Bachelorthesis

Selbstavatare in virtueller Realität: Automatisierte Validierung von Körperform und Animation mit wenigen Markern

Johannes Schirm

732571

Hochschule Reutlingen

Fakultät Informatik

Medien- und Kommunikationsinformatik

Eingereicht am 20. Juli 2017

Erstprüfer:	Prof. Dr. rer. nat. Uwe Kloos
Zweitprüfer:	Prof. Dr.-Ing. Cristóbal Curio
Betreuerin am Max-Planck-Institut für biologische Kybernetik:	Dr. Betty Mohler

Kurzfassung

Besonders in virtueller Realität lohnt sich der Aufwand, einen individuellen Avatar zur Repräsentation des eigenen Körpers bereitzustellen. Zwar gibt es immer die Möglichkeit, diesen mittels eines 3D-Scans für jeden einzelnen Benutzer manuell zu erstellen, jedoch ist diese Vorgehensweise nur für einige spezielle Szenarien praktikabel, da für einen 3D-Scan nicht nur teure Hardware, sondern je nach Methode und gewünschter Qualität auch einige Zeit zur Berechnung und Optimierung der Ergebnisse benötigt werden. Das Hauptziel dieser Arbeit besteht darin, erste Schritte in Richtung einer möglichst realistischen Einbindung von hochauflösenden Selbstavataren in Echtzeitanwendungen für virtuelle Realität zu gehen und Werkzeuge für die weitere Verbesserung des vorgestellten Ansatzes zu entwickeln. Aufgebaut wird dabei auf eine vorhandene Prozedur zur Generierung von Avataren aus Körpermaßen und statistischen Informationen über menschliche Körper.

Abstract

Especially in virtual reality, it is worth the effort to provide a personalized avatar in order to represent the own body. Although it is possible to manually create such an avatar from a 3D scan of every user, this is only feasible for special scenarios. A 3D scan requires not only expensive hardware, but also takes some time for calculation and optimization of the results depending on the desired quality. The main goal of this thesis is to make first steps towards realistically including high-resolution self-avatars into realtime applications for virtual reality and also to develop tools for further optimization of this approach. In doing so, it is built upon an already existing procedure to generate avatars from body measurements and statistical information about human bodies.

Danksagungen

Zuerst danke ich natürlich Uwe Kloos für die Betreuung von Seiten der Hochschule Reutlingen und Cristóbal Curio für die Zweitprüfung. Es freut mich, dass sich beide Prüfer meines Themas angenommen haben.

Betty Mohler danke ich für die Betreuung von Seiten des Max-Planck-Instituts für Biologische Kybernetik und für das Angebot dieses sehr spannenden Themas. Außerdem danke ich Joachim Tesch, der einen großen Einfluss auf die Entwicklung meines technischen Verständnisses hatte.

Ich möchte auch Naureen Mahmood, Sergi Pujades und Timo Bolkart aus der Gruppe „Perceiving Systems“ des Max-Planck-Instituts für Intelligente Systeme dafür danken, dass sie sich immer wieder Zeit genommen haben, um mir viele Dinge zum Körpermodell SMPL und den zugehörigen Python-Skripten zu erklären.

Ebenfalls danke ich Anne Thaler für ihre hilfsbereite Unterstützung bei Fragen der Statistik und des wissenschaftlichen Schreibens und Alessia Cavallo für die zuverlässige Durchführung meiner Studie mit zehn Teilnehmenden.

Zuletzt möchte ich mich auch für die wunderbare Möglichkeit der Arbeit am Max-Planck-Campus bei den beiden Direktoren Heinrich H. Bülthoff und Michael J. Black bedanken, die sich für eine äußerst interessante Forschung einsetzen.

Inhaltsverzeichnis

1 Einleitung

Besonders aus Videospielen und Online-Systemen dürfte die Erzeugung von Avataren zur Repräsentation der eigenen Person auf virtueller Ebene bereits vielen bekannt sein. Dort ist es in der Regel möglich, eine beliebige Kombination aus vordefinierten Merkmalen auszuwählen, beispielsweise Frisur, Körpergröße und Hautfarbe. Dies ist jedoch nicht mehr ausreichend, sobald ein Avatar außer der Wiedererkennbarkeit vor allem die Immersion in eine virtuelle Umgebung verbessern soll: Hierbei spielen nämlich Faktoren wie zum Beispiel Statur, Armlänge und Schulterbreite eine größere Rolle. Speziell im Fall des „Selbstavatars“ (mehr dazu Kapitel 2.5), der immer aus der Ich-Perspektive mithilfe von *„Head-mounted Displays“* (HMDs) erlebt wird, rücken diese Faktoren besonders in den Vordergrund. Im wissenschaftlichen Umfeld wird daher für Studien, die aus Gründen der Genauigkeit einen möglichst präzisen Avatar voraussetzen, ein vollständiger, hochauflösender 3D-Scan jedes Teilnehmenden erstellt, sodass dieser dann genauestens in der virtuellen Umgebung repräsentiert werden kann.

Leider ergeben sich bei dieser Vorgehensweise nicht nur organisatorische Hürden, die den Ablauf des Experiments verzögern können, sondern auch einige technische Limitationen: Neben der hohen Informationsdichte von vollständigen Körperscans, die die Leistung mancher Anwendungen spürbar verringern kann, ist es vergleichsweise aufwändig, den statischen 3D-Scan für interaktive Experimente animierbar zu machen, da für jeden Teilnehmenden auch ein passendes Rig von Hand erstellt werden müsste. Automatische Lösungen hierfür sind im Moment noch zu ungenau oder nur online verfügbar, was gerade im experimentellen Umfeld auch datenschutzrechtliche Probleme mit sich bringen würde. Gesucht ist also eine Lösung, welche die Proportionen des jeweiligen Körpers durch eine weniger aufwändige Prozedur als die des vollständigen Körperscans ermittelt und dann in Echtzeit einen zur realen Person möglichst passenden Avatar generiert.

Als Grundlage dafür würde sich das 2015 von M. Loper et al. vorgestellte Modell „SMPL“ [1] zur einfachen Darstellung von dreidimensionalen, menschlichen Körpern sehr gut eignen, da es direkt kompatibel mit aktuellen Spiel-Engines ist und schon ein eigenständiges Rig mit sich bringt, das nach automatischer Anpassung an die jeweilige Körperform zur Animation des Modells verwendet werden kann. Zudem ist SMPL aufgrund seiner Einfachheit sehr gut geeignet für Anwendungen, die anspruchsvolle virtuelle Umgebungen darstellen und daher keine Rechenleistung verschwenden dürfen. Mithilfe dieses Modells ist es momentan schon möglich, nur auf Basis einiger Körperabmessungen einen Avatar zu generieren, der in seiner Körperform eine gute Annäherung an die reale Person darstellen kann.

1.1 Zielsetzung

Um die jedoch noch nicht ganz ausgereifte Generierung von Avataren aus Körperabmessungen besser weiterentwickeln zu können, sollen im Rahmen dieser Arbeit in der Spiel-Engine Unity einige Werkzeuge entwickelt werden, mit deren Hilfe generierte Avatare automatisiert analysiert und verglichen werden können. Damit sollen Aussagen über die Genauigkeit der hierbei erstellen Avatare getroffen werden können, sodass die für die Generierung verwendeten statistischen Grundlagen in Zukunft entsprechend optimiert werden können. Neben einer Ausgabe von den gemessenen Daten sollen auch Bilder von den Körpern berechnet werden, die einen visuellen Vergleich verschiedener Varianten erleichtern.

Parallel soll die Umsetzung von Selbstavataren in virtueller Realität auf Basis von SMPL untersucht und verbessert werden. Ziel ist die Entwicklung eines Prototypen für eine Projektvorlage in Unity, nach deren Weiterentwicklung die einfache Erstellung neuer Szenarien mit einem bereits integrierten Selbstavatar möglich sein wird. Damit wären zum Beispiel Wissenschaftler imstande, den Avatar in ihren Experimenten aus wenigen Körperabmessungen der Teilnehmenden automatisiert generieren zu lassen und diesen gegebenenfalls zu animieren. Mit einer solchen Alternative könnte der bisher häufig notwendige 3D-Scan aller Teilnehmenden für viele Experimente vermieden werden, sofern die erzielte Genauigkeit dieses Verfahrens zur Beantwortung der Forschungsfrage des jeweiligen Experiments ausreicht. Auf dem Weg zu einem System, das diesen Anforderungen genügt, werden auch einige technische Lösungen entwickelt, mit denen die aktuelle Vorgehensweise bei der Umsetzung eines Selbstavatars verbessert werden soll.

Allen Überlegungen zur Ableitung einer Körperform aus Körperabmessungen geht ein gutes Verständnis des SMPL-Modells voraus, weshalb die Grundlagen hierfür in Kapitel 2 etwas ausführlicher erklärt werden.

1.2 Aufbau der Thesis

Einen großen Teil dieser Arbeit wird die Zusammenfassung des aktuellen Stands der Forschung einnehmen, da schon der Startzustand des untersuchten Problems einige statistische und computergrafische Kenntnisse voraussetzt und in direkter oder indirekter Verbindung zu mehreren wissenschaftlichen Veröffentlichungen steht, die schon in diesem Forschungsbereich erschienen sind. Nach der kompakten Erklärung des grundlegenden Wissens soll im gleichen Hauptkapitel dann auch ein Blick auf alternative Verfahren und die Geschichte von menschlichen Körpern in der Computergrafik (Kapitel 2.3) und den aktuellen Stand von Avataren in virtueller Realität (Kapitel 2.5) geworfen werden. Besonders ausführlich wird dabei

das SMPL-Modell zur Repräsentation menschlicher Körper beschrieben, das wohl als eine der wichtigsten Grundlagen dieser Arbeit angesehen werden kann. Kapitel 2.6 soll am Ende als Zusammenfassung des Hauptkapitels dienen, das die erklärten Grundlagen auf den Hauptteil der Arbeit bezieht.

Das nachfolgende Kapitel 3 wird beschreiben, wie das vorhandene Modell mit einigen Probanden, die ihre Körperscans für diesen Zweck zur Verfügung gestellt haben, für diese Arbeit validiert wurde und in welchen Belangen das Verfahren zur Generierung von Avataren aus Körpermaßen verbessert werden könnte. Auch wenn sich dieses Kapitel nicht ausschließlich der Implementierung widmet, werden hier ebenfalls einige technische Details besprochen, die bei der Umsetzung der automatischen Validierung eine Rolle gespielt haben.

In Kapitel 4 soll zusammengefasst werden, wie auf technischer Ebene bei der Umsetzung eines Selbstavatars mit SMPL vorgegangen wurde. Mithilfe von einigen Auszügen aus dem Quellcode der entwickelten Unity-Komponenten werden als Ergänzung zur allgemeinen Dokumentation des Unity-Projekts bemerkenswerte Implementierungsdetails genauer dargestellt.

Neben den beiden genannten Hauptteilen wurde für diese Arbeit eine kurze Studie entworfen, deren Durchführung und Ergebnisse in Kapitel 5 beschrieben werden. Zufällige Probanden sollten bei diesem Experiment das SMPL-Standardmodell in virtueller Realität so anpassen, dass es sie am besten repräsentiert. Verglichen mit den aus den Körperabmessungen der Probanden generierten SMPL-Modellen kann der erzeugte Avatar einige Hinweise darauf geben, auf welche Dinge die Teilnehmenden bei der Erstellung des Avatars besonders geachtet haben und auch, welche Toleranz sie bezüglich der Körperform ihrer virtuellen Repräsentation zeigen.

Abschließend werden in Kapitel 6 weiterführende Gedanken zur durchgeführten Studie strukturiert und bewertet. Auch werden hier die bei der Implementierung erreichten Ergebnisse besprochen und mögliche Ansatzpunkte eingebracht, um das Projekt in Zukunft noch weiter zu verbessern.

2 Stand der Forschung

Um den Stand der Forschung besser erklären zu können, werden in den ersten beiden Unterkapiteln einige grundlegende Themen kurz zusammengefasst, die für das Verständnis des Aufgabenbereichs sehr wichtig sind. Die nachfolgenden Kapitel sollen näher auf die Voraussetzungen dieser Arbeit und den bisherigen Stand der Forschung bezüglich der besprochenen Hauptthemen eingehen.

2.1 Animation computergrafischer Modelle

Dieses Unterkapitel soll eine kompakte Erklärung der computergrafischen Grundlagen darstellen, die generell für animierte Avatare in 3D-Anwendungen benötigt werden. Alle in diesem Abschnitt gezeigten Bilder zur Verdeutlichung der Beispielfälle wurden mit der 3D-Software Autodesk 3dsMax [2] erstellt.

2.1.1 Rigging und Skinning

Auch wenn in vorherigen Kapiteln schon Bezug darauf genommen wurde, soll hier noch einmal anhand eines Beispiels gezeigt werden, was die beiden Prozesse Rigging und Skinning unterscheidet und wie diese Techniken kombiniert werden, um menschliche 3D-Modelle darzustellen.

Grundsätzlich teilt sich ein animierbares Charaktermodell in zwei unterschiedliche Systeme auf: Zum einen besteht jedes dieser Modelle, wie andere 3D-Objekte auch, aus vielen Vertices (Punkte im 3D-Raum), die wiederum zu Polygonen verbunden werden, sodass sich eine darstellbare Oberfläche ergibt. Zum anderen gehört zu jedem animierbaren Charaktermodell auch ein individuelles Gerüst (auf Englisch „Rig“), welches aus Gelenken und Knochen besteht und in vereinfachter Form das Skelett des Charakters repräsentiert. Die Tätigkeit des Erstellens eines solchen logischen Gerüsts nennt sich Rigging. Die Tätigkeit der Zuordnung von Vertices aus dem Modell zum erstellten Gerüst nennt sich hingegen Skinning. Dabei werden auch Wichtungen vergeben, die bestimmen, wie stark die Veränderung des Gerüsts sich auf den jeweiligen Vertex auswirkt. In Abbildung 1 ist in weißer Farbe das unterliegende Rig des SMPL-Modells sichtbar. Die um 45°gedrehten Quadrate stellen dabei die Gelenke dar, deren Rotation später in der virtuellen Umgebung in Echtzeit den Bewegungen des Benutzers angeglichen wird.

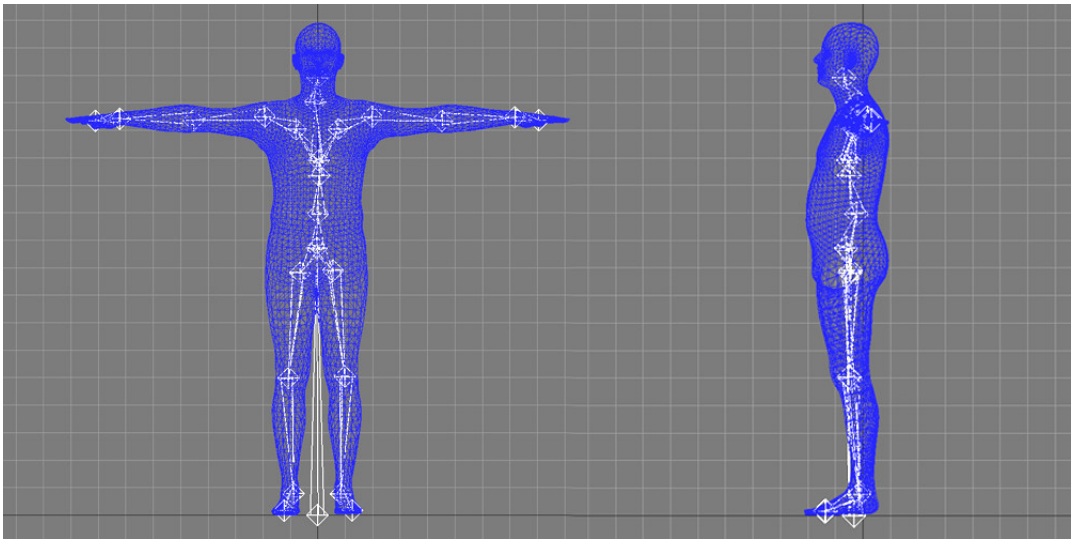


Abbildung 1: Rigging und Skinning - Das SMPL-Rig

Als ein Prozess, der für jedes 3D-Objekt in individueller Weise ausgeführt werden muss, ist es eine vergleichsweise schwierige Aufgabe, das Rigging und Skinning von Modellen zu automatisieren. Eine der zurzeit bekanntesten Lösungen ist hierfür „Mixamo“ [3] von Adobe, eine Onlineanwendung mit recht hoher Genauigkeit, die jedoch auch die Weitergabe des 3D-Modells an den Anbieter erfordert.

2.1.2 Modellvarianten

Die Kernidee des SMPL-Modells besteht in der Praxis zu einem großen Teil aus Modellvarianten, zumindest was die Darstellung menschlicher Körperform angeht. Als Modellvariante (auf Englisch „Blendshape“) wird in der Computergrafik eine spezielle lineare Deformierung eines 3D-Modells bezeichnet. Damit die Technik eingesetzt werden kann, müssen außer dem Basismodell, das als Ausgangspunkt dient, weitere Modellvariationen vorliegen, die in den jeweiligen Anwendungen meist als Kanäle durch einen Prozentwert anzusprechen sind. Da die Idee der Modellvarianten wichtig für diese Arbeit ist und noch häufig referenziert werden wird, soll das folgende Beispiel die Technik detailliert erklären:

Auch wenn es später um das Deformieren von komplexen 3D-Körpermodellen mit vielen Tausend Vertices gehen wird, sollte man zuerst ein klares Bild von Modellvarianten im Kleinen bekommen haben. Daher stehen im Mittelpunkt dieses Beispiels drei einfache 3D-Quader, deren Unterseite mittig im Weltursprung liegt. In Abbildung 2 ist auf der rechten Seite eine Drahtgitterdarstellung der Ausgangs-

szene zu sehen, die auch einen genaueren Blick auf die Ausmaße der drei Quader erlaubt. Durch die drei sehr verschiedenen Formen kann am besten demonstriert werden, was bei der linearen Überblendung der einzelnen Objekte passiert.

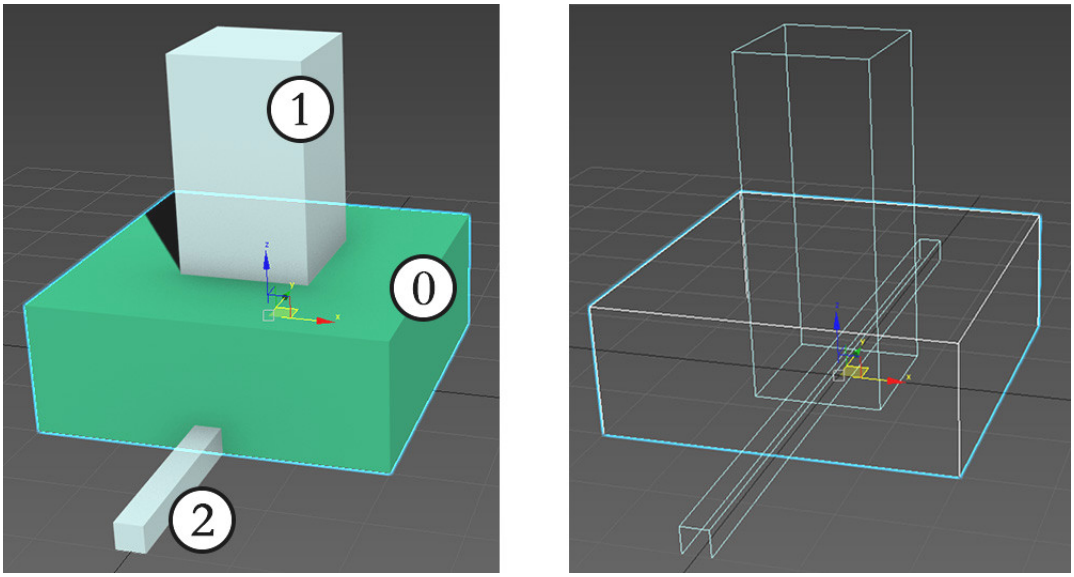


Abbildung 2: Modellvarianten - Drei Quader sollen das System veranschaulichen

Auf der linken Seite wurde schon eine Nummerierung vergeben, die gleichzeitig auch die Zuordnung der Objekte zu Kanälen und somit auch zu Modellvarianten repräsentiert: Der Quader mit der Nummer 0 soll als das Basismodell für dieses Beispiel dienen, weshalb ihm in der hierfür verwendeten Software eine Komponente hinzugefügt wurde, die seine Deformierung zu Modellvarianten ermöglicht. Die anderen beiden Quader werden daraufhin als Kanal 1 und Kanal 2 in der gerade erstellten Komponente registriert. Sehr wichtig hierfür ist, dass die beiden anderen Quader nur zu einer Modellvariante werden können, da sie die **gleiche Topologie** wie das Basismodell haben. Das bedeutet, dass alle Modellvarianten eines Quaders ebenfalls aus exakt acht Vertices bestehen müssen, die idealerweise auch die gleiche semantische Bedeutung innerhalb des Modells haben.

Hat ein Kanal den Wert 100 %, wird das Basismodell komplett zur entsprechenden Modellvariante des Kanals deformiert. Hat der Kanal jedoch 0 %, ist dessen Einfluss auf das Endergebnis komplett erloschen. Abbildung 3 und Abbildung 4 zeigen, wie das Basismodell bei einer entsprechenden Deformierung vollständig zur Modellvariante hinter den jeweiligen Kanälen wird.

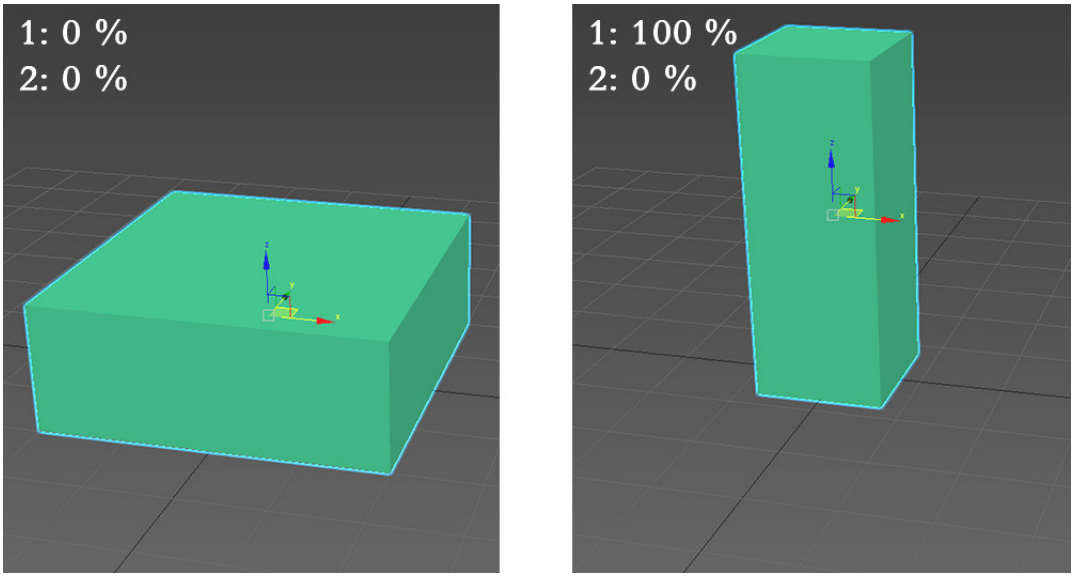


Abbildung 3: Modellvarianten - Kanal 1 auf 100 %

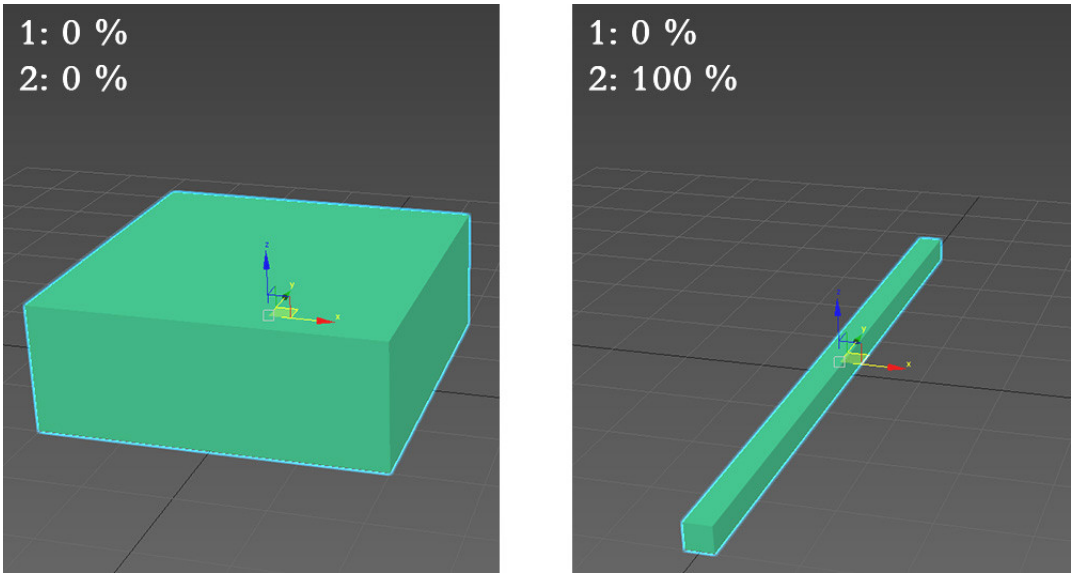


Abbildung 4: Modellvarianten - Kanal 2 auf 100 %

Komplexer wird jedoch schon das Zusammenspiel verschiedener Kanäle: Zwei Kanäle auf jeweils 50 % stellen beispielsweise die lineare Mitte zwischen beiden Modellvariationen dar. In Bezug auf das aktuelle Beispiel zeigt Abbildung 5, wie eine solche Mischung zwischen Kanal 1 und Kanal 2 aussehen würde.

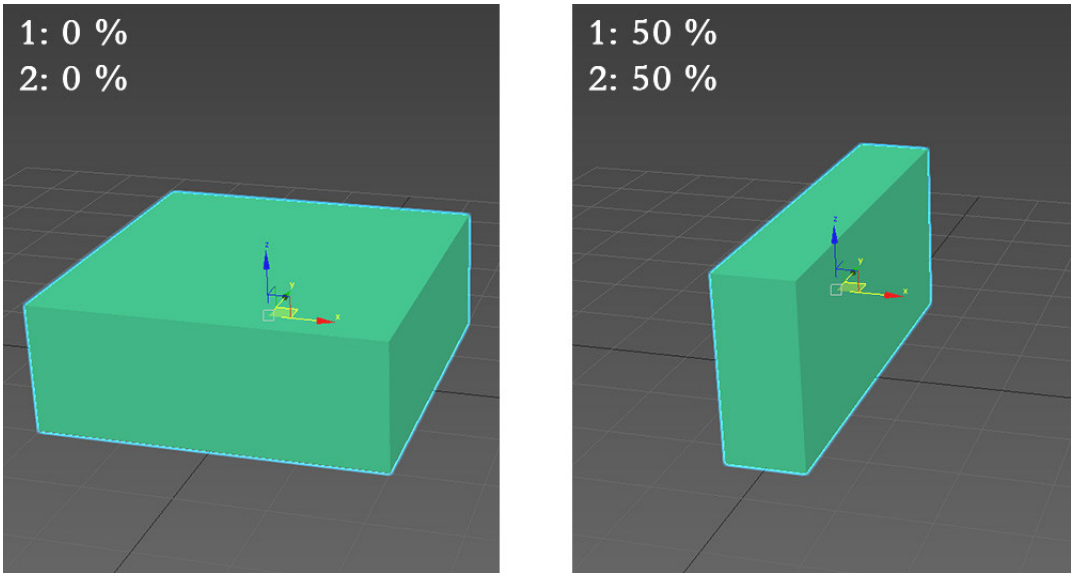


Abbildung 5: Modellvarianten - Kanal 1 und 2 Teilen sich den Einfluss

Da sich die Abweichung hier insgesamt auf 100 % summiert, hat das Basismodell in diesem Fall keinen Einfluss mehr auf das Ergebnis. Wäre jedoch insgesamt nur ein einziger Kanal auf 50 %, bekommt das Basismodell „den Rest des Einflusses“, sodass dann - wie man es natürlicherweise erwarten würde - die Abweichung dieses Kanals vom Basismodell in Erscheinung tritt. Um diesen Sachverhalt etwas besser zu veranschaulichen, wird in Abbildung 6 das Basismodell etwas verändert. Für dieses Beispiel werden die vier rechten Vertices um einige Längeneinheiten nach rechts verschoben, sodass das Basismodell etwas breiter wird.

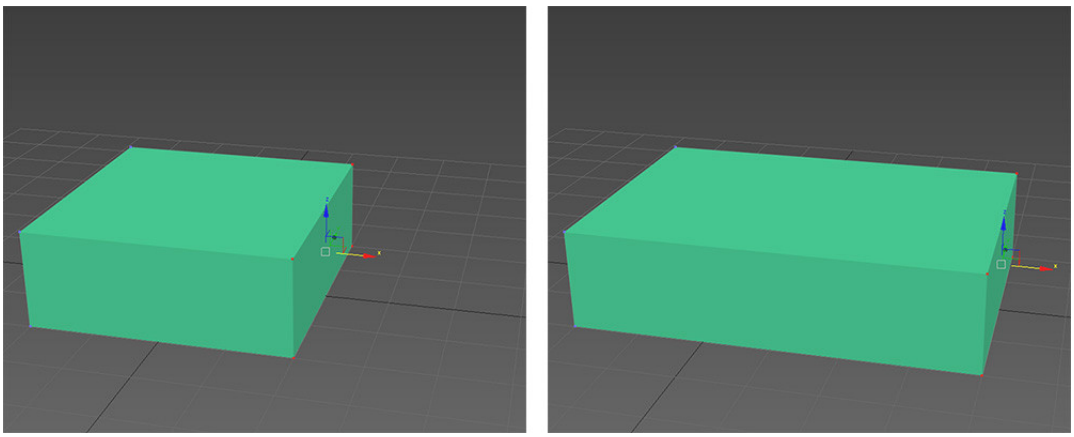


Abbildung 6: Modellvarianten - Modifikation des Basismodells

Vergleicht man nun den Zustand des deformierten Objekts vor und nach der Verschiebung der vier Vertices, fällt bei unterschiedlicher Kanalkonfiguration sofort die Differenz auf. Abbildung 7 zeigt, dass die Änderung des Basismodells bei einer vollständigen Balance der Modellvarianten keinen Effekt hat. In Abbildung 8, in der 50 % Einfluss „übrig bleiben“, wirkt sich die Änderung hingegen aus.

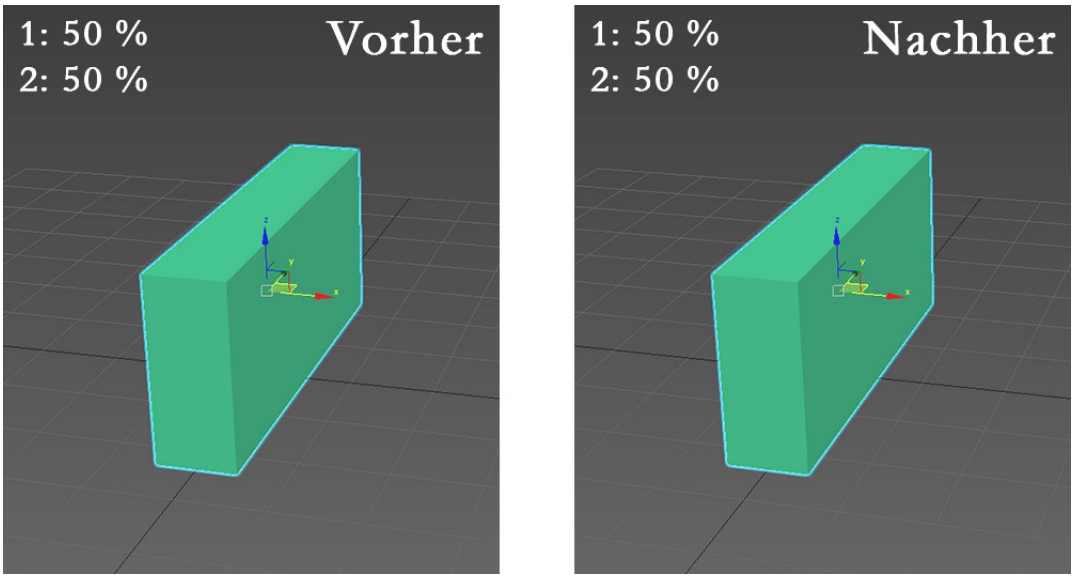


Abbildung 7: Modellvarianten - Kein Einfluss des Basismodells

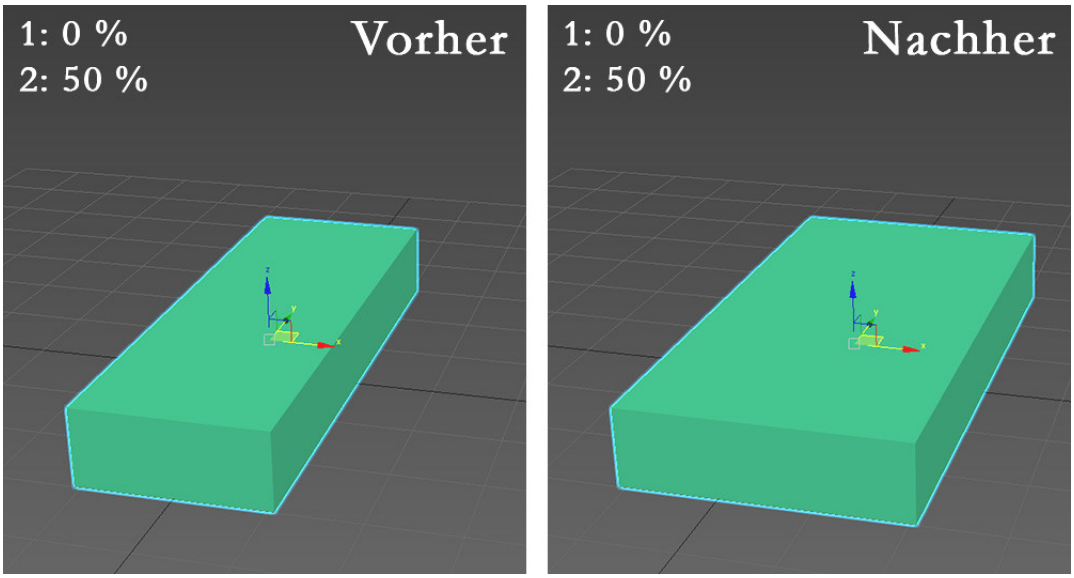


Abbildung 8: Modellvarianten - „Übriger Einfluss“ geht an das Basismodell

Besonders bekannt sind Modellvarianten durch ihre Verwendung bei der Animation von menschlichen Gesichtsausdrücken geworden. Im Jahr 1972 veröffentlichte F. Parke bereits eine erste Arbeit [4] zu den Grundlagen der Technik.

2.2 Statistische Verfahren für Körpermodelle

Dieses Unterkapitel soll eine kompakte Zusammenfassung der mathematischen und statistischen Grundlagen darstellen, die vor allem bei der späteren Arbeit mit den verwendeten Körpermodellen vorausgesetzt werden.

2.2.1 Hauptkomponentenanalyse

Eine häufig verwendete Vorgehensweise, die auch im Kontext dieser Arbeit Anwendung findet, ist die Hauptkomponentenanalyse. Mit diesem mathematischen Verfahren versucht man, das umgebende System einer Ansammlung an Datenpunkten so zu transformieren, dass dessen Hauptkomponenten nun nach der Wichtigkeit für diesen speziellen Datensatz sortiert werden können (vgl. [5, S. 196]). Dafür wird innerhalb des Datensatzes für alle Dimensionen des Systems eine Richtung gesucht, in der sich die stärkste Varianz der Daten ausmachen lässt. In der Praxis bildet man dann das ursprüngliche System auf ein transformiertes ab, sodass sich Vorteile bei der Adressierung von Datenpunkten ergeben (vgl. [6, S. 196]). Diese bestehen zum größten Teil aus der Komprimierung der Daten, da durch die Sortierung nach Wichtigkeit erfahrungsgemäß ein großer Teil der Dimensionen am Ende der Liste nicht mehr berücksichtigt werden muss.

Als mathematische Vorgehensweise kann bei der Hauptkomponentenanalyse die Singulärwertzerlegung eingesetzt werden. Nach einer Zentrierung des Datensatzes durch Subtraktion des Mittelwerts und weiteren notwendigen Transformationen wie Skalierung und Rotation der Basisvektoren - ohne dabei deren Orthogonalität aufzuheben - kann die Abbildung des ursprünglichen Koordinatensystems auf das an den Datensatz angepasste System als Matrix dargestellt werden. Variablen, die vor der Transformation voneinander abhängig waren, sind dies danach unter Umständen nicht mehr, weil das transformierte Koordinatensystem nur hinsichtlich der größten Varianzen in den Daten ausgerichtet wurde.

In Abbildung 9 ist die Transformation eines zweidimensionalen Koordinatensystems mit X- und Y-Komponente in ein Koordinatensystem mit X'- und Y'-Komponente bildlich dargestellt. Dieses Beispiel wird aufgrund seiner Einfachheit wegen nur zwei verwendeten Dimensionen zwar keine Komprimierung der Daten ohne größere Verluste zulassen, aber zumindest wäre folgende Aussage gültig:

„Wenn die Y-Komponente aufgrund der geringeren Varianz auf der roten Y-Achse ignoriert würde, wiese das rote System im Vergleich zum schwarzen einen durchschnittlich deutlich geringeren Fehler für diesen Datensatz auf“.

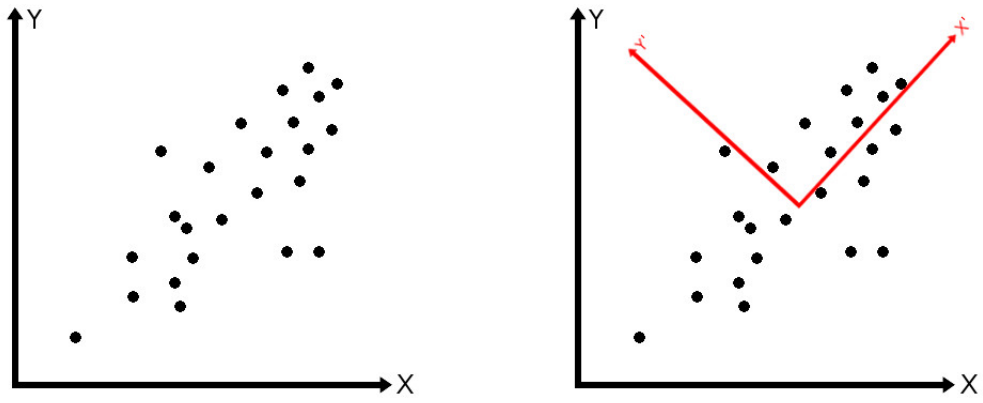


Abbildung 9: Hauptkomponentenanalyse - Skizze der Funktionsweise

Wie man an diesem Beispiel auch sehen kann, hängt es also stark vom Szenario ab, wie gut man seine Daten komprimieren kann. Müsste man sich hier auf exakt eine Koordinate beschränken, wäre es im roten System offensichtlich, dass man die X-Komponente dafür verwendet, während dies im schwarzen System einen ungefähr gleichen Verlust an Genauigkeit bedeuten würde.

