Bachelor of Science

A gesture-based interface to VR

Vincent Roy Kühn

Abgabedatum: 25. September 2018

Goethe-Universität Frankfurt am Main
Prof. Dr. A. Mehler
Prof. Dr. V. Ramesh

Erklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen Quellen oder Hilfsmittel als die in dieser Arbeit angegebenen verwendet habe.

_____

Ort, Datum

_____

Unterschrift

Zusammenfassung

Die folgende Arbeit handelt von einem *Human Computer Interaction Interface*, welches es gestattet, mit Hilfe von Gesten zu schreiben. Das System ermöglicht seinen Nutzern, neue Gesten hinzuzufügen und zu verwenden. Da Gesten besser erkannt werden können, je genauer die Darstellung der Hände ist, wird diese durch Datenhandschuhe an den Computer übertragen. Die Hände werden einerseits in der *Virtual Reality* (VR) dargestellt, damit sie der Nutzer sieht. Andererseits werden die Daten, die die Gestenerkennung benötigt, an das Interface weitergeleitet. Die Erkennung der Gesten wird mit Hilfe eines *Neuronales Netz* (NN) implementiert. Dieses ist in der Lage, Gesten zu unterscheiden, sofern es genügend Trainingsdaten erhalten hat.

Die genutzten Gesten sind entweder einhändig oder beidhändig auszuführen. Die Aussagen der Gesten beziehen sich in dieser Arbeit vor allem auf relationale Operatoren, die Beziehungen zwischen Objekten ausdrücken, wie beispielsweise „gleich“ oder „größer gleich“.

Abschließend wird in dieser Arbeit ein System geschaffen, das es ermöglicht, mit Gesten Sätze auszudrücken. Dies betrifft das sogenannte *gestische Schreiben* nach Mehler, Lücking und Abrami 2014. Zu diesem Zweck befindet sich der Nutzer in einem virtuellen Raum mit Objekten, die er verknüpfen kann, wobei er Sätze in einem relationalen Kontext manifestiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Aufgabenstellung

Das Ziel dieser Arbeit soll ein System sein, das Gesten lernen und erkennen kann. Das ganze System steht im Kontext des gestenbasierten Schreiben, das heißt, Sätze werden durch Gesten dargestellt.

1.2. Motivation

Wenn heutzutage Gesten in der Virtuellen Realität eingesetzt werden, dann meistens um einen Charakter im Spiel zu steuern oder um Objekte zu manipulieren. Dabei hat sich gezeigt, dass sie eine Alternative zur klassischen Controllersteuerung bieten und eine noch realistischere Umsetzung ermöglichen. Aber heutige Technologien bieten viele Möglichkeiten über dieses Szenario hinaus. Neue Eingabemöglichkeiten sind ein großer Fortschritt im Bereich der *Human Computer Interaction*. Nachdem die Spracheingabe inzwischen standardmäßig in vielen Betriebssystemen enthalten ist, könnte dasselbe mit einer Steuerung durch Gesten geschehen. Doch das ist nicht der einzige Nutzen, den uns diese neue Möglichkeit der Interaktion bietet. Gesten sind ein essenzieller Bestandteil der menschlichen Interaktion und Kommunikation (McNeill (1992, vgl.), zitiert nach Badelt (2013)). Jeder Mensch führt sie nicht nur beim Sprechen mit anderen Menschen aus, sondern sie bieten auch eine Möglichkeit mit Menschen zu kommunizieren, ohne dabei auf die Sprache angewiesen zu sein. Dies kann helfen, wenn die Sprachbarriere zu groß ist.

