf mehr als 17 Segmente).

`VertexObj.java`

Beschreibt die Eigenschaften eines Vertex (eines Scanpunktes). Das sind:

- x/y/z-Koordinate
- r/g/b-Wert der Farbe (falls vorhanden)
- Winkel und Radius für die Darstellung in Polarkoordinaten
- `faces` – Die Faces an denen dieses Vertex beteiligt ist.
- `interpolated` – Ob dieses Vertex bei der Scanlochfüllung errechnet wurde.
- `index` – Die Indexnummer dieses Vertex im eingelesenen File.

`compareTo()` ist eine Vergleichsfunktion um Vertices nach ihrem Winkel sortieren zu können. Sie wird gebraucht, wenn das VertexObj die Klasse Comparable erweitert.

Für andere Sortierungsfunktionen gibt es:

`compare()` - sortiert ebenfalls nach dem Winkel
`Y_order` – sortiert Vertices nach der Y-Koordinate (von links nach rechts)
`X_order` – sortiert Vertices nach der X-Koordinate (von hinten nach vorne)

`isEqual` – ermöglicht die Aussage, ob zwei Vertices gleich sind. Das ist dann der Fall, wenn die x, y und z-Koordinate übereinstimmen.

`VertexObj()` - Verschiedene creator-Funktionen (mit x,y,z-Koordinate, mit Radius und Winkel etc.) Die Funktionen werden an verschiedenen Stellen zur Erzeugung eines VertexObj verwendet und wurden an den Bedarf angepasst.

`toPolar()` - Berechnet die Polarkoordinaten aus den x/y/z-Koordinaten bezogen auf einen Mittelpunkt (wird übergeben).

`toCartesian()` - Wandelt die Polarkoordinaten in x/y/z-Koordinaten, bezogen auf einen Mittelpunkt (wird übergeben).

`addFace()` - Fügt ein FaceObj in die interne faces-Liste des VetexObj ein.

`dumpFaces()` - Erzeugt einen lesbaren String in dem alle Faces dieses VertexObj gelistet sind. Dies wurde zum Debuggen des Programmes gebraucht.

`hasCommonFace()` - Vergleicht die faces-Liste dieses VertexObj's mit der eines übergebenen und gibt `TRUE` zurück, wenn beide VertexObjekte mindestens ein gemeinsames Face haben (also beide an einer Fläche beteiligt sind).

`VertexList.java`

Implementiert eine ArrayList aus VertexObjekten und vereinfacht die Schreibweise im Programm. Außerdem werden Funktionen zur Verfügung gestellt, die auf alle Objekte einer VertexListe angewandt werden.

`toPolar()` - Wandelt alle VertexObjekte in einer VertexListe in Polarkoordinaten um.

`toCartesian()` – Wandelt alle VertexObjekte in einer VertexListe in kartesische Koordinaten um.

`setColor()` - Setzt eine bestimmte Farbe (RGB-Wert) in alle VertexObjekte einer Liste.

`meltVOs()` - Verschmilzt alle VertexObjekte einer Liste, die näher als einen bestimmten Abstand zusammenliegen (Constants.MAX_DISTANCE). Die Koordinaten dieses neuen Punktes sind der Mittelpunkt der verschmolzenen Punkte. Es gibt die Anzahl der verschmolzenen Punkte zurück.

`meltAllVOs()` - Verschmilzt alle VertexObjekte einer Liste, die näher als einen bestimmten Abstand (Constants.MAX_DISTANCE) zusammenliegen zu einem einzelnen neuen Punkt. Die Koordinaten dieses neuen Punktes sind der Mittelwert der Koordinaten der verschmolzenen Punkte. Die Routine gibt die Anzahl der verschmolzenen Punkte zurück.

`splitFrontRear()` - Teilt eine Liste in einen Vorder- und einen Rückteil auf. Die Listen, die Vorder- bzw. Rückseite aufnehmen, müssen vor Aufruf dieser Methode erzeugt und dann übergeben werden. Diese Methode zieht eine Trennlinie zwischen den äußersten VertexObjekten (ganz links und ganz rechts). Alle Objekte deren X-Wert größer als der interpolierte X-Wert an der Y-Koordinate des Objektes ist, werden nach vorne sortiert, alle anderen nach hinten.

Die Methode gibt den k-Wert (Steigung) der Trennlinie zurück. Dies erwies sich im Programm als nützlich.

`FaceObj.java`

Ein `FaceObj` beschreibt die Eigenschaften einer Fläche, also wie bestimmte Punkte (Vertices) zusammenhängen. Diese Information wird nur dann erzeugt, wenn ein .obj-File eingelesen wird. In einem .asc-File gibt es diese Daten nicht.

`vertices` – Die Punkte dieser Fläche.

`index` – Die Indexnummer dieses Faces im eingelesenen File.

`FaceObj()` - Die Konstruktoren, wahlweise mit einem übergebenen Index.

`addVertex()` - Füge ein VertexObj zur Liste hinzu.

`FaceList.java`

Eine ArrayList aus `FaceObj` – zur Vereinfachung der Schreibweise. Außerdem könnten Methoden, die für alle Objekte einer `FaceList` benötigt werden, hier eingefügt werden.

`zObj.java`

Ein `zObj` entspricht einer Scanebene, d.h. enthält alle Vertices mit gleichem z-Wert. Das `zObj` wird in zwei Zusammenhängen verwendet:

1.) Für den Scan des gesamten Körpers. In diesem Falle gibt es VertexListen für jedes einzelne Segment, d.h. es wird angegeben welche Vertices dieser Ebene in welchem Segment liegen.

2.) Für ein SegmentObjekt. Hier gibt es nur eine VertexListe, die alle Vertices einer Ebene für dieses Segment enthält.

`zPos` – Die Z-Koordinate dieser Ebene.

`levelNr` – Eine Indexnummer der Ebene. Beginnt bei 0 für die oberste Ebene des Scans und enthält ansteigende Werte nach unten hin.

`vertexList` – Alle Vertices dieser Ebene.

`distance` – Der Abstand zur vorigen Ebene in mm.

`envelopeList` – Der Umfang einer Ebene wird mit Hilfe eines „virtuellen Maßbandes" gemessen. D.h. Punkte, die konkav zum Mittelpunkt der Ebene liegen, werden übersprungen. Es wird also über den Nabel gemessen und nicht in den Nabel hinein.

`meanX` – Mittelpunkt der Ebene, X-Koordinate.

`meanY` – Mittelpunkt der Ebene, Y-Koordinate.

`area` – Berechnete Fläche der Ebene.

`meanVert2[]` - Vertexlisten, aufgeteilt nach den Segmenten in dieser Ebene.

`meanX2[]` - Mittelpunkt der Ebene, aufgeteilt nach Segmenten, X-Koordinate.

`minX[]` /`maxX[]` - Kleinster bzw. größter X-Wert jedes Segmentes in dieser Ebene.

`meanY2[]` - Mittelpunkt der Ebene, aufgeteilt nach Segmenten, Y-Koordinate.

`minY[]` /`maxY[]` - Kleinster bzw. größter Y-Wert jedes Segmentes in dieser Ebene.

`meanAngle[]` - Der mittlere Winkelabstand zwischen zwei Punkten, aufgeteilt nach Segmenten und bezogen auf deren Mittelpunkt.

`meanRadius[]` - Der mittlere Radius aller Punkte eines Segmentes, bezogen auf den Mittelpunkt dieses Segmentes.

`minRadius[]` / `maxRadius[]` - Der kleinste und größte Radius jedes Segmentes.

`leftArmY` / `rightArmY` – Zwischen Torso und Arm gibt es normalerweise eine Lücke. ArmY enthält die Y-Koordinate des Mittelpunktes dieser Lücke für den linken

bzw. rechten Arm.