Zwar ist die Hauptkomponentenanalyse mit zwei Dimensionen sehr anschaulich und einfach zu verstehen - in der Praxis wird jedoch eine Vielzahl an Dimensionen miteinbezogen, was diese Vorgehensweise schnell schwer visualisierbar macht.

2.2.2 Regressionsanalyse

Bei der Regressionsanalyse wird ein mathematisches Modell verwendet, um einen vorhandenen Satz an Daten möglichst genau repräsentieren und Aussagen über noch nicht abgedeckte Wertebereiche treffen zu können. Dabei werden häufig auch Variablen miteinbezogen, die keine systembedingte, sondern nur eine beobachtete Abhängigkeit voneinander haben. Ein anschauliches Beispiel dafür zeigen Urban und Mayerl in ihrem Lehrbuch zum Thema: Über einen fiktiven Datensatz, der von zehn Schülern Notendurchschnitt, Schulweglänge und Körpergewicht enthält, lässt sich zur Überraschung des Lesers die mathematisch korrekte Aussage treffen, „dass ein Schüler mit einem Körpergewicht von 150 kg und einer Länge des Schulwegs von 1 km einen Notendurchschnitt von 1,25“ [7, S. 18] erzielen sollte. Dass das Beispiel

wenig Sinn ergibt, betonen die Autoren natürlich im gleichen Absatz und geben dem Leser den Hinweis: „Vielmehr muss zuerst ein Regressionsmodell aufgestellt werden, das sinnvolle Verknüpfungen von Variablen vornimmt“ [7, S. 18]. Es geht also zuerst um eine genaue Beobachtung des Forschungsgegenstandes, bevor man anfangen kann, diesen zu formalisieren.

Wie der Begriff „Regression“ schon vermuten lässt, geht durch die statistische Komprimierung der Daten in eine mathematische Darstellung abhängig vom verwendeten Modell ein Teil der Genauigkeit verloren. Einer der Vorteile einer Regressionsanalyse kann jedoch die Übertragung einer Beobachtung in ein kompaktes, austauschbares Format und das Füllen von Lücken im Datensatz sein. Besonders das einfache Austauschen und Verwenden eines Regressors wird für diese Arbeit relevant sein, denn Echtzeitanwendungen kommt diese Art der Komprimierung und Abstraktion von Komplexität sehr zugute.

Die lineare Regression ist ein spezieller Weg, eine Regressionsanalyse auszuführen. Als eine der unkompliziertesten Vorgehensweisen wird hier versucht, beobachtete Korrelationen im Ausgangsdatensatz durch einen linearen Regressor darzustellen. Dieser kann mithilfe einer Matrix repräsentiert werden, die durch eine einfache Matrixmultiplikation eine beliebige Eingangskonfiguration in die zugehörige Ausgabekonfiguration umwandeln kann. Ist diese Matrix einmal gefunden, kann sie leicht weitergegeben und verwendet werden, da der Berechnungsaufwand linear ist. Ein Nachteil dieser Methode ist die vergleichsweise niedrige Genauigkeit. Gibt es in einem Datensatz größere Schwankungen, werden diese je nach Stärke immer ungenauer durch den Regressor dargestellt, weshalb sich diese Methode trotz ihrer Einfachheit für entsprechende Datensätze nicht sehr gut eignet.

Ist der Verlust der Genauigkeit bei einer linearen Regression eines entsprechend strukturierten Datensatzes zu hoch, gibt es viele Alternativen, die einen deutlich genaueren, dann jedoch auch komplexeren Regressor möglich machen. Ein Weg hierfür wäre das 1951 von D. Krige entwickelte verfahren „Kriging“ [8], welches ursprünglich aus der Geostatistik kommt und dort zur mathematischen Repräsentation von geografischen Topologien dient. Jedoch lässt sich damit natürlich auch jeder andere Datensatz vereinfacht darstellen - im Prinzip werden dabei mehrere mathematische Funktionen so kombiniert, dass sie sich dem ungefähren Verlauf der Daten möglichst gut anpassen, was in vielen Fällen deutlich genauer ist als ein linearer Regressor. Ein Nachteil bei diesem Verfahren ist wiederum, dass der Regressor selbst nicht mehr als kompakte Matrix dargestellt werden kann.

2.3 Menschliche Körper in der Computergrafik

Ein zentraler Gegenstand dieser Arbeit ist die Repräsentation menschlicher Körper im dreidimensionalen Raum. Es wurden hierfür in den letzten Jahren schon zahlreiche computergrafische Modelle aufgestellt, die in diesem Kapitel kurz angesprochen werden. Etwas ausführlicher beleuchtet wird das 2015 veröffentlichte SMPL-Modell, welches sich insbesondere durch eine unkomplizierte Integration in aktuell verwendete Animationssoftware und auch Spiel-Engines auszeichnet.

2.3.1 Die CAESAR-Datenbank

Die CAESAR-Datenbank (Kurz für „Civilian American and European Surface Anthropometry Resource“) ist ein Produkt von SAE International und enthält über 2000 als 3D-Scan registrierte menschliche Körper aus der Zivilbevölkerung in Nordamerika mitsamt der zugehörigen Körpermaße (vgl. [9]). Durch präzise spezifizierte Messpunkte und die große Menge an bereits gewonnenen Daten ist dieser Datensatz sehr gut für statistische Analysen geeignet, welche die wichtigste Grundlage einiger in diesem Kontext besprochener Körpermodelle darstellen.

2.3.2 Dreidimensionale Körpermodelle

Norman Badler beschäftigte sich schon 1993 im Rahmen seines Animationssystems „Jack“ [10] unter vielen anderen Dingen mit der Darstellung von Menschen in der dreidimensionalen Computergrafik. Dabei wurden einige Grundsteine für die heute bekannten Animationssysteme gelegt, die schon seit Jahren in vielen Bereichen der Industrie und Forschung Anwendung finden. Zwar sind diese Überlegungen natürlich nur schwer vergleichbar mit den heutigen Definitionen von Körpermodellen, doch vieles davon hat auch Einfluss auf die heutige Forschung zu diesem Thema, sodass seine Pionierarbeit durchaus noch relevant ist.

2003 veröffentlichten Allen et al. eine erste Arbeit [11], die als statistisches Körpermodell bezeichnet wurde. Das von ihnen vorgestellte Modell basiert auf statischen Körperscans, die mithilfe der Hauptkomponentenanalyse verarbeitet wurden.

Das 2005 von einer Gruppe der Stanford University veröffentlichte Verfahren „SCAPE“ [12] beruht auf einem Modell menschlicher Körper, das dem SMPL-Modell schon recht nahe kommt. Aus einem statischen Körperscan und zusätzlich aufgenommenen Bewegungsdaten kann mit diesem Verfahren ein zweiteiliges 3D-Modell einer Person erstellt werden, das aus einem unbeweglichen Teil, ähnlich

eines Rigs, und einem verformbaren Teil zum Beispiel für Haut und Muskeln besteht (vgl. [12, S. 1]). Durch diese Trennung werden bei der Benutzung des Modells individuelle Abwandlungen wie beispielsweise eine Variation des Körpergewichts oder der Körpergröße möglich. Eine Anwendung dieser Idee lässt sich an vielen Stellen finden, nicht nur bei Veränderungen des BMI in einer Veröffentlichung von Piryankova et al. [13] oder bei Veränderungen der Körperproportionen bei Feng et al. [14], sondern auch bei der Deformierung des Körpers hin zu einem bestimmten Stil, wie es von Fleming et al. [15] durchgeführt wurde.

Die zeitlich gesehen nächste Veröffentlichung in diesem Forschungsgebiet stammt von der Forschungsgruppe „Perceiving Systems“ [16] am Max-Planck-Institut für Intelligente Systeme in Tübingen: Mit einem neuen Modell namens „MoSh“ [17] wurde von M. Loper et al. 2014 ein Verfahren vorgestellt, durch das eine Schätzung von menschlichen Körpern ausschließlich auf Basis einer Sequenz an Bewegungsdaten von der entsprechenden Person erstellt werden kann. Die Daten können mit einem üblichen, meistens auch in der Filmindustrie verwendeten System zur Bewegungserfassung aufgenommen werden - in der Veröffentlichung wurden zum Beispiel 67 Marker genutzt. Je mehr Marker genutzt werden, desto genauer wird logischerweise das Ergebnis, sodass mit genügend Markern im Normalfall nur Abweichungen um wenige Zentimeter entstehen (vgl. [17, S. 8]). Das Verfahren kann daher als günstige Alternative zum vollständigen Körperscan angesehen werden, sollte man keinen teuren Körperscanner zur Verfügung haben, auch wenn natürlich bei MoSh vergleichsweise viele Details der Körperoberfläche verloren gehen. Ein Nachteil dieser Methode in der Praxis wiederum ist, dass die vielen Marker zuerst korrekt am Körper der jeweiligen Person angebracht werden müssen, was bei der empfohlenen Menge auf jeden Fall länger dauern würde als ein normaler Körperscan mit anschließender Datenverarbeitung.

Das 2015 von G. Pons-Moll et al. aus der gleichen Gruppe veröffentlichte Körpermodell „Dyna“ [18] beschäftigt sich primär mit der dynamischen Darstellung von menschlichem Weichgewebe und baut auf das bereits besprochene SCAPE-Modell auf. Dabei werden 4D-Scans analysiert, die mit 60 Einzelscans pro Sekunde aufgenommen wurden. Auf diesem Weg müssen - anders als in der Veröffentlichung zu SCAPE - keine Marker mehr an der Versuchsperson angebracht werden, da der zweiteilige Aufbau des SCAPE-Modells nur anhand der einmalig aufgenommenen Sequenz an Einzelscans errechnet wird.

Eine weitere interessante Idee, ebenfalls aus Tübingen, wurde 2015 von S. Zuffi und M. Black mit dem Körpermodell „Stitched Puppet“ [19] präsentiert. Dieser Ansatz unterscheidet sich von anderen Körpermodellen vor allem darin, dass hier vorerst nur mit Körpersegmenten und deren Modellvarianten gearbeitet wird, die dann durch einen Optimierungsprozess verbunden werden. Zwar wird dieser Prozess in

der Veröffentlichung anhand von menschlichen Körpern gezeigt, jedoch ist dieser Ansatz aufgrund seiner Modularität durch die verschiedenen Körperteile auch zur Modellierung von Tieren verwendbar (vgl. [19, S. 8]).

Eine aktuelle Veröffentlichung von L. Pishchulin et al. [20] nimmt sich ebenfalls der Aufgabe an, ein statistisches Modell für menschliche Körperform zu entwickeln, das vor allem möglichst viele verschiedene Körperformen darstellen kann.

Zuletzt sind in diesem Kontext auch die Veröffentlichungen von Rachel McDonnell [21] und Carol O'Sullivan [22] zu nennen, die sich mit der Wahrnehmung virtueller Menschen in vielen computergrafischen Aspekten befassen. Im Fokus liegen dabei auch häufig Fragen der Darstellung, beispielsweise welchen Einfluss das Rendering auf die Wahrnehmung der virtuellen Charaktere hat.

Nach dieser kurzen Rundschau im Bereich der dreidimensionalen Körpermodelle fehlt nur noch die Erklärung des in dieser Arbeit verwendeten Modells, der sich das nächste Unterkapitel widmet.

2.3.3 SMPL

Als die vielleicht wichtigste Grundlage dieser Arbeit soll in diesem separaten Unterkapitel das SMPL-Modell zur Darstellung menschlicher Körper vorgestellt werden. Es wurde 2015 von M. Loper et al. [1] veröffentlicht und basiert auf den mehr als 2000 3D-Scans menschlicher Körper, die in der CAESAR-Datenbank vorhanden sind. Die Idee hinter diesem Modell macht sich beide am Anfang dieses Hauptkapitels erklärte Techniken zunutze: Zuerst wurde eine Hauptkomponentenanalyse im Raum beinahe aller in der CAESAR-Datenbank verfügbaren 3D-Körper durchgeführt. Aus mathematischer Sicht interessant dabei ist, dass jeder vorhandene Körper eine eigene Dimension der Analyse darstellt. Der Ursprung jeder dieser Dimensionen ist das Durchschnittsmodell, das sich aus allen verfügbaren Körpern bildet, während positive und negative Werte entsprechende Interpolationen zwischen dem Durchschnitt und dem jeweiligen Körper der Dimension repräsentieren. Um sich besser vorstellen zu können, was hierbei passiert, kann man sich den durch diesen Schritt gewonnen Vorteil vor Augen führen: Ohne eine Hauptkomponentenanalyse könnte ausschließlich zwischen den schon bekannten Körpern mithilfe von Modellvarianten interpoliert werden. Dies hätte durch das verwenden sämtlicher verfügbarer Körper erhebliche Leistungsverluste bei der Darstellung einer speziellen Kombination tausender Modellvarianten zur Folge. Das Ergebnis einer Hauptkomponentenanalyse mit diesen Daten ist ein transformierter Raum an Modellvarianten, dessen Achsen nun in absteigender Reihenfolge sortiert werden können, sodass diese die immer schwächer werdenden Varianzen in den Daten

repräsentieren. Durch diese Sortierung ist es nun möglich, anstatt der über 2000 CAESAR-Modellvarianten zum Beispiel nur noch zehn der transformierten Modellvarianten zu verwenden und dabei die vorhandenen CAESAR-Körper immer noch mit einer Genauigkeit von 95 % (vgl. [1, S. 8]) darstellen zu können. Ein weiterer Vorteil ist die einfache Speicherung neuer Körper: Wurde das Modell durch die obige Prozedur einmal definiert, kann man sämtliche darstellbare Körperformen nur durch wenige Fließkommazahlen in einem leicht austauschbaren Format ausdrücken und vermeidet große Dateien, die komplette 3D-Modelle enthalten.

Würde man eine praktische Limitation des SMPL-Modells nennen wollen, wäre das die Abhängigkeit von den statistischen Ursprungsdaten. Die Wahl der äußerst sorgfältig gepflegten CAESAR-Datenbank ist gut begründet, da diese Quelle eine breite Zusammenstellung vieler durchschnittlicher Körperformen darstellt. Erzeugt man auf Basis dieser Datenbank das SMPL-Modell, erhält man die in Abbildung 10 gezeigten Durchschnittsmodelle für Mann und Frau. Das auf der offiziellen Website erhältliche SMPL-Modell wird im FBX-Format für die meisten aktuellen Grafikpipelines und auch für Python-Programmierung angeboten.

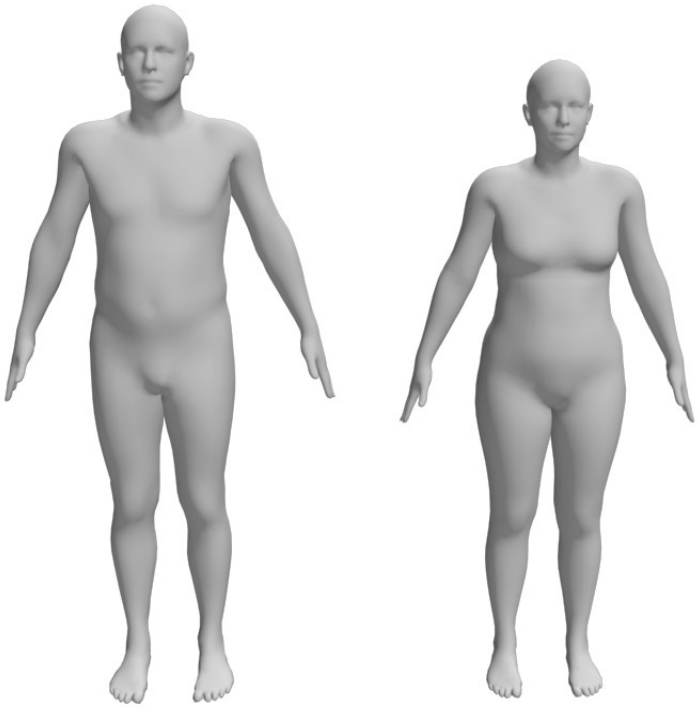


Abbildung 10: SMPL - CAESAR-Durchschnittsmodelle

Je mehr eine Körperform jedoch von diesen Durchschnittsdaten abweicht, desto weniger genau kann sich das verwendete SMPL-Modell daran anpassen. Würde man also auch extremere Körperformen darstellen wollen, müsste man separate SMPL-Modelle zum Beispiel für Kleinwüchsige, Hochwüchsige und natürlich auch Bodybuilder auf Basis anderer, spezialisierter Datenbanken erstellen. Zwar wird für die Generierung jedes neuen SMPL-Modells auch wieder eine größere Menge entsprechender qualitativer Daten benötigt, jedoch ist durch die Einfachheit des Modells die auch vergleichsweise einfache Adaption an beliebige Ursprungsdaten generell ein großer Vorteil (vgl. [1, S. 12]).

Was bis zu dieser Stelle in diesem Unterkapitel erklärt wurde, bezieht sich auf das individualisierte SMPL-Modell (Englisch „Fit“). Dabei werden zehn Faktoren, die den Einfluss der jeweils zehn wichtigsten transformierten Modellvarianten des SMPL-Modells auf CAESAR-Basis bestimmen, so weit wie möglich an die vorliegenden Ausgansinformationen angepasst. In Anlehnung an die Hauptveröffentlichung zu SMPL werden diese zehn Faktoren von nun an „β-Werte“ genannt. Die Ausgangsinformationen werden bisher üblicherweise durch einen Körperscan gesammelt. Da ein roher Körperscan aber nur als Punktwolke mit zufälliger Topologie vorliegt, muss vor der Findung der zehn passendsten β-Werte eine SMPL-Angleichung an den Körperscan erstellt werden. Eine solche Angleichung (Englisch „Alignment“) erhält man, wenn das durchschnittliche SMPL-Modell durch die Verschiebung jedes einzelnen Vertex in die Punktwolke - also die Rohdaten des Körperscans - eingepasst wird. Nach diesem Prozess sind natürlich weiterhin die körperspezifischen Details der Person sichtbar - Gesicht, Kopfform, Haare, Körperbau und eventuell sichtbare Knochen. Das liegt daran, dass der resultierende Körper einer SMPL-Angleichung an den Körperscan im Gegensatz zum individualisierten SMPL-Modell nicht auf den β-Raum auf CAESAR-Basis limitiert ist, sondern jeder Vertex einfach zur passendsten Stelle auf der Punktwolke verschoben wurde. Der Vorteil ist nun, dass man hiermit jeden Körperscan in eine einheitliche Repräsentation überführen kann, in der sich bestimmte Punkte auf dem Modell per gleichbleibendem Vertexindex ansprechen lassen. Außerdem werden auf diese Weise durch die geschlossene SMPL-Topologie etwaige Löcher im rohen 3D-Scan gefüllt, die unter ungünstigen Bedingungen manchmal entstehen können. Nicht nur für die später in dieser Arbeit durchgeführte Validierung, sondern auch für die automatische Findung der β-Werte anhand eines Körperscans für ein individualisiertes SMPL-Modell sind diese beiden Vorteile sehr wichtig. Man kann also zum Beispiel sicher sein, dass der in Abbildung 11 sichtbar gemachte Vertex mit der Nummer 2446 immer auf der linken Fingerspitze der Körpers liegen wird.

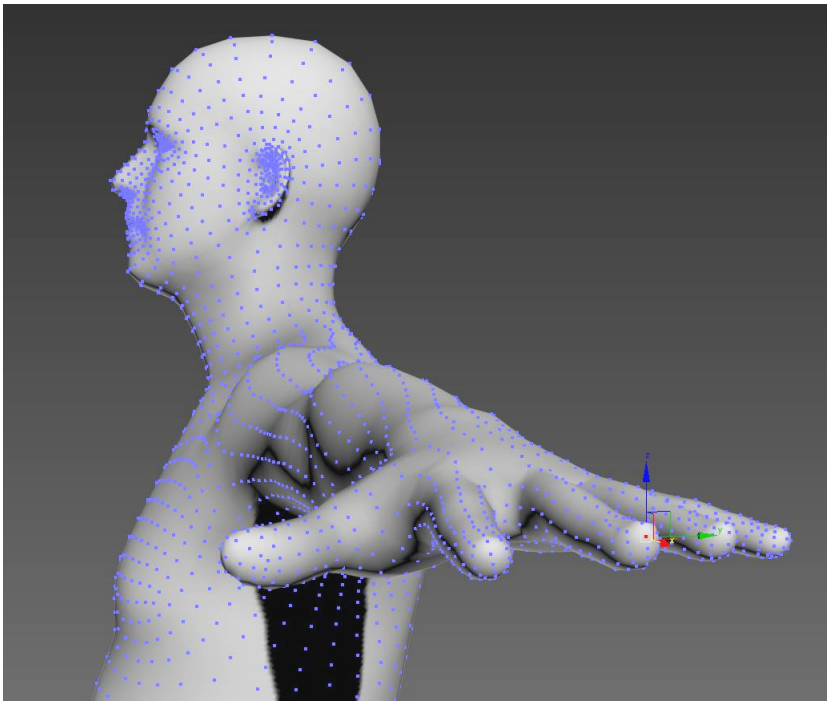


Abbildung 11: SMPL - Vertex 2446 an der linken Fingerspitze

Für eine der Hauptideen, auf die diese Arbeit aufbaut, ist die SMPL-Angleichung an den Körperscan jedoch nur ein Zwischenschritt, den es zu ersetzen gilt: Am Ende sollen die notwendigen Ausgangsinformationen so minimiert werden, dass beispielsweise schon einige Handabmessungen des Körpers anstelle eines vollen Körperscans für einen gut mit der realen Person übereinstimmenden Avatar ausreichen. Dieses Verfahren ist momentan schon in Benutzung und basiert auf einem linearen Regressor für die Abbildung von Handabmessungen des Körpers auf die β-Werte, muss jedoch noch verbessert werden.

2.4 Körperscans und Körpermaße

Eine praktische Grundlage dieser Arbeit bilden 3D-Körperscans und Körperabmessungen. Zum Erstellen der in dieser Arbeit verwendeten 3D-Scans konnte der 3D-Scanner am Max-Planck-Institut für Intelligente Systeme in Tübingen genutzt werden, der zur Zeit seines Aufbaus sogar die höchste Auflösung weltweit erzielte. In Sekundenbruchteilen werden Bilder von 22 Stereokameras geschossen, die dann im Nachhinein durch eine etwas rechenintensivere Prozedur in eine Punktwolke umgewandelt werden. (Vgl. [23])

Um jedoch den aufwändigen Körperscan für spätere Experimente zu umgehen, sollen im hier besprochenen alternativen Prozess Körperabmessungen genommen werden, aus denen dann auf statistisch begründete Weise der Avatar erzeugt wird, der am besten zu diesen Maßen passt. Daher hängt das Ergebnis auch von der Genauigkeit der genommenen Körpermaße ab, was einen Standard für solche Messungen notwendig macht. Hierfür werden die Messtechniken benutzt, die schon in den technischen Berichten zu CAESAR [24] definiert sind. Folgende Maße sind erfahrungsgemäß besonders wichtig für den Selbstavatar in virtueller Realität und werden daher auch bei der späteren Validierung berücksichtigt:

- Körpergröße (29)
- Körpergewicht (40)
- Armlänge (4)
- Schulterbreite (27)
- Brustumfang (8)
- Unterer Brustumfang (9)
- Taillenumfang (37)
- Taillenhöhe (39)
- Hüftumfang (23)
- Hüfthöhe (24)
- Schritthöhe (12)
- Oberschenkelumfang (31)
- Knöchelumfang (2)

In Klammern ist für jede Messung die Identifikationsnummer aus der Liste der CAESAR-Messungen [24, S. 31ff] aufgeführt, sodass die hier verwendete deutsche Übersetzung korrekt zugeordnet werden kann.

2.5 Avatare in virtueller Realität

Im Kontext dieser Arbeit wird verstärkt an der Verbesserung des Selbstavatars in virtueller Realität gearbeitet. Dieser unterscheidet sich von „normalen“ Avataren hauptsächlich darin, dass zusätzliche, technisch nicht ganz einfach umzusetzende Merkmale erfüllt sein sollten. Ein Selbstavatar wird aus der Ich-Perspektive erlebt, was meistens mit einem Spiegelszenario verbunden wird, sodass man den eigenen Körper auch ohne einen Blick nach unten besser wahrnehmen kann. Außerdem sollte ein Selbstavatar teilweise oder vollständig animiert und in seinen Dimensionen so gut wie möglich an die reale Person angepasst sein. Dass es sich lohnt, den Selbstavatar zu optimieren, zeigten B. J. Mohler et al. im Jahr 2010 durch Experimente, in welchen sie feststellten, dass Probanden mit einem personalisierten und animierten Avatar - also einem Selbstavatar - deutlich bessere Ergebnisse bei verschiedenen Aufgaben der räumlichen Einschätzung in virtueller Realität erzielen

konnten (vgl. [25, S. 11]). Aktuell kann die Anpassung des Avatars an die Person in mehreren Stufen vorgenommen werden: Im einfachsten Fall werden ein oder zwei Messungen verwendet, typischerweise Körperhöhe und Armspannweite, um einen vorgefertigten animierbaren Avatar uniform zu skalieren. Eine Balance zwischen Aufwand und Genauigkeit stellt das in dieser Arbeit besprochene Verfahren dar, das ein Körpermodell auf statistischer Grundlage an eine etwas größere Menge an Körperabmessungen anpasst. Zuletzt könnte ein vollständiger Körperscan durch manuelles Rigging und Skinning animierbar gemacht werden, was einen verhältnismäßig großen organisatorischen und technischen Aufwand darstellt.

Damit sich der Avatar jedoch natürlich bewegt, ist je nach Anzahl der vorhandenen Tracker zusätzlich eine Software für die Errechnung der Position und Rotation von nicht verfolgten Gelenken notwendig. Unter dem Begriff „IK“ (Inverse Kinematik) existieren zur aktuellen Zeit mehrere große Projekte, die diese Aufgabe sehr natürlich und zuverlässig erledigen können. Zum Zeitpunkt dieser Arbeit werden in einer Großzahl der Anwendungsfälle FinalIK [26] von RootMotion oder IKinema [27] verwendet, da diese beiden Softwarepakete aktuell als die besten in diesem Bereich gelten und es kaum Alternativen des gleichen Umfangs gibt.

Im April 2016 wurde das Forschungsfeld des Selbstavatars in virtueller Realität durch die Veröffentlichung des Head-mounted Displays HTC Vive [28] vor allem in praktischer Hinsicht revolutioniert: Damit war es fortan möglich, das durch die mitgelieferten Basisstationen ermöglichte Tracking für das Headset gleichzeitig noch für die Verfolgung von mehreren Trackern einzusetzen, die dann als Marker auf der realen Person angebracht werden können. Dies war der erste Weg, der vollständiges Körpertracking ohne ein zweites, oft sehr kompliziertes und teures System möglich machte und damit eine alltagstaugliche und einfache Alternative zu Systemen wie zum Beispiel Vicon [29] bot.

2.6 Zusammenfassung

Hauptgegenstand dieser Arbeit sind also SMPL-Körpermodelle, die eine lineare Kombination von Körpern aus der CAESAR-Datenbank darstellen. Aus Körperscans kann das entsprechende SMPL-Körpermodell schon mit hoher Genauigkeit ermittelt werden - auf Basis von Körperabmessungen hingegen entstehen zurzeit noch größere Fehler, die durch weitere Optimierung des Verfahrens ausgebessert werden müssen. Ein sehr interessanter Verwendungszweck für Avatare, die durch die letztere Methode generiert werden, sind Echtzeitszenarien wie das des Selbstavatars, bei welchen die Auslassung des vollen Körperscans mitunter eine große organisatorische und technische Erleichterung verspricht.

Mit dem in diesem Hauptkapitel erklärten Wissen ist es nun möglich, die genauen Anforderungen an diese Arbeit auf wissenschaftlicher und technischer Ebene noch einmal mit höherem Detailgrad zu definieren.

Die erste Aufgabe besteht darin, Werkzeuge zum Vergleich und zur Validierung der vorgestellten Arten von SMPL-Körpern zu entwickeln. Als Ausgangspunkt dienen die Körperscans und Körperabmessungen von zehn Probanden. Die Werkzeuge sollen der weiteren Entwicklung des Verfahrens zur Erzeugung von Avataren nur anhand von Körperabmessungen dienen und sind daher eine wichtige Grundlage für den technischen Teil der Arbeit. Am Anfang von Kapitel 3 wird als Vorbereitung darauf noch einmal ein genauer Überblick über die verschiedenen Arten von SMPL-Modellen geworfen, die dann für die Referenzierung im Text praktischere Namen erhalten. Aufgrund der festen Topologie von SMPL werden diese Werkzeuge lediglich die maximalen Ausmaße des 3D-Objekts und vordefinierte Distanzen zwischen Vertices vergleichen, die in speziellen Eingabedateien definiert werden können. Die Messung selbst liegt als eher einfache Aufgabe bei diesem Schritt also nicht im Fokus - vielmehr geht es um die Strukturierung und Aufbereitung der Eingabedaten und Messergebnisse für eine möglichst reibungslose Analyse.

Die zweite Aufgabe ist zum größten Teil eine praktische und technische Herausforderung, bei der ein Prototyp einer Vorlage für beliebige Szenarien mit Selbstavatar in virtueller Realität entwickelt werden soll. Einige bisher ungelöste Probleme bei der Umsetzung eines solchen Szenarios sollen dabei gelöst und dokumentiert werden, sodass das Projekt unkompliziert weiterentwickelt werden kann.

Die letzte Aufgabe besteht aus der Durchführung einer Studie, die untersucht, was die Teilnehmenden von einem SMPL-Avatar erwarten und welche Toleranz sie bezüglich seiner Körperform zeigen. Ein Bezug dieser Ergebnisse auf die Ergebnisse der Validierung soll dann in den letzten beiden Hauptkapiteln Aufschluss über weitere mögliche Anforderungen an einen SMPL-Selbstavatar geben.

3 Validierung des SMPL-Modells

Für die zukünftige Verbesserung von Selbstavataren basierend auf SMPL ist die erste praktische Aufgabe dieser Arbeit die Entwicklung von Werkzeugen für die Validierung, die einen einfachen Vergleich der verschiedenen Detailstufen bei der Erzeugung der Avatare zulassen. Das gesamte Projekt wird mit der populären Spiel-Engine Unity 3D [30] umgesetzt, für die unter anderem in der Programmiersprache C# eigene Komponenten entwickelt werden können. Die hauptsächliche Anforderung an die Validierungswerkzeuge besteht im automatisierten Vergleich von verschiedenen Körpern und dem Erzeugen von passenden Bildern, welche die Unterschiede zwischen den Körpern sichtbar werden lassen. Außerdem sollen die Ergebnisse im CSV-Format in eine Datei geschrieben werden, sodass die exakten Differenzen in späteren Schritten analysiert werden können.

Im Folgenden werden zur einfacheren Formulierung für die verschiedenen Detailstufen der erzeugten Avatare die am Ende von Kapitel 2.3.3 erwähnten englischen Begriffe verwendet. Zudem werden die personalisierten Avatare, deren β-Werte mithilfe eines linearen Regressors aus einer jeweils anderen Anzahl an Körperabmessungen errechnet wurden, im Folgenden als „Takes“ bezeichnet.

- **Alignment**:
  SMPL-Angleichung an einen Körperscan (Vertex-basiert)
- **Fit**:
  Personalisiertes SMPL-Modell (Zehn β-Werte auf Basis eines Körperscans)
- **Take 1**:
  Personalisiertes SMPL-Modell (Zehn β-Werte aus 12 Abmessungen)
- **Take 2**:
  Personalisiertes SMPL-Modell (Zehn β-Werte aus 6 Abmessungen)
- **Take 3**:
  Personalisiertes SMPL-Modell (Zehn β-Werte aus 4 Abmessungen)
- **Take 4**:
  Personalisiertes SMPL-Modell (Zehn β-Werte aus 2 Abmessungen)

Wichtig für die Validierung ist das Bewusstsein, dass sämtliche Takes maximal so genau werden können wie der Fit. Momentan sind jedoch die Abweichungen der Takes vom Fit teilweise signifikant. Trotzdem wird es in Zukunft mithilfe der hier entwickelten Werkzeuge hoffentlich möglich sein, sich nur auf Basis von Körperabmessungen immer weiter an die β-Werte des Fits anzunähern.

Welche Körperabmessungen bei der Berechnung der einzelnen Takes berücksichtigt wurden, ist in Abbildung 12 aufgeführt.

Körperabmessung	**Take 1**	**Take 2**	**Take 3**	**Take 4**
Körpergröße	✓	✓	✓	✓
Körpergewicht	✓	✓	✓	✓
Schritthöhe	✓	✓	✓	
Armlänge	✓	✓	✓	
Brustumfang	✓	✓		
Taillenumfang	✓	✓		
Hüftumfang	✓			
Taillenhöhe	✓			
Schulterbreite	✓			
Oberschenkelumfang	✓			
Knöchelumfang	✓			
Hüfthöhe	✓			

Abbildung 12: Validierung - Zuordnung von Körperabmessungen zu Takes

3.1 Erstellung von Avataren für einige Probanden

Um einen Probedatensatz zu haben, mit welchem die Validierungen durchgeführt werden können, wurden Körperscans von zehn Teilnehmenden erstellt, von denen fünf weiblich und fünf männlich sind. Alle Teilnehmenden haben zu Beginn des Experiments eine schriftliche Einverständniserklärung für ihre freiwillige Teilnahme an dieser Studie abgegeben. Der Mittelwert für das Alter aller Teilnehmenden liegt bei 29,5 Jahren mit einer Standardabweichung von 11,19 Jahren. Die Prozedur des Experiments wurde durch das lokalen Ethik-Komittee der Universität Tübingen genehmigt und befindet sich in Übereinkunft mit der Deklaration von Helsinki. Um sich in diesem Kapitel eindeutig auf die einzelnen Teilnehmenden beziehen zu können, werden im Folgenden Text und in allen Graphen die männlichen jeweils mit der Bezeichnung M1, M2, M3, M4 und M5 und die weiblichen jeweils mit der Bezeichnung W1, W2, W3, W4 und W5 referenziert.

Die Rohdaten der Körperscans liegen im OBJ-Format vor, können aber teils noch Löcher und andere Unregelmäßigkeiten wie geringfügige Verformungen enthalten. Das Alignment wird dann mithilfe eines Python-Skripts durchgeführt, das von der

Gruppe „Perceiving Systems“ am Max-Planck-Institut für Intelligente Systeme (im Folgenden PS-Gruppe genannt) intern zur Verfügung gestellt wird. Das Ergebnis ist ein 3D-Modell im PLY-Format, das für die weitere Verwendung ins OBJ-Format konvertiert wird, sodass es direkt in Unity importiert werden kann.

Auf Basis des Alignments wird im nächsten Schritt der Fit erstellt, ebenfalls mit einem Python-Skript, das intern von der PS-Gruppe zur Verfügung gestellt wird. Das Resultat des Fits ist dann kein 3D-Modell mehr, sondern besteht lediglich aus zehn β-Werten, die in einer einfachen Textdatei abgelegt werden.

Zudem werden die zwölf in Kapitel 2.4 aufgelisteten Körperabmessungen von allen Teilnehmenden registriert. Mit einem weiteren internen Skript der PS-Gruppe, das sich derzeit noch in der Entwicklung befindet, werden dann gemäß Abbildung 12 vier Takes errechnet, die jeweils eine andere Anzahl an Messungen berücksichtigen. Das Ergebnis dieser Takes sind wie beim Fit ebenfalls nur zehn β-Werte, die nach der Anwendung auf ein vorhandenes SMPL-Modell im FBX-Format den fertigen Körper im zugehörigen SMPL-Raum ergeben.

Die gerade beschriebenen Daten bilden die Grundlage für die folgenden Schritte bei der Validierung in diesem Hauptkapitel.

3.2 Validierung mit Skripten

Zunächst soll eine grobe Validierung der erzeugten Daten erfolgen, um mögliche Fehlerquellen durch die Skripte der PS-Gruppe oder deren falsche Benutzung ausfindig zu machen. Hierbei wird geprüft, wie sehr der Fit - also die bestmögliche Repräsentation des Körperscans im β-Raum - vom Alignment abweicht. Ein praktischer Weg hierfür sind Skripte, die den kompletten Datensatz in wenigen Minuten automatisiert gegenüberstellen und visuell vergleichbare Bilder berechnen. Dabei wird auch der in Abbildung 13 rot markierte Hüllkörper beider Varianten berechnet, um einen ersten ungefähren Eindruck der Abweichungen in Hinsicht auf Armlänge und Körpergröße bekommen zu können. Für diesen Teil der Validierung wurden Python-Skripte von Joachim Tesch aus der Gruppe Raum- und Körperwahrnehmung am Max-Planck-Institut für biologische Kybernetik verwendet, die für diese Zwecke auf eine Bibliothek der 3D-Software Blender [31] zugreifen.

Abbildung 13 zeigt exemplarisch ein Bild, das von den verwendeten Blender-Skripten für den Körper von Teilnehmer M1 erzeugt wurde.

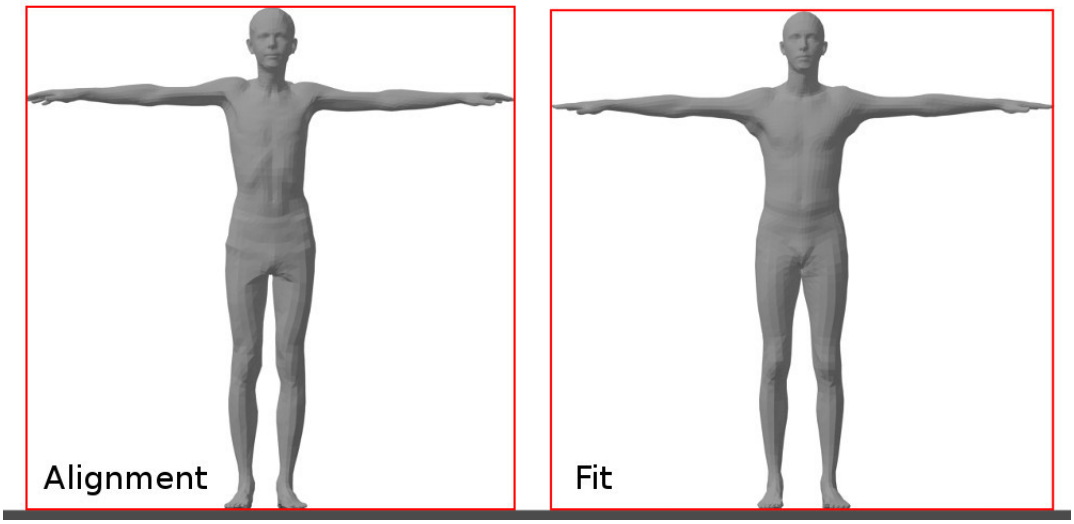


Abbildung 13: Validierung - Beispiel eines visuellen Vergleichs mit Skripten

Der rote Rahmen ist nicht Teil der originalen Bilder, sondern wurde hier lediglich zur Veranschaulichung eingefügt. Eine tabellarische Zusammenfassung der durch die Skripte errechneten Abweichungen in Zentimetern zwischen den Hüllkörpern von Alignment und Fit ist für alle Teilnehmenden in Abbildung 14 gezeigt.

