Fingertracking durch neuronale Netze anhand reduzierter Markersets und Motion-Capture-Daten

Tobias Picker, Bastian Dewitz, Christian Geiger

Hochschule Düsseldorf
Münsterstraße 156
40476 Düsseldorf
tobias.picker@study.hs-duesseldorf.de, {bastian.dewitz, geiger}@hs-duesseldorf.de

Zusammenfassung: In diesem Paper wird ein neuartiger Ansatz eines Fingertracking-Systems auf Basis neuronaler Netze vorgestellt. Dazu werden mit Hilfe eines externen, optischen Trackingsystems kleine Marker (Ø 4 mm) an den Fingerspitzen erkannt, womit zwei neuronale Netze die Positionen aller Fingergelenke bestimmen. Durch die hohe Genauigkeit und den geringen Rechenbedarf ist diese Vorgehensweise für den Echtzeiteinsatz, wie z.B. VR-Anwendungen, geeignet. Mit dieser Methode wird eine durchschnittliche Abweichung von 1,61 Millimetern pro Gelenk bei hoher Bewegungsfreiheit erreicht.

Stichworte: Fingertracking, Handtracking, Neuronale Netze, Tensorflow, Motion Capturing, OptiTrack, Virtual Reality

1 Einleitung

Interaktion in der virtuellen Realität wird, insbesondere im Consumer-Bereich, durch verschiedenartige Eingabegeräte durchgeführt. Auch wenn der Nutzer so das Gefühl hat, das Geschehen zu beeinflussen, wird die Natürlichkeit der Interaktion reduziert, da zuerst erlernt werden muss, welche Funktion hinter jedem Button steckt.

Um diese Nachteile zu vermeiden wird aktuell verstärkt an Datenhandschuhen geforscht, die durch elektrische Bauteile wie Biegesensoren oder IMUs die Krümmung der Finger erkennen. Dabei wird zumeist ein normaler Handschuh an jedem Finger mit solchen Sensoren ausgestattet, sodass mit Hilfe einer inversen Kinematik die Ausrichtung aller Fingergelenke angenähert werden kann. Dies ermöglicht dem Nutzer eine natürlichere Interaktion, bringt hingegen aber auch eine Einschränkung der Bewegungsfreiheit mit sich. Auch das Ansprechen der haptischen Sinne ist durch den umschließenden Handschuh erschwert. Durch die Annäherung an ein normiertes Handmodell kann es hier auch zu Abweichungen zwischen realer Hand und dem virtuellen Abbild kommen.

In diesem Paper wird ein neuartiger Fingertracking-Prototyp vorgestellt, der als natürliches und uneingeschränktes Eingabegerät für Computersysteme, beispielsweise in VR-Anwendungen, eingesetzt werden kann. Dabei steht neben der Präzision auch die freie Nutzung der Haptik im Vordergrund. Um dies zu erreichen, wird ein externes optisches Trackingsystem verwendet, welches an den Fingerspitzen angebrachte retroreflektierende Marker

nutzt, um ein virtuelles Modell der Hand zu steuern. Dafür werden zwei neuronale Netze verwendet, welche durch das Training mit Motion-Capture-Daten von realen Handbewegungen menschliche und realitätsnahe Bewegungen produzieren sollen. Dabei besteht die Herausforderung in der von der Handform und -größe unabhängigen Bewegungserzeugung, sodass eine glaubhafte virtuelle Darstellung erzeugt werden kann.

2 Aktueller Forschungsstand

Echtes Fingertracking in der virtuellen Realität wird bei der aktuellen Generation von kommerziellen VR-Brillen nicht eingesetzt. Stattdessen werden zum Beispiel bei der HTC Vive nur die Controller-Bewegungen erkannt um eine grundlegende Interaktion zu ermöglichen. Hingegen bietet die Oculus Rift erste Ansätze eines Fingertrackings, indem mit Hilfe von kapazitiven Buttons im Controller die virtuellen Finger gesteuert werden können [Mat16]. Dabei beschränkt sich allerdings die Fingerbewegung auf drei Fingergruppen (Daumen, Zeigefinger, Mittel- & Ring- & kleiner Finger) und drei diskrete Fingerpositionen (Drücken, Aufliegen, Abspreizen) welche an den vorgesehenen Stellen über den Knöpfen platziert werden. HTC zeigt mit ersten Prototypen des *Knuckles* Controllers ein für jeden Finger individuelles Tracking durch Näherungssensoren, welches eine Fingerbewegung durch deutlich mehr diskrete Positionen erlaubt [Kid18]. Aber auch hier handelt es sich nur um ein simuliertes Fingertracking, welches mit einem an der Hand befestigten Controller durchgeführt wird.