`leftArmYF` / `leftArmYR` / `rightArmYF` / `rightArmYR` – Im Bereich der Achsel verläuft die Lücke nicht parallel zur X-Achse (hinten-vorne). Daher wird in diesem Bereich die Y-Koordinate getrennt für die Vorderseite (Front – F) und Rückseite (Rear – R) errechnet.

`legY` – Y-Koordinate des Mittelpunktes der Lücke zwischen den Beinen.

`Konstruktoren` – Erzeugen ein `zObj` mit verschiedenen Ausgangswerten (leeres Objekt, mit z-Koordinate und dem ersten VertexObjekt oder mit z-Koordinate und der VertexListe der Ebene). In jedem Fall werden die Mittelwerte gelöscht.

`clear()` – Löscht alle VertexListen.

`clearMeans()` – Löscht die berechneten Mittelwerte.

`addVertex()` - Hängt ein VertexObjekt an die Liste aller Vertices dieser Ebene an.

`calcMean()` - Berechnet die Mittelwerte aller Segmente dieser Ebene. Diese Funktion teilt die Vertices dieser Ebene auch in die verschiedenen Hauptsegmente (Torso, Arme und Beine) auf. Weiters werden die Oberflächen geglättet, Punkte, deren Abstand unter `Constants.MAX_DISTANCE` liegt, miteinander verschmolzen, Scanlöcher aufgefüllt und Überlappungen korrigiert. Die Mittelpunkte werden abschließend neu berechnet und die Polarkoordinaten aller Vertices ermittelt.

`calcMeanAll()` - Berechnet die Mittelwerte für die VertexListe mit allen Vertices dieser Ebene und die Polarkoordinaten aller Vertices. Diese Methode löscht alle Mittelwerte, nicht nur den für die Gesamtliste.

`calcMean()` - Berechnet nur die Mittelwerte für die VertexListe mit allen Vertices dieser Ebene. Es werden keine weiteren Berechnungen durchgeführt.

`calcMeanList2()` - Berechnet die Mittelwerte einer übergebenen VertexListe, nicht einer im `zObj` gespeicherten. Es kann eine bestimmte Mindestanzahl an Punkten vorgegeben werden. Wird diese Anzahl unterschritten, wird keine Berechnung durchgeführt. Die Mittelwerte werden im `zObj` gespeichert. Der Index des Segmentes wird auch übergeben.

`getLimbY()`/`setLimbY()` - Sind getter/setter für die Y-Werte der Lücken zwischen Armen und Torso bzw. zwischen den Beinen. Für die Berechnungen in den anderen Modulen hat es sich als zweckmäßig erwiesen, nicht für jeden Y-Wert ein eigenes getter/setter-Paar zu erzeugen, sondern sie über einen `index`-Wert anzusprechen.

`meanExists()` - Gibt `TRUE` zurück, wenn für dieses Segment ein Mittelwert berechnet wurde (ungleich 0).

`smoothPolar()` - Glättet die Oberflächen, basierend auf den Polarkoordinaten der übergebenen VertexListe. Es wird ein einfacher Glättungsalgorithmus verwendet, indem der Radius durch den Mittelwert der Radien der beiden benachbarten Vertices ersetzt wird. Dies setzt natürlich voraus, dass die Liste vorher nach Winkeln sortiert wurde.
Der nächste Nachbar für das letzte VertexObjekt in der Liste ist das erste und umgekehrt („wrap around").

`calcCircumPolar()` - Berechnet den Umfang eines Segmentes basierend auf den Polarkoordinaten der entsprechenden VertexListe. Die Polarkoordinaten werden dazu zuerst in kartesische umgewandelt und dann einfach die Distanz zwischen zwei benachbarten Vertices aufaddiert.

`checkForHoles()` - Versucht Scanlöcher in einem Segment zu finden: Aus den Winkelabständen aller Punkte wird ein Mittelwert gebildet. Überschreitet der Winkelabstand zweier Punkte das Zweifache dieses Mittelwertes, wird ein Scanloch angenommen. Es werden dann Punkte mit dem mittleren Winkelabstand eingefügt, der Radius wird linear zwischen den beiden Eckpunkten interpoliert.

`removeSingle()` - Entfernt einzelne Punkte. Die verwendete Methode hängt vom eingelesenen Dateiformat ab. Für .obj-Files werden alle Punkte entfernt, die keiner Oberfläche (Face) angehören. Für .asc-Files wird der Minimalbstand jedes Punktes zu allen anderen Punkten dieser Ebene ermittelt. Überschreitet er einen bestimmten Wert (`Constants.SINGLE_DISTANCE`), wird der Punkt entfernt. Die zweite Methode ist zeitaufwändiger, da alle Punkte mit allen anderen verglichen werden müssen. Daher ist die Verwendung des .obj-Fileformates vorzuziehen.

`calcVarValues()` - Diese Methode dient zur Berechnung verschiedener statistischer Werte einer Ebene. Die Werte können dann in anderen Modulen verwendet werden. Die Werte sind:

mittlerer Winkel
maximaler Radius

Diese Methode gibt es für ein bestimmtes Segment oder für alle Segmente einer Ebene.

`correctOverlap()` - Durch das Auffüllen von Scanlöchern im Bereich der Oberarme bzw. Oberschenkel kann es zu Überlappungen kommen. D.h. es werden Punkte mit einem Radius derartig erzeugt, dass sie innerhalb des benachbarten Segmentes zu liegen kommen. Diese Methode ersetzt solche Punkte, indem die Y-Koordinaten der überlappenden Teile gemittelt werden.

`createEnvelope()` - Erzeugt eine Hüllkurve um die Vertices einer Vertexliste. Diese Kurve ist so beschaffen, dass jeweils drei benachbarte Vertices konvex zum Mittelpunkt der Kurve liegen („virtuelles Maßband"). Wenn der Umfang eines Körperteils gemessen wird, wird das Maßband normalerweise über Körpervertiefungen wie den Nabel gespannt.

Diese Methode untersucht drei benachbarte Vertices (drei Vertices, die in der Vertexliste unmittelbar aufeinander folgen, wenn diese Liste nach dem Winkel der Polarkoordinaten sortiert ist). Liegen diese Vertices konvex zum Mittelpunkt, wird um ein Vertex weiter gesucht. Wenn das nicht der Fall ist, wird das mittlere Vertex aus der Liste entfernt und ein Vertex zurückgegangen.

`isConvex()` - Untersucht, ob drei Vertices konvex zum Mittelpunkt des untersuchten Segmentes liegen. Dazu werden zwei Geraden errechnet. Eine zwischen dem ersten und dem dritten Vertex, eine zweite zwischen dem zweiten Vertex und dem Mittelpunkt. Liegt der Schnittpunkt zwischen den beiden Geraden näher zum Mittelpunkt als der Abstand des zweiten Vertex (Radius der Polarkoordinaten), dann verläuft die Kurve konvex.

`getVO()` - Liefert des `VertexObj` an einer bestimmten Position in einer gewünschten Liste zurück. Diese Methode behandelt auch „wrap around" Fälle, liefert also die Position `index mod` (Länge der Liste).

`toPolar()` - Berechnet die Polarkoordinaten alle Punkte der Vertexliste dieses zObjekts bezogen auf den Mittelpunkt der Liste.

`calcArea()` - Berechnet die Größe der Fläche, die die Vertexliste aufspannt. Die Vertexliste wird nach den Winkeln der Polarkoordinaten sortiert. Damit liegen zwei aufeinander folgende Vertices nebeneinander. Zusammen mit dem Mittelpunkt spannen sie ein Dreieck auf, dessen Fläche mit dem Satz nach Heron berechnet werden kann. Alle Dreiecksflächen aufsummiert ergeben die Gesamtfläche der Vertexliste.