Nachdem schon die wichtigste Komponente - die Sprache - in die Interaktion zwischen Mensch und Computer mittels Spracherkennung integriert wurde, soll dies nun auch mit der Gestik geschehen. Dafür muss als erstes ein „Gestenalphabet“ entwickelt werden, das eine Grundlage für weitere Gesten bildet. Dies kann je nach Anwendungsbereich angepasst und überarbeitet werden. Gesten können in vielen unterschiedlichen Bereichen des *Human Computer Interaction* genutzt werden. Daher muss man zwischen Gesten zur Steuerung und zum Schreiben (*Gestural Writing* **Mehler:Luecking:2012**; Mehler, Lücking und Abrami 2014)) unterscheiden. In dieser Arbeit werden Gesten, die dem *Gestural Writing* (2.1) dienen, umgesetzt. Diese Gesten bilden eine von fünf Kategorien, die in (Ekman und W. 1969) aufgestellt wurden. Hierbei kann natürlich nur ein Ausschnitt an möglichen Gesten betrachtet werden, da es viel zu viele Gesten gibt. Es soll daher in dieser Arbeit die Möglichkeit geschaffen werden, Gesten zur relationalen Ordnung benutzen zu können.

Nachdem man ein Alphabet an Gesten entwickelt hat, das umgesetzt werden soll, braucht man ein System, das diese möglichst genau erkennt. Dafür hat sich die Arbeit für ein *Neuronales Netz* (NN; 2.2) entschieden, da es sich einfacher erweitern lässt, wenn man neue Gesten

hinzufügen möchte. Ein Neuronales Netz hat den Vorteil, dass das Alphabet an genutzten Gesten schnell erweitert werden kann, ohne dass viel Programmiererfahrung notwendig ist. Da man bei Mehler, Abrami u. a. (2018) Gesten einzeln programmiert und konfiguriert hat, müsste man sich tiefergehend in das Thema einarbeiten und viele Tests durchführen, bis die Geste von vielen Nutzern gemacht werden kann. Diese Schritte sollen in dieser Arbeit wegfallen und es wird versucht, es dem Nutzer so einfach wie möglich zu machen, Gesten zu ändern oder das Alphabet zu erweitern.

2. Grundlagen

2.1. Gestenbasiertes Schreiben

„Gestural writing is defined as a sort of coding in which propositions are only expressed by means of gestures “ (Mehler, Lücking und Abrami 2014, S. 1). Man kann die Gesten dafür mit Wörtern vergleichen, die zusammengesetzt einen Satz bilden. Jede Geste hat bereits eine komplexere Bedeutung und dient nicht dazu, nur mit den Händen Buchstaben zu formen. Die Gebärdensprache kann man ebenfalls zum *Gestural Writing* zählen, da die einzelnen Gesten weitergehende Bedeutungen haben. Der Vorteil gegenüber dem händischen Formen von Buchstaben ist, dass wenige Gesten ausreichen, um eine Beziehung zwischen Objekten herzustellen. Da die Anzahl an möglichen Themengebieten innerhalb des *Gestural Writing* sehr groß ist, werden in dieser Arbeit, wie bereits erwähnt (1.2), hauptsächlich Gesten zur relationalen Ordnung umgesetzt.

2.2. Neuronale Netze

2.2.1. Biologischer Hintergrund

Jedes biologische Gehirn besitzt Neuronen. Diese werden auch Nervenzellen genannt und sind untereinander durch Synapsen verknüpft. Ein kleiner Stromimpuls dient als Reiz. Wenn dieser von einer vorgeschalteten Zelle eingeht, wird er in der betrachteten Zelle verarbeitet. Entweder reicht der Impuls aus und das Neuron leitet das Signal weiter oder er war zu schwach und der Reiz wird nicht weitergegeben. Dieses Entscheidungsverfahren bezeichnet man als Aktionspotenzial. Ein Neuron kann mehrere Nachbarzellen besitzen, die Reize senden, aber auch Reize empfangen. Wenn der Mensch lernt, werden die Verknüpfungen im Gehirn verstärkt und ein dichteres Netz an Neuronen entsteht. Lernt der Mensch nicht, werden die Verbindungen kaum genutzt und dementsprechend auch nicht weitergebildet (Risse 2018, vgl.).