Zusätzliche kommerzielle Produkte setzen zumeist auf Handschuh-basierte Ansätze, welche zum Beispiel durch Biegesensoren die Krümmung der einzelnen Finger bestimmen und mit dem Tracking-System des HMDs eine absolute Positionierung ermöglichen [She+16]. Auch die Nutzung von Kamera-basierten System ist äußerst beliebt. Hier wird mit einer Infrarot-, Tiefen- oder RGB-Kamera die Hand gefilmt und durch Bildverarbeitung in ein virtuelles Modell umgewandelt [HSC14][Tog+18][TH16].

Weitere experimentelle Fingertracking-Systeme setzen zum Beispiel auf akustische Systeme [Zha+17] oder nutzen kodiertes Licht um die Ausrichtung der Finger zu erfassen [Ma+17]. Auch eine Ergänzung durch haptisches Feedback mit Mckibben Muskeln anstatt Vibrationsmotoren ist hier möglich [Jad+17].

3 Systemaufbau

Die Positionserfassung der Finger wird im Live-Betrieb mit fünf Markern und einem RigidBody [1], entsprechend Abb. 2 (oben), durchgeführt. Durch ein externes, optisches Trackingsystem (OptiTrack mit 10 Prime 17W Kameras) werden die absoluten Positionen der Fingerspitzen, sowie die Position und Rotation des RigidBody am Handrücken berechnet und an

[1] Zusammenschluss aus mehreren retroreflektierenden Markern, mit denen zusätzlich zur Position auch die Rotation erfasst werden kann.

die Game-Engine (Unity 2017.2) weiter gegeben. Hier werden die Daten in Kugelkoordinaten relativ zum RigidBody umgewandelt und normiert, sodass eine allgemeine Betrachtung unabhängig der Handform bzw. -größe möglich ist. Ein erstes neuronales Netz ordnet die Koordinaten den einzelnen Fingern zu, sodass in der weiteren Datenverarbeitung eine gleichbleibende Reihenfolge der Gelenke garantiert ist. Die sortierten Markerpositionen werden komponentenweise in ein weiteres neuronales Netz mit 15 Eingabeneuronen (3 Komponenten pro Marker) gegeben, welches durch mehrere Berechnungsschichten am Ende 45 Werte (x, y, z Komponenten der Gelenke) ausgibt. Diese werden dann zu neuen Positionsdaten zusammengefasst, welche die nicht getrackten 15 Fingergelenke widerspiegeln. Die daraus resultierenden Daten steuern in Kombination mit den Trackingdaten ein abstraktes, virtuelles Handmodell, bei dem die Gelenke als Kugeln dargestellt und wie in Abbildung 5 mit Zylindern verbunden werden. Der zugehörige Workflow ist in Abbildung 1 dargestellt.