`zList.java`

Definiert eine `ArrayList` aus `zObj`'s. Dies dient hauptsächlich der besseren Lesbarkeit und erlaubt die Implementierung von Funktionen, die ganze `zObj`-Listen betreffen:

`calcVolume()` - Berechnet das Volumen einer `zList`.

`getVolume()` - Liefert ein zuvor mit `calcVolume()` berechnetes Volumen zurück.

`SegmentObj.java`

Das SegmentObjekt enthält alle Daten, die ein Segment beschreiben. Dies sind:

`zLevels` – Eine Liste von `zObj`'s, die dieses Segment enthält.

`volume` – Das Volumen dieses Segments.

`length` – Eine Segmentlänge. Bei manchen Segmenten werden im Hatze-Modell zwei Längen angegeben. Hier muss also eine zu vergleichende Länge ausgewählt werden.

`lenName` – Der Name dieses Längenmaßes.

`addZObj()` - Fügt ein zObj in dieses Segment ein.

`addZVL()` - Fügt eine Vertexliste mit einer bestimmten z-Koordinate in das SegmentObjekt ein. Existiert ein `zObj` mit dieser Koordinate bereits, wird die Vertexliste an die Vertexliste des bestehenden `zObj`'s angehängt. Ansonsten wird ein neues zObj erzeugt und die Vertexliste an dieses angehängt.

`getVL()` - Liefert die Vertexliste dieses Segments zurück. Dazu wird eine neue Liste erzeugt und die Vertexlisten aller `zObj`'s an diese Liste angehängt.

`getFrontVL()` - Liefert die Vertexliste dieses Segments zurück mit allen `VertexObj`'s, die an der Vorderseite des Meshes liegen.

`clear()` - Löscht die zObjekte dieses Segmentes.

`calcMean()` - Berechnet die Mittelpunkte aller `zObj`'s dieses Segmentes.

`toPolar()` - Berechnet die Polarkoordinaten aller Vertices dieses Segmentes, bezogen auf den Mittelpunkt des entsprechenden `zObj`'s.

`calcVolume()` - Berechnet das Volumen des Segmentes.

`MeshObj.java`

Ein MeshObjekt fasst alle Daten zusammen, die einem eingelesenen Scan entsprechen.

`vertexList` – Eine Liste aller eingelesenen Vertices. Die Elemente dieser Liste werden grundsätzlich nicht entfernt, allerdings werden die Werte im Zuge der Datenbereinigung verändert.

`faceList` – Eine Liste aller eingelesenen Faces. Diese Liste wird lediglich für die Information verwendet, ob es Vertices gibt, die keinem Face angehören.

`vertexCount` – Anzahl aller eingelesenen Vertices.

`faceCount` – Anzahl aller eingelesenen Faces.

`zList` – Alle eingelesenen Vertices werden auf Grund ihres z-Wertes entsprechenden `zObj`'s zugeordnet. Um festzustellen, ob eine `zObj` für einen

bestimmten z-Wert bereits existiert und um die Sortierung zu erleichtern, wird für die Speicherung der `zObj`'s eine Hashtabelle verwendet. Der Schlüssel (`key`) ist dabei der z-Wert.

`sortZ` – Ist eine sortierte Tabelle der in zList verwendeten z-Werte, um einen Scan in aufsteigender Reihenfolge durchsuchen zu können.

`min_x, min_y, min_z,`
`max_x, max_y, max_z` – Geben die kleinsten bzw. größten x/y/z-Werte aller eingelesenen Vertices an. Dies ist die sog. *boundary box* des Meshes und wird für die Zentrierung verwendet. Nachher liegen diese Werte i. A. symmetrisch um den 0-Punkt.

`result` – Ein Results-Objekt mit allen Ergebnissen.

`startLevel, endLevel` – Erste bzw. letzte verwendete Ebene des Scans. Diese Werte sind bereits um nicht verwendbare Ebenen (die zu wenige Vertices enthalten) bereinigt.

`segObjs` – Die Segmente dieses Meshs.

`pathName, fileName` – Pfad und Name der eingelesenen Datei.

`MeshObj()` - Konstruktor mit Filenamen. Wird dieser Konstruktor verwendet, erzeugt er ein `inFile`-Objekt, welches wiederum das angegebene File einliest und die Daten in das `MeshObj` einfügt.

Grundsätzlich unterstützt die Software zwei Arten, ein `MeshObj` zu erzeugen und mit Daten zu füllen:

1.) Den Konstruktor mit Filenamen.
2.) Den normalen Konstruktor und dann das Einlesen von Daten mittels des InputFile-Objektes.

Welche Version verwendet wird, hängt vom Einsatzzweck der Software bzw. der Module ab. Werden künftig PlugIns benötigt oder geschrieben, ist die 2. Variante besser, denn wie das `MeshObj` schlussendlich mit Daten befüllt wird, hängt vom Plugin ab.

`addVertex()` - Fügt ein eingelesenes Vertex in die `vertexList` und in das entsprechende zObj ein, wobei eventuell ein neues `zObj` erzeugt wird. Erhöht den `vertexCount` und aktualisiert die boundary box. Nach dem Einlesen aller Vertices ist die boundary box auf dem aktuellen Stand.

`addFace()` - Fügt ein eingelesenes Face in die `FaceList` ein und erhöht den `faceCount`.

`cleanUp()` - Führt die Datenbereinigung durch:

- Aus jeder Ebene werden einzelstehende Vertices entfernt
- Alle Ebenen, die zu wenige Vertices enthalten, werden entfernt
- Alle Vertices, die näher als einen Minimalabstand nebeneinander liegen, werden verschmolzen und die Koordinaten des resultierenden Vertex durch den Mittelwert der Koordinaten der verschmolzenen Vertices ersetzt.
- Aktualisierung der boundary box mit Zentrierung des Meshes. Danach liegt der Mittelpunkt des Meshes etwa bei 0/0/0 und die boundary box liegt symmetrisch dazu.
- Alle zObj-Werte inkl. der sortierten Liste werden aktualisiert.

`analyze()` - Die Hauptanalyse-Methode. Sie wird nach dem Einlesen des Meshes aufgerufen. Alle erforderlichen Werte werden hier berechnet.

- Stellt zunächst fest, wie viele Scanebenen gültig sind und setzt die Höhe des Scans.
- Kopferkennung (`findHead()`)
- Schultererkennung (`findShoulders()`)
- Armerkennung (`findArms(), refineArms()`)
- Schulterblatterkennug (`findAcro()`)
- Für die weiteren Erkennungen werden u.a. die Mittelwerte der bisher erkannten Ebenen und Segmente benötigt, deshalb werden hier die SegmentObjekte erzeugt sowie mit `calcMean()` die Mittelwerte aller bisher erkannten Segmente berechnet.
- Abspaltung von Unterarmen und Händen aus dem Armsegment (`calcForeArms()`).
- Abspaltung von Unterschenkeln und Füßen aus dem Fußsegment (`calcLegs()`).
- Berechnung der Schultersegmente (`calcLeftShoulder(), calcRightShoulder()`)
- Nabelerkennung und Aufteilung der abdomino-thoractic und abdomino-pelvic Segmente (`calcUmbi()`).
- Verfeinerung der Kopferkennung (`refineHead()`)
- Verfeinerung der Beckensegmentierung (`calcPelvic()`)
- Ausgabe aller Berechnungen: Ausgabe von Werten in der Java-Konsole, Ausgabe des exploded views, sowie Ausgabe der Segmentlisten in ein Textfile.

Die Benennung der Methoden erfolgte nach folgendem Schema:

`find...()` führt eine erste Grobunterteilung durch
`calc...()` teilt die Grobsegmente in die finalen Segmente auf
`refine...()` korrigiert bereits durchgeführte Ergebnisse mit neuen Erkenntnissen.