2.2.2. Übersicht zu Neuronalen Netzen

Künstliche Neuronale Netze sind ein Teilbereich des *Machine Learning*. Sie beschreiben eine Klasse von Systemen, die das Gehirn und die Arbeitsweise möglichst genau als Algorithmus für den Computer darstellen und damit die Möglichkeit bieten, eigenständig zu lernen.

Es gibt verschiedene Arten von Neuronalen Netzen, die für unterschiedliche Anwendungen geeignet sind. Sie lassen sich in zwei grundlegende Gruppen einteilen. Die erste Gruppe beinhaltet Netze, die das menschliche Gehirn und dessen Arbeitsweise widerspiegeln sollen. Damit möchte man die biologischen Zusammenhänge des Gehirns und die Verknüpfungen

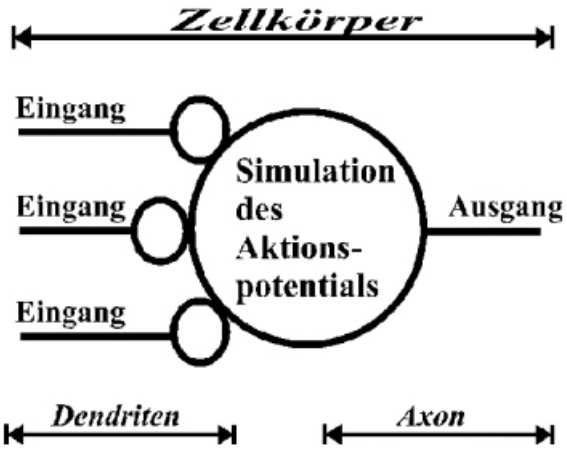


Abbildung 2.1.: Schematische Darstellung einer Nervenzelle (Risse 2018)

zwischen den Neuronen besser verstehen. Des weiteren besteht die Möglichkeit das Gehirn zu simulieren. Zur zweiten Gruppe gehören Netze, die nicht zum Verstehen des Gehirns beitragen müssen, sondern zum Lösen von Problemen genutzt werden. Hierbei ist wichtig, dass sich die Funktionsweisen eventuell von bekannten bzw. vermuteten Arbeitsweisen des Gehirns unterscheiden oder Verbesserungen aufweisen (Rey und Beck 2018, vgl.).
Es gibt vier Obergruppen an Netztypen (Rey und Beck 2018, vgl.):

- *Pattern Associator*
- *Rekurrente Netze*
- *Kompetitive Netze*
- *Kohonen-Netze*

Da das in dieser Arbeit genutzte Netz zur Klasse der Rekurrenten Netze gehört, wird im Folgenden vor allem diese Gruppe genauer betrachtet.

2.2.3. Aufbau Neuronaler Netze

Wie bereits ausgeführt (2.2.2), dient das Gehirn als Vorbild für Neuronale Netze. Sie bestehen ebenfalls aus einzelnen Neuronen, die miteinander verknüpft sind. Ein Neuron kann, wie das biologische Vorbild (siehe Abbildung 2.1), eins bis n Eingabeparameter und eins bis n Ausgaben ($n \epsilon \mathbb{N}$) besitzen (siehe Abbildung 2.2). Im Gegensatz zum Gehirn, nutzt ein Neuronales Netz jedoch keine Stromimpulse als Reiz oder als Eingabeparameter, sondern Zahlen. Jedes Neuron besitzt eine Aktivierungsfunktion und entscheidet, ob die Eingaben stark genug waren, damit ein weiterer Reiz an das nächste Neuron gesendet werden kann. Wie in Abbildung (2.2) zu sehen ist, werden alle Eingaben gewichtet und somit bereits gefiltert, bevor sie die Aktivierungsfunktion erreichen. Der Vorteil an der Gewichtung ist, dass potenziell schlechtere bzw. unwichtige Informationen ein geringes Gewicht erhalten. Eingänge mit guten und