Proband	**Breite**	**Höhe**	**Tiefe**
M1	5,11	1,62	0,33
M2	4,48	0,74	4,47
M3	0,81	1,32	3,07
M4	2,84	0,19	2,54
M5	4,61	0,76	5,49
W1	4,37	0,12	6,12
W2	4,12	0,52	2,08
W3	3,86	0,00	2,28
W4	0,71	0,35	1,96
W5	2,11	1,73	5,82

Abbildung 14: Validierung - Durch das Blender-Skript für alle Teilnehmenden der Validierung ermittelte Hüllkörperabweichungen in Zentimetern

Dass einige der Abweichungen bei ungefähr 5 cm liegen, lässt sich durch die Abstammung der beiden verglichenen Modelle begründen: Der Fit ist im Gegensatz zum Alignment, das exakt den 3D-Scan repräsentiert, hinsichtlich seiner Pose normalisiert - es werden lediglich β-Werte auf das Standardmodell angewandt. Waren die Teilnehmenden also nicht in perfekter Standardpose, was wohl bei den meisten 3D-Scans der Fall sein dürfte, wirkt sich die resultierende Abweichung der Pose auf das Messergebnis aus. In den meisten Fällen wurden die Arme nicht exakt waagerecht gehalten oder die Haltung der Person unterscheidet sich etwas vom Durchschnitt. Trotz dieser Faktoren, die sich nicht ohne Weiteres aus der Messung entfernen lassen, da ein Alignment kein Rig zur Veränderung der Pose hat, werden vor allem in der Höhe erstaunlich gute Werte unter 2 cm erreicht. Damit wurde gezeigt, dass der Fit-Vorgang von den Skripten der PS-Gruppe zuverlässig ausgeführt wird und als Grundlage für die Verbesserung der Takes dienen kann.

3.3 Validierung in Unity

Nach der ersten Validierung von Alignment und Fit der Probanden kann nun die Entwicklung der Unity-Werkzeuge beginnen, die als Grundlage für die weitere Verbesserung der Takes genutzt werden sollen.

Das Ziel bei diesem Schritt ist - ähnlich zum vorhergehenden Unterkapitel - der Vergleich der verschiedenen Körperrepräsentationen. Dabei soll jedoch nicht nur auf den Hüllkörper geachtet werden, sondern auch auf vordefinierte Abstände von Vertices. Dies gibt nicht nur einen detaillierteren Einblick in die Unterschiede der Genauigkeit für verschiedene Körperregionen, sondern erlaubt auch die erneute Messung mit beliebig vielen, eigens definierten Vertexdistanzen. Doch vor der Entwicklung dieser Funktionalität steht die Visualisierung der Messpunkte, anhand derer die Distanzen ermittelt werden. Dies ist vor allem notwendig, da die verwendeten Vertices allein aufgrund ihres Vertexindex ermittelt werden und Unity standardmäßig Optimierungen an importierten 3D-Modellen vornimmt, die unter Umständen eine völlig andere Vertexreihenfolge zur Folge haben. Um diese Funktionalität wiederverwendbar zu gestalten, werden die anzuzeigenden Vertexindizes aus einer Textdatei geladen, die mit folgenden Werten gefüllt wurde:

FingerTipLeft = 2446
FingerTipRight = 5906
HeadTop = 412
HeelLeft = 3466
ShoulderLeft = 3012
ShoulderRight = 5291
ShoulderTop = 4272
EllbowRight = 5130
EllbowLeft = 1658
BellyButton = 3501
NippleRight = 6489
NippleLeft = 3041

Crotch = 1210	HipRight = 4165
BackBellyButton = 3022	HipLeft = 677
ChestRight = 4892	WaistRight = 4963
ChestLeft = 1420	WaistLeft = 1489

Die Wahl dieser Vertexpunkte muss nicht perfekt mit der anatomischen Position auf dem Körper übereinstimmen und kann nach Augenmaß durchgeführt werden, da es bei der Auswertung nur um relative Unterschiede zwischen den Körpern geht und die SMPL-Topologie sich immer in gleicher Weise auf den Körper verteilt. Führt man die entsprechende Szene des Unity-Projekts aus, sieht man wie in Abbildung 15 an den vorher definierten Vertexpositionen kleine, rote Punkte platziert. So kann man erkennen, dass die Vertexpositionen auch wirklich mit den durch Vertexindizes angegebenen Messpunkten übereinstimmen, die zuvor in einer beliebigen 3D-Software herausgesucht wurden.

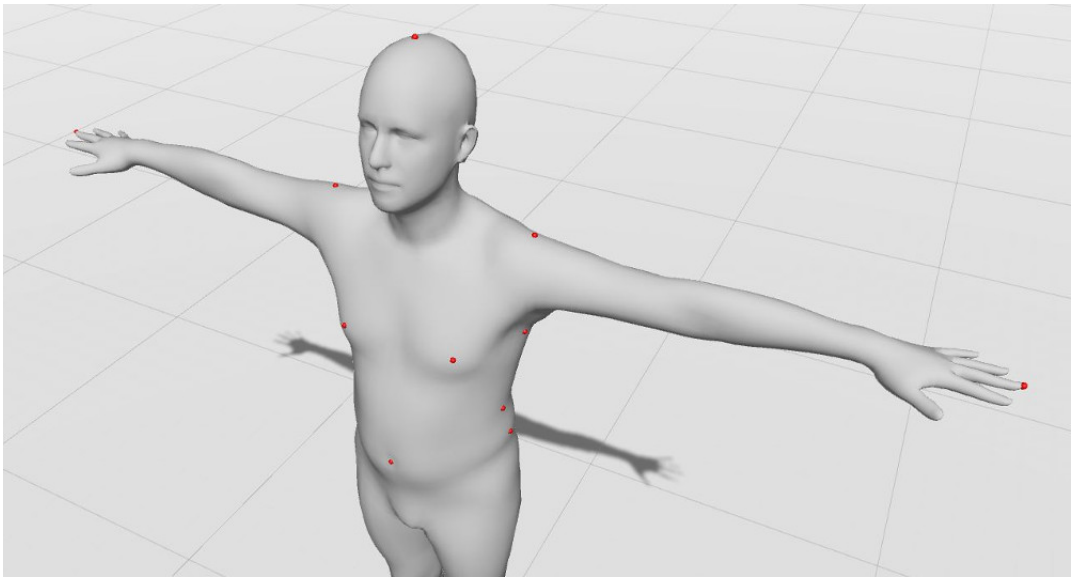


Abbildung 15: Validierung - Das SMPL-Standardmodell auf Basis von CAESAR mit Markierungen für die ausgewählten Vertices

Ein Problem auf dem Weg zu dieser Visualisierung war die Transformation des 3D-Objekts in der Hierarchie. Der erste Ansatz wäre natürlicherweise, die Position der definierten Vertexpunkte zu ermitteln und an die gleiche Position auch das rote Markerobjekt zu setzen. Mit einem *MeshFilter*, der alle Daten zum 3D-Modell beinhaltet, würde man also diese Funktion formulieren:

```
Vector3[] GetVertices(GameObject meshObject)
{
    return meshObject.GetComponent<MeshFilter>().mesh.vertices;
}
```

Das Problem ist jedoch, dass diese Vertices im lokalen Raum des Objekts liegen. Um die absoluten Weltkoordinaten der Vertices zu bekommen, musste daher eine Funktion zur Konvertierung der Vertexpositionen geschrieben werden, die Zugriff auf eine Transform-Komponente zur Umrechnung der Vertices hat:

```
Vector3[] GetWorldSpaceVertices(Vector3[] local, Transform trans)
{
    // Transform each local vertex into world space:
    return local.ToList().ConvertAll<Vector3>(
        vertex => trans.TransformPoint(vertex)
    ).ToArray();
}
```

Sinnvollerweise wird als Transform-Komponente im aktuellen Problemfall immer die nächstgelegene übergeben, weil diese auch beim Rendering des Körpers dessen Vertices transformiert. Nachdem man sich nun sicher sein kann, dass in Unity die Vertexindizes auch auf die richtigen Vertices verweisen, die zur Messung benutzt werden sollen, kann der Vergleich selbst programmiert werden.

In einer weiteren Textdatei sind hierfür Distanzen zwischen jeweils zwei der vorher definierten Vertexpunkte anzugeben, die für die spätere Anwendung auch ihren eigenen Namen bekommen. Folgendes ist danach in der Textdatei enthalten:

HeelLeft, HeadTop, **Height**
HeelLeft, Crotch, **Inseam**
WaistLeft, HeelLeft, **WaistHeight**
HipLeft, HeelLeft, **HipHeight**
FingerTipRight, FingerTipLeft, **ArmSpan**
EllbowRight, EllbowLeft, **ElbowSpan**
ShoulderRight, ShoulderLeft, **ShoulderWidth**
NippleRight, NippleLeft, **BreastWidth**
ChestRight, ChestLeft, **ChestWidth**
HipRight, HipLeft, **HipWidth**
WaistRight, WaistLeft, **WaistWidth**
BackBellyButton, BellyButton, **BellyDepth**

Zur besseren Verdeutlichung der Zugehörigkeit wurden die definierten Namen der Distanzen fett gedruckt. Jeder in dieser Datei verwendete Name für einen Vertexpunkt muss vorher in der am Anfang gezeigten Datei definiert worden sein.

Nun fehlen nur noch die Körper selbst, die verglichen werden sollen. Zum Zeitpunkt der Erstellung dieser Arbeit konnte leider keine funktionierende Lösung für den Import von OBJ-Dateien in Unity zur Laufzeit ausfindig gemacht werden, bei der die Reihenfolge der Vertexpunkte nicht manipuliert wird. Der einzige Weg schien die Erstellung eines AssetBundles zu sein, einer Datei, in der importierte Unity-Objekte im bereits für Unity optimierten Format vorliegen. Diese Datei kann dynamisch zur Laufzeit geladen werden und die enthaltenen 3D-Meshes - sofern sie bei der Erstellung des AssetBundles entsprechend importiert wurden - stehen mit der korrekten Vertexreihenfolge zur Verfügung. Um den Prozess der AssetBundle-Erstellung zu vereinfachen, wurde ein separates Unity-Projekt angelegt, das durch passende Editor-Skripte den Import der Quelldaten und das Verpacken der AssetBundles übernimmt. Für jeden Teilnehmenden wird dabei ein AssetBundle erstellt. Der folgende Code weist Unity an, bei jedem Importvorgang die Optimierung der Meshes zu deaktivieren und damit die Vertexreihenfolge beizubehalten:

```
class DisableOptimizeOnImport : AssetPostprocessor
{
    public void OnPreprocessModel()
    {
        (base.assetImporter as ModelImporter).optimizeMesh = false;
    }
}
```

Die generelle Idee hierbei ist, dass von *AssetPostprocessor* abgeleitete Klassen bei bestimmten Ereignissen des Importvorgangs benachrichtigt werden und darauf über die Basisklasse in den Prozess eingreifen können.

Durch weitere Funktionen der Programmierschnittstelle des Unity-Editors werden dann automatisch alle SMPL-Varianten eines Teilnehmenden gefunden und in ein AssetBundle verpackt. So können beliebig viele Varianten hinzugefügt werden. Hat man das jeweilige AssetBundle für jeden Teilnehmenden in den Ordner mit den Eingabedateien des Hauptprojekts kopiert, kann der Vergleich beginnen.

Um den Vergleichsmechanismus zu kapseln, wurde dieser in eine Klasse namens *SMPLComparer* verlagert, die im Konstruktor zwei *GameObject*s übergeben bekommt, welche dann beim Aufruf der Funktion *Compare* verglichen werden. Das Ergebnis wird vorerst in einer internen Struktur vorgehalten, sodass je nach Notwendigkeit entweder eine CSV- oder TXT-Ausgabe generiert werden kann.

Folgender Codeausschnitt ist ein Beispiel für die Benutzung dieser Klasse:

```
Dictionary<string, int> measurementVertices =
    InputFileParser.ParseVertexNumbers("Input/MeasurementVertices.txt");
List<string[]> distancesToMeasure =
    InputFileParser.ParseDistanceMeasurements("Input/Distances.txt");
List<GameObject> participantVariations = new List<GameObject>();
// ...
participantVariations.ForEach(
    variation1 => participantVariations.ForEach(
        variation2 => {
            SMPLComparer comparison = new SMPLComparer(
                variation1,
                variation2
            );
            comparison.Compare(
                measurementVertices,
                distancesToMeasure
            );
            Debug.Log(comparison.GetTXTComparison());
        }
    )
);
```

Zuerst werden in diesem Beispiel die definierten Vertexpunkte und die zu prüfenden Distanzen aus der jeweiligen Textdatei gelesen. Zur Liste *participantVariations* werden dann alle Variationen des gerade behandelten Teilnehmenden als *GameObject* hinzugefügt, was durch einen Kommentar ersetzt wurde, da dieser Vorgang für das Beispiel nicht relevant ist. Für alle Kombinationen der zu vergleichenden Varianten wird dann ein *SMPLComparer* erstellt, der Vergleich ausgeführt und eine leserliche Repräsentation der Ergebnisse auf der Konsole ausgegeben.

Eine größere Herausforderung stellte die Berechnung von passenden Bildern mit einer variablen Anzahl an SMPL-Körpern dar. Hierfür werden orthogonale Kameras verwendet, deren Bildausschnitt dynamisch an die aufsummierte Breite oder Tiefe aller Körper angepasst wird. Die Körper werden zu Beginn des Vergleichs mit den Füßen auf dem Weltursprung zentriert und dann mit dem konfigurierten Abstand so im Raum verteilt, dass die Szene von drei Kameras aufgenommen und in eine Bilddatei gespeichert werden kann. Es sind hiermit Aufnahmen von vorne, links und hinten möglich. In Abbildung 16 wurde die Kamera für die frontale Ansicht bereits auf die gerade geprüften Körper eingestellt.

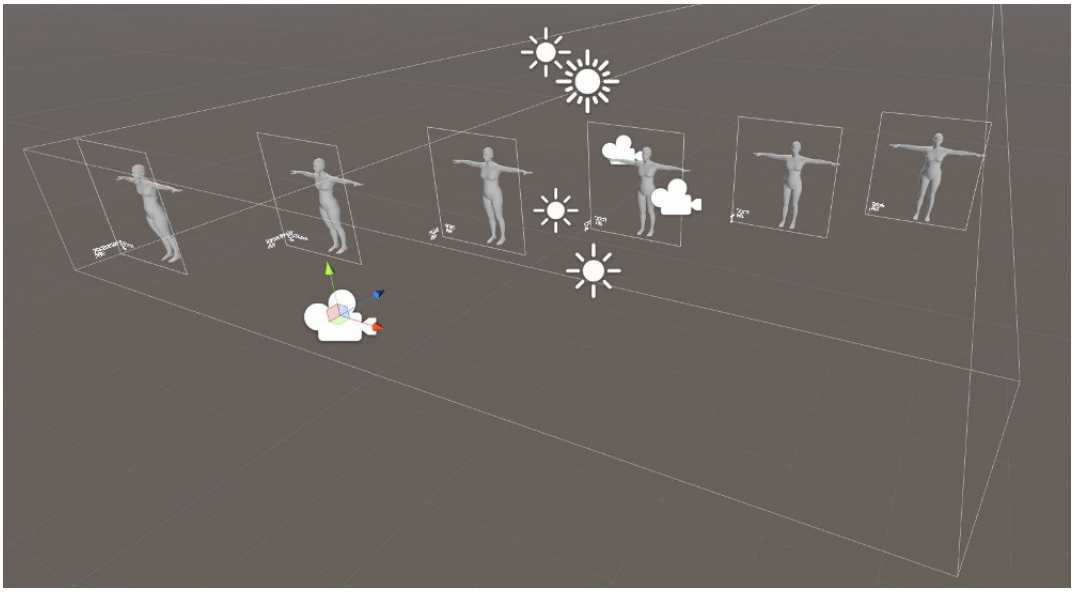


Abbildung 16: Validierung - Die frontale Kamera wurde auf die Körper eingestellt

Bei orthogonalen Kameras spielt die Versetzung der Körper in der Tiefe innerhalb der vorderen und hinteren Clippingebene keine Rolle. Daher erlaubt eine einmalige Aufstellung wie in Abbildung 16 direkt ein Rendering von allen drei Kameras. Die Körper werden nach einem ersten, horizontalen Durchgang für einen zweiten Durchgang vertikal im Raum verteilt. Ein Beispiel für ein horizontales Bild, das auf diese Weise berechnet wurde, zeigt Abbildung 17.

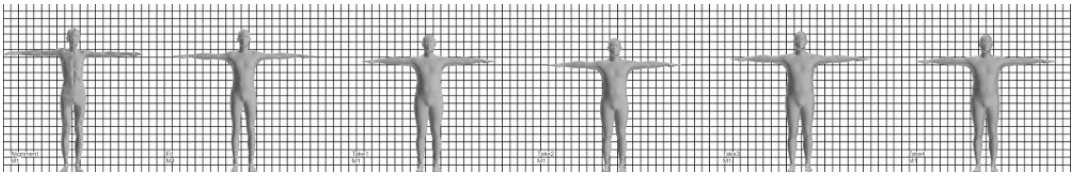


Abbildung 17: Validierung - Das Ergebnis eines Unity-Vergleichs für M1

Zusätzliche Optionen erlauben die Einstellung der verwendeten Auflösung und die Ersetzung des normalerweise transparenten Hintergrunds durch ein Gitter, das den visuellen Vergleich der Körper erleichtern soll. Um die Beschriftungen an den Körpern besser sichtbar zu machen, wurde das Unity-Skript *Outline* verwendet, welches den weißen Rand um den angezeigten Text dynamisch erzeugt. Zusätzlich wird eine geringfügige Abwandlung des UI-Text-Shaders mit Backface Culling verwendet. UI-Texte mit diesem Shader sind nur noch von vorne sichtbar, sodass sich

die Beschriftungen für die vordere und hintere Kamera nicht mehr überlagern, da sie nun wegen ihrer Ausrichtung nur noch für die jeweilige Kamera sichtbar sind. Damit die Körper ungeachtet ihres Ursprungskoordinatensystems immer in die richtige Richtung zeigen, muss der Index eines SMPL-Vertex angegeben werden, der auf der vorderen, vertikalen Mittellinie des Modells liegt. Dafür könnten zum Beispiel Vertices auf der Nase oder auf dem Bauchnabel ausgewählt werden. Mit dieser zusätzlichen Information können alle importierten Körper mit dem Gesicht zur vorderen Kamera zeigend ausgerichtet werden.

Die gerade erklärten Einstellungen lassen sich in einer Hauptkomponente namens *ComparisonController* tätigen, die für den kompletten Ablauf des Unity-Vergleichs zuständig ist. Eine mögliche Konfiguration ist in Abbildung 18 gezeigt.

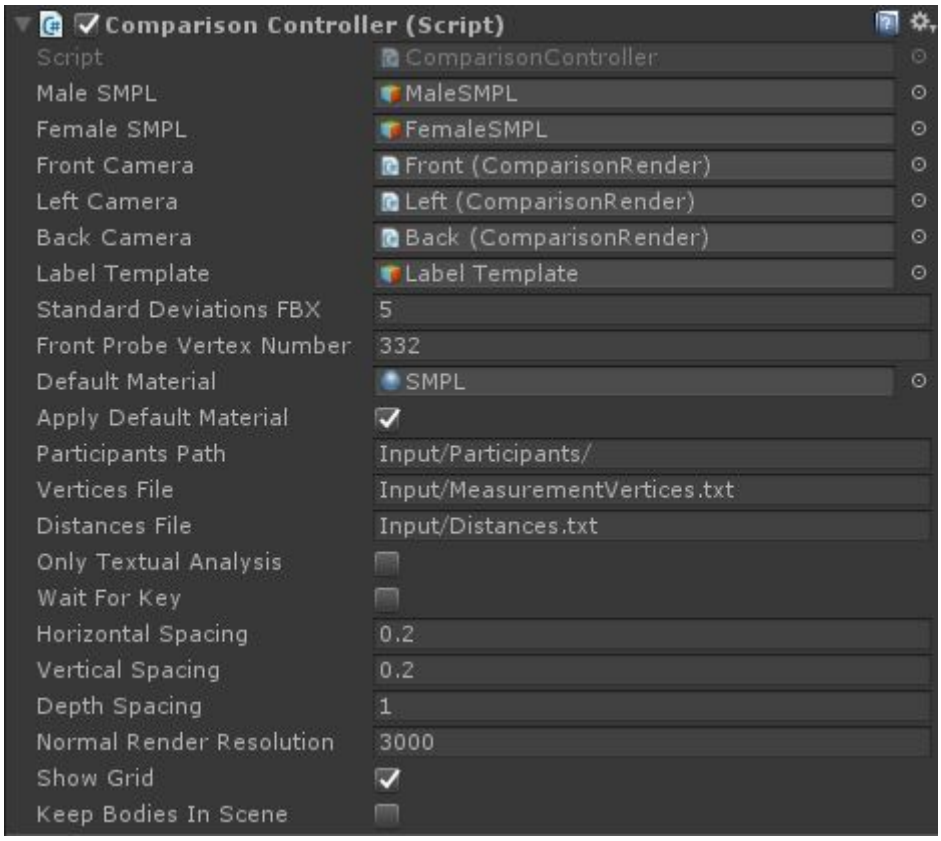


Abbildung 18: Validierung - Eine Hauptkomponente steuert den Vergleich in Unity

3.4 Ergebnisse der Validierung

Nachdem in Unity die verschiedenen Detailstufen der Körper aller Teilnehmenden verglichen wurden, können die Ergebnisse der Analyse nun besprochen werden. Als Grundlage dafür dienen Zusammenstellungen der generierten Bilder und Graphen, die Statistiken zu den gemessenen Abweichungen zeigen.

Für einen genauen Blick auf die unterschiedlichen Detailstufen der Körper zeigt Anhang 1 die Teilnehmerinnen und Anhang 2 die Teilnehmer in horizontaler Aufstellung. Da manche Körperregionen visuell besser in der Breite als in der Höhe verglichen werden können, zeigen Anhang 3 die Teilnehmerinnen und Anhang 4 die Teilnehmer nochmals in vertikaler Aufstellung.

Auffallend ist zunächst die generelle Abrundung der Körperform beim Fit, die durch die Repräsentation des Alignments im CAESAR-Raum entsteht. Dadurch gehen wie erwartet einige individuelle Merkmale wie zum Beispiel Weichgewebe am Körper der Frauen oder die Rippenknochen von M1 verloren. Trotzdem erzielt der Fit allgemein gute Ergebnisse bei der Körpergröße und auch hinsichtlich des optischen Eindrucks des Körpergewichts. Einen detaillierten Vergleich der Differenzen zwischen Alignment und Fit getrennt nach Geschlecht enthält Anhang 5. Dort liegen die Abweichungen in den meisten Fällen zwischen 0 cm und 3 cm, abhängig von der Person und der gemessenen Distanz kommt es teilweise auch zu Abweichungen zwischen 4 cm und 6 cm. Wie man jedoch an den Bildern der ersten vier Anhänge sehen kann, fallen diese mit bloßem Auge kaum auf.

Ein visueller Vergleich der verschiedenen Takes mit dem Fit deutet bei vielen der Teilnehmenden darauf hin, dass die Körpergröße innerhalb der Takes mitunter sichtbar variiert. Bei den Frauen scheint sich die Körperform kaum zu verändern und das Körpergewicht visuell dasselbe zu sein. Bei den Männern scheint das Körpergewicht bei der Generierung der Takes zuzunehmen, besonders sichtbar wird dies bei M1. Eine Erklärung hierfür könnte der generell relativ geringe BMI der Teilnehmer sein. Allgemein fällt auf, dass die Takes sich tendenziell wieder dem SMPL-Standardmodell für das jeweilige Geschlecht nähern. Dieses Verhalten lässt sich auf den linearen Regressor für die Berechnung der Takes zurückführen, der die besten Ergebnisse für Körper liefert, die dem Durchschnitt ähnlich sind.

Ein Blick auf die gemessenen Abweichungen zwischen Fit und Take4 in Anhang 6 macht deutlich, dass Take4, der nur aus Körpergröße und Körpergewicht errechnet wurde, teils auch entsprechend stark vom Fit mit den optimalen β-Werten abweicht. Ein zusammenfassender Vergleich beider Abweichungen, sowohl zwischen Alignment und Fit als auch Fit und Take4, ist in Anhang 7 für die Teilnehmerinnen und in Anhang 8 für die Teilnehmer gezeigt. Damit wird bestätigt, dass Take4 bis

auf wenige Ausnahmen stärker vom Fit abweicht als der Fit vom Alignment. Auch erweist sich durch Abweichungen bis zu 12 cm der Eindruck als richtig, dass die Körpergröße der Takes vor allem bei den Männern sichtbar verändert ist. Bei den Frauen weichen vor allem Messungen am Oberkörper zwischen Fit und Take4 mit 0 cm bis 4 cm vergleichsweise wenig ab. Bei den Männern sind diese Werte ebenfalls überdurchschnittlich gering, jedoch sind auch hier individuelle Abweichungen zu beobachten, beispielsweise bei der Hüftbreite von M1, deren Abweichung sehr wahrscheinlich mit dem niedrigen BMI des Teilnehmers zusammenhängt.

Für eine detailliertere Analyse der in diesem Kapitel betrachteten Abweichungen enthält Anhang 9 für jeden Teilnehmenden der Validierung eine direkte Gegenüberstellung von Alignment und Fit und von Fit und Take4. In diese Zusammenstellungen wurden auch Seitenansichten einbezogen, die einen genaueren optischen Vergleich ermöglichen. Zusätzlich sind die Körperabmessungen der entsprechenden Person sichtbar, sodass zum Beispiel auch der BMI berechnet werden kann.

Das momentane Vorgehen scheint - zumindest auf Basis des optischen Vergleichs in den ersten vier Anhängen - nicht merklich von den unterschiedlichen Anzahlen an Körperabmessungen zu profitieren. Es wäre daher vermutlich am sinnvollsten, den Regressor, der die Körperabmessungen in β-Werte umwandelt, für unterschiedliche Verwendungszwecke zu optimieren, sodass am Ende mehrere Regressoren mit je speziellen Teilmengen der verfügbaren Körperabmessungen arbeiten würden.

Weitere Gedanken zur Optimierung dieses Verfahrens und zu alternativen Fragestellungen bezüglich der Validierung sind in Kapitel 6 aufgeführt.

4 Implementierung eines Selbstavatars

Als einer der interessantesten Anwendungsfälle eines personalisierten SMPL-Modells soll in dieser Arbeit der Selbstavatar behandelt werden. Wie im vorherigen Kapitel bei der Validierung wird zur Umsetzung eines prototypischen Selbstavatars ebenfalls die Spiel-Engine Unity verwendet. Die folgenden Unterkapitel gliedern die Vorgehensweise bei der Entwicklung nach gelösten Problemen.

Bis auf die Einrichtung der inversen Kinematik, die einmalig vorgenommen wird, müssen bei der Initialisierung jedes Selbstavatars bestimmte Schritte durchgeführt werden. Der fertige Prototyp kombiniert alle diese Schritte in einem zentralisierten Ablauf mit passenden Textanweisungen, die in virtueller Realität angezeigt werden. Wiederverwendbare Komponenten dieses Teils der Implementierung wurden in ein Unity-Paket namens *ViveIKHelpers* ausgelagert, sodass sie später direkt in anderen Projekten Verwendung finden können.

Abbildung 19 zeigt im rechten Teil das fertige Szenario mit einem virtuellen Spiegel, in welchem man seinen Avatar betrachten kann. Im linken Teil ist das zum Moment der Aufnahme des Selbstavatars passende Bild des Benutzers, der im HMD das rechts gezeigte Szenario erlebt.

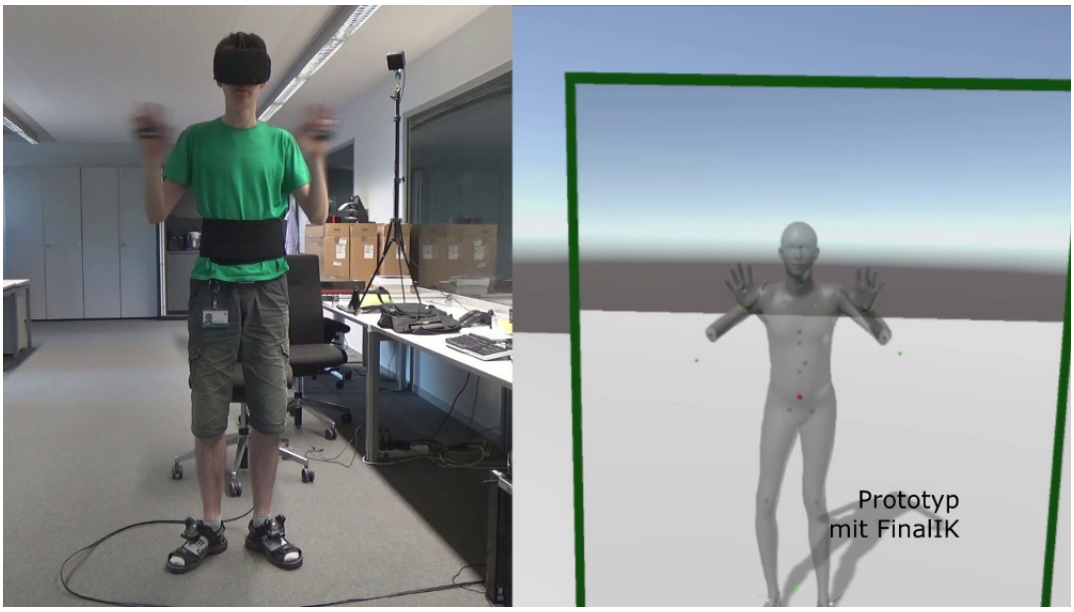


Abbildung 19: Selbstavatar - Das fertige Szenario vor dem virtuellen Spiegel

4.1 Zuordnung der Tracker

Für dieses Projekt wird das HTC Vive als HMD zusammen mit fünf der separat erhältlichen Tracker eingesetzt, die mit demselben System verfolgt werden können. Eine Zusammenstellung der verwendeten Hardware zeigt Abbildung 20.

Abbildung 20: Selbstavatar - Die zur Umsetzung verwendete Hardware

Links oben in Abbildung 20 ist eine der beiden Basisstationen auf einem Stativ zu sehen, die zusammen mit dem links unten abgebildeten Headset ausgeliefert werden. Die im rechten Teil der Abbildung gezeigten Tracker sind als zusätzliches Zubehör seit März 2017 für Entwickler erhältlich.

Üblicherweise bekommen die Tracker in der Reihenfolge des Einschaltens Indizes zugewiesen, anhand derer sie durch die SteamVR-API referenzierbar werden. In Unity können diese Indizes durch Komponenten aus dem Paket namens „SteamVR Plugin“ beliebigen Objekten zugewiesen werden, die dann an am Anfang jedes Einzelbilds an die aktuelle Position des Trackers platziert werden. Das Paket wird kostenlos im *Unity Asset Store* zum Download angeboten und erweitert Unity-Projekte um Funktionalität für die erweiterte Interaktion mit HMDs.

Abbildung 21 zeigt, wie die Tracker an einer Person angebracht werden können. An die Tracker für die Hände (links oben) wurden Klettbänder angebracht, mit denen das Gerät auf dem Handrücken der Person befestigt wird. Für die Befestigung an den Füßen (links unten) haben zwei der Tracker zusätzliche Metallklemmen,

mit denen sie entweder fest am eingeflochtenen Schnürsenkel eines geschlossenen Schuhs oder wie im Bild am Verschluss einer Sandale befestigt werden können. Zuletzt wird der im rechten Teil der Abbildung gezeigte Tracker mithilfe eines Gurts um den Bauch geschnallt, sodass der Tracker sich immer etwa auf der gleichen Höhe und möglichst mittig am Rücken befindet.

Abbildung 21: Selbstavatar - Anbringung der HTC Vive Tracker

Um aber bei der Anbringung der Tracker an der realen Person nicht auf eine spezielle Reihenfolge des Einschaltens achten zu müssen, werden die automatisch vergebenen Indizes in einer Kalibrierungspose neu zugeordnet. Dafür muss die Person mit den bereits angebrachten Trackern sich in der T-Pose aufstellen, bei der die Arme auf Schulterhöhe nach rechts und links ausgestreckt werden. Die für diesen Schritt zuständige Unity-Komponente namens *TPoseCalibrator* bietet eine Funktion zur Vorbereitung, bei der so lange gewartet wird, bis mindestens fünf der verwendeten Tracker aktiv sind. Um diese Funktionalität möglichst wiederverwendbar umzusetzen, wird im folgenden Code zuerst die Möglichkeit gegeben, alle der fünfzehn möglichen Geräteindizes als Array für die einfachere Iteration über alle theoretisch ansprechbaren Geräte zusammenzufassen. Die unterhalb definierte Funktion *GetGenericTrackersAvailable* prüft dann für jeden dieser Indizes, ob der „SteamVR Runtime“ dafür ein verbundenes, valides, getracktes und generisches Gerät vorliegt und gibt die passenden Geräte in einer Liste zurück.

```
private int[] DeviceIndices
{
    get
    {
        return new int[] {
            (int)SteamVR_TrackedObject.EIndex.Device1,
            (int)SteamVR_TrackedObject.EIndex.Device2,
            // ...
            (int)SteamVR_TrackedObject.EIndex.Device15
        };
    }
}

private List<SteamVR_Controller.Device> GetGenericTrackersAvailable()
{
    return DeviceIndices.Where(index => {
        SteamVR_Controller.Device device =
            SteamVR_Controller.Input(index);
        return device.connected && device.valid &&
            device.hasTracking &&
            Valve.VR.OpenVR.System.GetTrackedDeviceClass((uint)index)
            == Valve.VR.ETrackedDeviceClass.GenericTracker;
    }).ToList().ConvertAll(index => SteamVR_Controller.Input(index));
}
```

Eine zweite Funktion zur Durchführung der eigentlichen Kalibrierung erkennt anhand der Trackerpositionen die richtige Zuordnung der Indizes. Um für das Beispiel unwichtige Referenzen zu vermeiden, werden im folgenden Code die ermittelten Indizes einer neuen Variable statt den zugehörigen Unity-Objekten zugewiesen.

```
List<SteamVR_Controller.Device> devices =
    GetGenericTrackersAvailable();
devices.Sort((dev1, dev2) =>
    -dev1.transform.pos.y.CompareTo(dev2.transform.pos.y));
// Assign the new order to all tracked objects and swap left and
    right trackers for hand and feet if needed:
bool swapHandLR = devices[0].transform.pos.x <
    devices[1].transform.pos.x;
bool swapFeetLR = devices[3].transform.pos.x <
    devices[4].transform.pos.x;
int handRightIndex = (int)devices[swapHandLR ? 1 : 0].index;
int handLeftIndex = (int)devices[swapHandLR ? 0 : 1].index;
int pelvisIndex = (int)devices[2].index;
int footRightIndex = (int)devices[swapFeetLR ? 4 : 3].index;
int footLeftIndex = (int)devices[swapFeetLR ? 3 : 4].index;
```

Zuerst werden alle Geräte nach ihrer Position auf der Y-Achse sortiert. Für die Arme und Füße wird danach ermittelt, ob die jeweiligen Paare für eine eindeutige Sortierung noch getauscht werden müssen, da bei diesen je nach Haltung und Anbringung der Tracker die Reihenfolge vorerst unbestimmt ist.

4.2 Erkennung der anatomischen Gelenke

Eine technisch anspruchsvolle Aufgabe stellt es dar, sich bei der Initialisierung des Selbstavatars dynamisch an beliebige Positionierungen der Tracker auf Händen und Füßen anzupassen. Während man davon ausgehen kann, dass der Tracker am Rücken jedes Teilnehmenden durch den Gurt immer ungefähr auf gleicher Höhe, mit gleichem Abstand zur Mitte des Beckens und in gleichem Winkel zum Körper angebracht ist, kann ein fest definierter Abstand zwischen einem Tracker an Hand oder Fuß und dem zugehörigen anatomischen Gelenk zu sichtbaren Abweichungen führen, da man die Tracker dort auf verschiedene Arten befestigen kann und verschiedene Hände oder Füße sich teils deutlich in Größe und Proportion unterscheiden. Die Folge einer solchen Abweichung kann sein, dass sich die virtuellen Körperteile nicht an derselben Stelle wie die realen Körperteile befinden und eine Rotation um die anatomischen Gelenke sich entsprechend falsch auf den virtuellen Körper überträgt. Um das beste Ergebnis zu erzielen, muss also das Gelenk des virtuellen Körpers bei jeder Bewegung der Person möglichst exakt auf die Position des Gelenks am realen Körper gesetzt werden.

Da es zum Zeitpunkt dieser Arbeit scheinbar keine offizielle Lösung dieses Problems mit größerem Bekanntheitsgrad gibt, wurde ein eigener Algorithmus entwickelt, der dieses Problem während der Laufzeit löst. Dafür muss jeder Teilnehmende lediglich für wenige Augenblicke jedes der anatomischen Gelenke auf allen drei Achsen rotieren. Das Ergebnis ist der statistisch am wenigsten durch das Weltkoordinatensystem gereiste Punkt, der sich selbst im lokalen Koordinatensystem des Trackers befindet und mittels der durch Unity zur Verfügung gestellten Hierarchie stets zusammen mit dem Tracker transformiert wird.

Abbildung 22 zeigt in abstrahierter Form die hauptsächliche Funktionsweise des entwickelten Algorithmus als Aktivitätsdiagramm nach UML-Standard. Dieses Modell wird auch als Grundlage für folgende Codebeispiele verwendet.