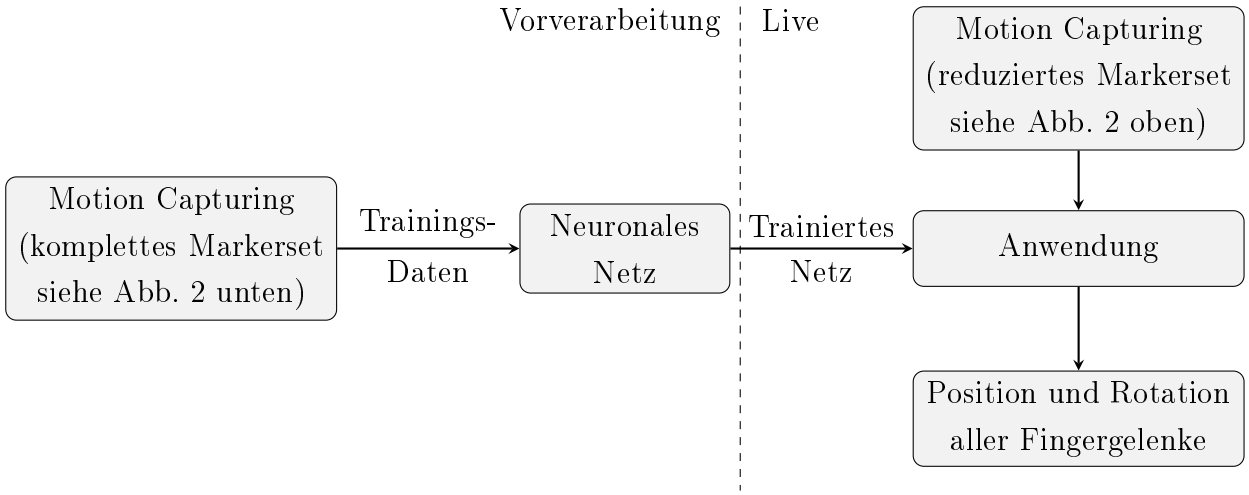

Abbildung 1: Workflow des Fingertrackings

3.1 Training des Finger-zuweisenden neuronalen Netzes

Mit Hilfe von Supervised Learning und Tensorflow wird ein neuronales Netz darauf trainiert zur Laufzeit die Positionsdaten der Marker den jeweiligen Fingern zuzuordnen, was dem darauf folgendem neuronalen Netz eine gleichbleibende Reihenfolge der Marker garantiert. Dafür werden im Training die Markerpositionen komponentenweise (x, y, z) als Input genutzt (15 Eingabeneuronen), um in den fünf Ausgabeneuronen die Reihenfolge auszugeben. Dazu entspricht z.B. der Daumen einem Ausgabewert von 0,1, der Zeigefinger 0,3 und so weiter. Dies ermöglicht im Live-Betrieb eine Feststellung der Sicherheit der Zuordnung, indem die Differenz der Ausgabe zu den Normwerten betrachtet wird. So können ggf. weitere Maßnahmen ergriffen werden, um eine korrekte Zuweisung zu garantieren.

3.2 Training des bewegungsschätzenden neuronalen Netzes

Ein weiteres neuronales Netz wird darauf trainiert, Motion-Capture-Daten aus einem reduzierten Markerset zu rekonstruieren. Dazu werden mit 20 Markern und einem RigidBody, die