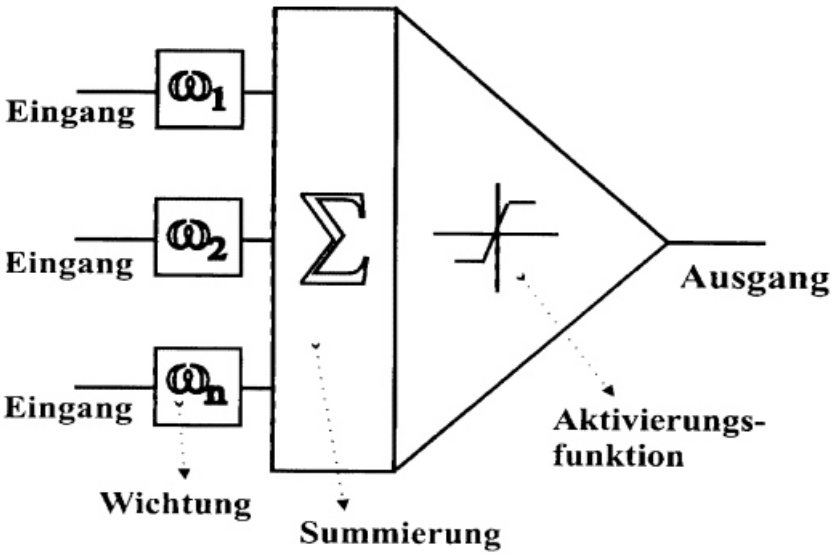


Abbildung 2.2.: Schematische Darstellung eines künstlichen Neurons (Risse 2018)

relevanten Informationen bekommen hingegen ein hohes Gewicht. Nur wenn genügend relevante Daten vorhanden sind, wird das Aktivierungslevel der Aktivierungsfunktion überschritten. Die Neuronen lassen sich verschiedenen Schichten zuordnen. Jedes Netz besteht aus einem Input Layer, das die Eingabewerte erhält und an die nächste Schicht weitergibt. Manchen Netztypen, wie ein *Recurrent Neural Network*, besitzen ein Hidden Layer, welches komplexere Verarbeitungen ermöglicht. Die letzte Schicht ist immer ein Output Layer, das die verarbeiteten Daten ausgibt, damit sie weiterverarbeitet werden können

2.2.4. Rekurrente Netze (RNN)

„Rekurrente Netze sind dadurch gekennzeichnet, dass Rückkopplungen von Neuronen einer Schicht zu [...] einer vorangegangenen Schicht existieren." (Rey und Beck 2018). Dadurch werden zusätzliche Verknüpfungen geschaffen und das einzelne Neuron erhält neben seinen normalen Inputs zusätzlich weiterverarbeitete Informationen und kann dementsprechend seine Entscheidung optimieren. Aus diesem Grund können die Netze vorherige Daten mit in die aktuelle Entscheidungen einbeziehen.

Je nach Kategorie gibt es verschiedene Verknüpfungsarten:

- direkte Verknüpfung
- indirekte Verknüpfungen
- seitliche Rückkopplungen
- vollständige Verbindungen

2.2.5. Long Short-term memory (LSTM)

Ein LSTM ist eine spezielle Art eines RNN. Wie der Name „Long Short-term“ bereits andeutet, liegt die Besonderheit darin, dass es ein Kurzzeitgedächtnis besitzt, das sich getroffene Entscheidungen über eine längere Periode merken kann. Analog zum RNN besitzt auch das *Long Short-term Memory* drei verschiedenen Arten von Schichten. Auf das Input Layer, das die Informationen erhält, folgt mindestens ein Hidden Layer. Der Aufbau eines Neurons im Hidden Layer wird in Abbildung (2.3) dargestellt. Ist die Verarbeitung im Hidden Layer abgeschlossen, wird die Information an das Output Layer weitergegeben, damit die Daten anschließend weiterverarbeitet werden können.