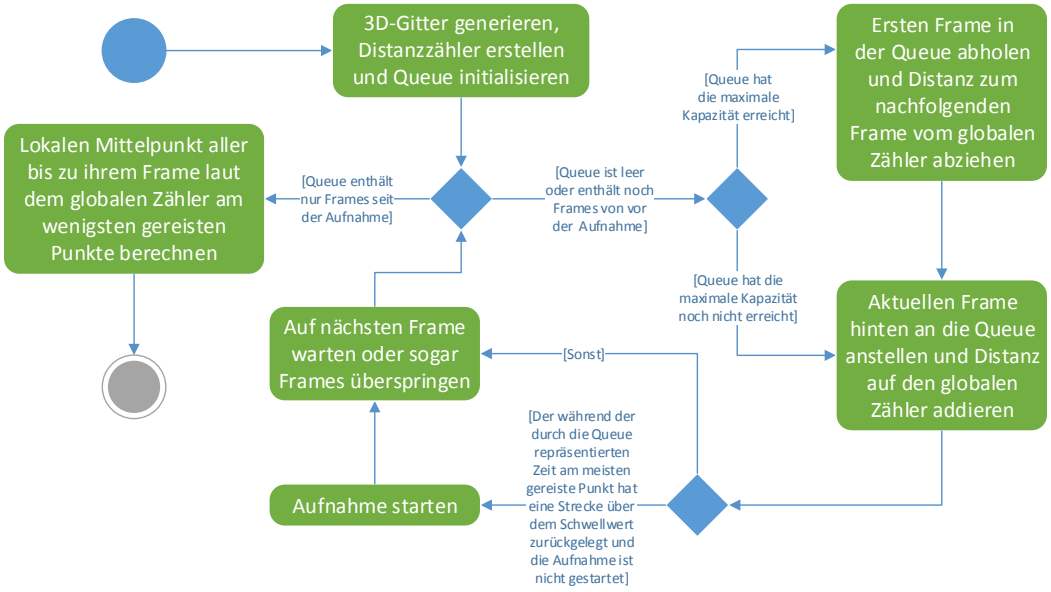


Abbildung 22: Selbstavatar - Ein Algorithmus zur Findung des Rotationszentrums

Vor der genauen Erklärung des gezeigten Ablaufs wird zuerst ein Blick auf die Parameter des Algorithmus geworfen. Viele dieser Variablen müssen sorgfältig und je nach Anwendungsfall gesetzt werden, ansonsten kann ein Verlust an Genauigkeit oder verbleibender Rechenleistung für die Grafikberechnung die Folge sein.

```
public Vector3 testingRangeCm3 = new Vector3(20, 20, 20);
```

Der Ansatz für die Lösung des Problems besteht darin, ein Gitter von Punkten im 3D-Raum dem Tracker unterzuordnen und in bestimmten Abständen aus diesen Punkten denjenigen zu ermitteln, der sich während einer bestimmten Zeit am wenigsten durch das Weltkoordinatensystem bewegt hat. Rotiert sich der Tracker also um ein Zentrum, das innerhalb des Gitters liegt, kann dieses sehr genau bestimmt werden. Wichtig bei der Kalibrierung von anatomischen Gelenken ist dabei, dass man das Gitter mit der richtigen Größe definiert, sodass das vermutete Zentrum auch sicher innerhalb des Gitters liegen wird. Der erste Parameter definiert daher das Volumen, in welchem das Rotationszentrum vermutet wird. Für Hand- und Fußgelenke reichen meist 20 cm^3 aus, je nach Befestigung des Trackers.

```
public int probeCountForEachAxis = 20;
```

Mit einer einzigen Variablen ist die Anzahl der Punkte definiert, die auf jeder der drei Achsen erzeugt werden. Im Normalfall werden zwanzig Prüfpunkte für jede Dimension gleichmäßig im 3D-Gitter verteilt, was bei einer Gittergröße von 20 cm^3

einen ungefähren Punktabstand von 1 cm bei insgesamt 8000 Punkten ergibt. Wird der Wert erhöht, fällt das Ergebnis zwar durch das dichtere Gitter etwas genauer aus, jedoch kann damit selbst ein moderner Rechner an seine Leistungsgrenze gebracht werden, worunter die Darstellung der virtuellen Realität leidet. Grund dafür ist die Auswirkung dieser Variable auf die Laufzeit einer Iteration über alle Messpunkte mit $O(n^3)$. 8000 Punkte stellen eine in der Praxis bewährte Anzahl dar, die selbst auf 20 cm^3 verteilt einen sehr geringen Fehler im Bereich weniger Millimeter aufweisen. Dies ist durch den gewichteten Mittelpunkt möglich, der in Abbildung 22 als letzte Aktion des Algorithmus errechnet wird.

```
public Vector3 gridOffset = default(Vector3);
```

Um nun das definierte Gitter an das vermutete Zentrum anzunähern, lässt sich der Gittermittelpunkt mithilfe dieser Variable im lokalen Koordinatensystem des Trackers beliebig verschieben. Der Standard ist keine Verschiebung, da diese abhängig von der Ausrichtung des Trackers ist und daher einzeln konfiguriert werden muss. Um dies besser zu veranschaulichen, ist in Abbildung 23 eine Aufnahme der Demoszene für die Kalibrierung neben die passende reale Situation gelegt.

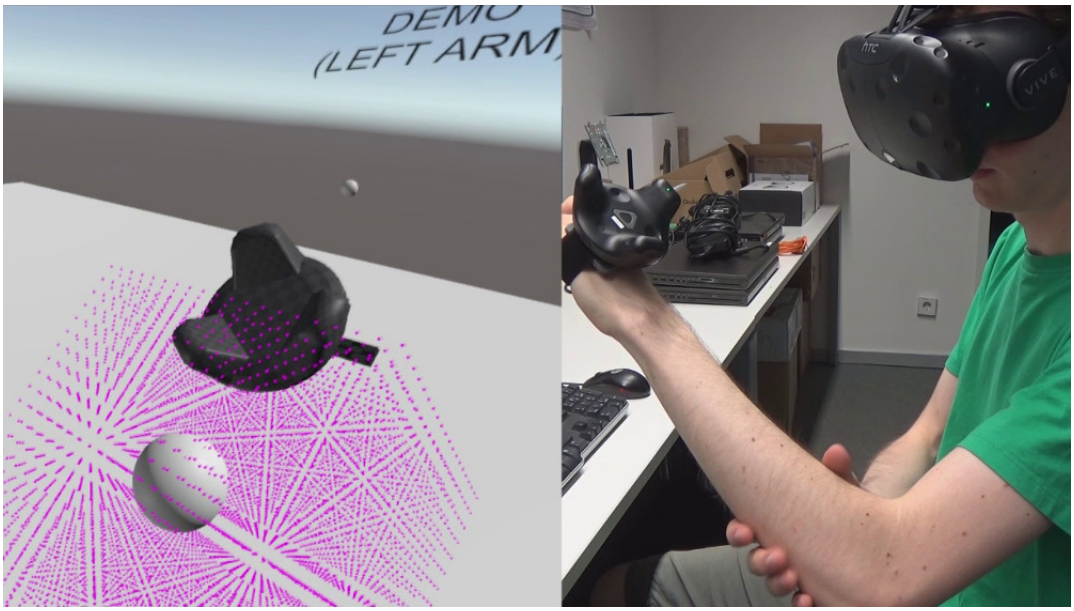


Abbildung 23: Selbstavatar - 3D-Gitter an der Stelle des vermuteten Gelenks

Im Fall der Arme ist durch eine eindeutige Befestigungsweise der Trackers bekannt, in welcher Richtung sich das jeweilige Gelenk am rechten und linken Arm befinden muss. In der Demoszene wird das Gitter durch magentafarbene Würfel mit einem Volumen von 1 mm^3 sichtbar gemacht. Auch wird durch zwei graue Kugeln zur

Sicherheit angedeutet, auf welcher lokalen Achse des Trackers sich der Arm befinden sollte. Die größere Kugel sollte dabei der Person zugewandt sein, während die kleinere Kugel auf dem auswärtigen Achsenteil liegt. Das Gitter für den in Abbildung 23 gezeigten linken Arm ist 10 cm in Richtung des Körpers und 5 cm nach unten verschoben, da bei dieser Befestigung des Trackers das Handgelenk stets unterhalb und hinter dem Trackerursprung liegen wird.

```
public int useEveryNthFrame = 2;
```

Dieser Parameter steuert den Abstand in von Unity berechneten Frames, in dem der Algorithmus einen eigenen Frame - bestehend aus den Positionsdaten der einzelnen Messpunkte im Weltkoordinatensystem - aufnehmen wird. Effektiv wird dadurch jeder n-te Unity-Frame in die statistischen Aufzeichnungen einbezogen. Je nach Anwendungsszenario kann es nützlich sein, nicht bei jedem der 60 oder gar 90 Einzelbilder, die für eine Anwendung mit virtueller Realität üblicherweise berechnet werden müssen, einen weiteren Berechnungsschritt ausführen zu lassen. Standardmäßig wird jeder zweite Unity-Frame berücksichtigt.

```
public int minimumFrameInterspaceMs = 20;
```

Eine Ergänzung zum minimalen Abstand von Frames stellt der minimale Zeitabstand dar. Je schneller der Computer die Grafikberechnung für jeden Unity-Frame abgeschlossen hat, desto weniger Zeit vergeht bis zum nächsten Zeitpunkt, an welchem der Algorithmus die gereiste Distanz der Messpunkte prüft. Mit einer geringeren Distanz pro Messung und einem zu kleinen Puffer schwindet auch die Aussagekraft der minimalen und maximalen gereisten Distanz, weshalb zusätzlich zu ausgelassenen Unity-Frames durch den minimalen Zeitabstand eine Mindestzeit in Millisekunden vor der nächsten Aufzeichnung abgewartet werden kann.

```
public int bufferFramesCount = 60;
```

Da der Algorithmus in Echtzeit arbeitet und dabei hauptsächlich Statistiken über eine bestimmte Zeit erstellt, kommt ein Puffer zum Einsatz, der für eine durch diesen Wert definierte Anzahl an Messungen (im Folgenden „Frames“ genannt) Platz bietet. Ist der Puffer voll, wird der älteste Frame verworfen und der gerade aufgenommene wird hinzugefügt. Ein möglichst aussagekräftiger Puffer sollte nicht zu klein sein, auf der anderen Seite dauert eine Aufnahme mit größerem Puffer auch insgesamt länger. Standard ist ein Puffer mit 60 Frames.

```
public float thresholdDistanceTravel = 0.002f;
```

Zwar ist das Tracking des HTC Vive unter guten Bedingungen auf den Millimeter genau, eine gewisse Schwankung um einige Bruchteile lässt sich dennoch vor allem bei komplett unbewegten Trackern beobachten, die zum Beispiel auf dem Boden

abgelegt wurden. Um mögliche Störungen aus den Berechnungen auszuschließen, kann durch diese Variable eine Mindeststrecke in Metern definiert werden, die von einem Punkt zwischen zwei Frames zur Berücksichtigung der Distanz zurückgelegt werden muss. Standardmäßig ist dieser Wert auf 2 mm für vergleichsweise schnelle Rotationsbewegungen eingestellt.

```
public float thresholdStartCapture = 0.3f;
```

Wie in Abbildung 22 gezeigt, versucht der Algorithmus in seiner ersten Phase, selbstständig den Start einer Kalibrierungsbewegung zu erkennen. Dafür wird während der laufenden Messung für jeden Frame der Punkt im 3D-Gitter ermittelt, der in der durch den Puffer abgedeckten Zeit den weitesten weg durch das Weltkoordinatensystem gereist ist. Ist dieser zurückgelegte Weg größer als der hier definierte Schwellwert, wird die eigentliche Kalibrierung gestartet.

Nachdem nun alle für die Ausführung wichtigen Parameter vorgestellt wurden, soll mit ausgewählten Ausschnitten aus dem Quellcode des Algorithmus dessen Kernfunktionalität zusammengefasst werden. Die erste Aufgabe besteht in der Erzeugung eines 3D-Gitters aus Punkten, die in der Unity-Hierarchie dem Tracker-Objekt untergeordnet sind, welches sich wiederum an der Position des realen Trackers befindet. An derselben Stelle in der Hierarchie wird sich nach der Kalibrierung auch das Zielobjekt befinden, dessen Position und Rotation das zugehörige virtuelle Gelenk einzunehmen versuchen wird.

Für die einfache Iteration über sämtliche Messpunkte wird im folgenden Abschnitt eine Aktion definiert, die eine übergebene Aktion mit allen möglichen Indexkombinationen der Messpunkte ausführt. Damit lässt sich die unveränderte Logik für die Iteration über das 3D-Gitter elegant kapseln.

```
Action<Action<int, int, int>> gridPointIterate =
    delegate(Action<int, int, int> action) {
    for (int x = 0; x < probeCountForEachAxis; x++)
    {
        for (int y = 0; y < probeCountForEachAxis; y++)
        {
            for (int z = 0; z < probeCountForEachAxis; z++)
            {
                action(x, y, z);
            }
        }
    }
};
```

Für die einmalige Erzeugung des Gitters ist der folgende Code zuständig:

```
Transform[,,] probeGrid = new Transform[probeCountForEachAxis,
    probeCountForEachAxis, probeCountForEachAxis];
gridPointIterate((x, y, z) => {
    // For each position in the 3D grid, create a new game object
       and adjust its transform.
    GameObject probe = new GameObject(x + "|" + y + "|" + z);
    probe.transform.parent = this.transform.parent;
    probe.transform.localPosition = new Vector3(
        (-(testingRangeCm3.x / 2f) + x * testingRangeCm3.x /
            probeCountForEachAxis) / 100f,
        (-(testingRangeCm3.y / 2f) + y * testingRangeCm3.y /
            probeCountForEachAxis) / 100f,
        (-(testingRangeCm3.z / 2f) + z * testingRangeCm3.z /
            probeCountForEachAxis) / 100f
    ) + gridOffset;
    probeGrid[x, y, z] = probe.transform;
});
```

Das Ergebnis ist ein multidimensionales Array, das für jeden der erstellten Punkte eine Referenz zum zugehörigen *Transform* enthält. Dadurch kann später direkt auf die von Unity in jedem Einzelbild aktualisierten Weltpositionen zugegriffen werden. Nun werden die nötigten Objekte zur Verwaltung des Puffers angelegt:

```
Queue<Vector3[,,]> bufferedPositions = new
    Queue<Vector3[,,]>(bufferFramesCount);
Queue<int[]> leastTravelledPointIndices = new
    Queue<int[]>(bufferFramesCount);

float[,,] totalDistanceTravelled = new float[probeCountForEachAxis,
    probeCountForEachAxis, probeCountForEachAxis];
```

Mit der in Abbildung 22 häufig referenzierten *Queue* sind mehrere Instanzen der gleichnamigen First-In-First-Out-Datenstruktur aus dem C#-Namensraum namens *System.Collections.Generic* gemeint, die zur Verwaltung des Puffers genutzt werden. Dieser besteht aus zwei dieser Warteschlangen - die erste enthält dreidimensionale Arrays mit den 3D-Positionen aller Messpunkte für jeden Frame, während die zweite für jeden Frame genau drei Indizes enthält, die zum in diesem Frame am wenigsten weit gereisten Punkt im entsprechenden Array aus der ersten Warteschlange führen. Als Anfangskapazität wird die maximale Anzahl an Frames übergeben, sodass später keine Anpassungen der Kapazität mehr nötig sind.

Neben den beiden beschriebenen Warteschlangen wird ein zusätzliches multidimensionales Array erstellt, das als Zwischenspeicher für die Distanz dient, die während der durch den Puffer definierten Zeit von den Messpunkten zurückgelegt wurde. Als wichtige Optimierung für eine akzeptable Laufzeit erspart dieses Array die andernfalls für jeden aufgenommenen Frame notwendige Aufsummierung der zurückgelegten Distanzen zwischen allen Frames des Puffers. Stattdessen können mit vergleichsweise wenig Aufwand beim Hinzufügen und Entfernen eines Frames die Distanzen der Messpunkte zu deren Positionen im benachbarten Frame von deren Gesamtwert im Zwischenspeicher subtrahiert oder auf diesen addiert werden.

Im Hauptteil des Algorithmus wird dann zuerst bei einem vollen Puffer das älteste Element entfernt, welches sich immer an der Spitze der Warteschlange befindet. Zuvor wird die aktuelle Warteschlange für die *Vector3*-Arrays in eine Liste umgewandelt, durch die per Index in beliebiger Reihenfolge auf die Frames zugegriffen werden kann. Zwar scheint diese Vorgehensweise Laufzeit zu verschwenden, jedoch wird dafür intern nur eine Array-Referenz für jeden Frame in die neu erstellte Liste kopiert, was bei den verhältnismäßig kleinen Anzahlen an Frames vernachlässigt werden kann. Der gewonnene Vorteil ist neben einer Zugriffszeit von $O(1)$ auf die Liste die leicht zu verstehende Funktionsweise des Puffers auf Basis von Warteschlangen und deren für C# optimierte Implementierung.

```
List<Vector3[,,]> bufferedPositionsSnapshot =
    bufferedPositions.ToList();

if (bufferedPositions.Count >= bufferFramesCount)
{
    Vector3[,,] deletedBufferObject = bufferedPositions.Dequeue();
    // After removing the oldest frame, we also need to subtract the
       frame's travelled distance from the total travelled distance.
    gridPointIterate((x, y, z) => {
        float deletedTravelledDistance =
            Vector3.Distance(bufferedPositionsSnapshot[1][x, y, z],
            deletedBufferObject[x, y, z]);
        totalDistanceTravelled[x, y, z] -= deletedTravelledDistance
            < thresholdDistanceTravel ? 0f : deletedTravelledDistance;
    });
}
```

Wurde sichergestellt, dass nun mindestens ein freier Platz im Puffer verfügbar ist, kann der aktuelle Frame aufgezeichnet werden. Folgender Codeabschnitt ist dafür zuständig, das neue Array mit den Weltpositionen aller Messpunkte zu erstellen und neu zurückgelegte Distanz auf den Zwischenspeicher zu addieren.

```
Vector3[,,] createdBufferObject = new Vector3[probeCountForEachAxis,
    probeCountForEachAxis, probeCountForEachAxis];
gridPointIterate((x, y, z) => {
    createdBufferObject[x, y, z] = probeGrid[x, y, z].position;
    float frameTravelledDistance = 0f;
    if (bufferedPositionsSnapshot.Count > 0)
    {
        frameTravelledDistance = Vector3.Distance(
            bufferedPositionsSnapshot[bufferedPositionsSnapshot.Count
                - 1][x, y, z], createdBufferObject[x, y, z]);
    }
    totalDistanceTravelled[x, y, z] += frameTravelledDistance <
        thresholdDistanceTravel ? 0f : frameTravelledDistance;
});
bufferedPositions.Enqueue(createdBufferObject);
```

Am Ende des Hauptteils steht die Findung des für diesen Frame am wenigsten gereisten Messpunkts, wofür nun über den Zwischenspeicher iteriert werden kann. Trotz dieser Optimierung hat der Algorithmus eine Laufzeit von $O(2n^3)$, wenn der Puffer noch nicht voll ist und somit kein Element entfernt werden muss, und $O(3n^3)$, wenn dies der Fall ist. Daher zahlt es sich aus, nicht zu viele Messpunkte generieren zu lassen, vor allem, weil am Ende der räumliche Mittelpunkt aller in ihrem Frame am wenigsten weit gereisten Messpunkte berechnet wird, auf den ein dichteres Gitter bei angemessener Pufferlänge in der Praxis keinen größeren Einfluss mehr hat. Das kann folgendermaßen erklärt werden:

> Seien die durch den ersten Parameter definierten Gitterausmaße x, y und z und die durch den zweiten Parameter definierte Messpunktdichte auf jeder Dimension d. Im schlechtesten Fall liegt das reale Zentrum der Rotation eines Frames exakt zwischen vier Messpunkten. Dann findet eine Rundung statt, die das aufgezeichnete Zentrum um
>
> $$\frac{\sqrt{(\frac{x}{d})^2 + (\frac{y}{d})^2 + (\frac{z}{d})^2}}{2}$$
>
> cm verschiebt, also die Hälfte der Raumdiagonale durch den von den vier umliegenden Messpunkten gebildeten Quader.

Bei den Standardwerten entspricht dieser maximale Fehler ungefähr 8,66 mm. Wird dann, wie man es bei einer natürlichen Bewegung erwartet, auf verschiedene Messpunkte in verschiedene Richtungen gerundet, steigt die Wahrscheinlichkeit für einen deutlich genaueren Messpunkt. In Abbildung 24 ist eine weitere Situation aus der Demoszene der Kalibrierung zu sehen, in welcher der gefundene Gelenkpunkt zusammen mit den 60 ermittelten Messpunkten sichtbar gemacht wurde.

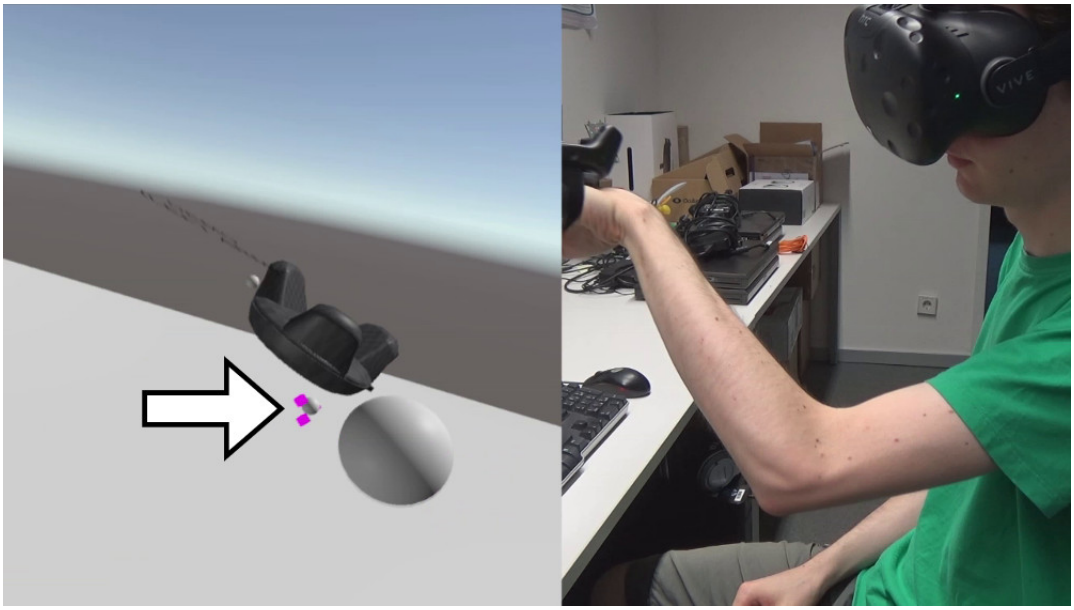


Abbildung 24: Selbstavatar - Markierungen für das ermittelte Gelenk

Die zum Zeitpunkt der jeweiligen Frames über den Pufferzeitraum am wenigsten weit gereisten Messpunkte sind als magentafarbene Würfel repräsentiert, das Endergebnis als kleine, graue Kugel in der Mitte dieser Würfel. In diesem Fall ist durch die in Abbildung 23 gezeigte Fixierung des Unterarms mit der anderen Hand keine große Streuung der Messpunkte entstanden.

Der folgende Codeabschnitt ist für die Ermittlung des gewichteten Mittelpunkts und die Neuplatzierung des Ziels verantwortlich:

```
Func<int[], Vector3> getLocalGridPointPosition =
    delegate(int[] indices) {
        return probeGrid[indices[0],
            indices[1],
            indices[2]].localPosition;
    };
Vector3 sum =
   getLocalGridPointPosition(leastTravelledPointIndices.Peek());
foreach (int[] indices in leastTravelledPointIndices.Skip(1))
{
    sum += getLocalGridPointPosition(indices);
}
Vector3 localCenterPointOffset = sum / bufferFramesCount;
this.transform.localPosition = localCenterPointOffset;
```

Als gewichtet wird der Mittelpunkt deshalb bezeichnet, weil der Mittelwert aller in ihrem Frame ausgewählten Punkte errechnet wird, was nicht ausschließt, dass einige Punkte mehrfach in diese Berechnung einbezogen werden. Folglich verlieren unerwünschte Sprünge an Einfluss, die das Ergebnis verfälschen könnten.

Der gesamte Prozess ist im fertigen Algorithmus von zusätzlichen Mechanismen umgeben, die an der passenden Stelle auf den nächsten aufzuzeichnenden Frame warten und erkennen, sobald der Benutzer eine Bewegung beginnt.

Um bessere Ergebnisse zu bekommen, sollte ausschließlich das Zielgelenk rotiert werden, gleichzeitig auf allen Achsen. Abbildung 25 zeigt für die linke Hand und den rechten Fuß, wie bei deren Kalibrierung der Unterarm und der Unterschenkel für eine besser kontrollierbare Rotation stabilisiert werden können.

Abbildung 25: Selbstavatar - Stabilisierung von Arm und Bein bei der Kalibrierung

4.3 Einrichtung der inversen Kinematik

Bevor die Komponente für die inverse Kinematik des Avatarmodells aktiviert wird, werden noch die β-Werte der Person darauf angewandt. Dieser Vorgang wurde schon in Unity-Komponenten von Joachim Tesch aus der Gruppe Raum- und Körperwahrnehmung am Max-Planck-Institut für biologische Kybernetik implementiert, sodass an dieser Stelle keine genauere Dokumentation dazu notwendig ist. Durch einige Funktionsaufrufe werden die β-Werte der darzustellenden Person aus

einer Textdatei mit festem Namen gelesen und auf das SMPL-Modell angewandt. Im zweiten Schritt wird dann durch einen linearen Regressor die neue Position jedes Gelenks im Rig des Modells in Abhängigkeit der verwendeten β-Werte ermittelt. Dieser Schritt ist notwendig, da sich zwar durch die neuen Gewichtungen der verschiedenen Modellvarianten die Vertexpositionen innerhalb des SMPL-Raums verändern, die einzelnen Gelenkpunkte des Rigs jedoch an ihrer ursprünglichen Position bleiben und damit nicht mehr korrekt im finalen Körper liegen.

Die Unity-Szene mit dem männlichen SMPL-Standardmodell und den Trackern mit Nummerierung ist in Abbildung 26 zu sehen. Zusätzlich hat der Avatar ein kostenlos verfügbares 3D-Modell [32] des verwendeten HMDs auf dem Kopf, um in entsprechenden Szenarien das nicht personalisierte Gesicht zu verdecken.

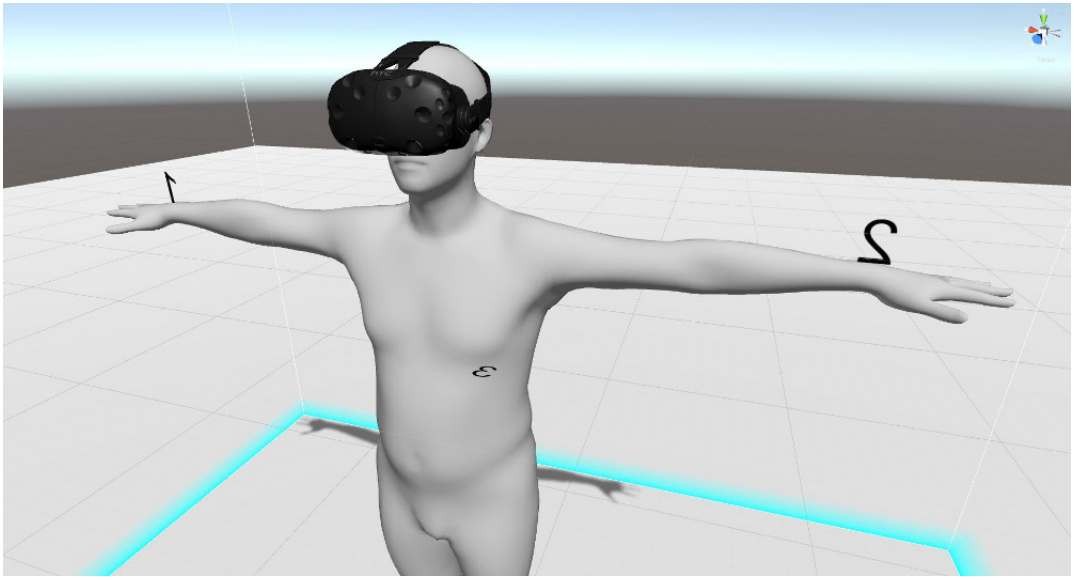


Abbildung 26: Selbstavatar - Mit einem Modell des HMDs und den Trackern

Für die schnelle Umsetzung des Prototypen wurde das Unity-Paket *FinalIK* [26] verwendet, da die alternative Lösung *IKinema* [27] zusätzliche Interaktion mit einer externen Laufzeitumgebung erfordert, während bei *FinalIK* schon der Import des Unity-Pakets zur Verwendung ausreicht.

Aus der Sammlung der zahlreichen von *FinalIK* zur Verfügung gestellten Komponenten, die für verschiedene Situationen optimiert sind, wird für dieses Projekt die Komponente *VRIK* verwendet. Leider wird die leicht vom Standard abweichende Struktur des SMPL-Rigs durch die Komponente nicht automatisch erkannt, sodass die Referenzen zu den passenden Gelenken im SMPL-Rig von Hand zugewiesen werden müssen. In Abbildung 27 ist die fertige Zuweisung dargestellt.

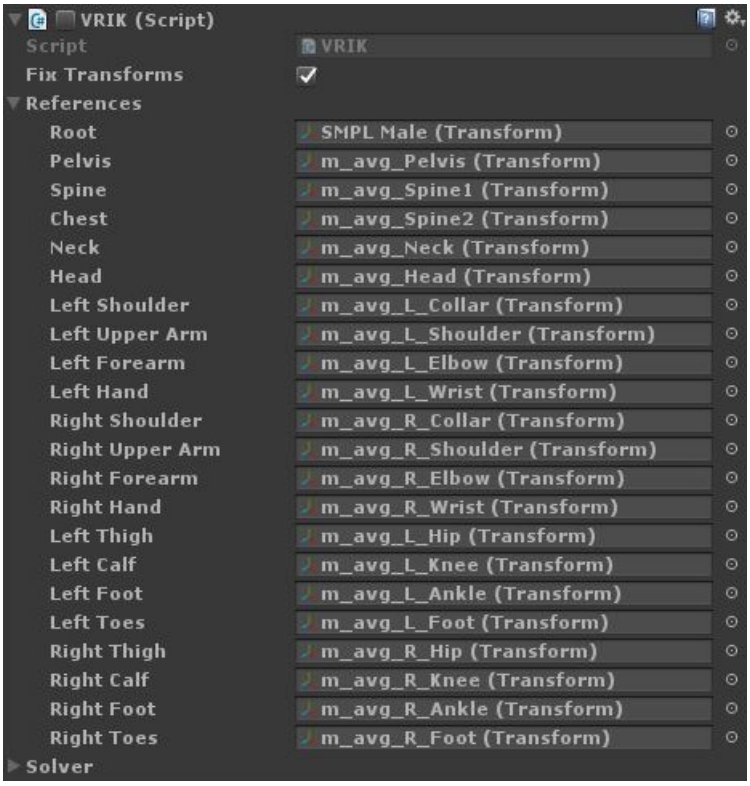


Abbildung 27: Selbstavatar - Referenzen zum SMPL-Rig für inverse Kinematik

Außer den Referenzen zum zugehörigen Rig muss der Komponente mitgeteilt werden, welche Zielobjekte für die Berechnung zum Einsatz kommen sollen und welche zusätzlichen Einstellungen für die inverse Kinematik gewünscht sind. Dazu werden die den Trackern untergeordneten Objekte verwendet, die sich durch den Schritt im letzten Unterkapitel nun an der Position des anatomischen Gelenks befinden. Diese Konfiguration ist jedoch für beide SMPL-Modelle (weiblich und männlich) die gleiche, weshalb sie in der entwickelten Szene der Hauptkomponente namens *Main* zugeordnet wurde, die dann in der laufenden Anwendung die Konfiguration des tatsächlich verwendeten Modells überschreibt.

Die sorgfältige Konfiguration der *VRIK*-Komponente und die Optimierung des kompletten Szenarios nimmt sehr viel Zeit in Anspruch und ist an die Limitationen der verwendeten Software gebunden. Diese Schritte werden daher nicht im Rahmen dieser Arbeit besprochen, können aber auf Basis des entwickelten Prototyps in Zukunft vorgenommen werden.

5 Durchführung einer Studie

Auf dem Weg zum personalisierten Selbstavatar ohne Körperscan geht es neben den exakten Proportionen und Abmessungen auch um die Wahrnehmung eines Avatars durch den Benutzer. Es stellt sich die Frage, wie tolerant die Benutzer bezüglich ihrer Repräsentation in virtueller Realität sind und ob es spezielle Merkmale gibt, auf die bei der Bewertung eines Avatars besonders Acht gegeben wird.

Um am Ende dieser Arbeit noch etwas Aufschluss zu dieser Frage geben zu können, wurde eine kurze Wahrnehmungsstudie mit zehn zufälligen Teilnehmenden aus der Region Tübingen durchgeführt, die das SMPL-Standardmodell des passenden Geschlechts durch die einzelnen β-Werte von Hand und in virtueller Realität so einstellen sollten, dass der entstandene SMPL-Körper ihren eigenen Körper so gut wie möglich darstellt. Am Ende sollte das Ergebnis bewertet werden.

5.1 Methodik

Da aus organisatorischen Gründen nicht alle der in Kapitel 3.1 beschriebenen Teilnehmenden der Validierung erneut für diese Studie zur Verfügung stehen konnten, wurden zehn zufällige Personen aus Tübingen und der Umgebung gefunden, von denen sechs weiblich und vier männlich sind. Alle Teilnehmenden haben zu Beginn des Experiments eine schriftliche Einverständniserklärung für ihre mit 8 Euro pro Stunde vergütete Teilnahme an dieser Studie abgegeben. Der Mittelwert für das Alter aller Teilnehmenden liegt bei 27,6 Jahren mit einer Standardabweichung von 4,43 Jahren. Die Prozedur des Experiments wurde ebenfalls durch das lokalen Ethik-Komittee der Universität Tübingen genehmigt und befindet sich in Übereinkunft mit der Deklaration von Helsinki. Um sich in diesem Kapitel eindeutig auf die einzelnen Teilnehmenden beziehen zu können, werden im Folgenden Text und in allen Graphen die männlichen jeweils mit der Bezeichnung 2-M1, 2-M2, 2-M3, 2-M4 und die weiblichen jeweils mit der Bezeichnung 2-W1, 2-W2, 2-W3, 2-W4, 2-W5 und 2-W6 referenziert. Durch dieses System der Nummerierung sollen Kollisionen mit den Teilnehmenden der Validierung vermieden werden.

Das Experiment fand in virtueller Realität statt, die mit dem bereits bei der Umsetzung des Selbstavatars in Kapitel 4 vorgestellten HMD *HTC Vive* erlebt wurde. Die Teilnehmenden haben in zwei Wiederholungen drei Mal alle zehn β-Werte des gegenüberstehenden SMPL-Modells in einer zufälligen Reihenfolge einzeln verändert. Die Anweisung dabei war, dass sie ihren eigenen Körper so gut wie möglich annähern sollen. Kontrolliert werden konnten die β-Werte mithilfe der HTC Vive Handcontroller, die am unteren Rand von Abbildung 28 sichtbar sind.

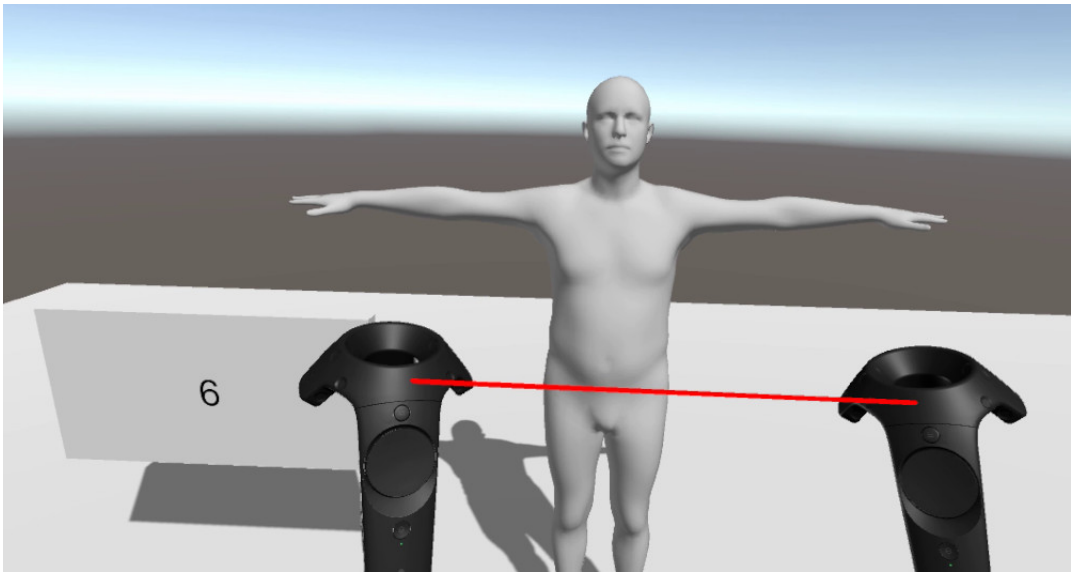


Abbildung 28: Studie - Mit den Handcontrollern kann das im HMD betrachtete SMPL-Modell verändert werden

Abbildung 28 zeigt eine Momentaufnahme des finalen Experiments aus Sicht eines Teilnehmenden, der gerade den β-Wert mit der Nummer sechs des männlichen SMPL-Standardmodells verändert. Sobald der Auslöser an der Rückseite beider Controller gedrückt wird, erscheint eine rote Verbindung zwischen den virtuellen Controllern und der Nutzer kann durch einfaches Vergrößern oder Verkleinern der Distanz zwischen den Handcontrollern eine Fließkommazahl zwischen -5 und +5 einstellen. Sobald ein β-Wert wie gewünscht eingestellt ist, kann durch einen gleichzeitigen Klick auf die beiden großen Knöpfe an den Controllern zum nächsten Wert gewechselt werden. Dies wiederholt sich bis zur nächsten Unterbrechung.

Nach jeder der beiden Wiederholungen gibt es eine Unterbrechung, in der die Teilnehmenden einen kurzen Fragebogen ausfüllen sollen. Dieser enthält hauptsächlich Fragen zur Zufriedenheit mit bestimmten Körperregionen des erstellten Avatars, welche mit ganzzahligen Werten zwischen 1 und 7 beantwortet werden können. Hierbei heißt der niedrigste Wert „sehr schlechte Repräsentation“, während der höchste Wert „sehr gute Repräsentation“ bedeutet. Nachdem dieser Teil des Experiments abgeschlossen war, wurden von jedem der Teilnehmenden abschließend die zwölf in Kapitel 2.4 aufgelisteten Körperabmessungen registriert.

5.2 Ergebnisse der Studie

Eine visuelle Repräsentation der gegebenen Antworten auf die Fragen des Fragebogens ist für beide Wiederholungen in Abbildung 29 zu sehen.

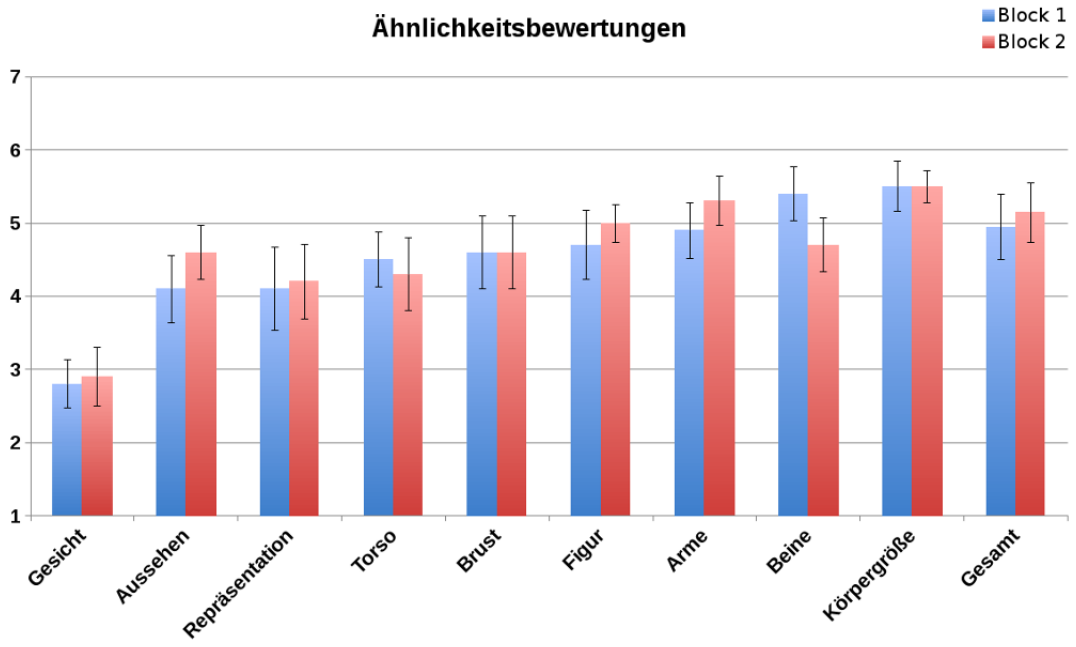


Abbildung 29: Studie - Ergebnisse des Fragebogens zur Ähnlichkeitsbewertung

Die Wiederholungen sind in Abbildung 29 als rote und blaue Balken dargestellt, in der Legende sind diese auch als „Block 1“ und „Block 2“ benannt.