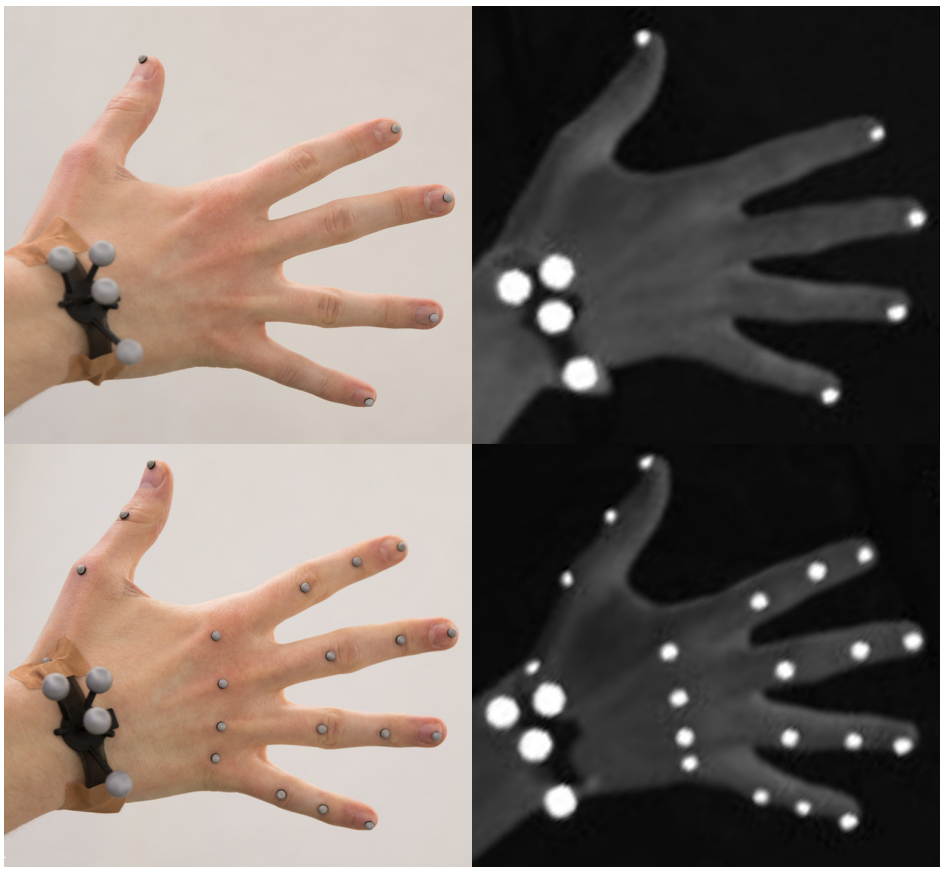

Abbildung 2: Markerpositionierung für Live- (oben) und Trainings-Betrieb (unten)

entsprechend Abb. 2 (unten) an definierten Gelenkpositionen platziert werden, unterschiedlichste Fingerbewegungen aufgezeichnet und in Trainingsdaten umgewandelt. Im Gegensatz zum Live-Betrieb werden hier die Marker über eine Kombination aus Nearest-Neighbor-Suche und Bewegungsvorhersage den jeweiligen Fingern zugewisen, was eine erhöhte Genauigkeit auf Kosten der Performance ermöglicht. Diese Algorithmen sind aufgrund ihres hohen benötigten Rechenbedarfs für den Echtzeit-Einsatz nicht geeignet.

4 Ergebnisse

Zur Analyse der künstlichen neuronalen Netze wurde mit Hilfe der Fehlerfunktion aus Gleichung 1 die Genauigkeit der Bewegungsschätzung bestimmt. Dabei wird der mittlere absolute Fehler betrachtet, der sich aus der durchschnittlichen Summe der Abweichungen aller Test-Datensätze n zusammensetzt. Die Abweichung berechnet sich aus der Distanz aller berechneten Markerpositionen Y_i zur zugehörigen realen Position $\hat{Y}_i$. Im Vergleich zu anderen Fehlerfunktionen wie dem MSE kann so die Gesamtabweichung eines Frames als Längenmaß angegeben werden.

$$E = \frac{1}{n} \sum^{n} \sum_{i=1}^{15} |\hat{Y}_i - Y_i| \tag{1}$$

Da bei neuronalen Netzen keine allgemeinen Aussagen über die Kombination der Parameter bezogen auf das resultierende Ergebnis möglich sind, werden im Folgenden unterschiedliche Konfigurationen getestet und miteinander verglichen.

Für das Training des bewegungsschätzenden neuronalen Netzes wurde die TensorFlow-GPU Low Level API genutzt, welche auf einem Desktop-PC mit einer Intel Xeon E3-1230v2 (4x 3.30 GHz) CPU und einer GTX 1070 GPU lief. Die folgenden Ergebnisse zeigen unterschiedlich konfigurierte neuronale Netze, welche jeweils für 10 Minuten mit 14.400 Datensätzen (entspricht 60 Sekunden Motion Capturing), bestehend aus unterschiedlichsten Fingerbewegungen einer männlichen Person, trainiert und mit weiteren 3.600 getestet wurden, wobei nach jeder Epoche der jeweilige Datensatz gemischt wurde. Dazu besteht die Basis aus 15 Eingabe-, zwei mal 64 versteckte und 45 Ausgabeneuronen, die mit einer Sigmoid-Funktion aktiviert werden. Das Training führt ein Adam-Optimizer mit einem Full-Batch-Learning durch. Um die richtigen Parameter für das endgültige neuronale Netz zu finden, werden diese Einstellungen einzeln ausgetauscht und dessen Genauigkeit miteinander verglichen.