2.2.6. Aufbau LSTM

Wie bereits erläutert (2.2.5), besteht ein LSTM aus drei Schichten. Die Schicht, die hier genauer betrachtet werden soll, ist das Hidden Layer. Die Abbildung (2.3) verdeutlicht, dass ein einzelnes Neuron aus mehreren Komponenten besteht. Jedes Neuron bekommt einen Input, den es verarbeitet. Dieser kann entweder von einem Neuron eines anderen Hidden Layers oder aus dem Input Layer erzeugt werden, je nachdem ob sich das betrachtete Neuron am Anfang oder in der Mitte des Netzes befindet. Zusätzlich kann sich das LSTM Daten aus früheren Durchläufen merken und diese verwenden (2.2.5). Diese Fähigkeit wird ermöglicht durch die Implementierung des Forget-Gates. Der Output wird dort abgespeichert und steht im nächsten Schritt zusätzlich zu den aktuellen Daten zur Verfügung (Zhu u. a. 2016, vgl.). Diese Bedingungen für ein Neuron lassen sich in einem mathematischen Modell beschreiben, welches aus fünf Formeln besteht. Hierbei werden die einzelnen Komponenten eines Neurons (siehe Abbildung 2.3) mathematisch beschrieben. Diese sind das *Input Gate* (i_t), das *Forget Gate* (f_t), das *Neuron* (c_t), das *Output Gate* (o_t) und der *Output* (h_t), der an das nächste Neuron weitergegeben wird.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t\text{-}1} + W_{ci}c_{t\text{-}1} + b_i),$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t\text{-}1} + W_{cf}c_{t\text{-}1} + b_f),$$

$$c_t = f_t \odot c_{t\text{-}1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t\text{-}1} + b_c),$$

$$o_t = \sigma(W_{xo}xt + W_{ho}h_{t\text{-}1} + W_{co}c_t + b_o),$$

$$h_t = o_t \odot \tanh(c_t),$$

$W_{x\beta}$ ist die Gewichtung der entsprechenden Information vom Input x zu β, wobei $\beta \epsilon \{i, f, c, o\}$. Die Sigmafunktion ist mit $\frac{1}{1+e^{-x}}$ gegeben (Zhu u. a. (vgl. 2016, übersetzt durch d. Verf.), zitiert nach Graves (2012)).

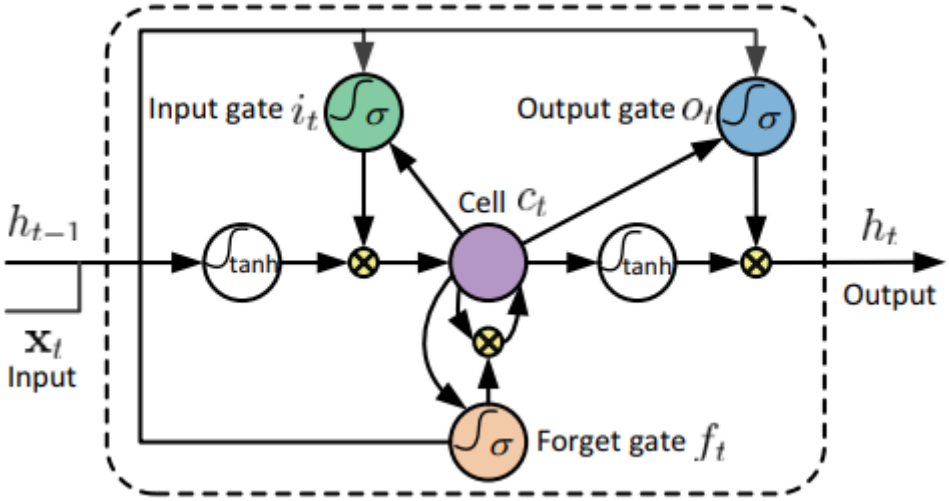


Abbildung 2.3.: Aufbau vom LSTM (Zhu u. a. 2016)