Direkt auffallend ist die niedrige Bewertung für das Gesicht, die in dieser Form zu erwarten war. Denn das SMPL-Modell konzentriert sich ausschließlich auf Körperform und Pose, weshalb das Gesicht immer ein von den β-Werten unabhängiges Durchschnittsgesicht darstellt. Doch zumindest wurde hierfür häufig ein Wert in der Nähe von drei von insgesamt sieben abgegeben, was vermutlich daran liegt, dass das Durchschnittsgesicht mit beinahe jedem Gesicht etwas gemeinsam hat.

Dafür, dass man für die Einstellung der Körpergröße einfach geradeaus schauen müsste, um den entsprechenden Wert zur kompletten Zufriedenheit einzustellen, liegt die dafür abgegebene Ähnlichkeitsbewertung mit Antworten zwischen fünf und sechs von insgesamt sieben bei einem vergleichsweise niedrigen Wert. Dies lässt sich durch die statistische Natur der zehn verwendeten β-Werte erklären: Verändert man einen der β-Werte, kann es durchaus sein, dass sich mehrere aus

menschlicher Sicht unabhängige Merkmale gleichzeitig verändern. Es ist deshalb davon auszugehen, dass die Teilnehmenden entweder zum Ausgleich für andere als wichtig empfundene Merkmale eine geringe Abweichung der Körpergröße in Kauf genommen haben oder die Aufgabe wegen der vielen Nebeneffekte der einzelnen β-Werte nicht zu ihrer Zufriedenheit lösen konnten.

Obwohl die Aufgabe der Einstellung von einzelnen β-Werten des SMPL-Modells durchaus als schwierig angesehen werden kann, spricht die geringe Abweichung zwischen Block 1 und Block 2 dafür, dass die Teilnehmenden es schafften, zwei Mal in Folge ein für sie in gleicher Weise zufriedenstellendes Ergebnis einzustellen. Auch befinden sich die Bewertungen aller restlichen Körperregionen meistens über der neutralen Bewertung von vier, was ein knappes positives Ergebnis darstellt.

Aus den in Anhang 10 aufgelisteten Körperabmessungen der Teilnehmenden für diese Studie wurden mit dem aktuellen, schon bei der Validierung verwendeten Regressor die vier Takes mit jeweils den Körperabmessungen generiert, die schon in Abbildung 12 definiert wurden. Einen visuellen Vergleich dieser Körper enthalten Anhang 11 und Anhang 12 in horizontaler Aufstellung. Dieselben Körper können in Anhang 13 und Anhang 14 in vertikaler Aufstellung verglichen werden.

Generell scheinen sich die Beobachtungen anhand der Ähnlichkeitsbewertungen nach einem Blick auf die generierten Bilder zu bestätigen: Die beiden generierten Avatare, in den Anhängen als „Erstavatar“ und „Zweitavatar“ bezeichnet, sehen optisch in den meisten Fällen sehr ähnlich aus. Ein Vergleich zwischen den durch die Teilnehmenden eingestellten Avataren und beispielsweise Take1, der aus zwölf realen Körperabmessungen generiert wurde, zeigt bei den Männern in manchen Fällen, dass diese sich selbst muskulöser eingestellt haben. Bei den Frauen lässt sich keine eindeutig sichtbare Tendenz ausmachen, optisch aber scheinen manche der eingestellten Avatare etwas zierlicher zu sein.

Abschließend kann mit Anhang 15 noch ein Blick auf einen zusammenfassenden Vergleich zwischen den beiden erstellten Avataren und zwischen „Zweitavatar“ und Take1 geworfen werden. Diese Abmessungen zeigen, dass Teilnehmerin 2-W4 scheinbar Probleme mit der Findung eines geeigneten Avatars hatte, in extremen Fällen entstehen bei ihr sogar Abweichungen von 30 cm.

Allgemein liegen die Abweichungen zwischen „Zweitavatar“ und Take1 bei 2 cm bis 10 cm, was darauf hindeutet, dass je nach Person die aktuellen Abweichungen der personalisierten Avatare möglicherweise nicht negativ wahrgenommen werden. Für genauere Aussagen müsste man die hier gewonnenen Ergebnisse jedoch wieder mit einem Körperscan vergleichen, um sich nicht ausschließlich auf den gerade in der Entwicklung befindlichen linearen Regressor zu stützen.

6 Ausblick

In diesem letzten Kapitel der Arbeit sollen Möglichkeiten zur weiteren Forschung im Bereich der Selbstavatare und Ideen für die zukünftige Entwicklung der im Rahmen dieser Arbeit umgesetzten Unity-Projekte eingebracht werden.

Eine sehr nützliche Erweiterung des nun implementierten Unity-Vergleichs würde zum Beispiel zusätzliche Funktionalität zur Messung von Umfängen der 3D-Körper auf der Höhe eines definierten Vertex darstellen. Damit könnten - bis auf das Körpergewicht natürlich - von beliebigen SMPL-Körpern dieselben Abmessungen ermittelt werden, die auch nach der CAESAR-Definition [24] am realen Körper für die bisherigen Experimente vorgenommen werden.

Der aktuelle Unity-Vergleich ist jedoch auch gut geeignet, um weitere Experimente mit SMPL durchzuführen: Man könnte zum Beispiel untersuchen, wie gut die in Flemings Veröffentlichung im Jahr 2015 [15] verwendeten stilisierten Avatare sich in den CAESAR-Raum übertragen lassen. Ebenfalls könnte man die Takes aus der Validierung als ganzes 3D-Modell skalieren, sodass sie keine Abweichung mehr in der Körpergröße aufweisen. Eine interessante Untersuchung wäre dann ein Vergleich der Abweichung der anderen Messwerte, die durch die Richtigstellung der Körpergröße möglicherweise zusätzlich verursacht wurde.

In experimenteller Hinsicht sollte geprüft werden, ob die momentan bestehenden Unterschiede zwischen den verschiedenen Takes von den Benutzern überhaupt wahrgenommen werden. Denn möglicherweise reicht lediglich eine Verfeinerung des Regressors aus, der Körpergröße und Körpergewicht in β-Werte umwandelt. In diesem Fall würde man noch etwas mehr Zeit sparen können, da nicht alle der zwölf Körperabmessungen ermittelt werden müssten. Das gilt natürlich nur für andere Szenarien als den Selbstavatar, denn vor allem Körpergröße und Armspannweite müssen bei einem gut funktionierenden Selbstavatar genau stimmen.

Eine im Kontext der durchgeführten Studie äußerst interessante Veröffentlichung von S. Streuber aus dem Jahr 2016 nennt sich „Body Talk“ [33]. Hier wurde durch die statistische Auswertung von entsprechenden Befragungen ein statistisches Modell entwickelt, mit dessen Hilfe verbale Beschreibungen von menschlichen Körpern auf die zehn β-Werte des SMPL-Modells abgebildet werden können. Mit dieser Möglichkeit könnten die Teilnehmenden zukünftiger Studien bei der Einstellung ihres eigenen Avatars aus verschiedenen sprachlichen Bezeichnungen auswählen, wodurch die zehn β-Werte nicht mehr direkt eingestellt werden müssten. Ebenfalls sollte von den Teilnehmenden einer weiteren Wahrnehmungsstudie auch ein vollständiger 3D-Scan erstellt werden, mit dessen Hilfe ein sehr genauer Vergleich des eingestellten Avatars mit der realen Person möglich wäre.

Nachdem nun eine prototypische Vorlage für einen Selbstavatar entwickelt wurde, können in Zukunft die restlichen Optimierungen vorgenommen werden. Während die Phase der Kalibrierung schon weitestgehend optimiert ist, müssen noch einige Verbesserungen an der Konfiguration der inversen Kinematik vorgenommen werden. Zuerst müsste ein Weg gefunden werden, auf welchem das Zielobjekt für das Beckengelenk des SMPL-Rigs dynamisch an den Benutzer angepasst werden kann. Denn für Hände und Füße ist eine Kalibrierungsbewegung eindeutig, für das menschliche Becken gibt es aber keine einfache Möglichkeit, eine gezielte Rotation um ein bestimmtes Zentrum auszuführen. Eine Idee zur Lösung dieses Problems wäre die Abstandsmessung eines speziellen Vertex zum Beckengelenk des SMPL-Rigs nach Anwendung der für den Benutzer individuellen β-Werte, wodurch auf den Abstand des Trackers zur anatomischen Körpermitte zu schließen wäre.

Ein Problem bei der Umsetzung des in Kapitel 4.3 besprochenen HMDs zur Überdeckung des durchschnittlichen Gesichts des SMPL-Modells ist die feste Position des 3D-Modells innerhalb des SMPL-Rigs. Dies führt dazu, dass das Modell bei einer Anwendung von β-Werten und damit einer Veränderung der Körperform sich nicht an die leichte Veränderung der Kopfform anpasst, die sich im Gegensatz zum Gesicht durchaus in geringen Maßen ändern kann. Das Resultat ist, dass das HMD zum Beispiel bei β-Werten, die einen vergleichsweise langen Kopf zur Folge haben, seinen ursprünglichen Abstand zum Genick des SMPL-Modells einhält und sich sichtbar mit dem Kopf überschneidet. Eine mögliche Lösung für dieses Problem wäre ein zusätzlicher Regressor, der nach der Anpassung des SMPL-Rigs an neue β-Werte ebenfalls eine neue Position und Skalierung für das Headset auswählt. Eine weniger aufwändige Lösung wäre die Messung der Distanzen von ausgewählten Vertices am Kopf des SMPL-Modells, nachdem die neuen β-Werte darauf angewandt wurden. Dadurch könnte man die Dimensionen des Kopfes berechnen und das Headset entsprechend platzieren.

Mit optimierter inverser Kinematik, einem unkomplizierten System zur Eingabe oder Echtzeitmessung der Körperabmessungen und einigen generischen Texturen auf den bisher immer einfarbigen Avataren wäre das Projekt SMPLVR als vielseitig einsetzbare Projektvorlage bereit für die Anwendung im Alltag.

Literatur

[1] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oktober 2015.

[2] Autodesk Inc. Autodesk 3dsMax Website. `https://www.autodesk.de/products/3ds-max/overview`, Mai 2017.

[3] Adobe Systems Inc. Mixamo Website. `https://www.mixamo.com/auto-rigger`, Mai 2017.

[4] F. I. Parke. Computer generated animation of faces. In *Proceedings of the ACM Annual Conference - Volume 1*, ACM '72, pages 451–457, New York, NY, USA, 1972. ACM.

[5] A. J. Izenman. *Modern multivariate statistical techniques: Regression, classification, and manifold learning.* Springer texts in statistics. Springer New York, 2008.

[6] I. N. Bronštejn, K. A. Semendjaev, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik.* Harri Deutsch, Frankfurt am Main, 2005. 6. Auflage.

[7] D. Urban and J. Mayerl. *Regressionsanalyse: Theorie, Technik und Anwendung.* SpringerLink: Bücher. Wiesbaden: VS Verlag für Sozialwissenschaften, 2011, 2011.

[8] D. G. Krige. A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society*, 52:119–139, 1951.

[9] SAE International. Produktwebsite der CAESAR-Datenbank. `https://store.sae.org/caesar`, Mai 2017.

[10] C. Phillips and N. I. Badler. Jack: A toolkit for manipulating articulated gures. In *Proceedings of ACM SIGGRAPH Symposium on User Interface Software*, pages 221–229, Oktober 1988.

[11] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Trans. Graph.*, 22(3):587–594, 2003.

[12] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: Shape Completion and Animation of People. *ACM Trans. Graph.*, 24(3):408–416, Juli 2005.

[13] I. V. Piryankova, J. K. Stefanucci, J. Romero, S. De La Rosa, M. J. Black, and B. J. Mohler. Can I Recognize My Body's Weight? The Influence of Shape and Texture on the Perception of Self. *ACM Trans. Appl. Percept.*, 11(3):13:1–13:18, September 2014.

[14] A. Feng, D. Casas, and A. Shapiro. Avatar Reshaping and Automatic Rigging Using a Deformable Model. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, MIG '15, pages 57–64, New York, NY, USA, 2015. ACM.

[15] R. Fleming, B. J. Mohler, J. Romero, M. J. Black, and M. Breidt. Appealing female avatars from 3D body scans: Perceptual effects of stylization. In *11th Int. Conf. on Computer Graphics Theory and Applications (GRAPP)*, Februar 2016.

[16] Max-Planck-Institut für Intelligente Systeme. Perceiving Systems Website. `https://ps.is.tuebingen.mpg.de/`, Mai 2017.

[17] M. M. Loper, N. Mahmood, and M. J. Black. MoSh: Motion and Shape Capture from Sparse Markers. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 33(6):220:1–220:13, November 2014.

[18] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black. Dyna: A Model of Dynamic Human Shape in Motion. *ACM Transactions on Graphics, (Proc. SIGGRAPH)*, 34(4):120:1–120:14, August 2015.

[19] S. Zuffi and M. J. Black. The Stitched Puppet: A Graphical Model of 3D Human Shape and Pose. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2015)*, pages 3537–3546, Juni 2015.

[20] L. Pishchulin, S. Wuhrer, T. Helten, C. Theobalt, and B. Schiele. Building Statistical Shape Spaces for 3D Human Modeling. *Pattern Recognition*, 67:276–286, 2017.

[21] R. McDonnell. Publikationen von Rachel McDonnell. `https://www.scss.tcd.ie/Rachel.McDonnell/portfolio.shtml`, Mai 2017.

[22] C. O'Sullivan. Publikationen von Carol O'Sullivan. `http://isg.cs.tcd.ie/cosulliv/Publications.html`, Mai 2017.

[23] MPI für Intelligente Systeme. Website zum 3D Scanner in Tübingen. `https://ps.is.tuebingen.mpg.de/pages/3dcapture`, Mai 2017.

[24] K. Robinette, S. Blackwell, H. Daanen, M. Boehmer, and S. Fleming. Civilian American and European Surface Anthropometry Resource (CAESAR), Final Report. Volume 1. Summary. Technical report, DTIC Document, 2002.

[25] B. J. Mohler, S. H. Creem-Regehr, W. B. Thompson, and H. H. Bülthoff. The effect of viewing a self-avatar on distance judgments in an hmd-based virtual environment. *Presence: Teleoperators and Virtual Environments*, 19(3):230–242, 2010.

[26] RootMotion. FinalIK Produktwebsite. `http://www.root-motion.com/final-ik.html`, Mai 2017.

[27] IKinema. IKinema Produktwebsite. `https://www.ikinema.com/`, Mai 2017.

[28] HTC Corporation. HTC Vive Website. `https://www.vive.com/de/product/`, Mai 2017.

[29] Vicon Motion Systems Ltd. Vicon Website. `https://www.vicon.com/`, Mai 2017.

[30] Unity Technologies. Unity 3D Website. `https://unity3d.com/unity`, Mai 2017.

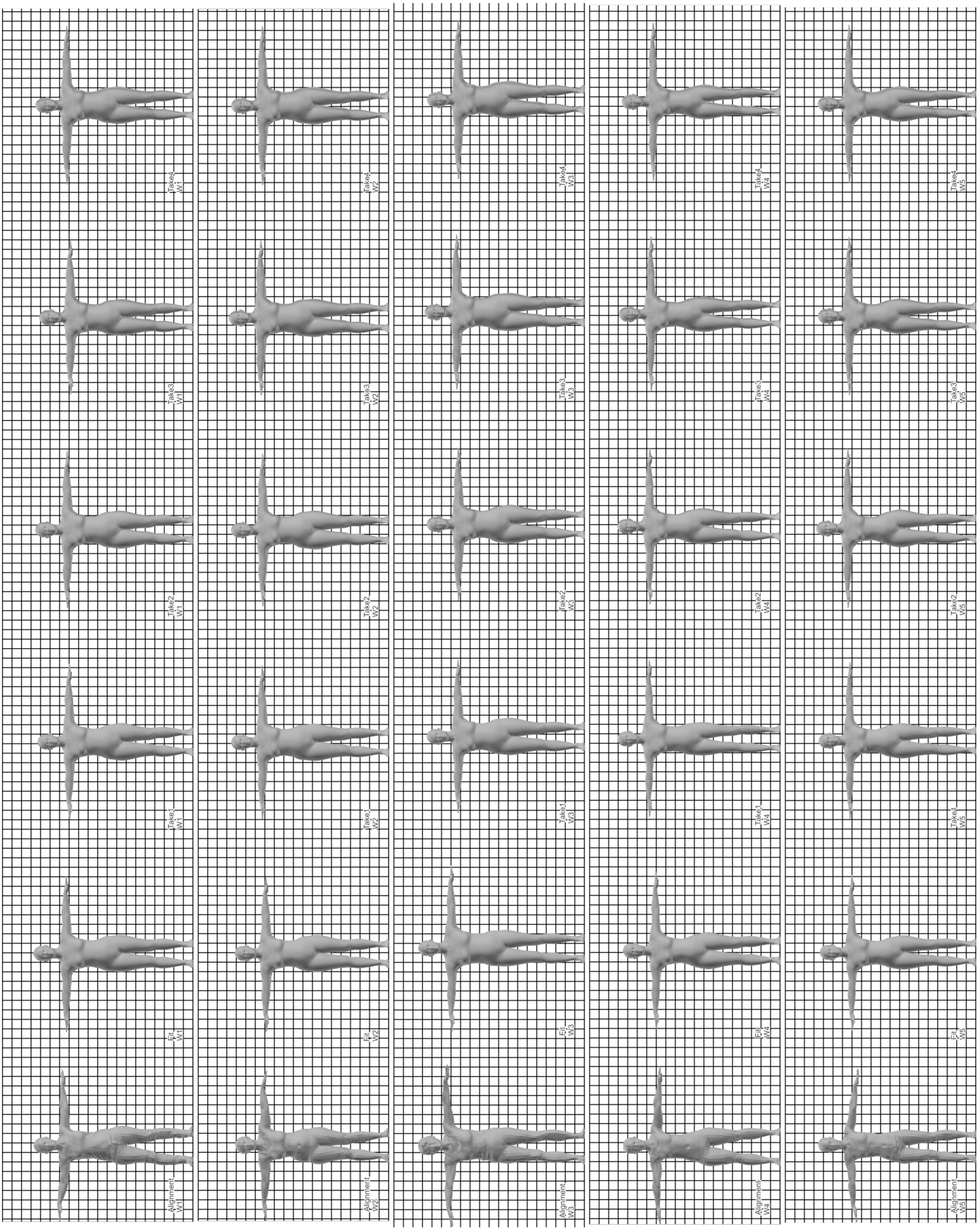
[31] Blender Foundation. Blender Website. `https://www.blender.org/`, Mai 2017.

[32] Nutzer namens Eternal Realm Sketchfab Portal. Sketchfab Website eines Modells des HTC Vive HMDs. `https://sketchfab.com/models/4cee0970fe60444ead77d41fbb052a33`, Mai 2017.

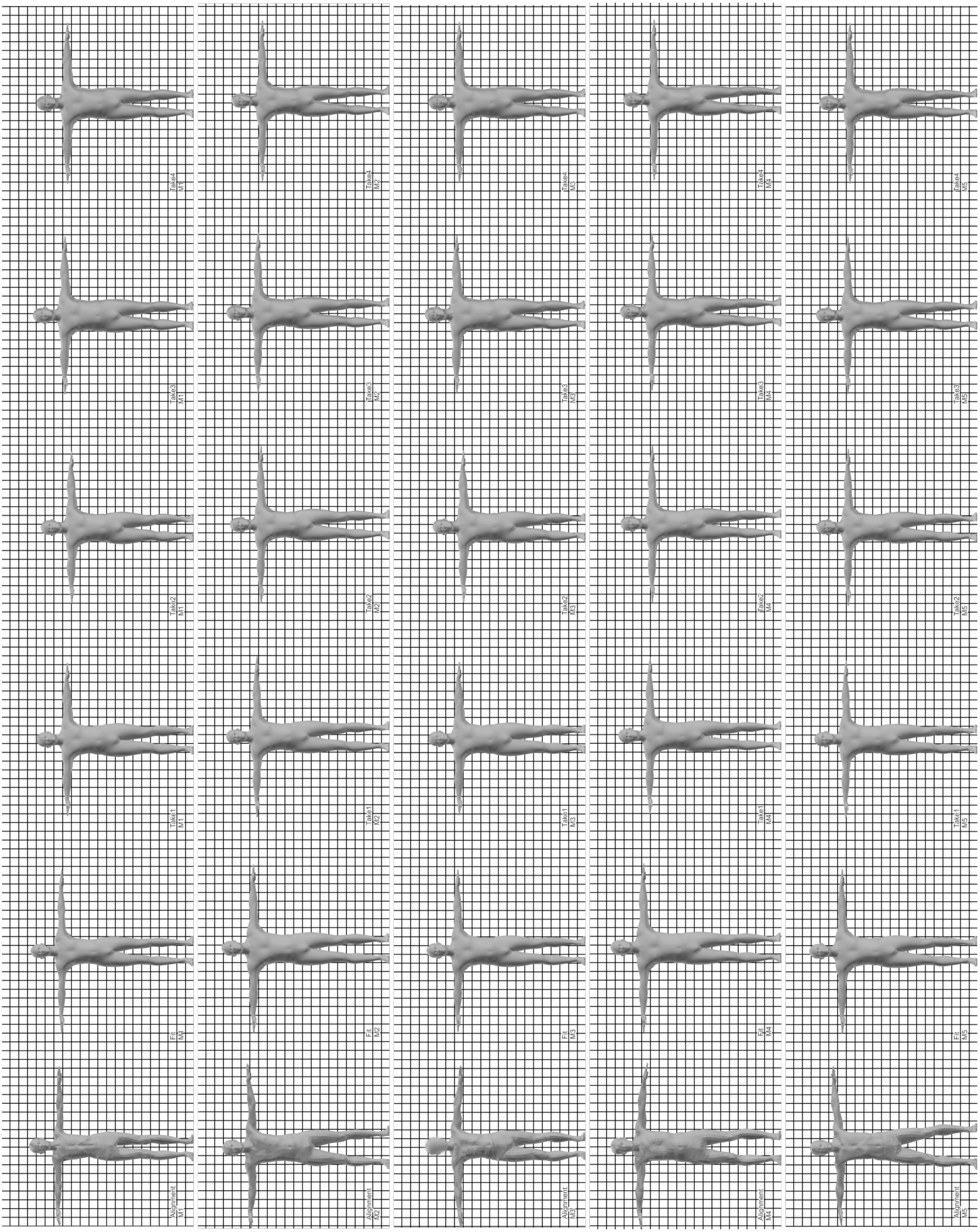
[33] S. Streuber, M. A. Quiros-Ramirez, M. Q. Hill, C. A. Hahn, S. Zuffi, A. O'Toole, and M. J. Black. Body Talk: Crowdshaping realistic 3D avatars with words. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 35(4):54:1–54:14, Juli 2016.

Abbildungsverzeichnis

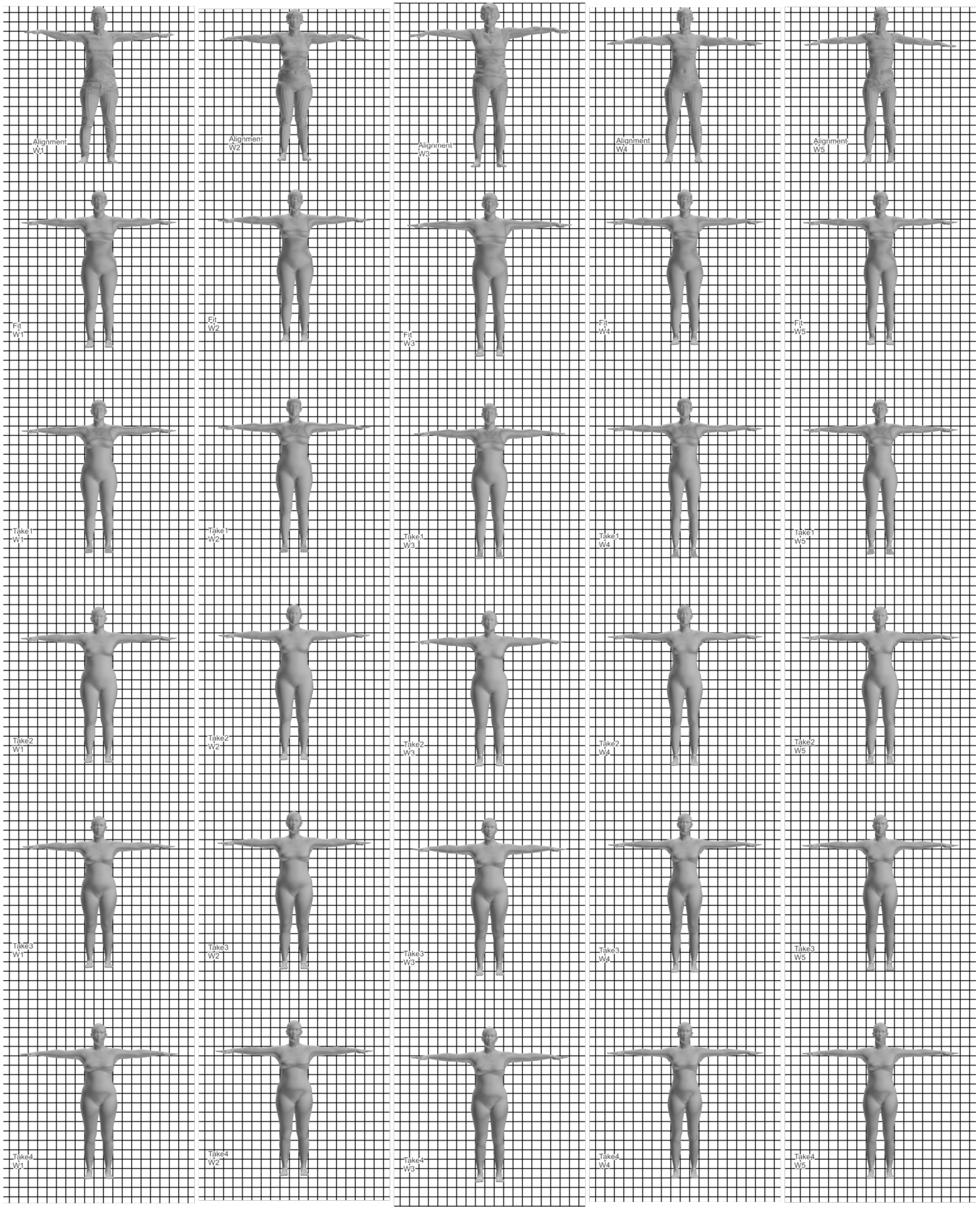
Anhang 1


Take4 W1
Take3 W1
Take2 W1
Take1 W1
Fit W1
Alignment W1
Take4 W2
Take3 W2
Take2 W2
Take1 W2
Fit W2
Alignment W2
Take4 W3
Take3 W3
Take2 W3
Take1 W3
Fit W3
Alignment W3
Take4 W4
Take3 W4
Take2 W4
Take1 W4
Fit W4
Alignment W4
Take4 W5
Take3 W5
Take2 W5
Take1 W5
Fit W5
Alignment W5

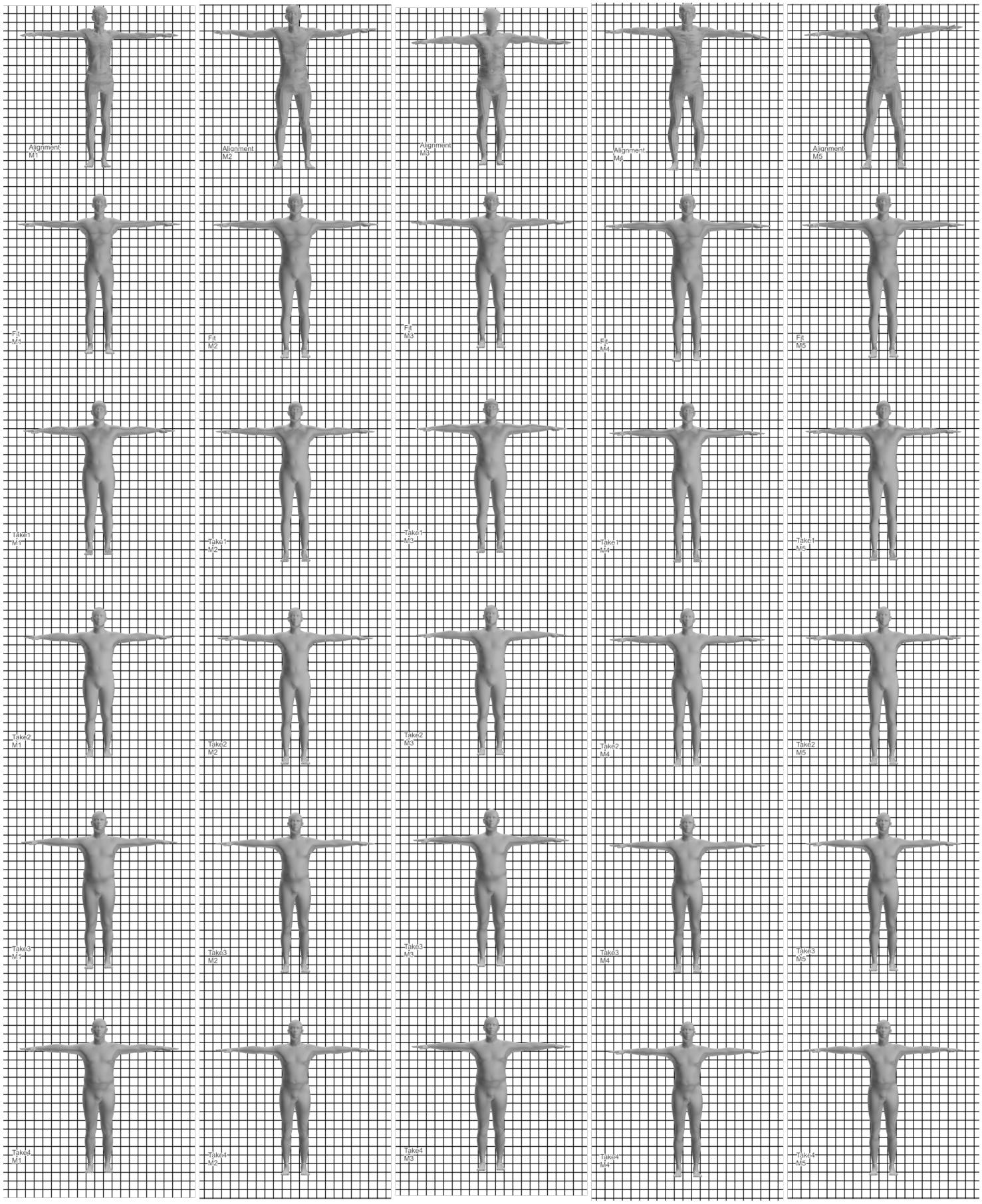
Anhang 2


Take4 M1
Take3 M1
Take2 M1
Take1 M1
Fit M1
Alignment M1
Take4 M2
Take3 M2
Take2 M2
Take1 M2
Fit M2
Alignment M2
Take4 M3
Take3 M3
Take2 M3
Take1 M3
Fit M3
Alignment M3
Take4 M4
Take3 M4
Take2 M4
Take1 M4
Fit M4
Alignment M4
Take4 M5
Take3 M5
Take2 M5
Take1 M5
Fit M5
Alignment M5

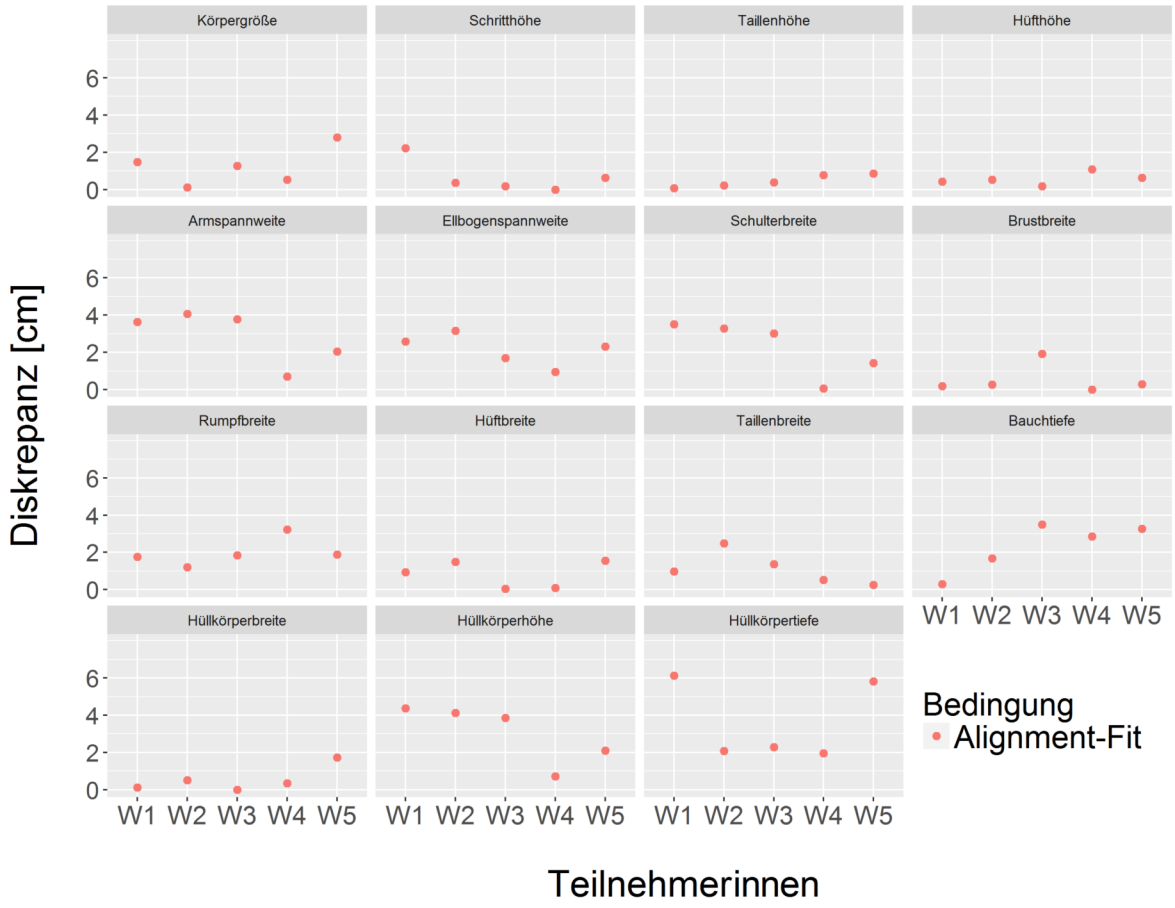
Anhang 3

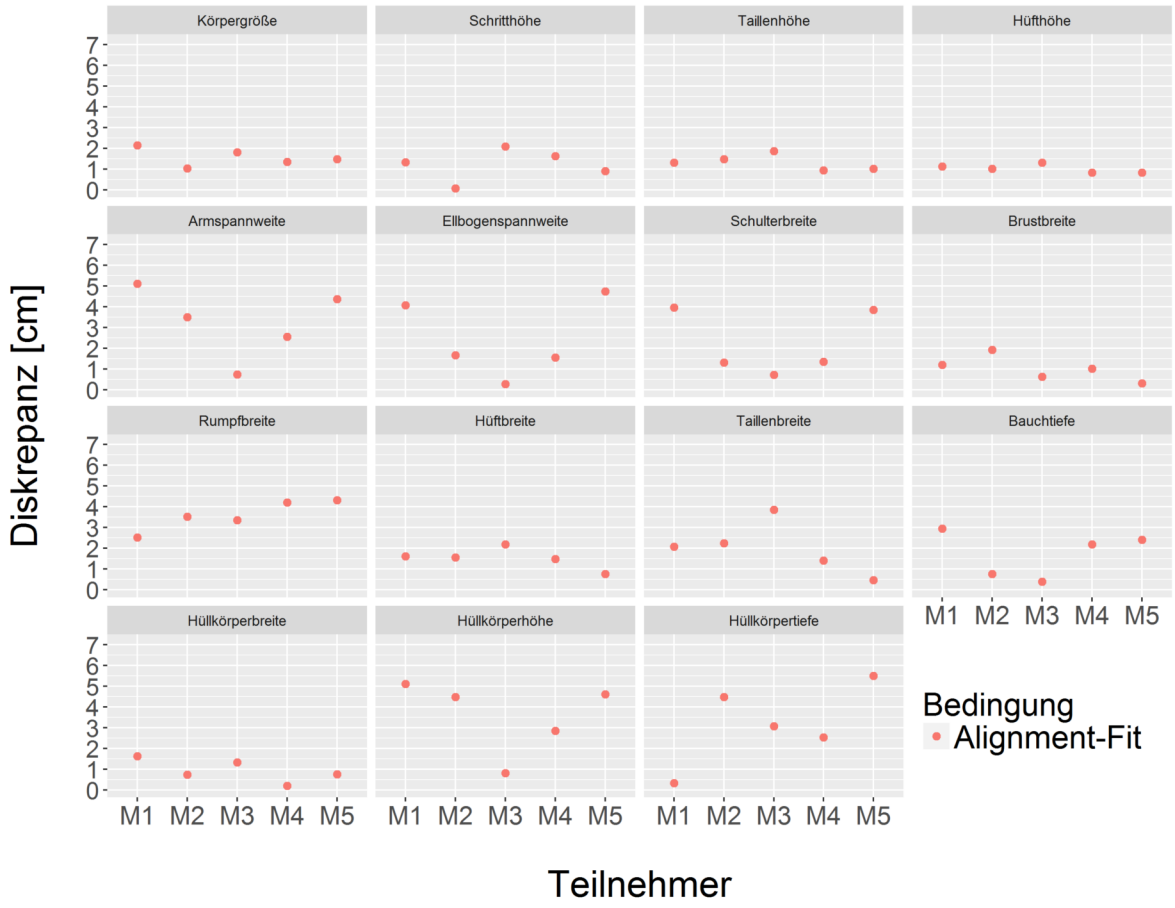

Alignment W1
Fit W1
Take1 W1
Take2 W1
Take3 W1
Take4 W1
Alignment W2
Fit W2
Take1 W2
Take2 W2
Take3 W2
Take4 W2
Alignment W3
Fit W3
Take1 W3
Take2 W3
Take3 W3
Take4 W3
Alignment W4
Fit W4
Take1 W4
Take2 W4
Take3 W4
Take4 W4
Alignment W5
Fit W5
Take1 W5
Take2 W5
Take3 W5
Take4 W5