`findHead()` - Sucht, beginnend am oberen Ende des Scans, nach dem Kopf. Das Kopfende ist durch den Adamsapfel definiert. Dieser ist jedoch ohne weitere Hilfe nicht auffindbar. Daher wird nach einer signifikanten Änderung des Mittelpunktes jeder Ebene gesucht. Das geschieht am Kopfende beim Halsbeginn. Ändert sich die X-Koordinate des Mittelpunktes über eine Anzahl an Ebenen um einen bestimmten Schwellwert, wird das Kopfende erkannt. Beide Werte können in den `Constants`

eingestellt werden.

`refineHead()` - Nach der Baucherkennung wird das Kopfende korrigiert. Dazu wird der am weitesten innen liegende (posteriorer) Punkt an der Vorderseite zwischen bisherigem Kopfende und Bauchebene gesucht. Das neue Kopfende wird auf halber Distanz zwischen altem Kopfende und diesem posterioren Punkt definiert. Bei den vorliegenden Scans hat sich diese Annahme als gute Näherung herausgestellt. Es muss jedoch angemerkt werden, dass hier weitere Arbeiten zur Verifizierung oder Falsifizierung dieser Annahme notwendig sind.

`findShoulders()` - Dazu wird die Silhouette in der Frontansicht verfolgt. Der Schulterbeginn befindet sich dort, wo die Tangente an die Silhouette vom Kopfende abwärts eine 45 Grad Steigung überschreitet. (D.h. die Steigung k den Wert von 1 überschreitet.)

`findAcro()` - Nach dem Übergang in die Schultern ist $k < 1$. Das Schulterblatt (Acromion) wird dort erkannt, wo der k-Wert wieder über 1 steigt.

`findArms()` - Zwischen Schulterbeginn und dem Ende (das Viertel der Distanz Schulter – Scanende über dem Scanende liegt) wird nach einer Lücke links und rechts von der Mittelachse des Körpers gesucht. Liegt diese Lücke etwa um die Mittelachse, wird der Beinbeginn festgestellt.

`refineArms()` - In der Achselgegend kann keine direkte Lücke festgestellt werden. Hier verläuft sie an Vorder- und Hinterseite des Körpers unterschiedlich. Daher wird die Suche aufgeteilt. Außerdem kann es einzelne Ebenen geben, in denen keine Lücke festgestellt werden kann. Zwischen dem bisherigen Armbeginn und der letzten Ebene mit Lücke wird der Verlauf der Lücke interpoliert.

`findMaxX()` - Diese Methode stellt den anterioren und posterioren Punkt an der Rückseite sowie den anterioren Punkt an der Vorderseite des Torsos fest.

`interExtend()` - Interpoliert Werte zwischen zwei Ebenen für ein gegebenes Segment. Der letzte Wert kann um eine Anzahl Ebenen weitergeführt bzw. ignoriert werden.

`calcCircumEnv()` - Berechnet den Umfang eines Körperteils mit Hilfe einer Envelope-Funktion (virtuelles Maßband).

`getMinYforLevel()` - Liefert den kleinsten Y-Wert einer gegebenen Ebene.

`getMaxYforLevel()` - Liefert den größten Y-Wert einer gegebenen Ebene.

`getMinVOYforLevel()` - Liefert das `VertexObj` mit dem kleinsten Y-Wert einer gegebenen Ebene.

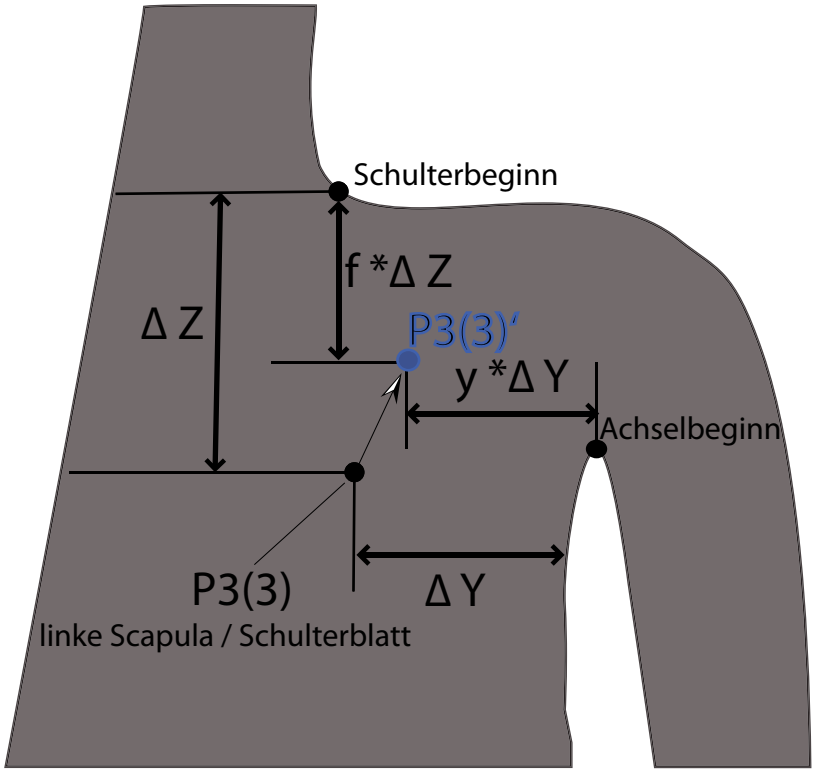
`getMaxVOYforLevel()` - Liefert das `VertexObj` mit dem größten Y-Wert einer gegebenen Ebene.

`calcForeArms()` - Sucht zwischen Beginn und Ende des linken Arms nach der Grenze Ober-/Unterarm und dem Handbeginn. Die Grenze Ober-/Unterarm wird durch den posterioren Punkt an der Vorderseite des Armes erkannt.

Der Handbeginn wird durch den kleinsten Umfang zwischen Unterarmbeginn und einer fixen Distanz vor dem Armende erkannt. Hier muss die Erkennung rechtzeitig vor den Fingern gestoppt werden.

`calcLeftShoulder()/calcRightShoulder()` - Die Grenze des Schultersegments zum abdomino-thorakalen Segment wird durch eine Kurve zwischen Schulterbeginn und Achsel definiert. Diese Kurve sollte an der Rückseite durch das Schulterblatt (posteriorer Punkt des Oberkörpers) verlaufen.

Obwohl dieser posteriore Punkt recht gut festgestellt werden kann, liefert das Ergebnis keine brauchbaren Daten. Daher wurden zwei Faktoren eingeführt, die das Delta-Z zwischen Schulterblatt und Schulterbeginn bzw. das Delta-Y zwischen Schulterblatt und Achselbeginn verändern können. Diese korrigierten Deltas legen einen dritten Punkt (P3(3)' für die linke Schulter und P7(3)' für die rechte Schulter) fest, der eine quadratische Interpolation der Trennkurve erlaubt (vgl. Abb. 29).



f: `Constants.SHOULDER_FZL` (`_FZR` für P7(3) rechts)

y: `Constants.SHOULDER_FYL` (`_FYR` für P7(3) rechts)

Abbildung 29 Schultersegment-Interpolationspunkt P3(3)'-Berechnung

`calcLegs()` - Unterteilt die Beinsegmente in Ober-/Unterschenkel und Füße. Ähnlich wie bei den Armen gibt es einen markanten Punkt in Kniehöhe. Er kann jedoch nicht als einfacher anteriorer/posteriorer Punkt festgestellt werden. Der Bereich zwischen Beinbeginn und einem Punkt über den Füßen (der durch eine bestimmte Anzahl Ebenen vom Scanende definiert wird) wird gedrittelt. Das mittlere Drittel enthält die Knieebene. Zwischen dem Beginn und dem Ende dieses Drittels wird an der Rückseite der Beine eine Verbindungslinie gezogen. Der Abstand zwischen dieser Linie und der tatsächlichen Rückseite der Beine erreicht auf Höhe der Knie ein Maximum. Die Füße werden auf Höhe der Knöchel an der Außenseite abgetrennt. Hier erreicht der Radius der Polarkoordinaten ein Maximum. Die Suche wird dabei auf den Quadranten eingeschränkt, der die Knöchel enthält (90-180 Grad für den linken bzw. 180-270 Grad für den rechten Knöchel). Die Außenseite wurde deswegen gewählt, weil sie bei einem Großteil der vorliegenden Scans weniger Scanlöcher enthält.