2.3. Lernalgorithmen für Neuronale Netze

2.3.1. Überblick

Da ein Neuronales Netz sehr allgemein verwendet werden kann und es nicht speziell auf eine Anwendung zugeschnitten ist, braucht man Lernalgorithmen, die das Netz auf bestimmte Aufgaben trainieren. Sie sind dazu da, dass das Netz die Verbindungen zwischen den einzelnen Neuronen anpasst, so wie es auch im Gehirn geschieht (2.2.1). Diese Anpassungen werden durch Änderungen der Gewichtungen einzelner Verbindungen umgesetzt. Dabei gilt die Grundlage, dass Neuronen, die wichtige Daten liefern, eine höhere Gewichtung bekommen als Neuronen mit weniger relevanten Daten. Für das entsprechende Training gibt es verschiedene Algorithmen, wobei drei davon in dieser Arbeit genutzt werden.

2.3.2. Backpropagation Learning

„Heutige neuronale Netze, die konkrete Anwendungsprobleme lösen sollen, greifen typischerweise auf das Backpropagation-Verfahren zurück“ (Rey und Beck 2018). Es dient zur Fehlerminimierung des Hidden Layers. Dazu muss überprüft werden, wie groß die Abweichung des Outputs vom RNN zum Sollzustand ist. Falls die Abweichung größer als ein bestimmter Schwellenwert ist, wird die Gewichtung der einzelnen Inputs angepasst, sodass sich der Fehler reduziert. Dies wird iterativ fortgeführt, bis man die Gewichtung des Input Layers erreicht und anpasst (Riedmiller und Braun 1993; Fu-Chuang 1990; Rey und Beck 2018, vgl.)

2.3.3. Reinforcement Learning

„Reinforcement learning can be viewed as a form of learning for sequential decision making“ Unity Technologies (2018a). Das Hauptaugenmerk liegt auf dem Erstellen von Regeln, die ein Agent befolgen soll. Der Agent bildet die Schnittstelle zwischen Neuronalen Netz und dem

restlichen System. Die Umsetzung erfolgt durch das Belohnungs- und Bestrafungsprinzip. Wenn der Agent eine Aktion ausführt, die „gut“ war, bekommt er eine Belohnung. Im Gegensatz dazu wird er bestraft, falls er etwas macht, dass die Lösung der Aufgabe behindert. Sein Anreiz besteht in der Maximierung der Belohnungen. Damit dies gelingen kann, muss er nach und nach ein Modell entwickeln, das ihm zeigt, welche Aktionen er wann ausführen soll. Dies ist insofern wichtig, da es nicht nur darum geht, zum Zeitpunkt t eine maximale Belohnung zu bekommen, sondern diese über die gesamte Zeit zu maximieren.

2.3.4. Imitation Learning

Das *Imitation Learning* ist eine spezielle Form des *Reinforcement Learnings* (2.3.3). Der grundlegende Unterschied dabei ist, dass in diesem Fall das System die Entscheidungen von Menschen in der selben Situation verwendet und deren Verhalten nachstellt. Dadurch liegen dem RNN schon Informationen vor, mit denen es weiterarbeitet und aus denen es sich die Entscheidungsregeln ableiten kann.