Anhang 4

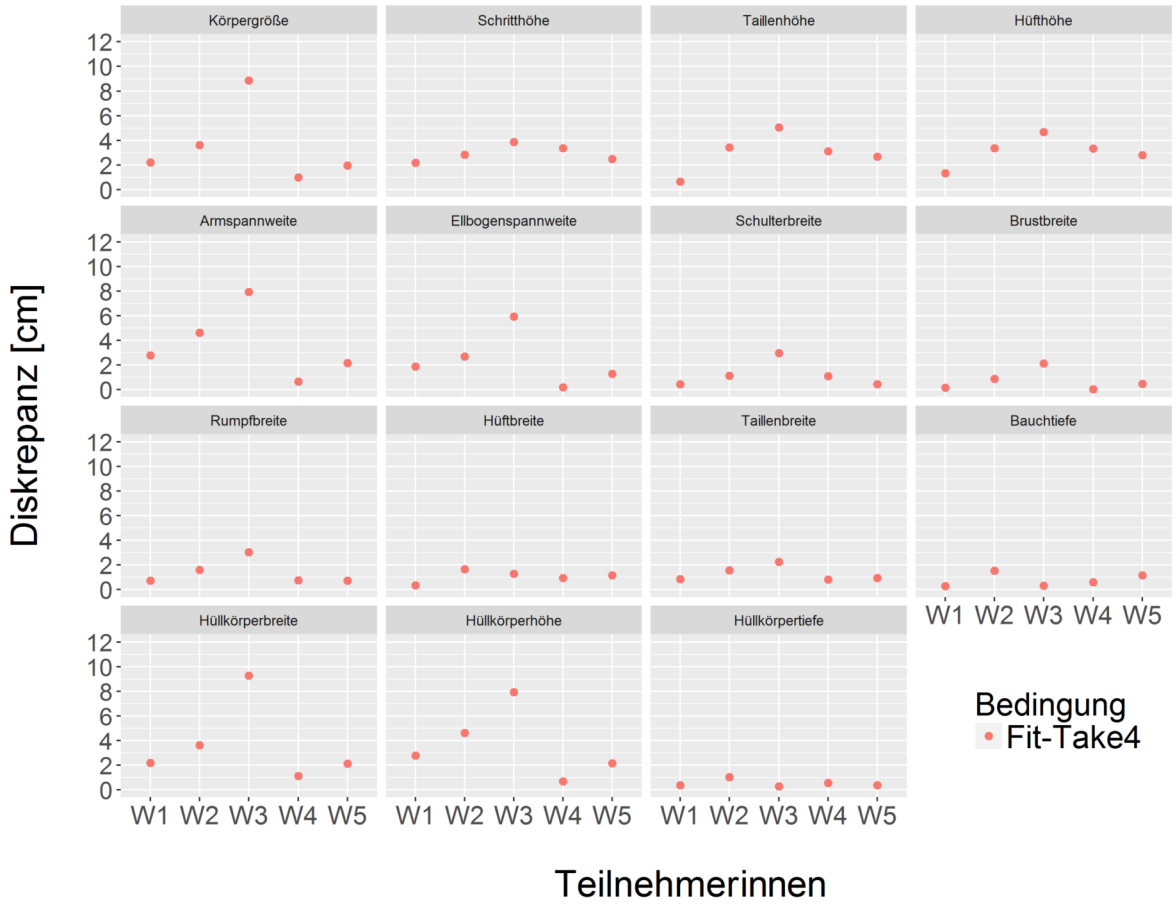

Alignment M1
Fit M1
Take1 M1
Take2 M1
Take3 M1
Take4 M1
Alignment M2
Fit M2
Take1 M2
Take2 M2
Take3 M2
Take4 M2
Alignment M3
Fit M3
Take1 M3
Take2 M3
Take3 M3
Take4 M3
Alignment M4
Fit M4
Take1 M4
Take2 M4
Take3 M4
Take4 M4
Alignment M5
Fit M5
Take1 M5
Take2 M5
Take3 M5
Take4 M5

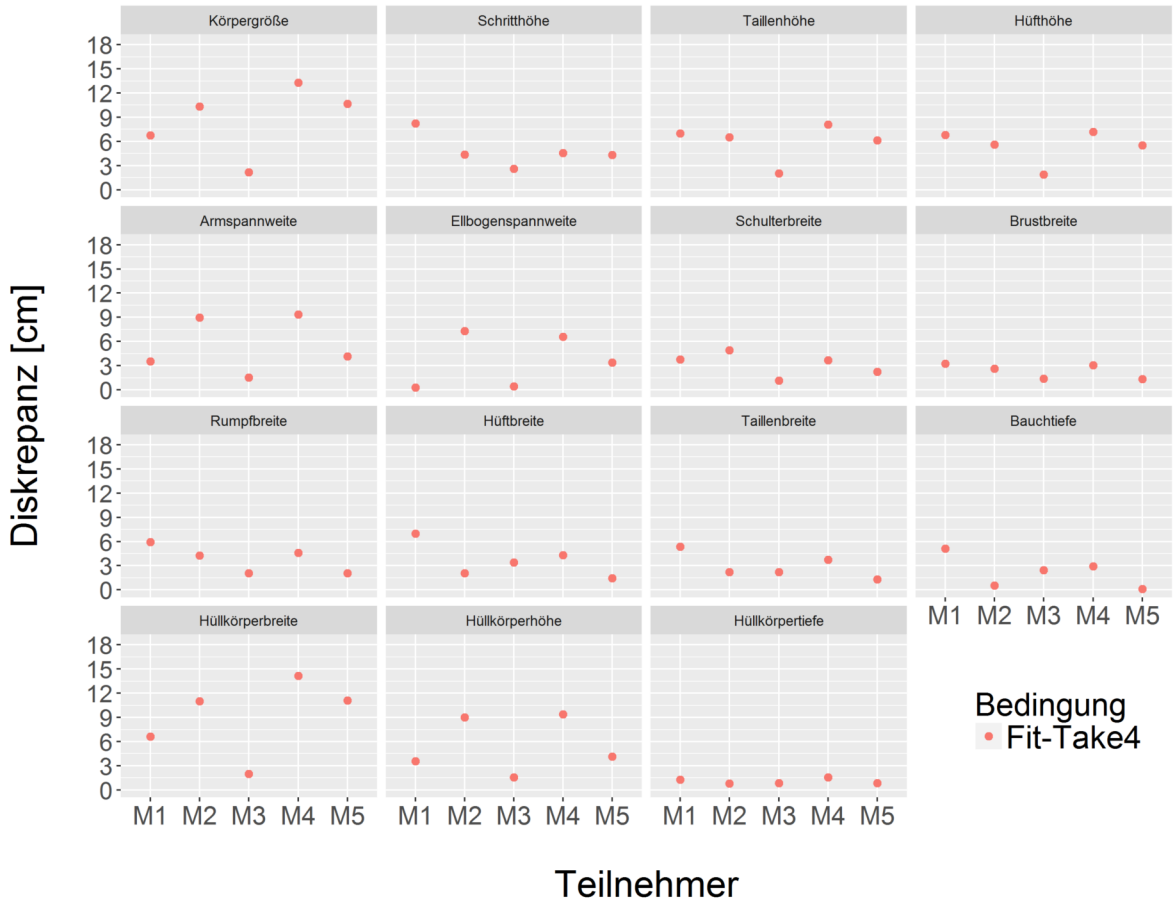
Anhang 5


Körpergröße
Schritthöhe
Taillenhöhe
Hüfthöhe
Armspannweite
Ellbogenspannweite
Schulterbreite
Brustbreite
Rumpfbreite
Hüftbreite
Taillenbreite
Bauchtiefe
Hüllkörperbreite
Hüllkörperhöhe
Hüllkörpertiefe
Diskrepanz [cm]
0
2
4
6
W1 W2 W3 W4 W5
Teilnehmerinnen
Bedingung
Alignment-Fit

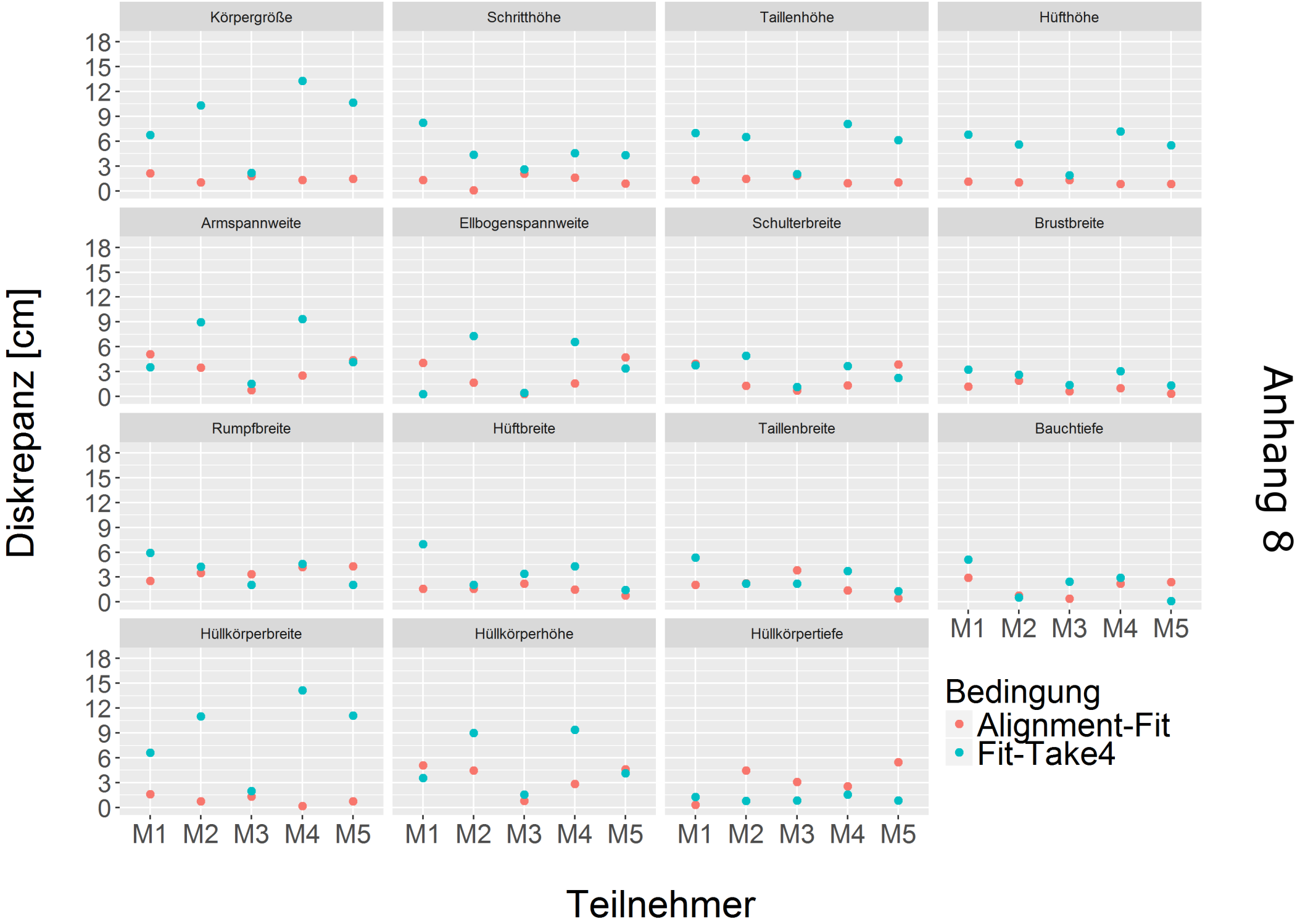

Körpergröße
Schritthöhe
Taillenhöhe
Hüfthöhe
Armspannweite
Ellbogenspannweite
Schulterbreite
Brustbreite
Rumpfbreite
Hüftbreite
Taillenbreite
Bauchtiefe
Hüllkörperbreite
Hüllkörperhöhe
Hüllkörpertiefe
Diskrepanz [cm]
0
1
2
3
4
5
6
7
M1 M2 M3 M4 M5
Teilnehmer
Bedingung
Alignment-Fit

Anhang 6


Körpergröße
Schritthöhe
Taillenhöhe
Hüfthöhe
Armspannweite
Ellbogenspannweite
Schulterbreite
Brustbreite
Rumpfbreite
Hüftbreite
Taillenbreite
Bauchtiefe
Hüllkörperbreite
Hüllkörperhöhe
Hüllkörpertiefe
Diskrepanz [cm]
W1 W2 W3 W4 W5
Teilnehmerinnen
Bedingung
Fit-Take4


Körpergröße
Schritthöhe
Taillenhöhe
Hüfthöhe
Armspannweite
Ellbogenspannweite
Schulterbreite
Brustbreite
Rumpfbreite
Hüftbreite
Taillenbreite
Bauchtiefe
Hüllkörperbreite
Hüllkörperhöhe
Hüllkörpertiefe
Diskrepanz [cm]
M1 M2 M3 M4 M5
Teilnehmer
Bedingung
Fit-Take4


Körpergröße
Schritthöhe
Taillenhöhe
Hüfthöhe
Armspannweite
Ellbogenspannweite
Schulterbreite
Brustbreite
Rumpfbreite
Hüftbreite
Taillenbreite
Bauchtiefe
Hüllkörperbreite
Hüllkörperhöhe
Hüllkörpertiefe
0
2
4
6
8
10
12
W1
W2
W3
W4
W5
Diskrepanz [cm]
Teilnehmerinnen
Bedingung
Alignment-Fit
Fit-Take4

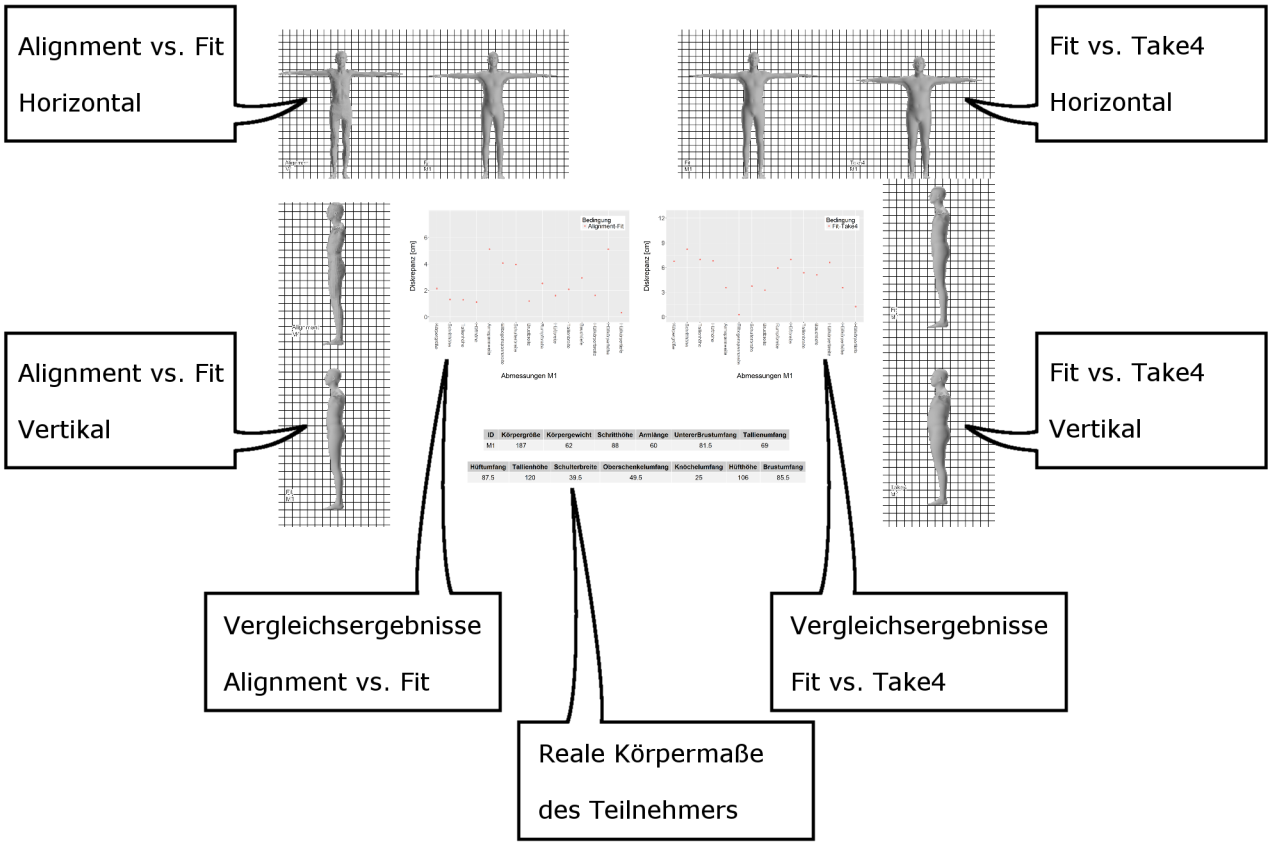

Körpergröße
Schritthöhe
Taillenhöhe
Hüfthöhe
Armspannweite
Ellbogenspannweite
Schulterbreite
Brustbreite
Rumpfbreite
Hüftbreite
Taillenbreite
Bauchtiefe
Hüllkörperbreite
Hüllkörperhöhe
Hüllkörpertiefe
18
15
12
9
6
3
0
M1
M2
M3
M4
M5
Diskrepanz [cm]
Teilnehmer
Bedingung
Alignment-Fit
Fit-Take4

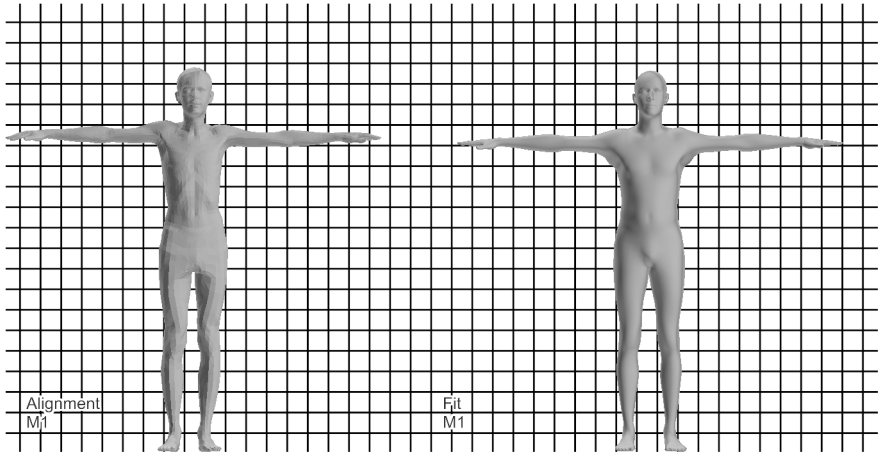
Anhang 8

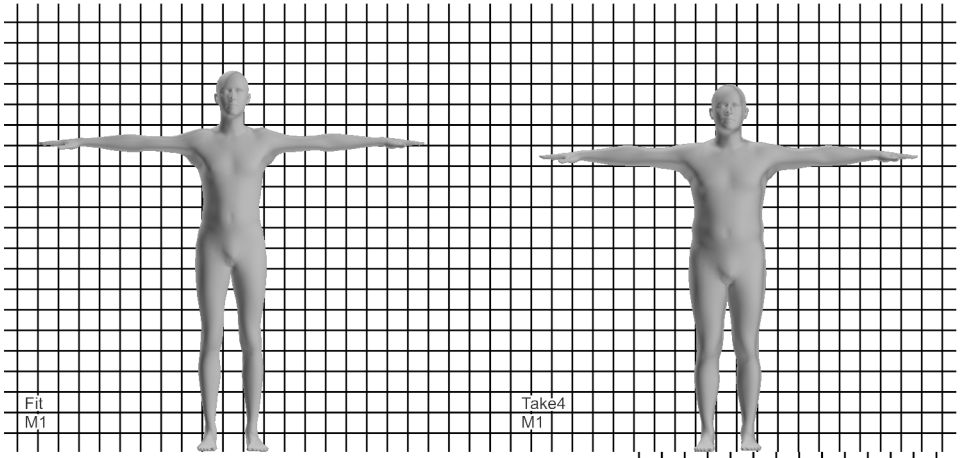
Anhang 9

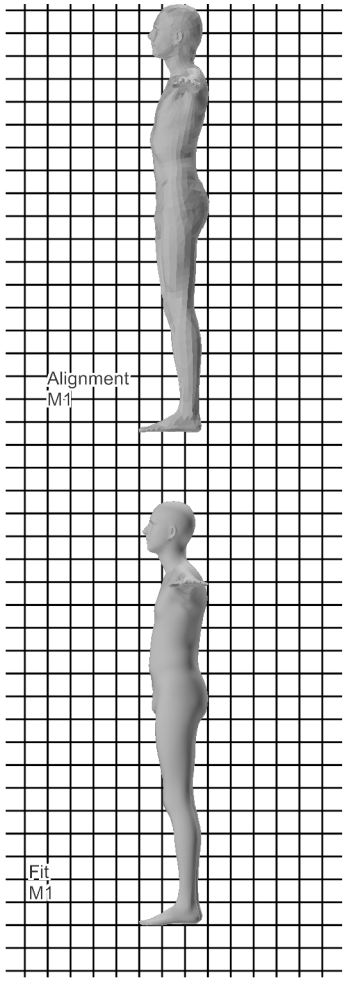
Es folgen die Einzelvergleiche für alle Teilnehmer der Validierung.

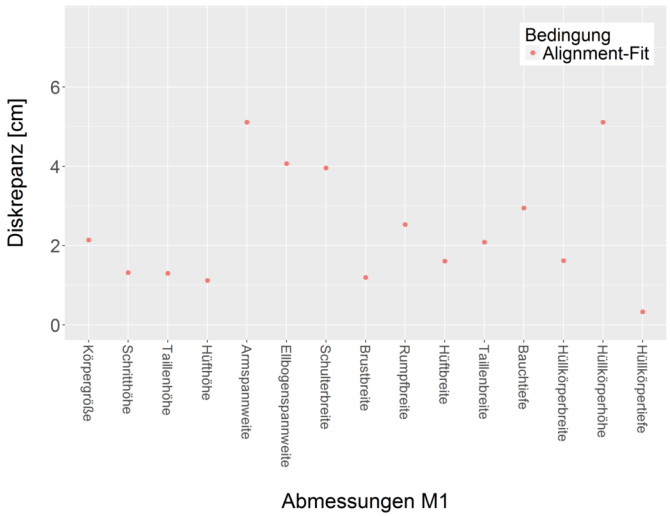
Der Aufbau der nächsten zehn Seiten ist wie folgt:

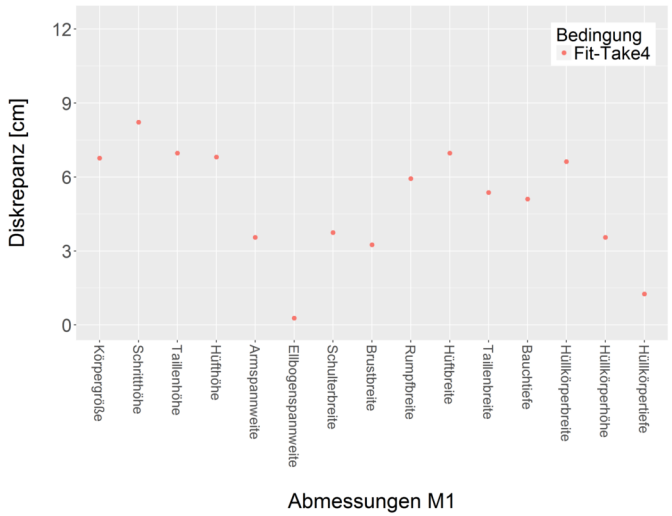


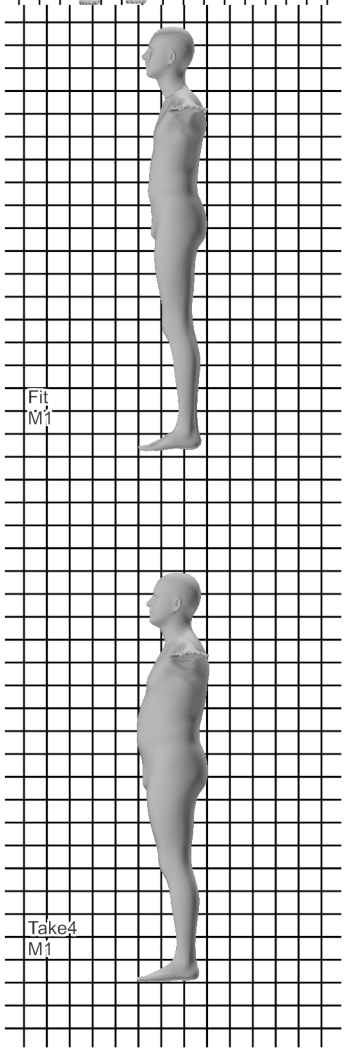






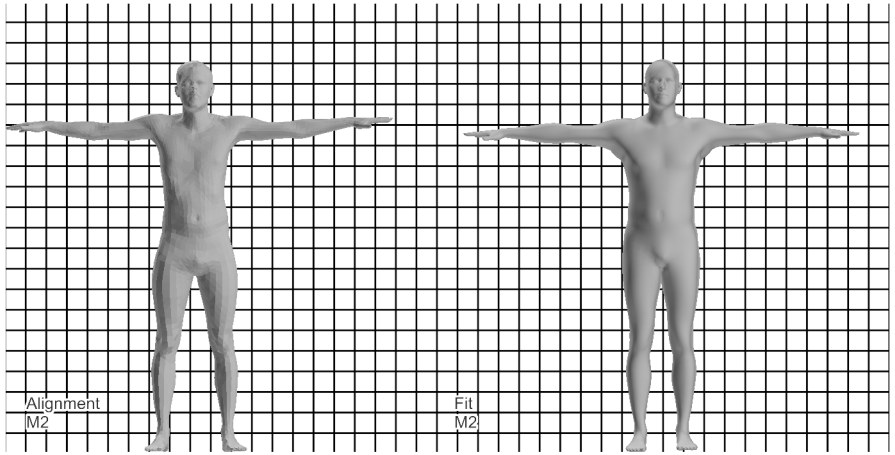


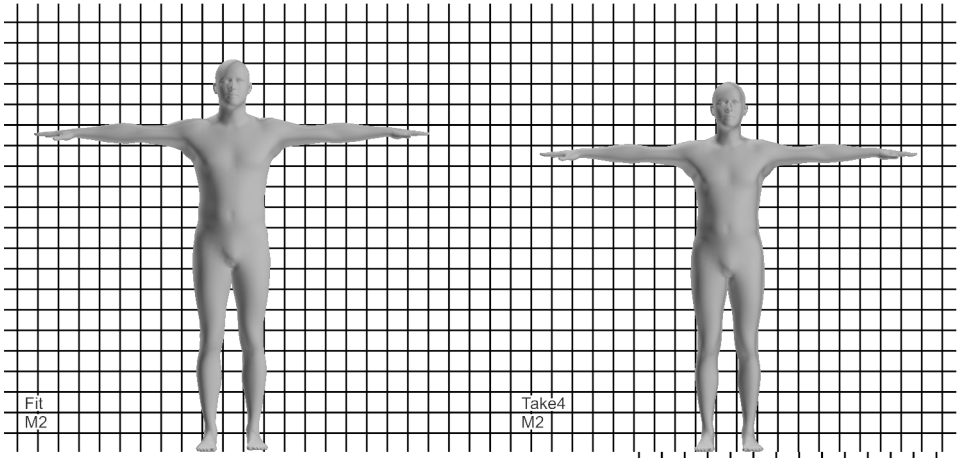


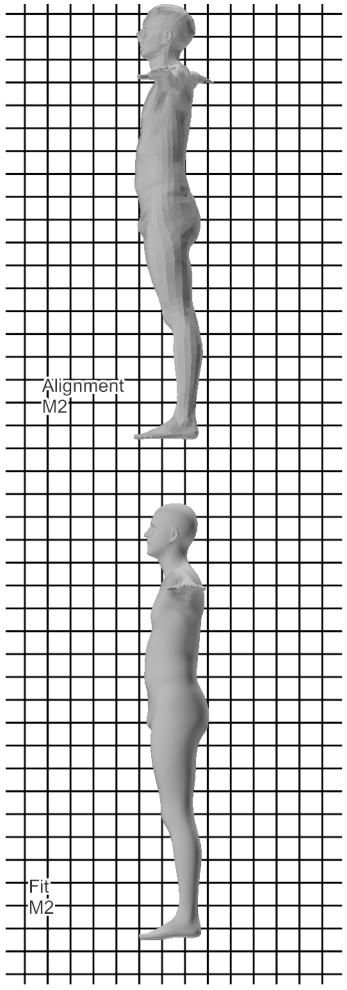


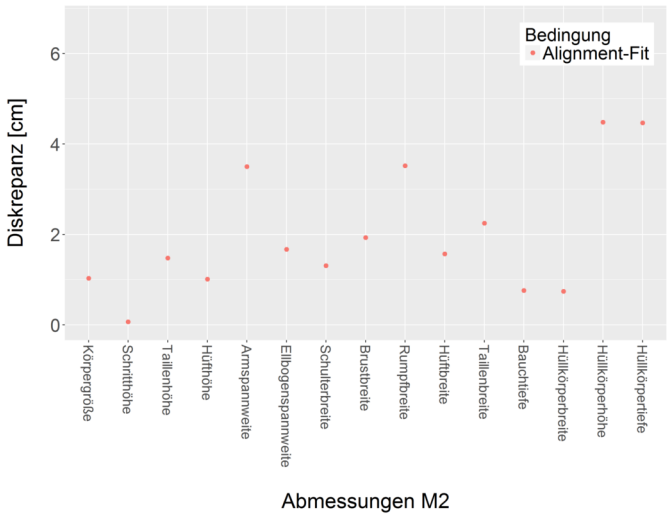
ID	Körpergröße	Körpergewicht	Schritthöhe	Armlänge	UntererBrustumfang	Taillenumfang
M1	187	62	88	60	81.5	69

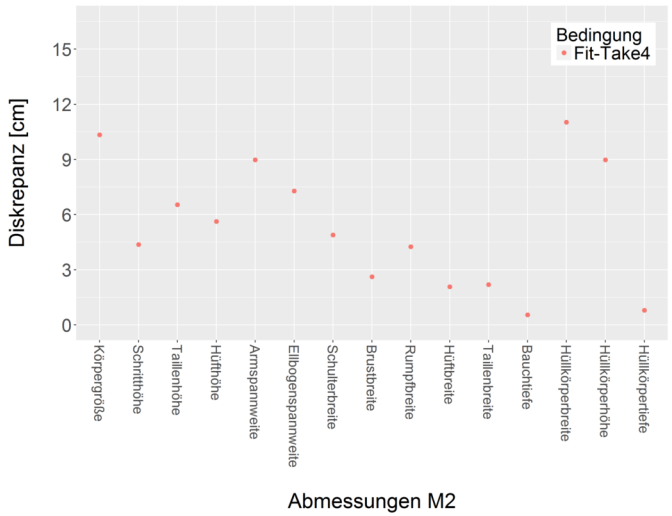
Hüftumfang	Taillenhöhe	Schulterbreite	Oberschenkelumfang	Knöchelumfang	Hüfthöhe	Brustumfang
87.5	120	39.5	49.5	25	106	85.5

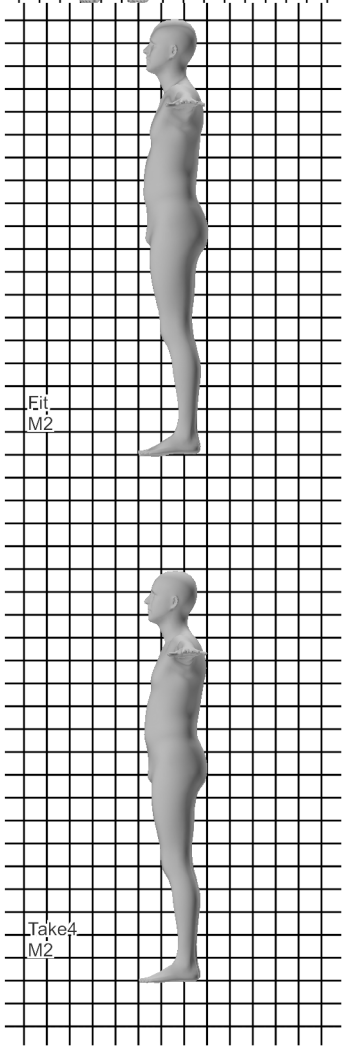






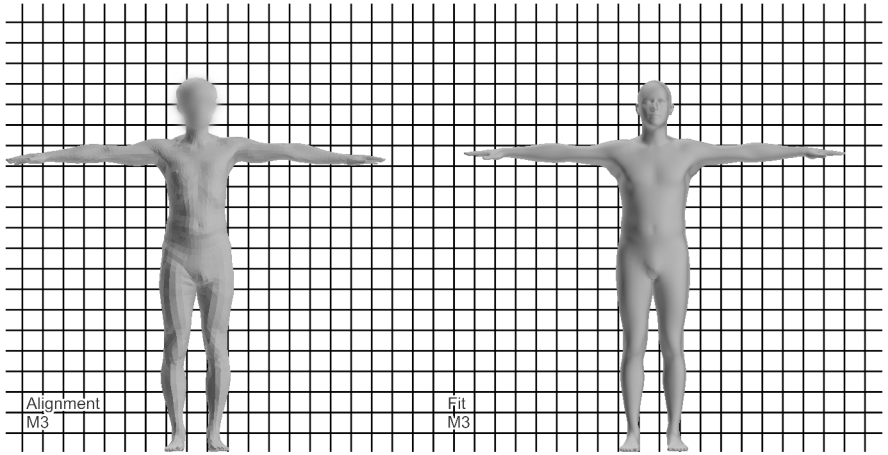


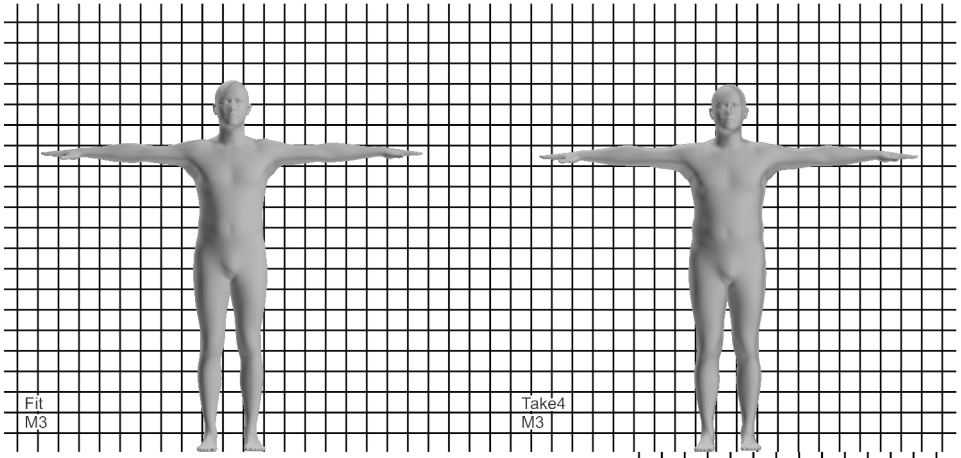


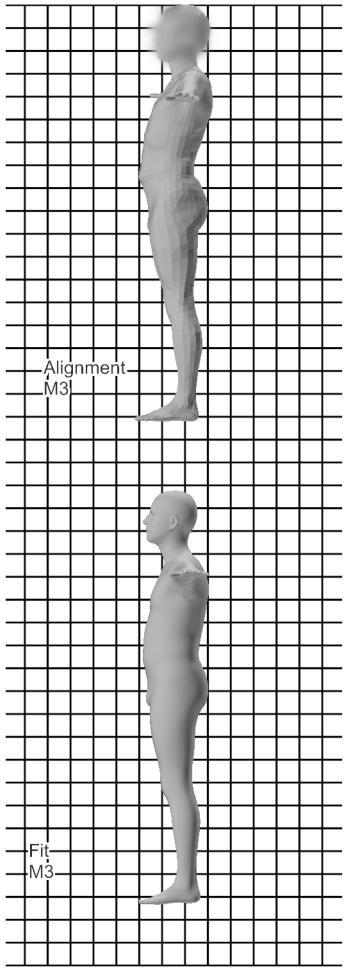


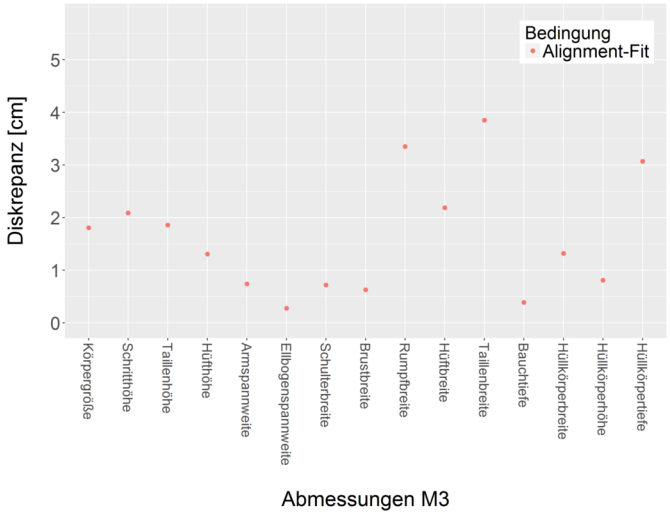
ID	Körpergröße	Körpergewicht	Schritthöhe	Armlänge	UntererBrustumfang	Taillenumfang
M2	189.5	82.8	92	62	93.5	82.5

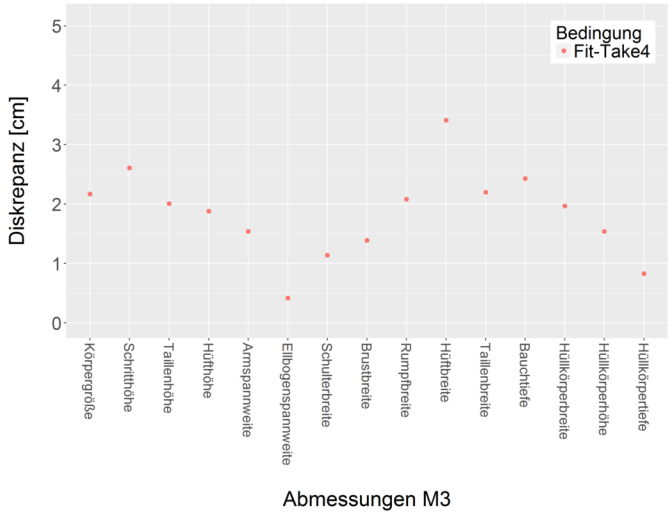
Hüftumfang	Taillenhöhe	Schulterbreite	Oberschenkelumfang	Knöchelumfang	Hüfthöhe	Brustumfang
102	122	48.5	61	25	100	95.5