`calcUmbi()` - Sucht den Nabel. Dieser definiert die Trennlinie zwischen abdomino-thorakalem und abdomino-pelvischen Segment. Von der Bauchebene (anteriorer Punkt der Vorderseite) wird innerhalb einer bestimmten Anzahl Ebenen nach einem lokalen Minimum gesucht. Dazu muss der X-Abstand zwischen dem anterioren Punkt und der untersuchten Ebene einen bestimmten Wert überschreiten, und auch der Abstand zwischen dieser Ebene und einer bestimmten Ebene darüber muss einen bestimmten Wert überschreiten. Die Anzahl der Ebenen sowie die Schwellwerte können im `Constants` angepasst werden. Wird keine passende Ebene über dem Bauch gefunden, wird darunter gesucht, denn der Nabel liegt manchmal über, manchmal unter diesem anterioren Punkt.

`calcPelvic()` - Nach der groben Segmentierung muss das Becken korrigiert werden. An der Vorderseite liegen Punkte der Oberschenkel im bisherigen Beckensegment. An der Rückseite hingegen liegen Punkte des Beckens in den Oberschenkelsegmenten. Der bisherige Beinbeginn definiert eine Ebene. Die Punkte an den Seiten der Schenkel (innen bzw. außen) definieren zwei Stützpunkte. An der Vorderseite liefert die Hautfalte zwischen Becken und Oberschenkel ein Merkmal, das verfolgt werden kann. Ab einer gewissen Ebene kann diese Falte wegen mangelnder Tiefe nicht mehr zuverlässig verfolgt werden, aber sie liefert an der letzten gültigen Stelle einen dritten Stützpunkt. Durch die so gefundenen drei Stützpunkte wird eine quadratische Interpolation durchgeführt, die Punkte aus dem Becken in die Oberschenkel transferiert. Ähnlich kann die Hautfalte an der Rückseite zwischen Gesäß und Oberschenkel verfolgt werden und liefert ebenfalls einen 3. Stützpunkt zur Überführung von Punkten aus den Oberschenkelsegmenten in das Beckensegment.

`calcSegLen()` - Berechnet die Länge der Segmente. Für jedes einzelne Segment wird die z-Distanz zwischen Start- und Endebene berechnet. Für die Füße gilt jedoch die x-Distanz. Hier wird in den Fußebenen der anteriore und der posteriore Punkt gesucht und die Differenz der x-Koordinaten zwischen diesen Punkten berechnet. Diese Methode ordnet auch jedem Segment den Text des Längenmaßes zu (z.B. „Z1(1)" für das abdomino-thorakale Segment).

`calcLength()` - Berechnet die z-Distanz zwischen zwei Ebenen. Alternativ kann diese Methode auch die Distanz der Mittelpunkte zwischen der Start- und der Endebene messen.

Die folgenden Objekte sind Hilfsobjekte mit Werten und Methoden, die an verschiedenen Punkten der Segmentierungs-Software verwendet werden:

`Results`

`BodySize` – Höhe des Meshs (Größe der Person).

`StartLevel,EndLevel` – Erste und letzte gültige Ebene des Scans.

`HeadLevels` – Kopfende

`HipLevel` – Hüftebene

`WaistLevel` – Taillenebene

`UpperArmLevel` – Vorläufiger Beginn der Arme.

`UpperLegLevel` – Vorläufiger Beginn der Beine.

`LShoulderLevel, RShoulderLevel` – Ebenen des linken bzw. rechten Schulterbeginns.

`LAcroLevel, RAcroLevel` – Ebene des linken bzw. rechten Acromions (Schlüsselbeines).

`BellyLevel` – Bauchebene

`UmbiLevel` – Nabelebene

`TorsoStart, TorsoEnd` – Beginn bzw. Ende des Torsos nach BodyScan (Kopf/Halssegment bzw. Beinbeginn).

`LeftArmStart, RightArmStart` – Beginn der Arme.

`LeftFArmStart, RightFArmStart` – Beginn der Unterarme.

`LeftHandStart, RightHandStart` – Beginn der Hände.

`LeftArmEnd, RightArmEnd` – Ende der Arme (Hände).

`LegStart` – Beginn der Beine.

`LLowerLeg, RLowerLeg` – Beginn der Unterschenkel.

`LFoot, RFoot` – Beginn der Füße.

Der Rest des Moduls enthält Getter- bzw. Settermethoden für jeden Wert.

`Utils.java`

`log()` - Ausgabe von Werten in der Java-Konsole von Eclipse.

`calcDist()` - Berechnet die Distanz zwischen zwei Punkten mit X/Y-Koordinaten.

`calcDist3D()` - Berechnet die Distanz zwischen zwei Punkten mit X/Y/Z-Koordinaten.

`round()` - Rundet einen double-Wert auf eine gewünschte Anzahl an Nachkommastellen.

`calcK()` - Berechnet die Steigung der Geraden zwischen zwei Punkten mit X/Y-Koordinaten. Sollte durch die entsprechende Methode im InterpolObj ersetzt werden.

`InterpolObj.java`

Führt verschiedene Interpolationen durch.

`mx1, mx2, mx3`,
`my1, my2, my3` – Die X/Y-Koordinaten von bis zu drei Stützpunkten.
`k, d` – Die Parameter der Geradengleichung y = k*x + d.
`a, b, c` – Die Parameter der quadratischen Gleichung y = a*x^2 + b*x + c.

`Konstruktoren` – Es gibt zwei Konstruktoren, einen für zwei Stützpunkte, der auch gleich die Parameter der Geraden zwischen diesen Punkten berechnet, und einen für drei Stützpunkte, der auch die quadratische Funktion durch diese drei Punkte berechnet.

`calcK()` - Berechnet die Steigung der Geraden zwischen zwei Punkten mit X/Y-Koordinaten. Statische Methode, die nicht die im InterpolObj gespeicherten Koordinaten verwendet.

`calcMyK()` - Berechnet die Steigung der Geraden zwischen den im InterpolObj gespeicherten Punkten 1 und 2 und speichert den Wert in k.

`calcD()` - Berechnet den Offset der Geraden zwischen den im InterpolObj gespeicherten Punkten 1 und 2 und speichert den Wert in d.

`calcY()` - Berechnet einen Y-Wert auf Grund des übergebenen X-Wertes und den k/d-Parametern (y = k*x + d). (Geradeninterpolation durch 2 Punkte.)

`calcQuadratic()` - Berechnet einen Y-Wert auf Grund des übergebenen X-Wertes und den a/b/c-Parametern (y = a*x^2 + b*x + c). (Quadratische Interpolation durch drei Punkte.)

`MeanObj`

Das `MeanObj` berechnet verschiedene Mittelwerte basierend auf einer Liste an double-Werten.

`ValueList` – enthält die double-Werte

`calcMean()` - Fügt einen double-Wert in die Werteliste ein und liefert den neuen Mittelwert zurück.

`resetMean()` - Löscht die Werteliste dieses Objekts.

`getMean()` - Berechnet den Mittelwert der Werteliste.