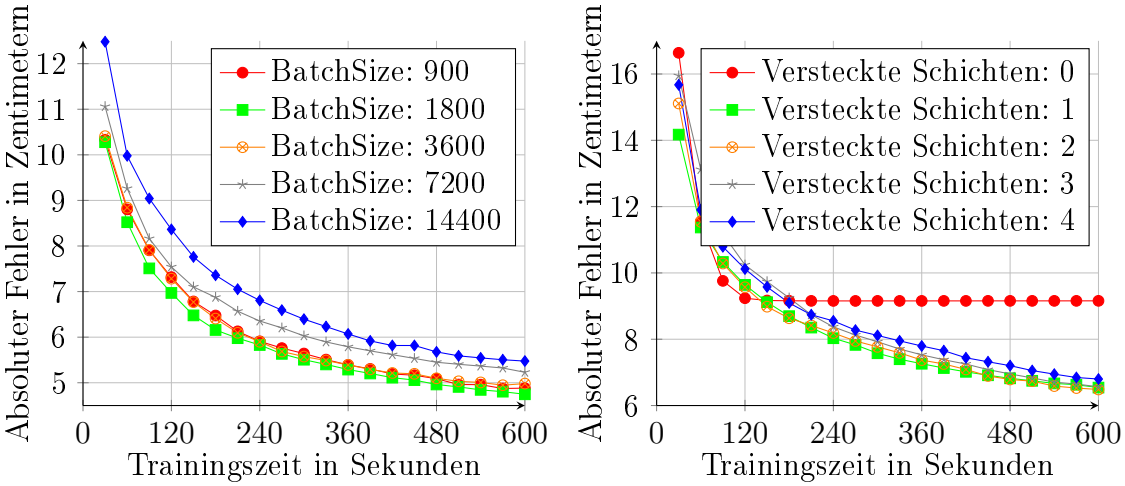

Abbildung 3: Vergleich Batch Sizes (links) & Anzahl Schichten (64 Neuronen) (rechts)

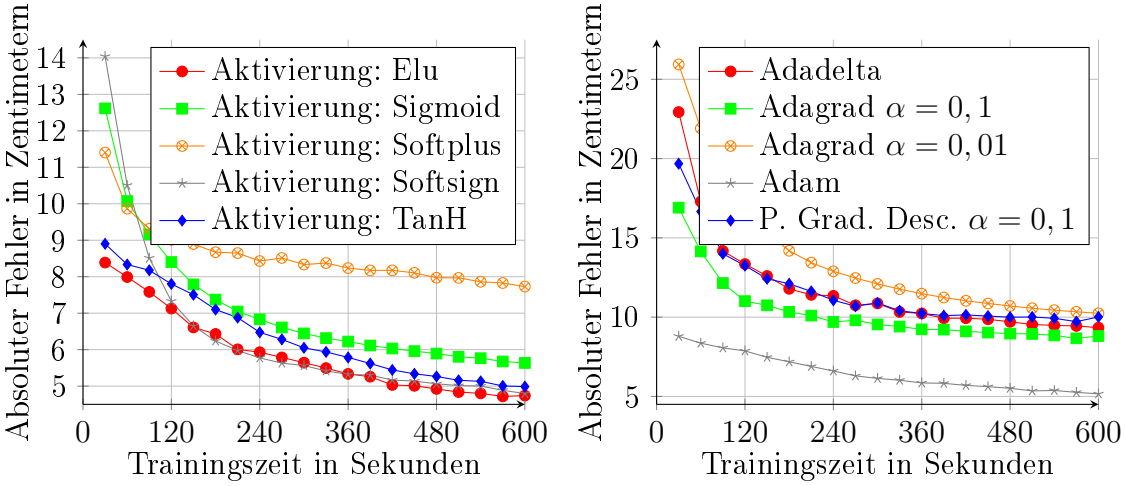

Abbildung 4: Vergleich Aktivierungsfunktionen (links) & Trainingsalgorithmen (rechts)

In Grafik 3 (links) wird das Training mit unterschiedlich großen Batch Sizes gegenüber gestellt. Dabei liefert das Full Batch Learning mit einem Fehler von circa 5,48 cm das schlechteste Ergebnis. Das beste Resultat wurde mit einer Batch Size von 1800, mit einer Abweichung von etwa 4,75 cm, erreicht.

In Abbildung 3 (rechts) sind verschiedene Netzwerktopologien zu sehen, welche sich durch die Anzahl der versteckten Schichten mit jeweils 64 Neuronen unterscheiden. Besonders auffällig ist der deutlich flachere Fehlerverlauf bei keiner versteckten Schicht, der sich mit dem schlechtesten Ergebnis von circa 8,15 cm auswirkt. Hingegen bieten eine, zwei und drei Schichten eine Genauigkeit von etwa 5,50 cm.