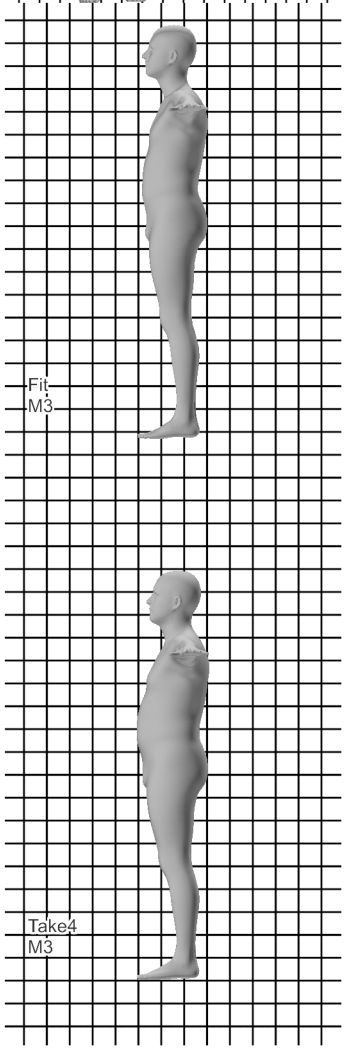






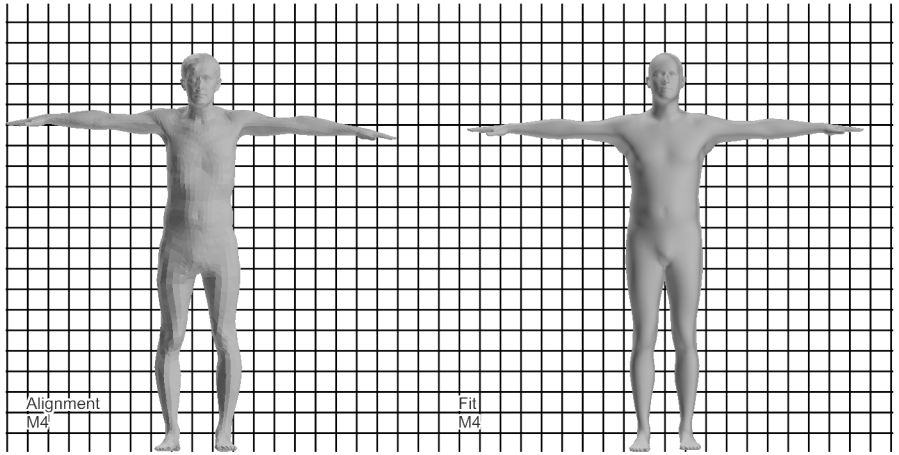


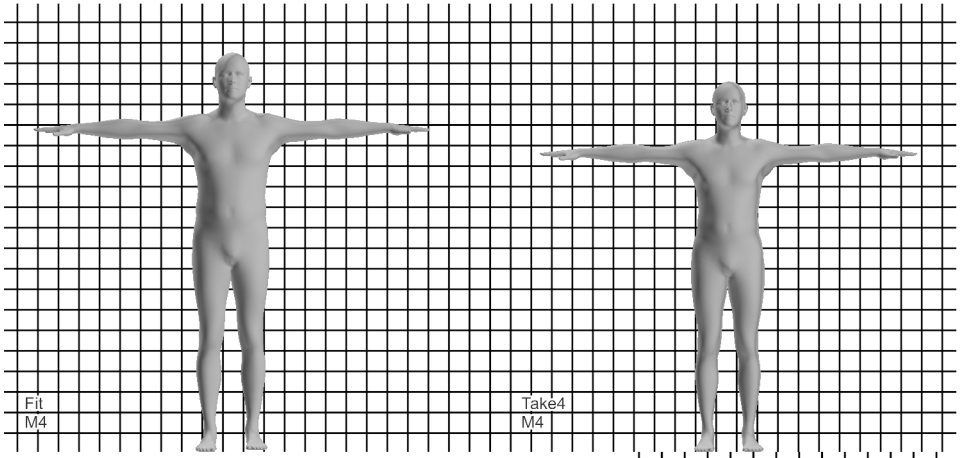


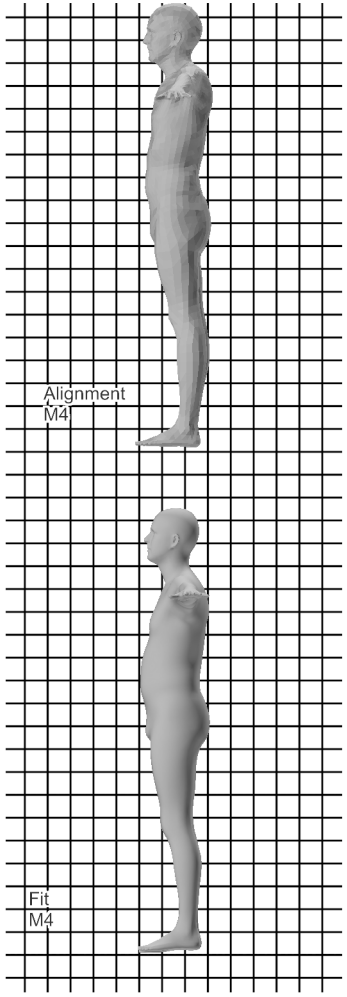


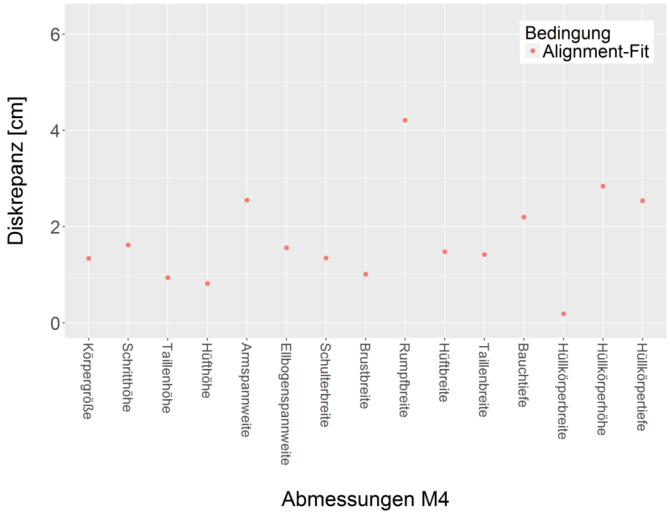
ID	Körpergröße	Körpergewicht	Schritthöhe	Armlänge	UntererBrustumfang	Taillenumfang
M3	181	78.2	83	63	92	87.5

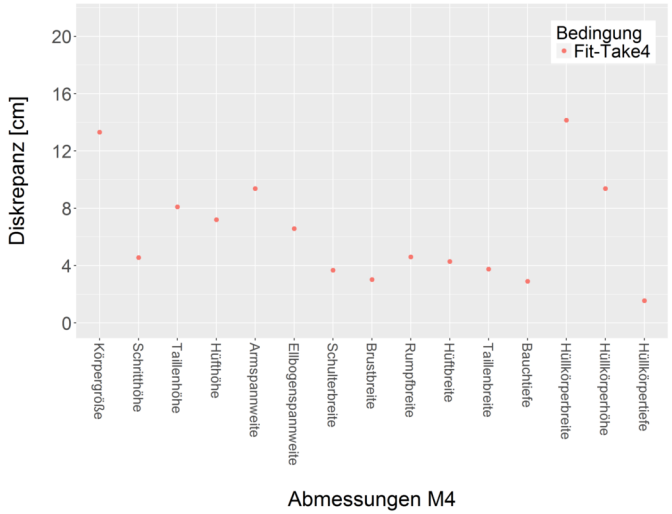
Hüftumfang	Taillenhöhe	Schulterbreite	Oberschenkelumfang	Knöchelumfang	Hüfthöhe	Brustumfang
92.5	112	46	56	25	101	92

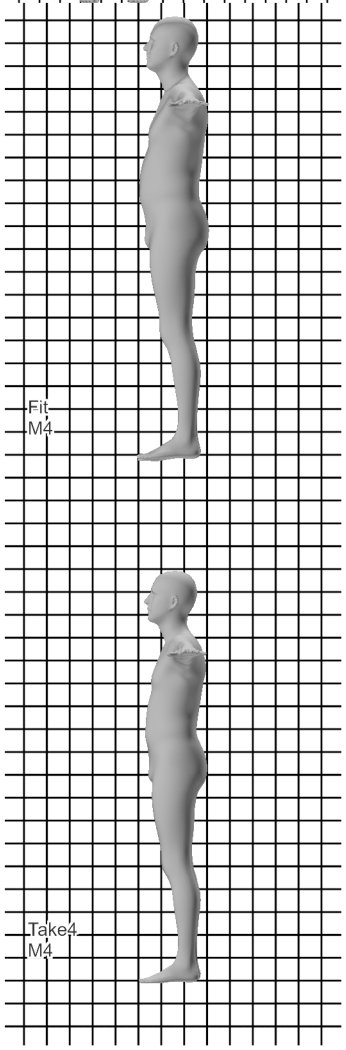






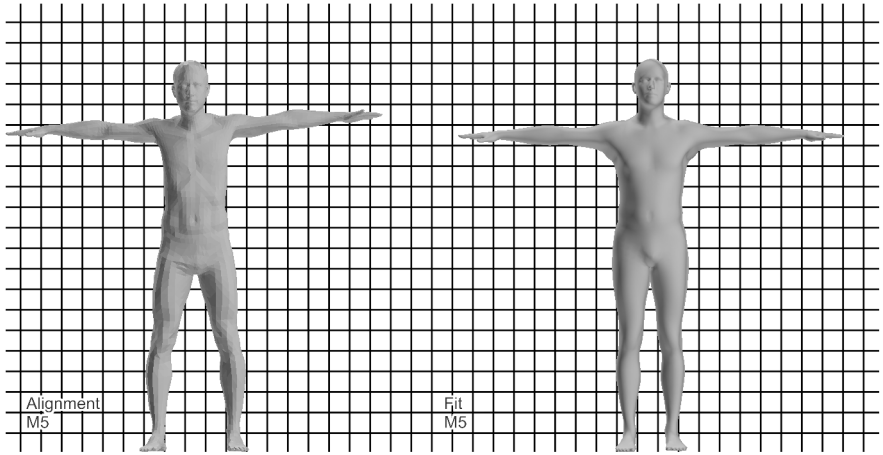


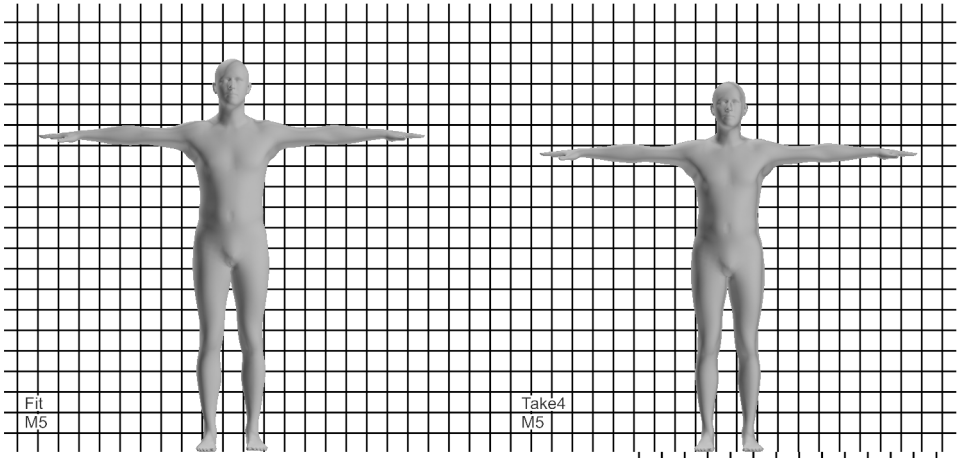


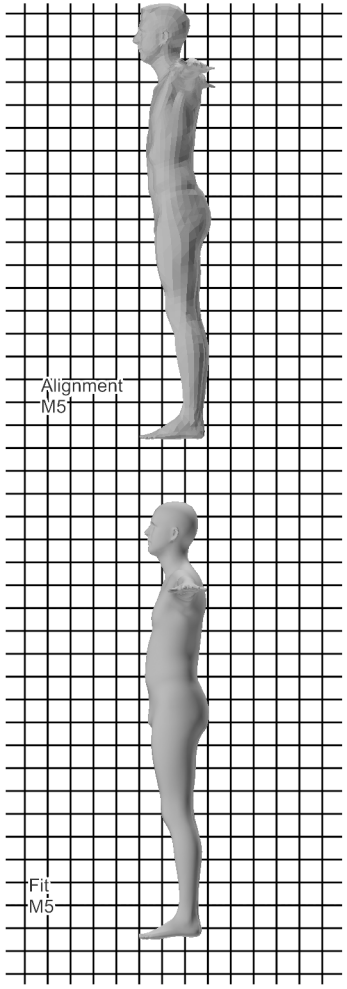


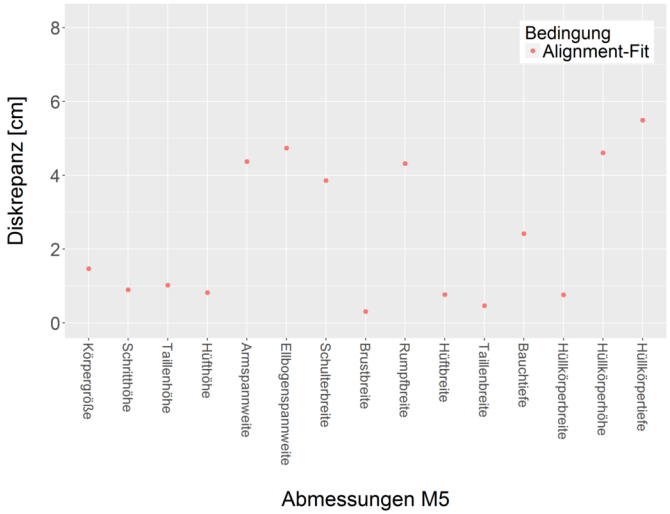
ID	Körpergröße	Körpergewicht	Schritthöhe	Armlänge	UntererBrustumfang	Taillenumfang
M4	194	86.5	91	62	98	87

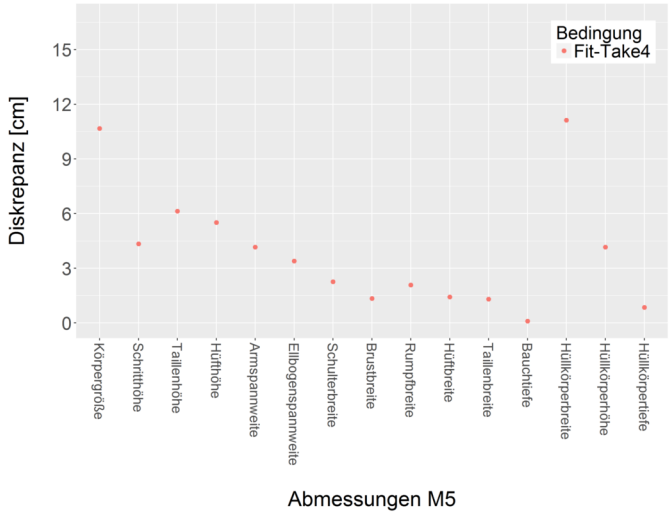
Hüftumfang	Taillenhöhe	Schulterbreite	Oberschenkelumfang	Knöchelumfang	Hüfthöhe	Brustumfang
101	121	46.5	56.5	25	99	99.5

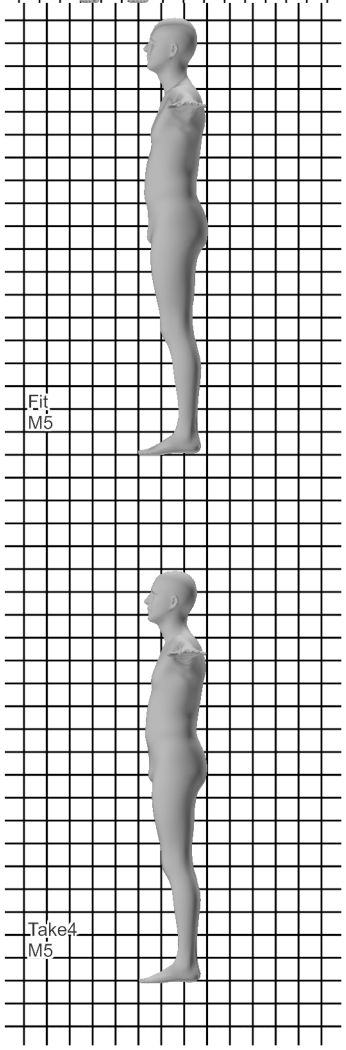






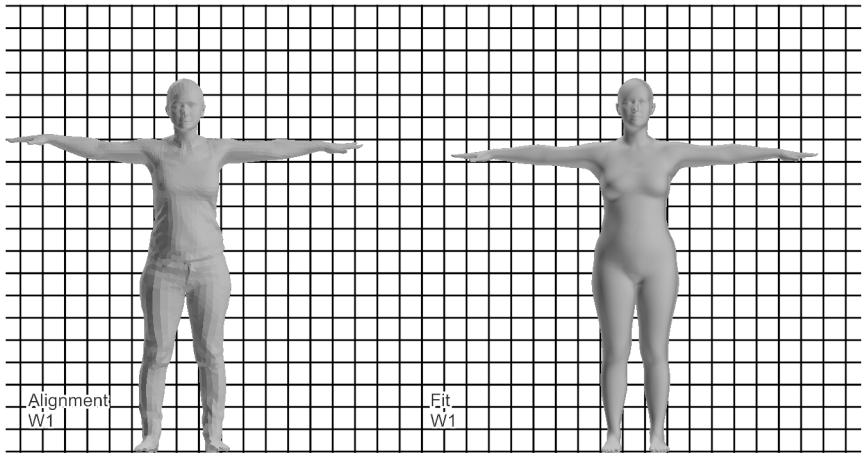


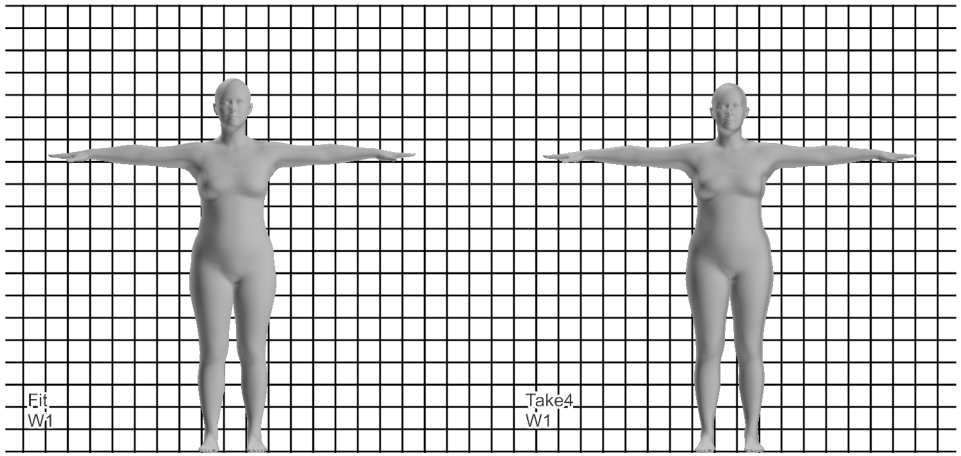


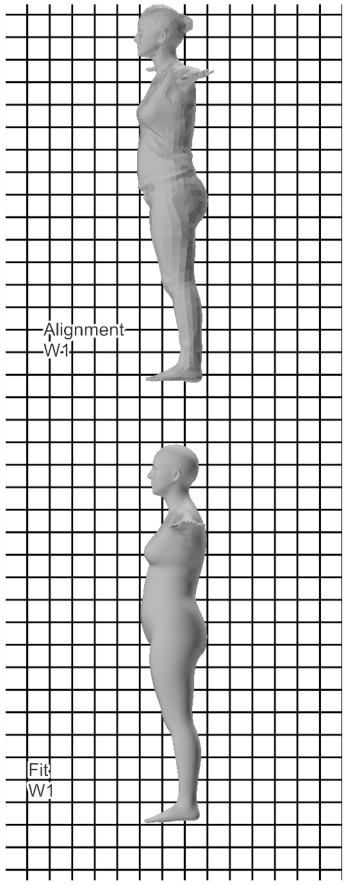


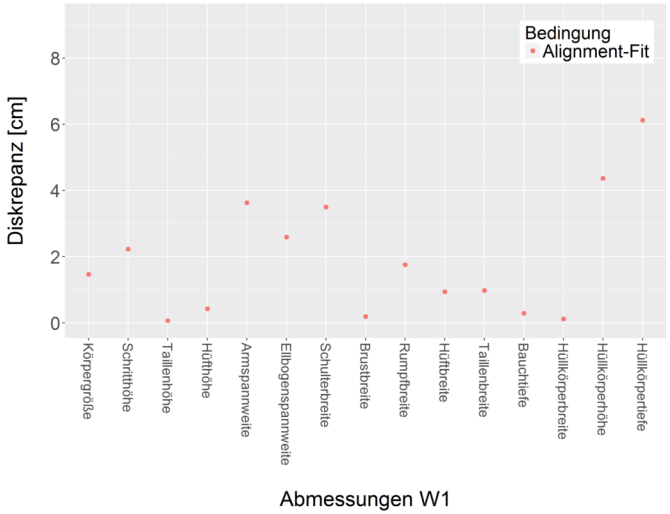
ID	Körpergröße	Körpergewicht	Schritthöhe	Armlänge	UntererBrustumfang	Taillenumfang
M5	189.5	80.1	92	59.5	99	91.5

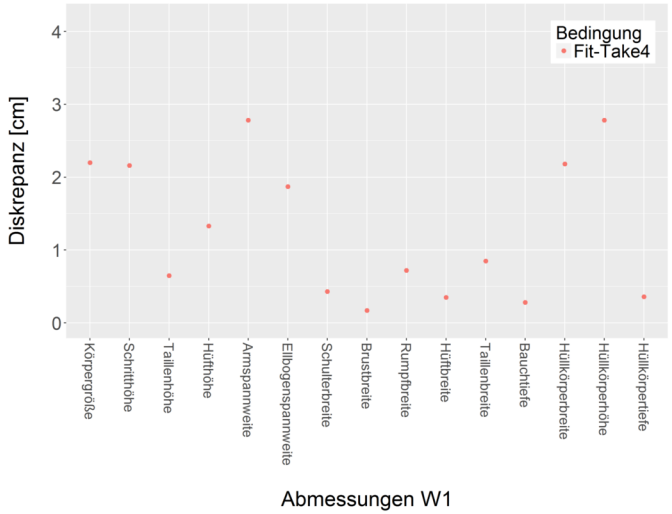
Hüftumfang	Taillenhöhe	Schulterbreite	Oberschenkelumfang	Knöchelumfang	Hüfthöhe	Brustumfang
101	114.5	54	58.5	24.5	92.5	105

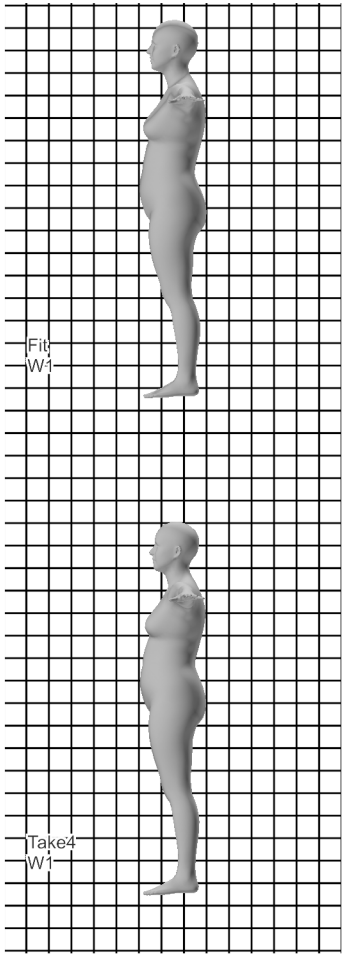






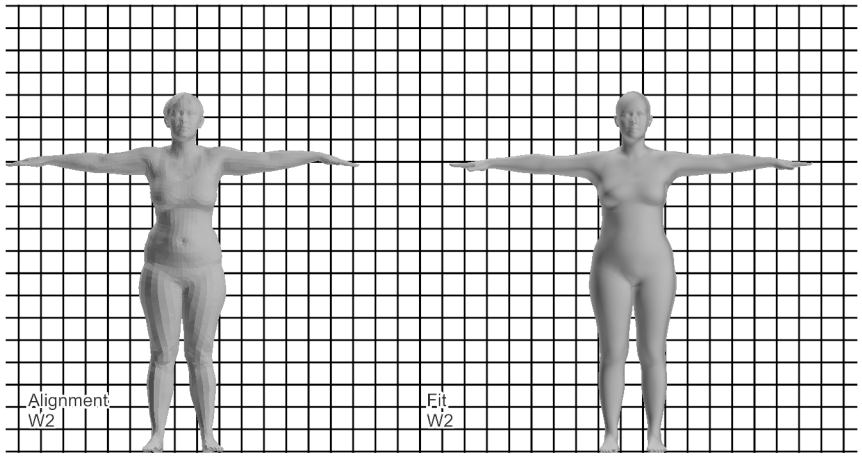


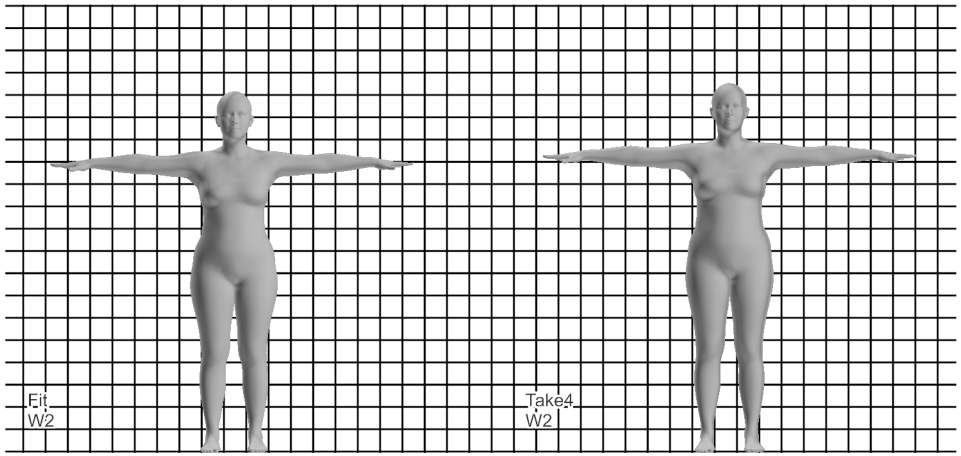


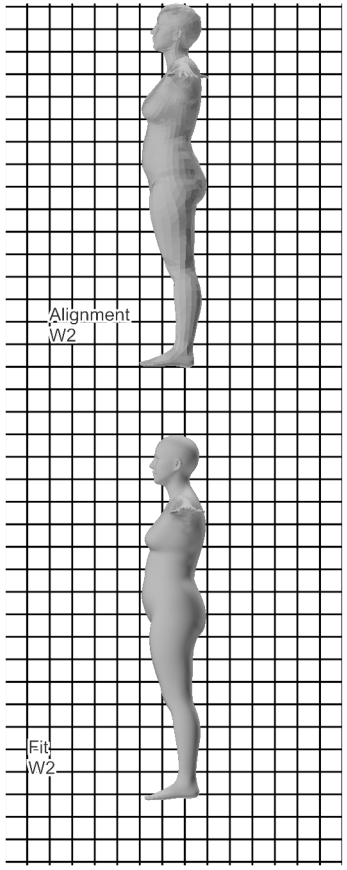


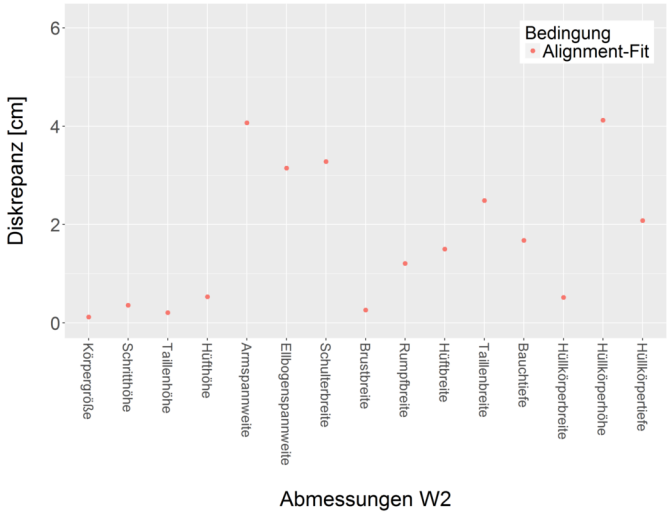
ID	Körpergröße	Körpergewicht	Schritthöhe	Armlänge	UntererBrustumfang	Taillenumfang
W1	166	71	73	58	94	95

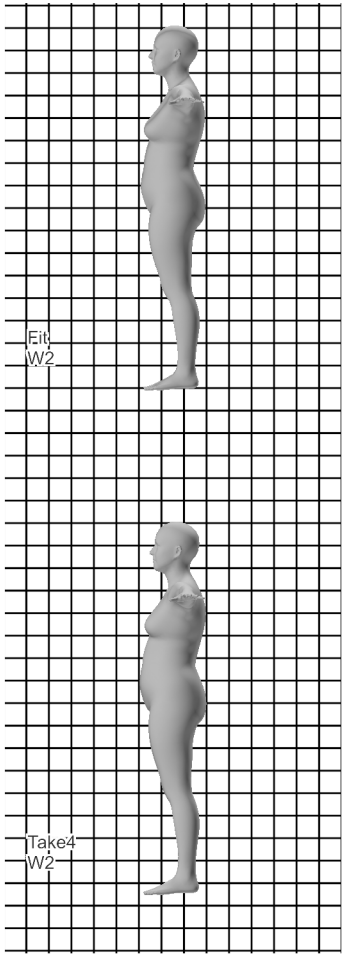
Hüftumfang	Taillenhöhe	Schulterbreite	Oberschenkelumfang	Knöchelumfang	Hüfthöhe	Brustumfang
108	92	44	63	25	77	96.2





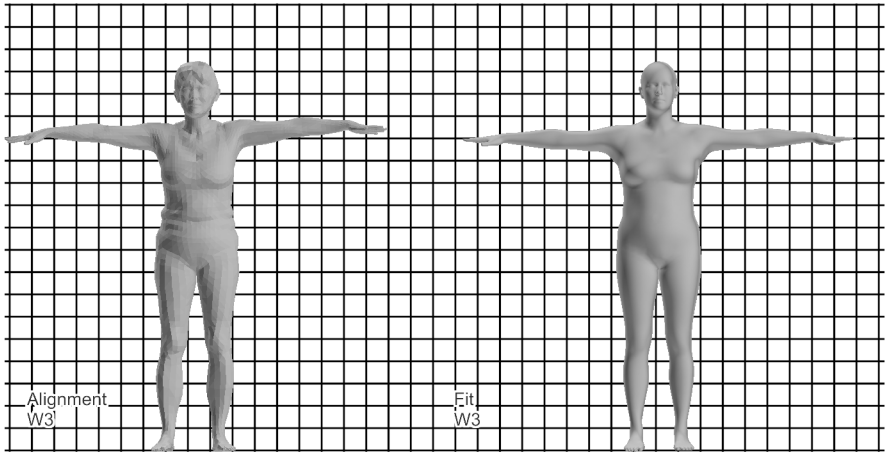


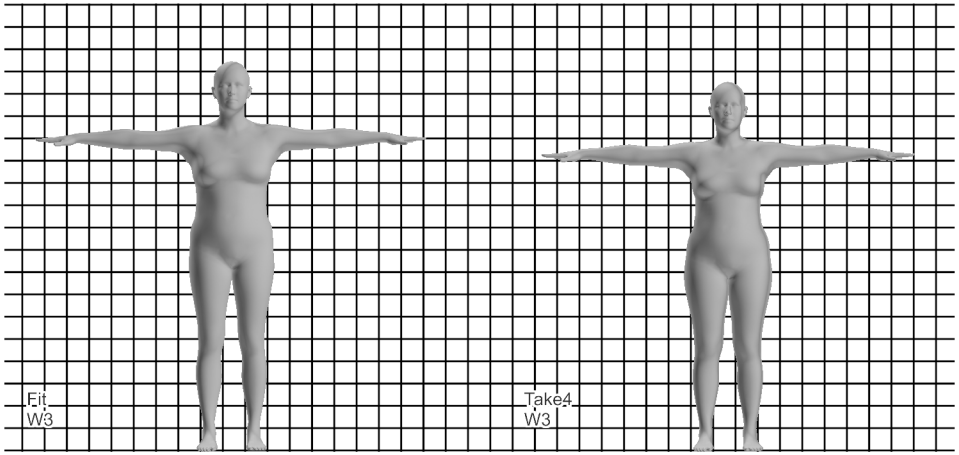


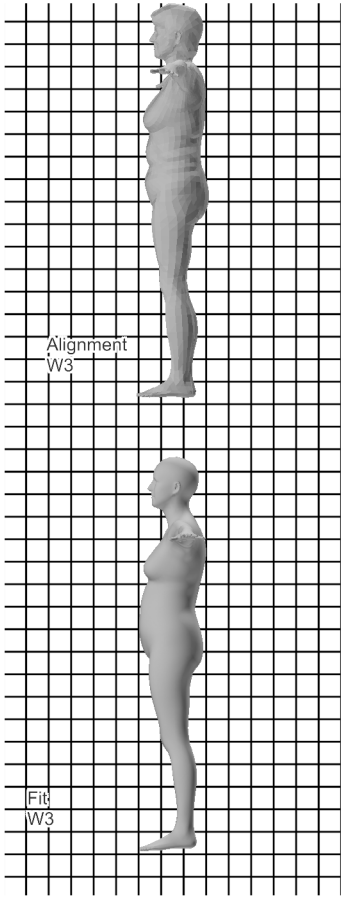


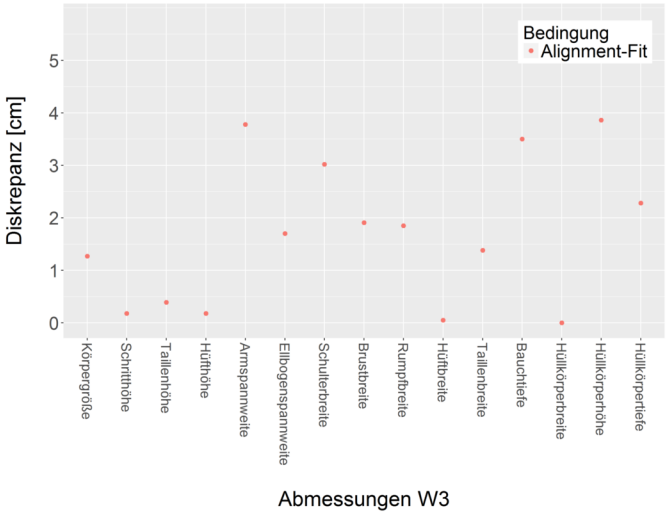
ID	Körpergröße	Körpergewicht	Schritthöhe	Armlänge	UntererBrustumfang	Taillenumfang
W2	159.5	64.3	73	50	93	71.5

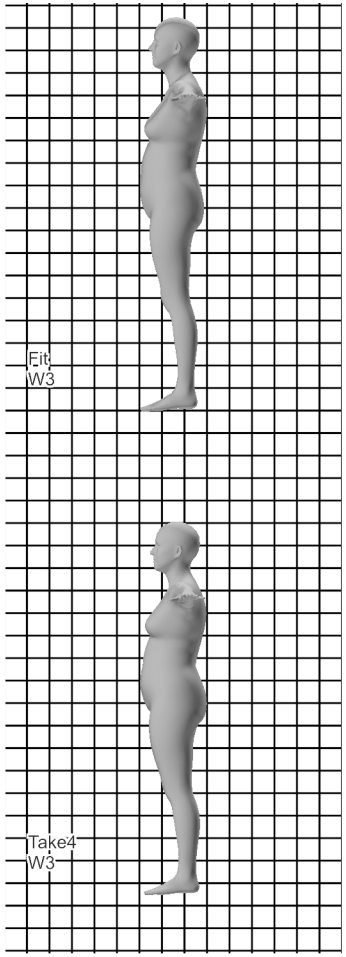
Hüftumfang	Taillenhöhe	Schulterbreite	Oberschenkelumfang	Knöchelumfang	Hüfthöhe	Brustumfang
96	102	41	62	25	83	91.5





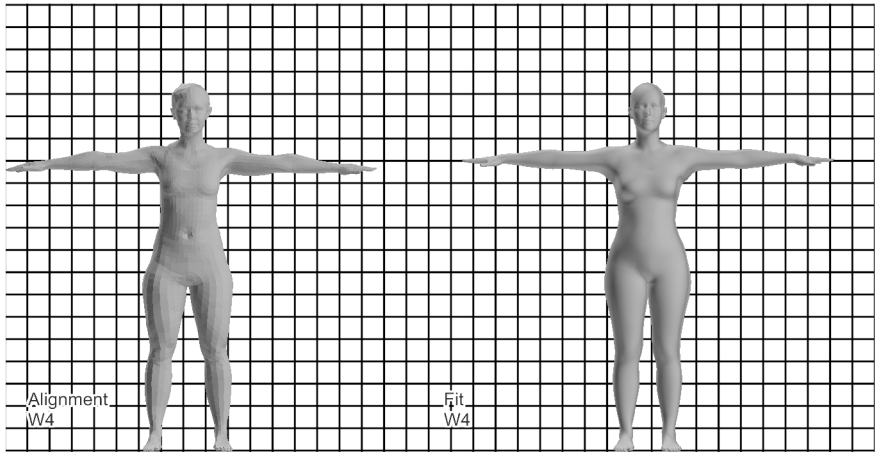


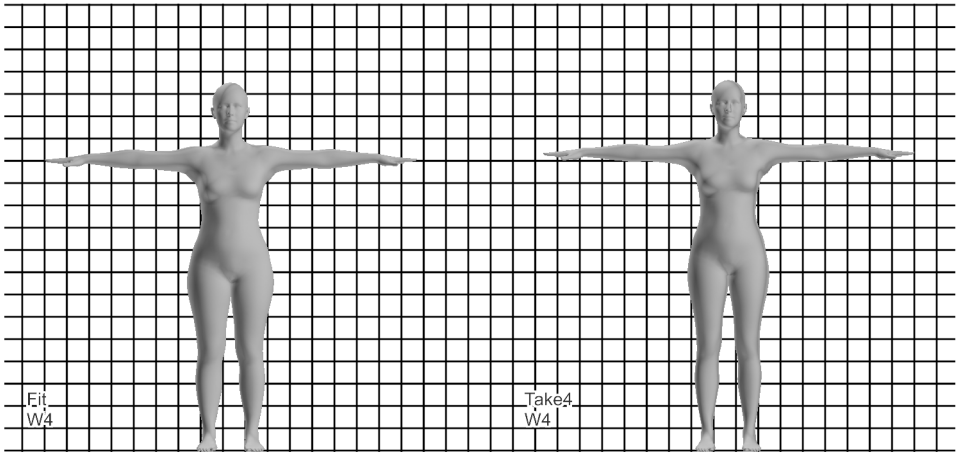


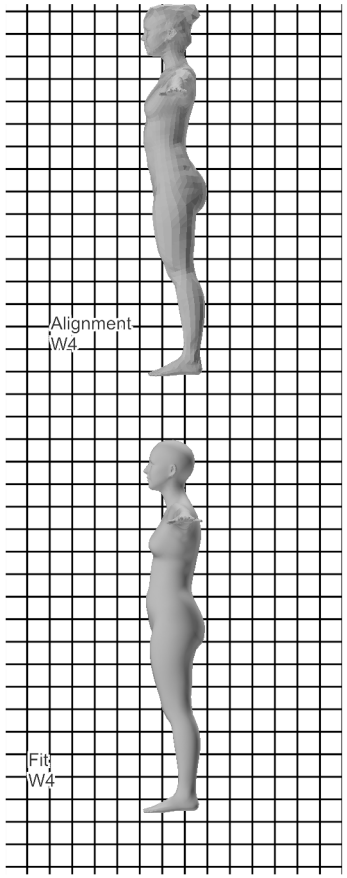


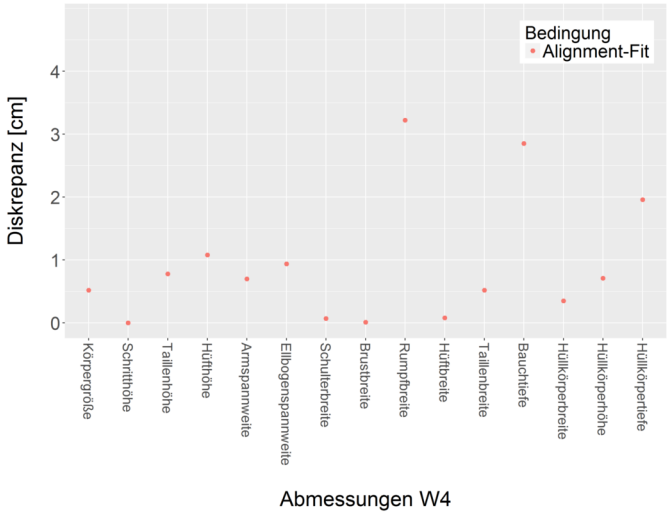
ID	Körpergröße	Körpergewicht	Schritthöhe	Armlänge	UntererBrustumfang	Taillenumfang
W3	173	72.8	83	58	99.5	83