`InputFile.java`

Liest ein MeshObjekt ein.

`fFile` – Das Fileobjekt des Datenfiles.

`fileType` – Gibt an, ob ein asc oder ein obj-File eingelesen wurde.

`error` – Fehlercode

`pattern` – Eine Regular-Expression zur Trennung von Werten durch Leerzeichen.

`face_pattern` – Eine Regular-Expression um Faces durch „/" zu trennen.

`mo` – Das MeshObj, das durch dieses File beschrieben wird.

`InputFile()` - Konstruktor mit `MeshObj` und dem `Filenamen`. Wenn dieser `Konstruktor` verwendet wird, wird das angegebene File eingelesen und die Werte im `MeshObj` gespeichert.

`processLineByLine()` - Arbeitet das Datenfile Zeile für Zeile ab.

`processLineOBJ()` - Verarbeitet eine Zeile eines .obj-Files. Es werden nur f- und v-Zeilen verarbeitet. Eine v-Zeile beschreibt ein `VertexObj`, das durch die x, y und z-Koordinaten beschrieben wird. Eine f-Zeile beschreibt ein `FaceObj`.

`processLineASC()` - Verarbeitet eine Zeile eines .asc-Files. Hier werden nur Vertices mit x, y und z-Koordinate und einem optionalen Farbwert (Grauwert) beschrieben.

`processVertex()` - Erzeugt ein neues `VertexObj` und fügt es zum `MeshObj` hinzu. Diese Methode gibt es zweimal, einmal nur für x/y/z-Koordinaten und einmal mit zusätzlichem Grauwert.

`processFace()` - Erzeugt ein neues `FaceObj`, fügt die notwendigen `Vertices` in das `FaceObj` ein und fügt es zum `MeshObj` hinzu.

`OutputFile.java`

Schreibt ein `MeshObj` als .asc-File.

`exp_x, exp_y, exp_z` – Für die Ausgabe einer explodierten Ansicht wird jedes Segment mit einem x/y/z-Offset geschrieben. Diese Offsets werden beim Erzeugen des `OutputFile`-Objekts mit Werten aus `Constants.java` initialisiert.

`OutputFile()` - Konstruktor mit dem `MeshObj`, dem Filenamen und einer Angabe, ob eine normale (`false`) oder eine exploded view (`true`) Ausgabe erfolgen soll.

`writeASC()` - Schreibt alle Vertices des MeshObjekts mit ihren x,y,z-Werten in das Ausgabefile.

`writeExpASC()` - Schreibt alles Vertices jedes Segmentes mit einem x,y,z-Offset in das Ausgabefile.

`OutputPoints.java`

`OutputPoints` arbeitet ähnlich wie `OutputFile`, jedoch werden die Vertices nach Segmenten getrennt. Es wird zunächst eine Zeile mit der Segmentnummer geschrieben, dann folgen die zu dem Segment gehörenden Vertices mit ihren x,y,z-Koordinaten. Abschließend wird eine Leerzeile geschrieben.

`writeTXT`() - Private Methode, die die Daten in eine Textdatei schreibt.

Alternativ wäre die Erweiterung dieses Moduls um andere Methoden denkbar, die verschiedene andere Formate unterstützen (z.B. XML).

Alle `GUI_...`-Module dienen zur grafischen Ausgabe am Bildschirm.

GUI_Drawing.java

erweitert eine `JComponent` zur Darstellung einer Ansicht.

`USE_...` - Konstanten, die beschreiben, wie eine Ansicht einer Vertexliste dargestellt wird. Von den 3 Koordinaten können für eine Ansicht 2 zur Darstellung ausgewählt werden (XY, YZ, XZ oder eine umgedrehte YZ-Ansicht. Dies dient zur Darstellung der Vorderseite anstatt der Rückseite).

`min_x, max_x, min_y, max_y` – Der kleinste bzw. größte darstellbare Wert in einem Drawing.

`ratio_x, ratio_y` – Das Verhältnis tatsächliche zu dargestellten Koordinaten.
`dist_x, dist_y` – Die Breite und Höhe des `Drawings`.

`min_depth, max_depth` - Die kleinste bzw. größte „Tiefe" einer Darstellung zur Einfärbung in verschiedenen Grauwerten und Erzeugung eines 3D-Effektes.

`oval` – Gibt an, ob Kreise oder Punkte zur Darstellung der Vertices verwendet werden sollen.

`mode3D` – Gibt an, ob die Grauwertedarstellung zur Erzeugung eines 3D-Effektes verwendet werden soll oder nicht.

`coordVert, coordHoriz` – Die Bezeichnungen der Koordinatenachsen.

`mode` – Welcher der Ausgabevarianten (`USE_...`) verwendet werden soll.

`points` – Eine Vertexliste mit den Vertices dieses `Drawings`.

`parent` – Das `GUI_Panel`, zu dem dieses `Drawing` gehört.

`GUI_Drawing()` - Konstruktor mit dem `GUI_Panel`, des `oval`-Flags sowie der Achsenbezeichnungen.

`paintComponent()` - Zeichnet alle Vertices in `points`. Aus Geschwindigkeitsgründen gibt es für jede Darstellungsvariante ein eigenes Codesegment.

`Calibrate()` - Es werden die min- und max-Koordinaten jeder Dimension übergeben. Daraus errechnet sich das Verhältnis, mit dem die Koordinaten der Vertexliste für die Darstellung skaliert werden müssen. Außerdem kann der 3D-Modus aktiviert werden.

`drawVertexList()` - Hängt eine Vertexliste an das `Drawing` an und zeichnet alle Punkte. Vorhergehende Listen werden überschrieben.

`drawAddVertexList()` - Die übergebene Vertexliste wird an die im `Drawing`-Objekt vorhandene Liste angehängt und alle Vertices neu gezeichnet.

`GUI_Panel.java`

Fasst die Elemente eines Panels zusammen. In jedem Panel wird ein `MeshObj` dargestellt.

`lstLog` – Das Listenelement zur Ausgabe der berechneten Werte.

`listModel` – Das ListModel für das `lstLog`-Element.

`scrollArea` – Das Panel, das das Listenelement enthält.

`buttonPanel` – Das Panel, das die Bedienelemente enthält.

`lastSelection` – Wird als workaround für ein Listenproblem der Swing-Bibliothek verwendet.

`buttonLoad` – Die Schaltfläche zum Laden eines Scans.

`buttonClose` – Die Schaltfläche zum Schließen eines Panels.

`drwFront, drwVert` – Die beiden Drawings für die beiden Ansichten des Meshes.

`tab` – Das Tab-Element, das das Panel enthält.

`parent` – Das Fenster zu diesem Panel.

`filePath, fileName` – Der Pfad und der Name des Files, das in diesem Panel dargestellt wird.

`mo` – Das Meshobjekt, das in diesem Panel dargestellt wird.

`r_col, g_col, b_col` – RGB-Komponenten der Farben, in denen jedes Segment dargestellt wird.

`GUI_Panel()` - Der Konstruktor erzeugt und initialisiert alle Elemente eines Panels. Hier gibt es auch die Methoden, die bei der Aktivierung einer Schaltfläche aufgerufen werden. Insbesondere gibt es den `ActionListener()` zum Laden eines Files. Er liest ein File ein, ruft die `analyze()`-Methode des zugehörigen Meshobjekts auf und zeichnet die resultierende Segmentierung.

`readFile()` - Zeigt den File-Dialog, erzeugt ein `MeshObj` und liest es ein. Anschließend werden Pfad und Namen des Files ins MeshObj geschrieben.

`loaded()` - Nach dem Einlesen eines Files wird die Schließen-Schaltfläche aktiviert und die Laden-Schaltfläche deaktiviert. Außerdem wird ein Tab mit der Beschriftung „(Neu)" erzeugt.