2.4. Hardware

Für eine gute Erkennung von Gesten ist eine genaue und präzise Hardware essenziell, denn sie limitiert die mögliche Anzahl an Gesten und gibt vor, inwieweit sich Gesten unterscheiden müssen. In diesem Bereich wurden in den letzten Jahren große Fortschritte gemacht. Für diese Arbeit werden VR-Handschuhe von Noitom International Inc. (Version 2018) genutzt. Sie arbeiten mit Sensoren an den Fingerkuppen und können damit die Fingerposition relativ präzise darstellen. Dadurch ist nicht nur die Erkennung einer Geste leichter, sondern die Hand des Nutzers wird zusätzlich natürlicher abgebildet und die Barriere zwischen virtueller und echter „Realität“ sinkt. Zwar sind die Handschuhe in dieser Hinsicht eine deutliche Verbesserung im Vergleich zu infrarotbasierten Systemen, wie der Microsoft Kinect (Microsoft Corporation Version 2018) oder LeapMotion (LeapMotion Inc. Version 2017), allerdings stellen sie ebenfalls keine perfekte Lösung dar. Ein Problem, welches vor allem beim Thema Gestenerkennung entscheidend ist, ist die fehlende Darstellung der Abstände zwischen zwei Fingern. Diese fehlt vollkommen und man muss dementsprechend beachten, dass die Gesten nicht dadurch klassifiziert und von anderen Gesten abgegrenzt werden können.

3. Stand der Wissenschaft

3.1. Gestenbasiertes Schreiben

Es gibt verschiedene Veröffentlichungen, die sich mit dem *Gestural Writing* und verwandten Systemen beschäftigen. Einige Arbeiten, beispielsweise Murakami und Taguchi (1991), nutzen das Alphabet der Gebärdensprache, unter der hier explizit nicht *Gestural Writing* verstanden wird. Allerdings gibt es auch andere Veröffentlichungen, die einen selbst entwickelten Satz an Gesten nutzen. Hierbei ist vor allem der Ansatz von Mehler, Lücking und Abrami (2014) für diese Arbeit relevant.

In dieser Veröffentlichung wird ein System vorgestellt, welches die Möglichkeit bietet, Museumsinhalte zu bewerten und zu verknüpfen. Die Erkennung erfolgt dabei über eine Microsoft *Kinect*. Dabei steht der Nutzer vor der Kinect-Engine und führt entsprechende Gesten aus. Diese dienen einerseits zur räumlichen Einordnung und andererseits zur Verknüpfung von Objekten. Da auch in dieser Arbeit Objekte verknüpft werden sollen, wurden drei Gesten aus dieser Veröffentlichung übernommen. Es handelt sich um die Link- (4.1.11), Contact- (4.1.12) und Part-whole-Geste (4.1.13). Die Unterschiede liegen hauptsächlich im Verwendungszweck und dem daraus resultierenden unterschiedlichen Alphabet an Gesten. Ein weiterer Unterschied ist die genutzte Hardware. Diese wirkt sich natürlich auf die Gesten aus, die man nutzt (siehe auch 2.4). Die *Kinect* kann nur erkennen, ob ein Finger ausgestreckt wird, aber nicht welcher. Daher kann man keine Gesten verwenden, die eine Unterscheidung zwischen der Anzahl an ausgestreckten Fingern erfordert. Folglich müssen sich die Gesten anders unterscheiden, als es in dieser Arbeit der Fall ist.

3.2. Gestenerkennung mittels Neuronaler Netze

3.2.1. Überblick

Es gibt verschiedene Arbeiten, die das Thema Gestenerkennung aufgreifen. Einige Veröffentlichungen verwenden dabei Neuronale Netze, die trainiert wurden verschiedenste Gesten zu unterscheiden. Bisher sind die meisten Systeme auf Bildeingaben spezialisiert. Bei einigen kommt jedoch auch andere Hardware zum Einsatz. Ein paar relevante und ähnliche Veröffentlichungen werden im Folgenden kurz vorgestellt.