Die Nutzung anderer Aktivierungsfunktionen wird in Grafik 4 (links) dargestellt, wobei die Softplus Funktion hier besonders negativ mit einem Fehler von circa 7,73 cm hervor sticht. Das beste Ergebnis erreicht die Elu Funktion, welche in einer Abweichung von etwa 4,74 cm resultiert. Die Relu Funktion wird in diesem Graph nicht dargestellt, da sie beim Training nicht konvergierte.

Der Vergleich verschiedener Trainingsalgorithmen wird in Abbildung 4 (rechts) gezeigt. Am besten schneidet der Adam Optimizer mit einer Abweichung von circa 5,16 cm ab. Das schlechteste Ergebnis wird mit dem Adagrad-Optimizer und einer Lernrate von 0,01 erreicht, der nach zehn Minuten Training eine Abweichung von 10,24 cm besitzt. Der klassische Gradient Descent Algorithmus wird hier nicht mit aufgeführt, da dieser deutlich langsamer konvergiert und nach zehn Minuten Training einen Fehlerwert von ungefähr 30 cm besitzt.

Batch Size	**Versteckte Schichten**	**Aktivierungs-Funktion**	**Trainings-Algorithmus**	**Fehler (in cm)**
900	2	Elu	Adam	3,075
900	2	Elu	Adagrad	3,841
900	2	Softsign	Adam	3,087
900	2	Softsign	Adagrad	5,077
900	3	Elu	Adam	2,515
900	3	Elu	Adagrad	3,307
900	3	Softsign	Adam	2,548
900	3	Softsign	Adagrad	3,993
1800	2	Elu	Adam	3,087
1800	2	Elu	Adagrad	3,974
1800	2	Softsign	Adam	3,117
1800	2	Softsign	Adagrad	5,193
1800	3	Elu	Adam	2,414
1800	3	Elu	Adagrad	3,422
1800	3	Softsign	Adam	2,431
1800	3	Softsign	Adagrad	4,267

Tabelle 1: Trainingsergebnisse nach 60 Minuten

Aus diesen Erkenntnissen wurden die jeweils besten Parameter miteinander kombiniert und für eine Stunde trainiert. Die Ergebnisse sind in Tabelle 1 zusammen getragen, wobei eine geringste Abweichung von circa 2,41 Zentimetern erreicht wird. Der durchschnittliche Fehler von etwa 1,61 Millimetern pro Fingergelenk entsteht bei einem neuronalen Netz mit drei versteckten Schichten bestehend aus jeweils 64 Neuronen, welches mit der Elu-Funktion aktiviert und durch den Adam-Optimizer mit einer Batch Size von 1800 Datensätzen trainiert wurde.

Bei dem Finger-zuweisenden neuronalen Netz werden drei versteckte Schichten mit jeweils 64 Neuronen und einer Softsign-Aktivierungsfunktion genutzt. Weiterhin wird der Adam-Optimizer mit einer Batch Size von 900 verwendet. Mit insgesamt 17674 korrekten Zuweisungen von 18000 Test-Daten (fünf pro Datensatz) erreicht dieses System eine Genauigkeit von etwa 98,19 Prozent.