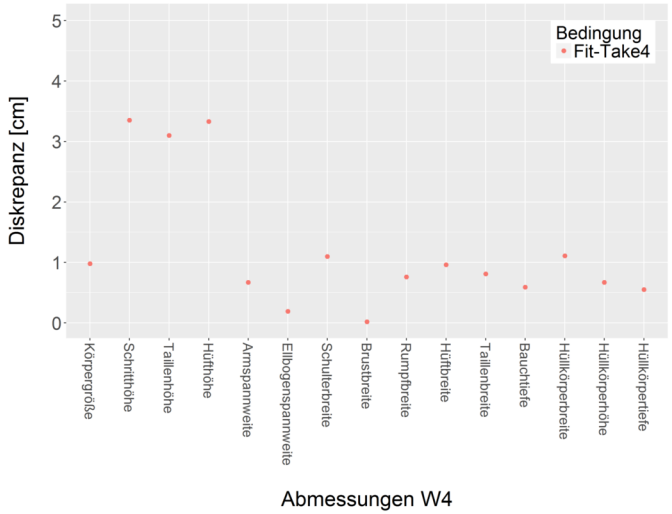
Hüftumfang	Taillenhöhe	Schulterbreite	Oberschenkelumfang	Knöchelumfang	Hüfthöhe	Brustumfang
98.5	113	42	56	25	102	95

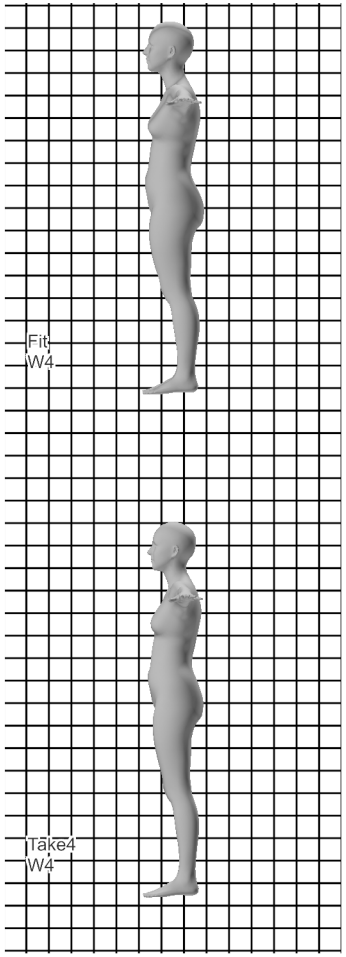






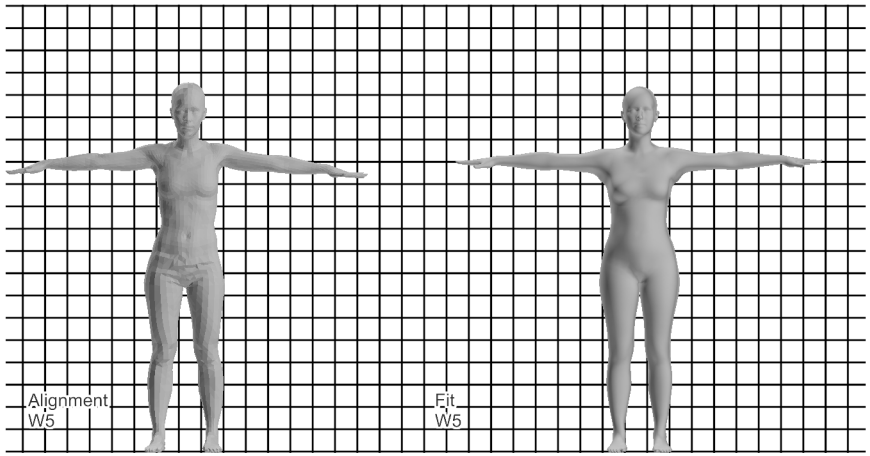


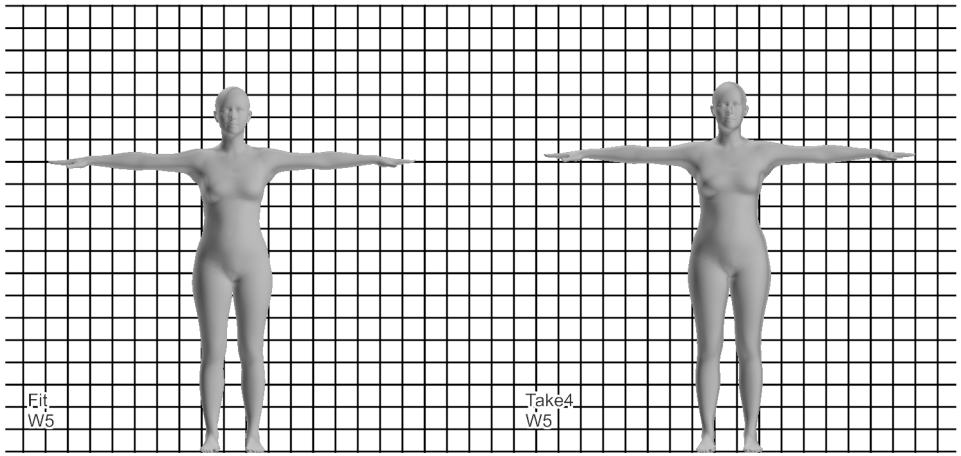


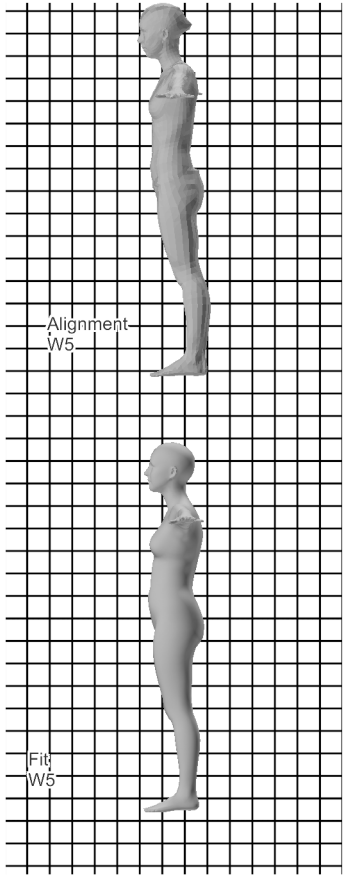


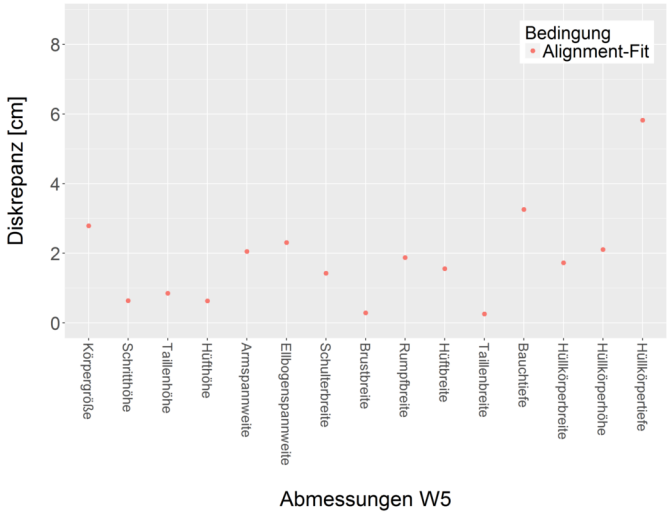
ID	Körpergröße	Körpergewicht	Schritthöhe	Armlänge	UntererBrustumfang	Taillenumfang
W4	164	63.2	79	54	84	75.5

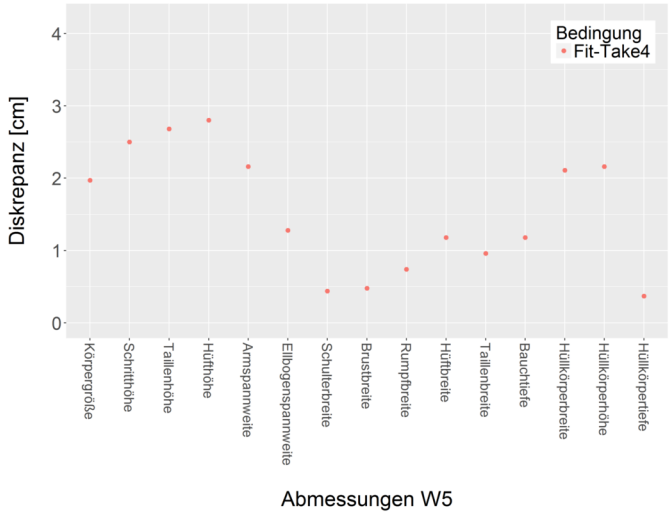
Hüftumfang	Taillenhöhe	Schulterbreite	Oberschenkelumfang	Knöchelumfang	Hüfthöhe	Brustumfang
103.5	100.5	41	59	23.5	83	80.5

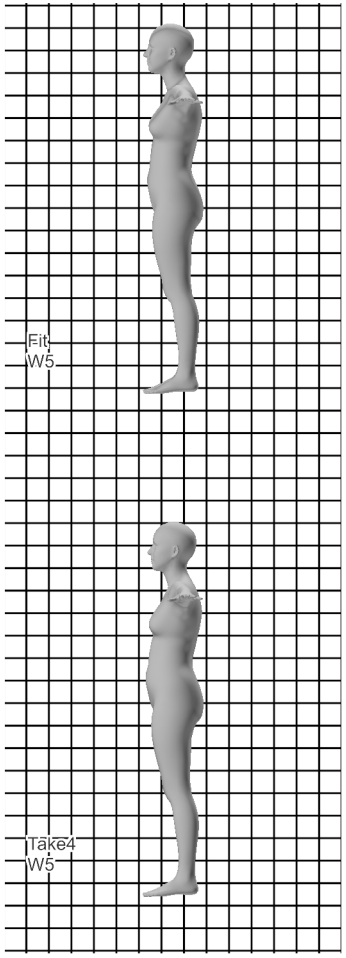










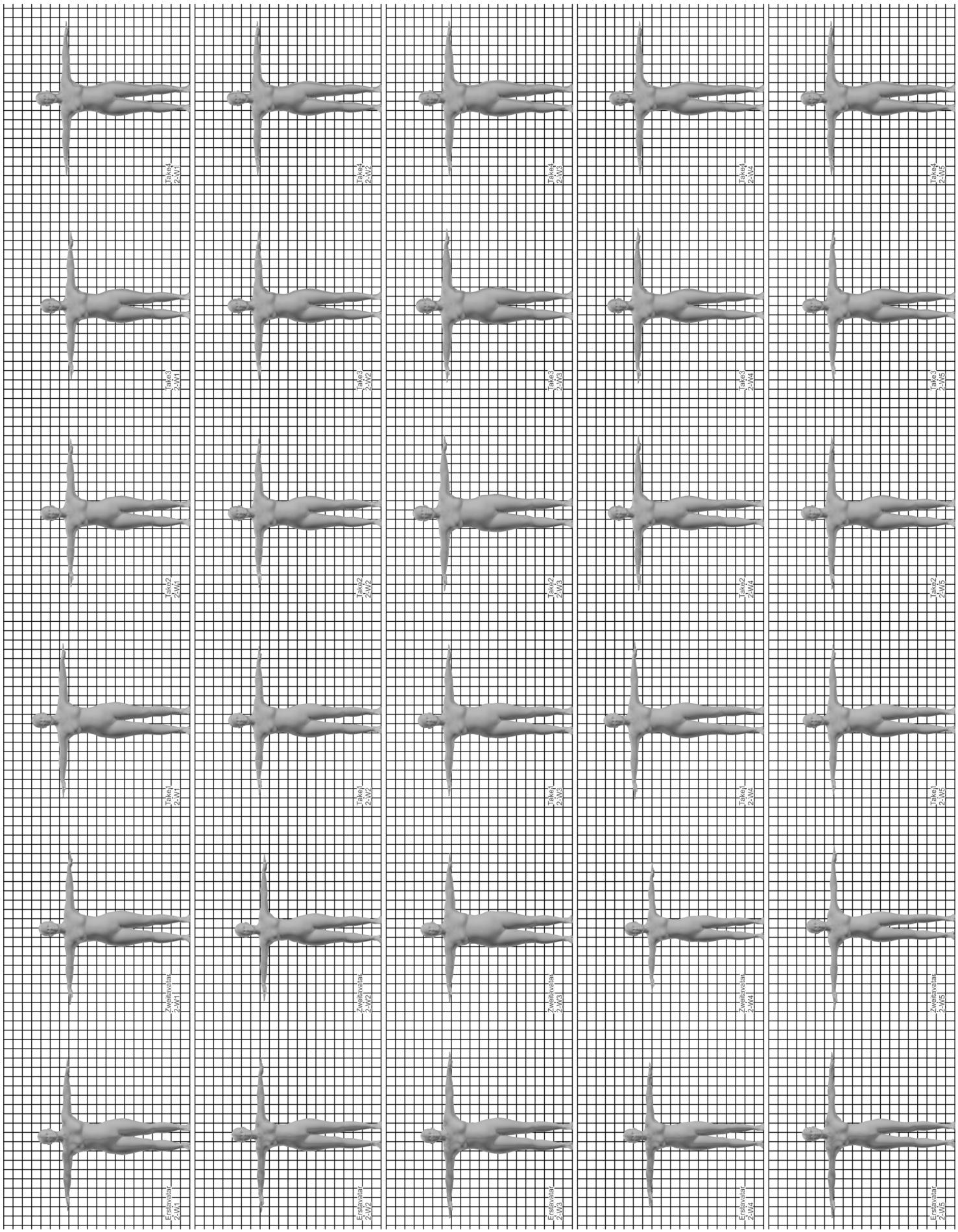


ID	Körpergröße	Körpergewicht	Schritthöhe	Armlänge	UntererBrustumfang	Taillenumfang
W5	165	55.3	77	58	75.5	73

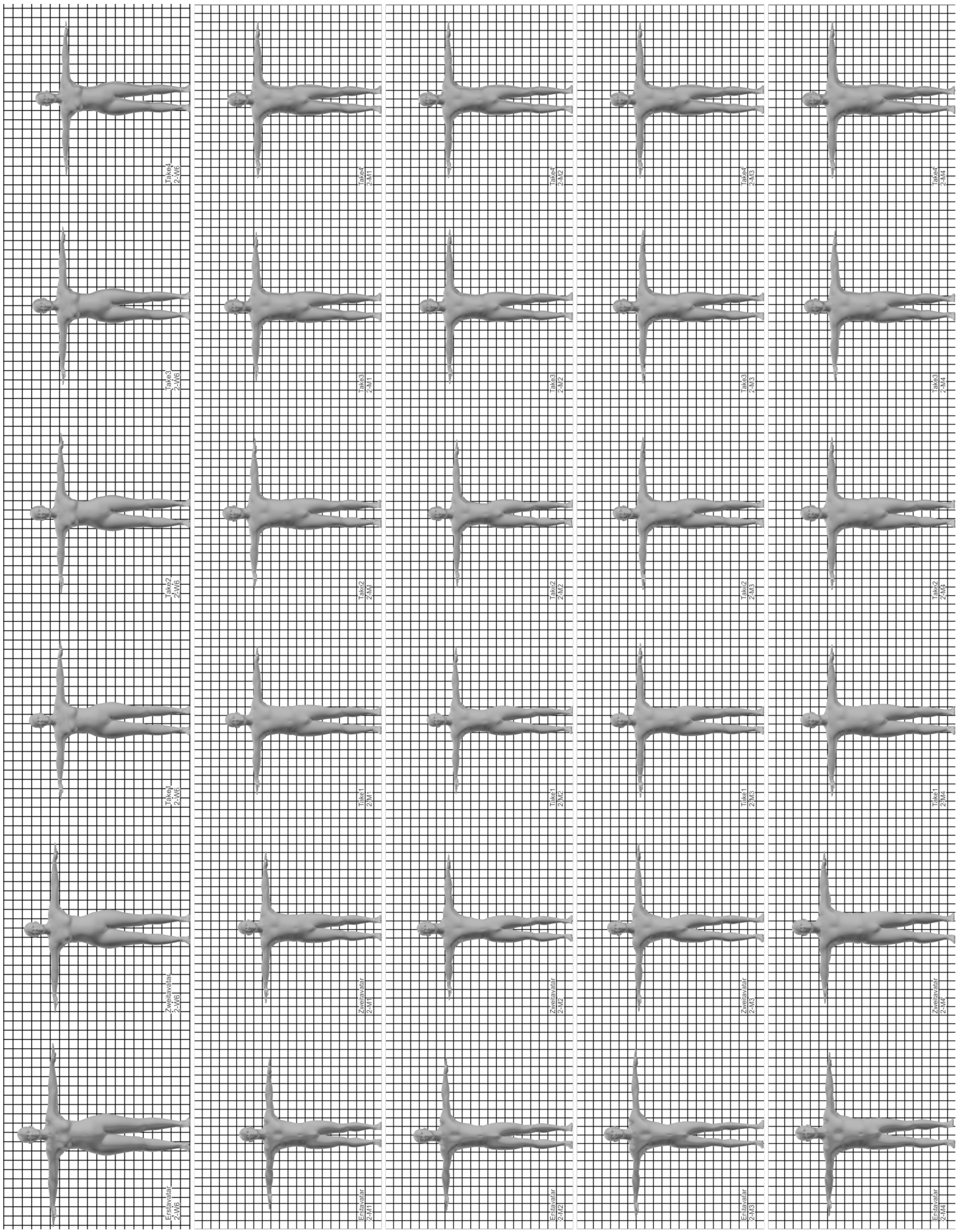
Hüftumfang	Taillenhöhe	Schulterbreite	Oberschenkelumfang	Knöchelumfang	Hüfthöhe	Brustumfang
93	97	38	52.5	23	82.5	83

ID	Körpergröße	Körpergewicht	Schritthöhe	Armlänge	UntererBrustumfang	Taillenumfang	Hüftumfang	Taillenhöhe	Schulterbreite	Oberschenkelumfang	Knöchelumfang	Hüfthöhe	Brustumfang
2-W1	161.0	56.3	70.0	55.0	87.0	71	96.0	110.0	36.0	56.0	23.0	80.0	81.0
2-W2	157.5	49.5	71.0	51.0	85.0	67	90.0	97.0	37.5	52.0	24.0	75.5	79.0
2-M1	176.0	72.8	79.0	50.0	97.0	89	102.0	98.5	44.0	51.5	26.0	85.0	97.0
2-W3	162.5	68.2	75.5	53.0	97.0	95	101.5	91.5	40.0	56.0	26.5	85.5	92.0
2-M2	189.0	78.4	86.5	67.0	92.0	65	103.0	111.0	46.0	57.5	27.5	96.5	92.0
2-W4	160.0	46.4	76.5	55.6	78.0	62	92.0	105.0	36.5	51.0	23.0	83.5	78.0
2-M3	175.5	66.7	77.5	63.0	85.5	81	92.0	103.5	43.5	54.0	27.0	94.0	90.5
2-W5	163.0	55.5	74.0	56.0	82.0	74	90.0	99.5	38.5	52.5	25.0	82.0	81.0
2-W6	162.5	52.2	79.0	57.5	81.5	78	99.0	101.0	37.5	52.0	25.8	84.0	80.5
2-M4	170.0	79.9	71.5	57.0	96.0	101	108.0	100.0	42.6	63.5	29.5	83.0	95.0

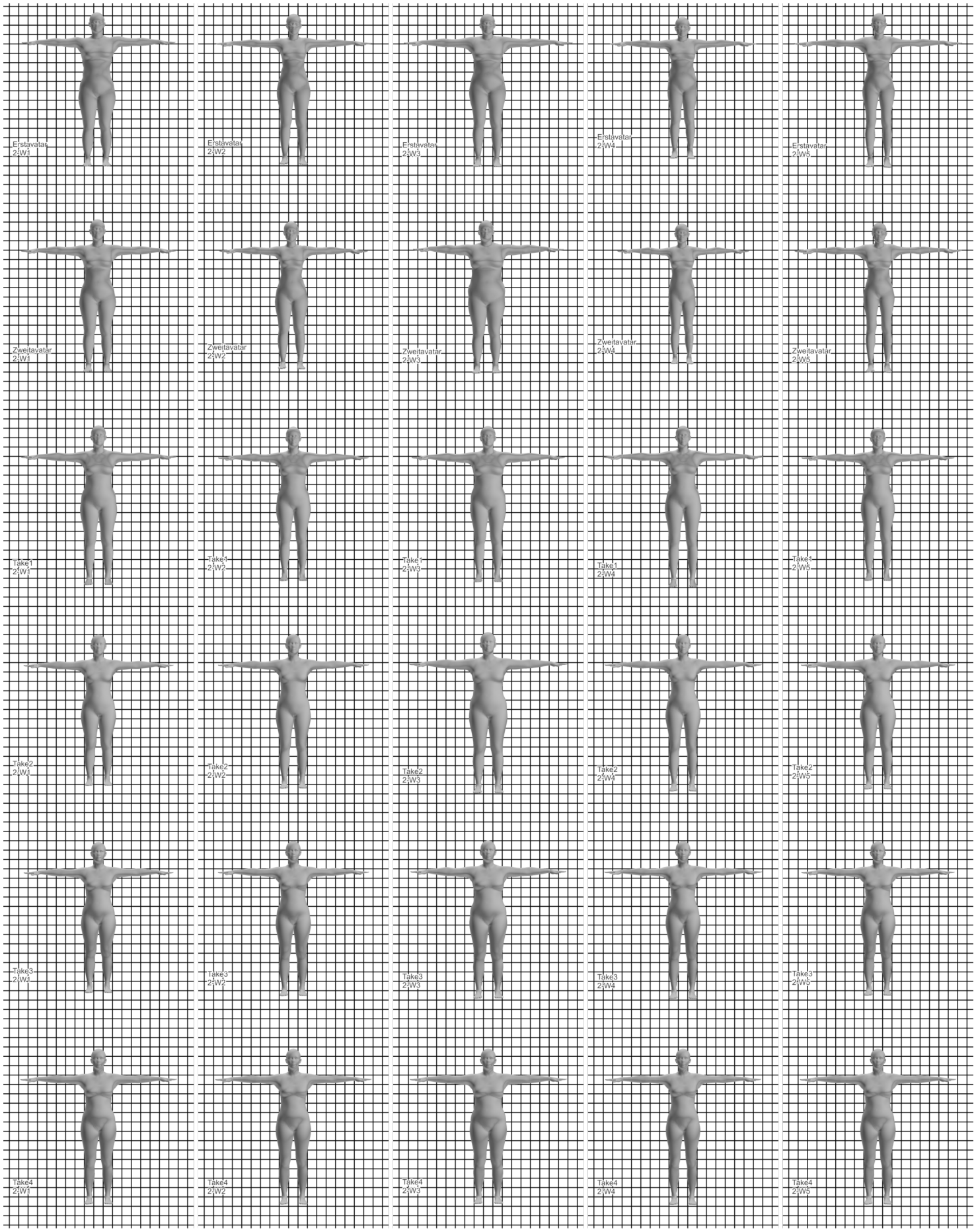
Anhang 11


Take4 2-W1
Take4 2-W2
Take4 2-W3
Take4 2-W4
Take4 2-W5
Take3 2-W1
Take3 2-W2
Take3 2-W3
Take3 2-W4
Take3 2-W5
Take2 2-W1
Take2 2-W2
Take2 2-W3
Take2 2-W4
Take2 2-W5
Take1 2-W1
Take1 2-W2
Take1 2-W3
Take1 2-W4
Take1 2-W5
Zweitavatar 2-W1
Zweitavatar 2-W2
Zweitavatar 2-W3
Zweitavatar 2-W4
Zweitavatar 2-W5
Erstavatar 2-W1
Erstavatar 2-W2
Erstavatar 2-W3
Erstavatar 2-W4
Erstavatar 2-W5

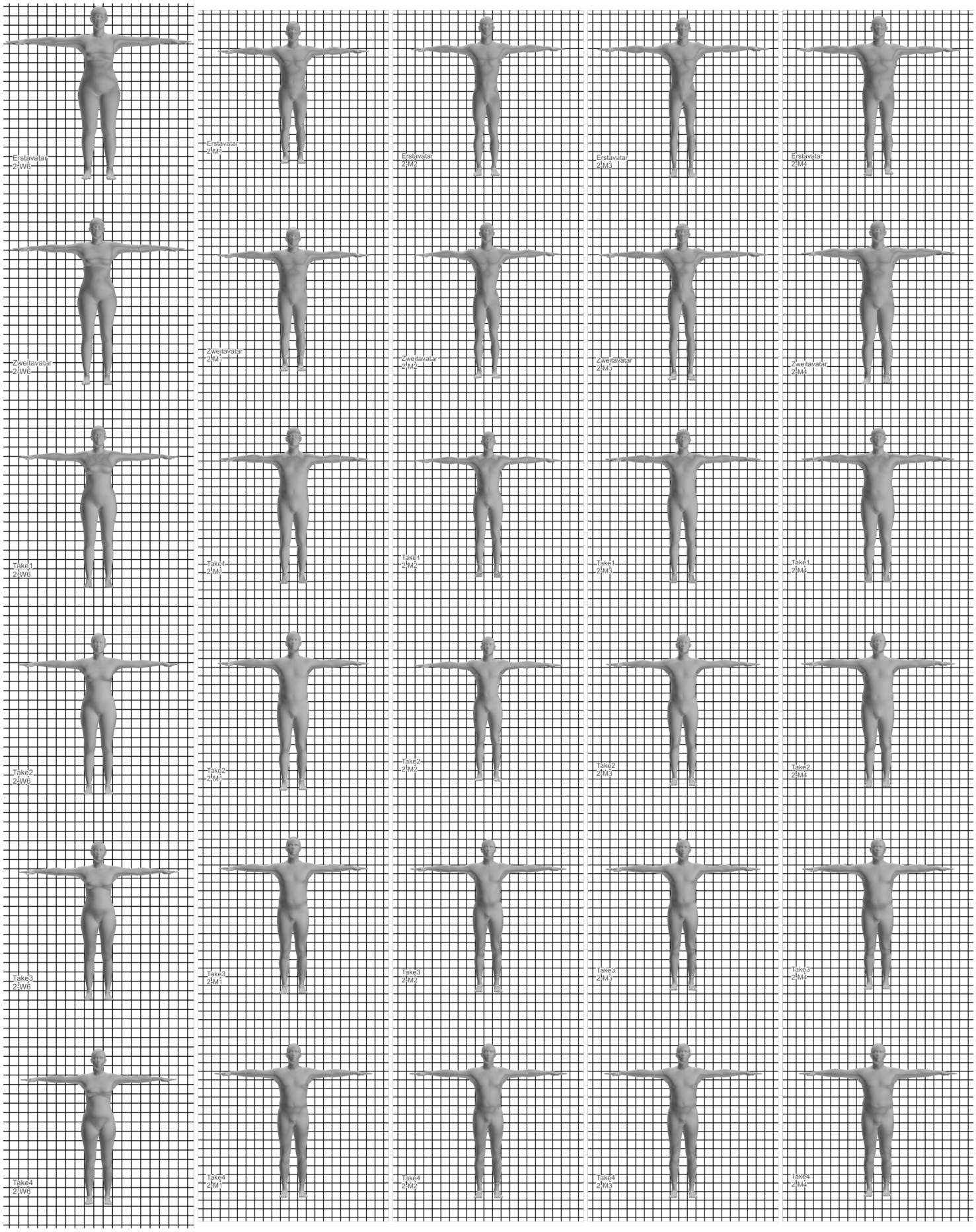
Anhang 12


Take4
2-W6
Take3
2-W6
Take2
2-W6
Take1
2-W6
Zweitavatar
2-W6
Erstavatar
2-W6
Take4
2-M1
Take3
2-M1
Take2
2-M1
Take1
2-M1
Zweitavatar
2-M1
Erstavatar
2-M1
Take4
2-M2
Take3
2-M2
Take2
2-M2
Take1
2-M2
Zweitavatar
2-M2
Erstavatar
2-M2
Take4
2-M3
Take3
2-M3
Take2
2-M3
Take1
2-M3
Zweitavatar
2-M3
Erstavatar
2-M3
Take4
2-M4
Take3
2-M4
Take2
2-M4
Take1
2-M4
Zweitavatar
2-M4
Erstavatar
2-M4

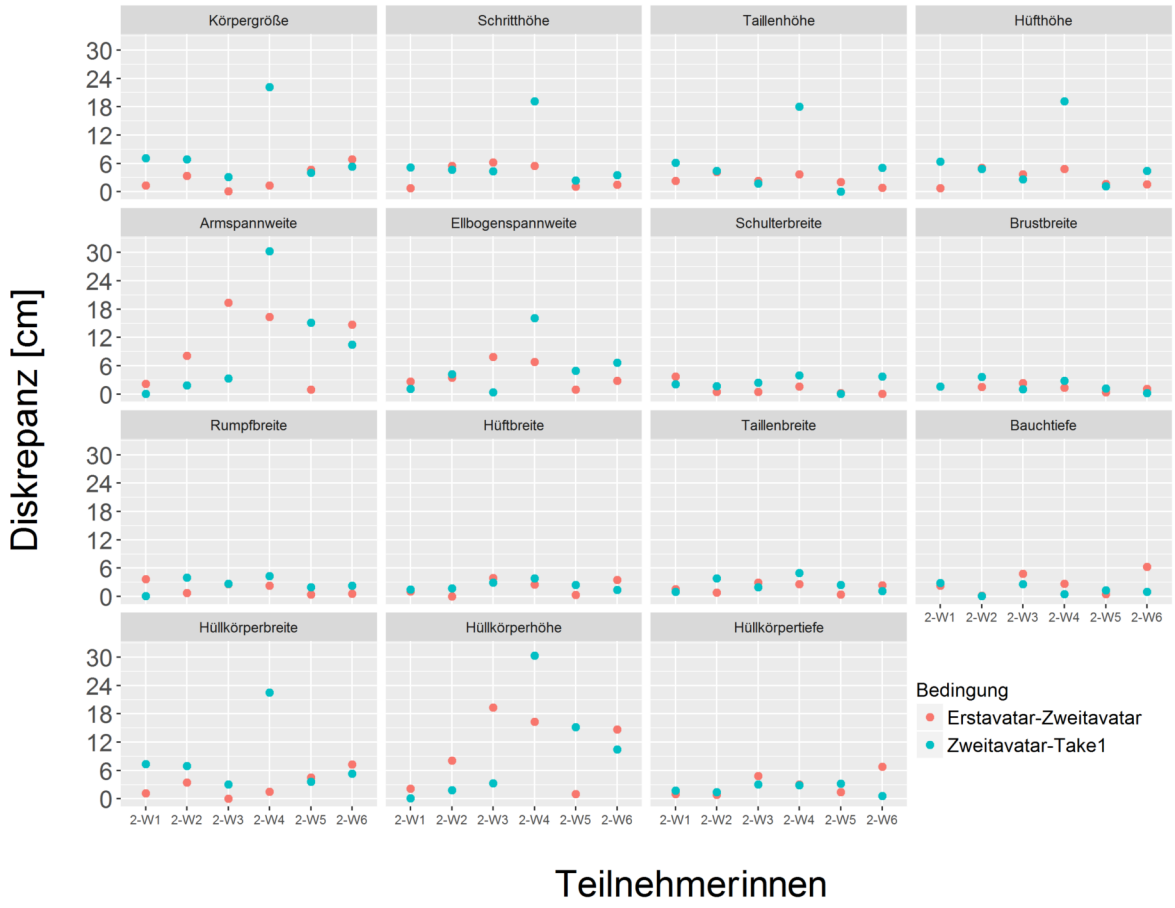
Anhang 13

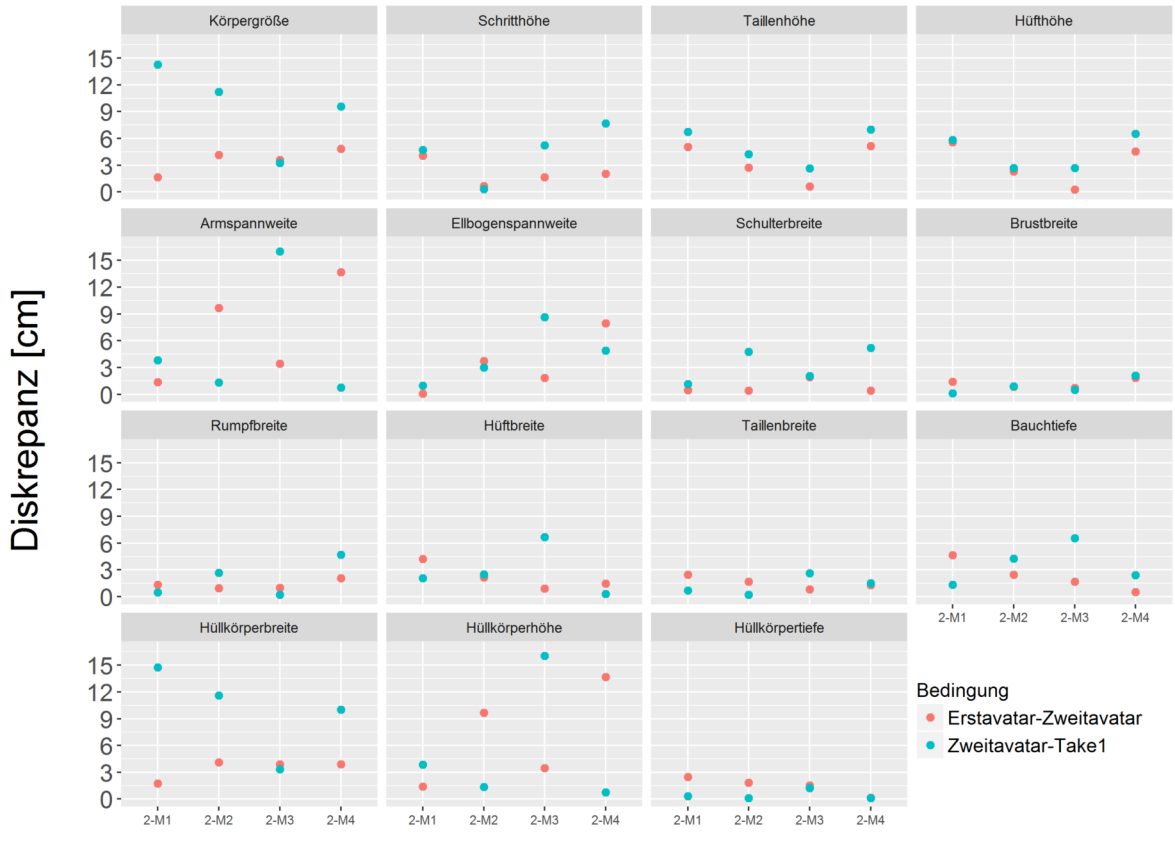

Erstavatar 2W1
Erstavatar 2W2
Erstavatar 2W3
Erstavatar 2W4
Erstavatar 2W5
Zweitavatar 2W1
Zweitavatar 2W2
Zweitavatar 2W3
Zweitavatar 2W4
Zweitavatar 2W5
Take1 2W1
Take1 2W2
Take1 2W3
Take1 2W4
Take1 2W5
Take2 2W1
Take2 2W2
Take2 2W3
Take2 2W4
Take2 2W5
Take3 2W1
Take3 2W2
Take3 2W3
Take3 2W4
Take3 2W5
Take4 2W1
Take4 2W2
Take4 2W3
Take4 2W4
Take4 2W5

Anhang 14


Erstavatar
2¦W6
Erstavatar
2¦M1
Erstavatar
2¦M2
Erstavatar
2¦M3
Erstavatar
2¦M4
Zweitavatar
2¦W6
Zweitavatar
2¦M1
Zweitavatar
2¦M2
Zweitavatar
2¦M3
Zweitavatar
2¦M4
Take1
2¦W6
Take1
2¦M1
Take1
2¦M2
Take1
2¦M3
Take1
2¦M4
Take2
2¦W6
Take2
2¦M1
Take2
2¦M2
Take2
2¦M3
Take2
2¦M4
Take3
2¦W6
Take3
2¦M1
Take3
2¦M2
Take3
2¦M3
Take3
2¦M4
Take4
2¦W6
Take4
2¦M1
Take4
2¦M2
Take4
2¦M3
Take4
2¦M4

Anhang 15


Körpergröße
Schritthöhe
Taillenhöhe
Hüfthöhe
Armspannweite
Ellbogenspannweite
Schulterbreite
Brustbreite
Rumpfbreite
Hüftbreite
Taillenbreite
Bauchtiefe
Hüllkörperbreite
Hüllkörperhöhe
Hüllkörpertiefe
Diskrepanz [cm]
0
6
12
18
24
30
2-W1
2-W2
2-W3
2-W4
2-W5
2-W6
Teilnehmerinnen
Bedingung
Erstavatar-Zweitavatar
Zweitavatar-Take1


Körpergröße
Schritthöhe
Taillenhöhe
Hüfthöhe
Armspannweite
Ellbogenspannweite
Schulterbreite
Brustbreite
Rumpfbreite
Hüftbreite
Taillenbreite
Bauchtiefe
Hüllkörperbreite
Hüllkörperhöhe
Hüllkörpertiefe
Diskrepanz [cm]
0
3
6
9
12
15
2-M1
2-M2
2-M3
2-M4
Teilnehmer
Bedingung
Erstavatar-Zweitavatar
Zweitavatar-Take1