`drawMO()` - Zeichnet das Meshobjekt dieses Panels.

`GUI_MainWindow.java`

Öffnet das Fenster des Segmentierungsprogramms.

`tab` – Die Pane dieses Fensters.

`GUI_MainWindow()` - Der Konstruktor erzeugt und initialisiert alle Komponenten des Fensters.

`AddTab()` - Erzeugt einen Tab mit der Beschriftung „(Neu)" und fügt ihn in das Fenster ein.

`Main.java`

Das `Main`-Objekt ist der Startpunkt der Segmentierungssoftware und erzeugt das Hauptfenster.

Anhang 2 ANSEPA-Output

ANTHROPOMETRIC DATA RECORD (DATA SEQUENCE AS ON EXPERIMENTAL RECORDING SHEET)

SUBJECT'S NAME: Proband 1 SEX: AGE(YEARS):F 1 MEASURED MASS(KG):6 55.2

IMALE AGE

0. 16.50

ABDOMINO-THORACIC SEGMENT

.220 .274 .257 .251 .252 .235 .263 .202 .115 .103 .142 .168 .187 .199 .206 .185 .182 .186 .182 .395 .217

HEAD-NECK SEGMENT

.125 .187 .199 .054

LEFT SHOULDER

.112 .217 .073 .015

RIGHT SHOULDER

.120 .214 .074 .016

LEFT ARM

.074 .084 .085 .076 .072 .070 .071 .070 .064 .055 .351 .294 .272 .257 .254 .253 .252 .249 .235 .229 .276

RIGHT ARM

.077 .078 .081 .071 .068 .065 .067 .065 .061 .056 .359 .305 .282 .267 .259 .256 .258 .254 .237 .239 .294

LEFT FOREARM

.079 .074 .069 .066 .058 .054 .053 .048 .047 .045 .245 .242 .234 .225 .205 .188 .174 .163 .154 .151 .253

RIGHT FOREARM

.078 .075 .071 .069 .063 .058 .055 .050 .047 .047 .245 .250 .244 .233 .213 .189 .171 .160 .149 .148 .250

LEFT HAND

.031 .070

RIGHT HAND

.030 .074

ABDOMINO-PELVIC SEGMENT

.270 .278 .281 .283 .277 .274 .279 .293 .284 .214 .763 .792 .812 .825 .822 .847 .841 .081 .043 .219 .019

LEFT THIGH

.142 .142 .135 .129 .123 .116 .104 .097 .094 .092 .501 .488 .482 .465 .444 .421 .399 .372 .360 .352 .440 .322

RIGHT THIGH

.135 .141 .135 .130 .122 .114 .107 .102 .093 .087 .515 .511 .499 .481 .462 .435 .416 .385 .360 .354 .419 .328

LEFT LEG

.093 .090 .088 .087 .085 .076 .065 .057 .048 .050 .319 .312 .320 .322 .311 .290 .255 .232 .201 .206 .406 .033

RIGHT LEG

.092 .085 .086 .086 .084 .079 .071 .059 .048 .055 .323 .310 .318 .326 .318 .300 .265 .235 .208 .210 .410 .034

LEFT FOOT

.076 .053 .013 .080 .208 .139

RIGHT FOOT

.082 .049 .012 .080 .212 .141

COMPUTED SEGMENT PARAMETER VALUES

(UNITS: VOLUME IN LITRE, MASS IN KG, COORDINATES IN M, MOMENTS OF INERTIA IN KG*M**2)

FORMAT AND SEQUENCE OF DATA PRESENTATION:

SEGMENT NAME

VOLUME, MASS, COORDINATES OF CENTROID(X,Y,Z)

PRINC. MOMENTS OF INERTIA W.R.T. CENTROID(IX,IY,IZ),AND LOCAL SYSTEMS ORIGIN(IOX,IOY,IOZ)

COORDINATES(OX,OY,OZ) OF ORIGIN OF DISTAL SEGM. RELATIVE TO LOCAL COORD.-SYSTEM OF PROXIMAL SEGM.

SPECIAL SEGMENT PARAMETERS

NOTE THAT COORDINATES OF CENTROIDS ARE GIVEN W.R.T. LOCAL (SEGMENT-FIXED) COORD. SYSTEMS, WHICH EMANATE FROM THE SEGMENT ORIGIN, AND WHICH ARE PARALLEL TO THE SEGMENT PRINCIPAL AXES. WHERE PRINCIPAL AXES DIFFER FROM ORIGINAL SEGMENT AXES (SEE MANUAL), THE CENTROID COORDINATES W.R.T. THESE AXES ARE ALSO GIVEN AS "SPECIAL SEGMENT PARAMETERS".

ABDOMINO-THORACIC SEGMENT

14.409 12.927 .000 .042 .176

.188638 .208449 .094067 .610554 .608006 .116427

.000 .000 .000

ANGLE (RADIANS) BETWEEN PRINCIPAL Z-AXIS AND ORIGINAL Z-AXIS OF THE SEGMENT:

.142

COORDINATES (X,Y,Z) OF CENTROID REL. TO ORIGINAL (NON-PRINCIPAL) SEGMENT AXES:

.000 .016 .180

HEAD-NECK SEGMENT

4.256 4.690 .000 .000 .123

.028124 .022853 .021136 .098921 .093651 .021136

.000 .056 .391

COORDINATES(VX,VY,VZ) OF VERTEX OF HEAD REL. HEAD-NECK COORD.-SYSTEM:

.000 .000 .223

LEFT SHOULDER SEGMENT

1.227 1.264 .000 .000 .155

.003335 .002270 .002969 .033849 .032785 .002969

.000 .046 .319

Y AND Z COORD. OF SYSTEM ORIGIN(REL. ABD.-THOR. SEGM.) FOR 2-DIM. MODEL MOVING IN SAGITTAL PLANE:

.045 .313

RESTING INCLINATION ANGLE (RADIANS) OF SEGMENT Z-AXIS TO HORIZONTAL:

.029

RIGHT SHOULDER SEGMENT

1.314 1.354 .000 .000 .159

.003725 .002639 .003131 .038082 .036997 .003131

.000 .045 .319

Y AND Z COORD. OF SYSTEM ORIGIN(REL. ABD.-THOR. SEGM.) FOR 2-DIM. MODEL MOVING IN SAGITTAL PLANE:

.045 .313

RESTING INCLINATION ANGLE (RADIANS) OF SEGMENT Z-AXIS TO HORIZONTAL:

-.026

LEFT ARM

1.494 1.583 .000 .000 -.121

.011320 .010889 .001500 .034453 .034022 .001500

.000 .000 .222

RIGHT ARM

1.633 1.731 .000 .000 -.128

.014158 .013498 .001737 .042418 .041757 .001737

.000 .000 .230

NOTE: FOR 2-DIM. MODEL ORIGINS OF ARM SEGMENTS COINCIDE WITH ORIGINS OF SHOULDER SEGM.