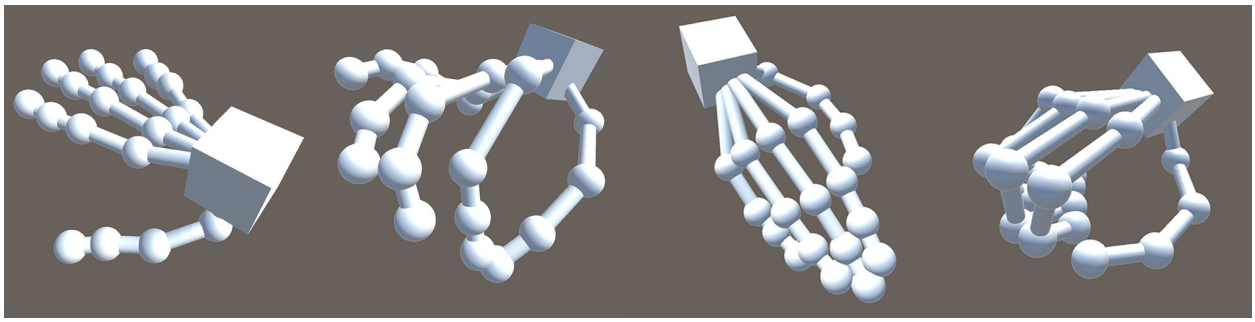

Abbildung 5: Ergebnisse mit abstraktem Handmodell

5 Diskussion

Für die Auswertung der neuronalen Netze wurden die geschätzten Positionen der Fingergelenke anhand eines Testdatensatzes mit den getrackten Positionen verglichen. Durch eine Fehlerfunktion wurde die mittlere Abweichung der Distanzen zwischen den beiden Positionen eines Datensatzes berechnet, um einen Vergleich unterschiedlicher neuronaler Netze zu ermöglichen. Dadurch, dass der Fehler der Abweichung in Metern entspricht, kann auch eine generelle Abschätzung zur Genauigkeit getätigt werden. Die getesteten Datensätze wurden zufällig aus einer Sammlung von Motion-Capture-Daten, welche unterschiedlichste Handposen beinhalten, gewählt.

Da noch keine auf den qualitativen Nutzereindruck bezogene Evaluation des Systems stattgefunden hat, ist die durchschnittliche Abweichung von 1,61 Millimetern pro Fingergelenk das einzige gemessene Qualitätskriterium. Erste Tests mit unterschiedlich großen Händen führten zu einer glaubhaften virtuellen Darstellung der Hand, bei der die Ungenauigkeit nicht spürbar aufgefallen ist. Auch eine Einschränkung durch die Marker oder den RigidBody wurden nach kurzer Eingewöhnungszeit nicht bemängelt.

Die Nutzung neuronaler Netze ermöglicht ein System mit geringem Leistungs- und Speicherbedarf, welches eine große Datenbank von Motion-Capture-Daten zu einem kleinen und effizienten Modell vereinfacht. Dieses erzeugt auch bei teilweise fehlerbehafteten Daten sinnvolle Ergebnisse, was zu einer generell geringen Fehleranfälligkeit führt. Dies ist gerade bei einem optischen Tracking-System von großem Vorteil, weil ein durchgängiges perfektes Tracking aufgrund von den räumlichen Gegebenheiten nicht garantiert werden kann. Auch die Verdeckung von Markern durch andere Finger ist hier ein Problem, was so teilweise umgangen werden kann. Allerdings bieten neuronale Netze kein vorhersehbares Verhalten in Extremfällen, also von den Trainingsdaten deutlich verschiedene Live-Daten, weshalb diese im Voraus getestet und ggf. verhindert werden müssen. Zusätzlich zum Training bei dem viele unterschiedliche Konfigurationen ausprobiert werden, entsteht ein hoher Zeitbedarf, der allerdings hauptsächlich aus Rechenzeit besteht.

Aufgrund des hohen technischen Aufwands und den damit verbundenen Kosten, ist ein Praxiseinsatz des Systems im Consumerbereich vorerst nicht möglich, sodass sich die Anwendbarkeit des Ansatzes auf den Forschungsbereich beschränkt. Um eine stabile VR-Erfahrung zu gewährleisten muss das Tracking-System sehr genau auf den Raum kalibriert werden, was einen mobilen Einsatz eher undenkbar macht. Auch der benötigte Platz von zusätzlich circa einem Meter pro Seite und die dadurch erforderliche Anzahl an Kameras muss beachtet werden.

Das vorgestellte System kann vor allem in haptischen VR-Anwendungen, z.B. bei der Interaktion mit virtuellen Replikas, sinnvoll eingesetzt werden. Generell sind Forschungsprojekte und konzeptionelle Entwicklungen mit Bezug zu Fingertracking bzw. Interaktion mit Händen in VR ein potenzielles Anwendungsfeld, bis entsprechende Systeme im Consumer-Bereich verfügbar sind.

6 Fazit & Ausblick

Der vorgestellte Prototyp verspricht großes Potenzial als präzises Fingertracking-System. Durch die Nutzung neuronaler Netze, welche durch Motion-Capture-Daten trainiert wurden, wird ein virtuelles Handmodell erzeugt, das Dank dem geringen Leistungsverbrauch ideal in Echtzeitanwendungen genutzt werden kann. Eine geringe Abweichung von 1,61 Millimetern pro Fingergelenk ermöglicht ein sehr genaues Abbild der realen Hand, welches sich an jede Handform und -größe anpasst. Zusätzlich wird durch das optische Trackingsystem eine volle Bewegungsfreiheit garantiert.

In Zukunft soll dieser Prototyp weiterhin in der Genauigkeit und Stabilität verbessert werden. Dazu wird der Einsatz von rekurrenten neuronalen Netzen angestrebt um eine genauere Analyse des Bewegungsmusters zu ermöglichen. Auch das Sammeln von deutlich mehr Motion-Capture-Daten von unterschiedlichen Personen ist hier ein wichtiger Schritt um die Präzision auch in schwierigen Situationen zu garantieren. Für die Verbesserung der Stabilität müssen dem Trainingsdatensatz weitere fehlerbehaftete Daten hinzugefügt werden, sodass die Fehleranfälligkeit des neuronalen Netzes sinkt. Auch eine andere Anordnung des

reduzierten Markersets könnte der Problematik der Verdeckung entgegen kommen, damit eine durchgängige realistische Virtual Reality Erfahrung ermöglicht wird. Weiterhin sollte in einer Evaluation getestet werden, welche Wirkung bei Nutzern, zum Beispiel im Bezug auf Embodiment und Presence, erzielt wird. Auch ein Vergleich von der aktuellen abstrakten Hand zu einer angepassten, realistisch aussehenden wäre sinnvoll, um den Einfluss des 3D-Modells zu messen.

Literatur

[HSC14] M. H. Hsu, T. K. Shih, and J. S. Chiang. "Real-Time Finger Tracking for Virtual Instruments". In: *2014 7th International Conference on Ubi-Media Computing and Workshops*. July 2014, pp. 133–138. DOI: 10.1109/U-MEDIA.2014.53.

[Jad+17] Saurabh Jadhav et al. "Soft robotic glove for kinesthetic haptic feedback in virtual reality environments". In: *Electronic Imaging* 2017.3 (2017), pp. 19–24. DOI: 10.2352/ISSN.2470-1173.2017.3.ERVR-102.

[Kid18] Essa Kidwell. *Valve 'Knuckles' controllers — Everything you need to know!* Jan. 20, 2018. URL: https://www.windowscentral.com/valve-knuckles-controllers (visited on 06/28/2018).

[Ma+17] S. Ma et al. "Lift: Using projected coded light for finger tracking and device augmentation". In: *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. Mar. 2017, pp. 153–159. DOI: 10.1109/PERCOM.2017.7917861.

[Mat16] Lucas Matney. *Review: Oculus Touch VR motion controllers*. Dec. 5, 2016. URL: https://techcrunch.com/2016/12/05/review-oculus-touch-motion-controllers (visited on 06/28/2018).

[She+16] Zhong Shen et al. "A soft stretchable bending sensor and data glove applications". In: *Robotics and Biomimetics* 3.1 (Dec. 2016), p. 22. ISSN: 2197-3768. DOI: 10.1186/s40638-016-0051-1.

[TH16] M. Tomida and K. Hoshino. "Finger-motion capture device with few restriction of users' movement for control of humanoid robot". In: *2016 IEEE International Conference on Industrial Technology (ICIT)*. Mar. 2016, pp. 78–83. DOI: 10.1109/ICIT.2016.7474729.

[Tog+18] Enkhtogtokh Togootogtokh et al. "3D finger tracking and recognition image processing for real-time music playing with depth sensors". In: *Multimedia Tools and Applications* 77.8 (Apr. 2018), pp. 9233–9248. ISSN: 1573-7721. DOI: 10.1007/s11042-017-4784-9.

[Zha+17] Cheng Zhang et al. "SoundTrak: Continuous 3D Tracking of a Finger Using Active Acoustics". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1.2 (June 2017), 30:1–30:25. ISSN: 2474-9567. DOI: 10.1145/3090095.