LEFT FOREARM

.808 .867 .000 .000 -.103

.004647 .004597 .000494 .013911 .013861 .000494

.000 .000 -.276

RIGHT FOREARM

.824 .883 .000 .000 -.100

.004466 .004419 .000528 .013336 .013289 .000528

.000 .000 -.294

LEFT HAND

.195 .217 -.035 .003 -.011

.000117 .000175 .000222 .000144 .000459 .000484

.000 .000 -.253

ANGLE (RADIANS) BETWEEN PRINCIPAL Z-AXIS AND ORIGINAL Z-AXIS OF THE SEGMENT:

-1.129

COORDINATES (X,Y,Z) OF CENTROID REL. TO ORIGINAL (NON-PRINCIPAL) SEGMENT AXES:

-.005 .003 -.036

RIGHT HAND

.206 .228 .036 .003 -.010

.000129 .000171 .000236 .000156 .000484 .000527

.000 .000 -.250

ANGLE (RADIANS) BETWEEN PRINCIPAL Z-AXIS AND ORIGINAL Z-AXIS OF THE SEGMENT:

1.157

COORDINATES (X,Y,Z) OF CENTROID REL. TO ORIGINAL (NON-PRINCIPAL) SEGMENT AXES:

.005 .003 -.037

ABDOMINO-PELVIC SEGMENT

8.661 8.679 .000 .075 -.044

.042586 .101679 .047893 .108954 .118611 .097330

.000 .000 .000

ANGLE (RADIANS) BETWEEN PRINCIPAL Z-AXIS AND ORIGINAL Z-AXIS OF THE SEGMENT:

-.802

COORDINATES (X,Y,Z) OF CENTROID REL. TO ORIGINAL (NON-PRINCIPAL) SEGMENT AXES:

.000 .021 -.085

THIS SUBJECT IS OBESE OR PREGNANT.

LEFT THIGH

5.894 6.223 .000 .000 -.197

.095829 .091353 .015684 .338381 .333905 .015684

-.076 .136 -.026

RIGHT THIGH

6.139 6.474 .000 .000 -.196

.098380 .092246 .017498 .347357 .341223 .017498

.079 .141 -.031

LEFT LEG

2.440 2.657 .000 .000 -.168

.033797 .033047 .002911 .108692 .107942 .002911

.000 .000 -.440

RIGHT LEG

2.530 2.756 .000 .000 -.173

.036147 .035272 .003083 .118432 .117557 .003083

.000 .000 -.419

LEFT FOOT

.645 .811 .000 -.042 -.033

.002326 .002257 .000565 .004648 .003117 .002027

.000 .000 -.406

ANGLE (RADIANS) BETWEEN PRINCIPAL Z-AXIS AND ORIGINAL Z-AXIS OF THE SEGMENT:

-.121

COORDINATES (X,Y,Z) OF CENTROID REL. TO ORIGINAL (NON-PRINCIPAL) SEGMENT AXES

.000 -.046 -.027

COORD. OF HEEL(XH,YH,ZH) AND TOE(XT,YT,ZT) GROUND CONTACT POINTS (REL. TO FOOT COORD. SYST.):

.000 -.086 .041 .000 -.064 -.139

RIGHT FOOT

.689 .871 .000 -.041 -.034

.002553 .002498 .000633 .005063 .003521 .002120

.000 .000 -.410

ANGLE (RADIANS) BETWEEN PRINCIPAL Z-AXIS AND ORIGINAL Z-AXIS OF THE SEGMENT:

-.132

COORDINATES (X,Y,Z) OF CENTROID REL. TO ORIGINAL (NON-PRINCIPAL) SEGMENT AXES:

.000 -.045 -.029

COORD. OF HEEL(XH,YH,ZH) AND TOE(XT,YT,ZT) GROUND CONTACT POINTS (REL. TO FOOT COORD. SYST.):

.000 -.086 .042 .000 -.062 -.142

TOTAL VOLUME AND MASS

53.362 54.213

*** END OF RECORD ***

Stop - Program terminated.

Anhang 3 HatSe-Output

Die Ausgabe des Programms besteht, neben der Liste im Hauptfenster selber, aus zwei Dateien:

```
NAME_exploded.asc
NAME_points.txt
```

Beide Dateien werden in dasselbe Verzeichnis geschrieben, aus dem die zu analysierende Datei ursprünglich geladen wurde.

Die asc-Datei wurde in ihrem Aufbau bereits beschrieben und kann direkt in ein 3D-Programm wie MeshLab zur Ansicht geladen werden.

`NAME_points.txt` enthält eine Liste der zu den jeweiligen Segmenten gehörenden Punkte. Das Format ist:

```
Segment Nummer x
x1 y1 z1
x2 y2 z2
...
[Leerzeile]
Segment Nummer y
x1 y1 z1
x2 y2 z2
...
```

Es wird also immer eine Zeile mit der Segmentnummer geschrieben, danach folgen die Koordinaten aller Punkte dieses Segments. Bevor die Daten eines neuen Segmentes aufgelistet werden, kommt eine Leerzeile.

Abbildungsverzeichnis

Tabellenverzeichnis

Pseudocodeverzeichnis

Literaturverzeichnis

[1] Baca, A. (1995), *Precise Determination of Anthropometric Dimensions by Means of Image Processing Methods for Estimating Human Body Segment Parameter Values*. J. Biomechanics 29, 563-567

[2] BIOMLIB (2001), *Determination of Anthropomorphic Segment Parameter Values*. User Reference Manual BIOMLIB-TR-79-UM-003. BIOMLIB, Muenchen

[3] Hanavan, E. P. (1964), *A Mathematical Model of the Human Body*, AMRL-TR-64-102, Wright-Patterson Air Force Base, Ohio

[4] Hatze, H. (1979), *A Model for the Computational Determination of Parameter Values of Anthropomorphic Segments*. National Research Institute for Mathematical Sciences, CSIR Techn. Report TWISK 79, Pretoria

[5] Hatze, H. (1980), *A Mathematical Model for the Computational Determination of Parameter Values of Anthropomorphic Segments.*, J. Biomechanics 13, 833-843

[6] Hatze H. (1986), *Methoden biomechanischer Bewegungsanalyse*. Österreichischer Bundesverlag, Wien

[7] Hemami, H., Farnsworth, R.L. (1977), *Postural and Gait Stability of a Planar Five Link Biped by Simulation*, IEEE Transactions on Automatic Control, AC-22(3):452-458

[8] Jensen, R.K. (1978), *Estimation of the Biomechanical Properties of Three Body Types Using a Photogrammetric Method*, J. Biomechanics 11, 349-358

[9] Ju, X., Siebert, P.J. (2000), *Individualising Human Animation Models,* 3D MATIC Laboratory, Department of Computing Science, University of Glasgow, Glasgow

[10] Jun-Ming Lu, Mao-Jiun J. Wang (2008), *Automated Anthropometric Data Collection Using 3D Whole Body Scanners*. Expert Systems with Applications: An International Journal archive. Volume 35, 407-414

[11] Kuntner, A., Meiller, N. & Schiffl, K. (2010), *Dokumentation zum Projektpraktikum BodyScan*, TU Wien

[12] Lohman, T., Roche, A., Martorell, R. (1988), *Anthropometric Standardization Reference Model*, Champaign, IL: Human Kinetics Books

[13] Piniel, P., Bredl, E. (2010), *Abschlussbericht Erfassung anthropometrischer Dimensionen 3D-Scanner und manuelle Körpervermessung im Vergleich*, MB2I - Bewegungswissenschaft und Sportinformatik - Abt. F, 350498, Wien

[14] SAE International, *CAESAR(TM) Executive Summary*, http://www.sae.org/standardsdev/tsb/cooperative/caesumm.htm (31.03.2011)

[15] Vitronic, *Vitus 3D-Bodyscanner Broschüre*, VITRONIC Dr.-Ing. Stein Bildverarbeitungssysteme GmbH, Wiesbaden, www.vitronic.de/uploads/media/vitus_d_02.pdf (08.01.2010)

[16] Vitronic (2009), *VITUS^{smart} 3D Body Scanning System*, VITRONIC Dr.-Ing. Stein Bildverarbeitungssysteme GmbH, Wiesbaden

[17] Vitronic (2008), *Vitus Height-Calibration Manual, VITRONIC Dr.-Ing. Stein Bildverarbeitungssysteme GmbH, Wiesbaden*

[18] Vukobratovic, M., Juricic, D. (1969), *Contribution to the Synthesis of Biped Gait*, IEEE Transactions on Biomedical Engineering, BME-16, 1-6