3.2.2. Gestenerkennung über Bilder

Die erste Veröffentlichungen Ghosh und Ari (2011) beschäftigt sich mit der Erkennung von statischen Gesten. Dazu wird ein *Radial Basis Function Neural Network* (RBFNN) genutzt. Dieses besitzt eine überschaubare Komplexität, da es zwischen Input und Output-Layer nur

ein Hidden-Layer gibt (vgl. Ghosh und Ari 2011, S. 2). Hierbei ist es interessant zu sehen, wie andere Arten von Neuronalen Netzen die Aufgabe der Gestenerkennung lösen und ob deren Erkennungsrate ähnlich ist oder nicht.
Der grundlegende Unterschied zu der Arbeit von (Ghosh und Ari 2011) ist die Art und Weise, wie die Hand dargestellt wird. Während in dieser Arbeit die Handschuhe von Noitom International Inc. (Version 2018) genutzt werden, wurde bei Ghosh und Ari (2011) Bilderkennung als Mittel der Wahl verwendet. Dementsprechend unterscheidet sich auch die formale Darstellung der Hand. Das Hauptaugenmerk liegt dabei auf drei Aspekten der Bilderkennung:

- Segmentation
- Rotation
- Morphological filtering

Dabei treten erwartungsgemäß andere Probleme auf, da nur ein zweidimensionaler Input zur Verfügung steht. Abschließend wurde herausgefunden, dass mit genügend Training eine Erkennungsrate von 99.6% erreicht werden kann (Ghosh und Ari 2011, vgl.).

3.2.3. Gestenerkennung mittels Datenhandschuhen

Zwei weitere Veröffentlichungen Murakami und Taguchi (1991) und Weissmann und Salomon (1999) haben ebenfalls Datenhandschuhe als Grundlage für die Handdaten eingesetzt. Die überprüften Parameter sind bei beiden Ansätzen sehr ähnlich. Es wird jeweils der Winkel an den einzelnen Fingergelenken überprüft, sowie die Lage der Hand im Raum. Der Fokus bei Murakami und Taguchi (1991) liegt in der Entwicklung eines Systems für die Gebärdensprache. Dabei handelt es sich um einen ähnlichen Ansatz, wie der in dieser Arbeit genutzte, da man Gebärdensprache auch zum *Gestural Writing* zählen kann. Das genutzte Netz wurde so eingebunden, dass es immer genau einen Outputwert gab, nämlich die Geste, die erkannt wurde (Murakami und Taguchi 1991).
Bei Weissmann und Salomon (1999) wurden zwei verschiedene Ansätze von Neuronalen Netzen verglichen. Genutzt wurden dazu ein *Radial-basis function neural network* (RBFNN), welches nur aus Input- und Outputlayer besteht, wobei jeder Output Zugang zu allen Inputs hat (vgl. Weissmann und Salomon 1999, S. 3). Diesem wurde ein RNN gegenübergestellt (2.2.4). Beide Arten an Netzen wurden jeweils mit fünf verschiedene Testsets trainiert. Dabei konnte festgestellt werden, dass das RNN sowohl die eigenen Daten, als auch alle anderen Testsets sehr gut erkennt. Das RBF hingegen war etwas schwächer (Erkennungsraten bei ca. 70%) beim Erkennen der eigenen Gesten aus dem Testset. Allerdings gab es beim Quervergleich der Testdaten, wie beim RNN, sehr gute Testdaten (ca. 98%).

4. Theoretische Vorarbeiten

4.1. Gesten

4.1.1. Einführung

Bei der Erstellung der Gesten ist darauf zu achten, dass sie für den Nutzer verständlich sind, damit er sie intuitiv verwenden kann. Zum Beispiel sollte man im westlichen Kulturkreis für eine Geste, die etwas Positives aussagt eher einen *Daumen nach oben* wählen und nicht den ausgestreckten Mittelfinger. Dies hängt nämlich auch vom Kulturkreis ab, in dem man sich befindet. Zeitgleich ist es sinnvoll, Gesten, die das Gegenteil aussagen, ähnlich zu konzipieren und mit einem „Nicht“ zu versehen, da der Nutzer dann weniger sehr unterschiedliche Gesten lernen muss. Außerdem steht dann ein Alphabet aus Gesten zur Verfügung, die sich weit genug unterscheiden und deren Distanz groß genug ist. Die Gesten beziehen sich nur auf die Hand und deren Stellung im Raum. Dabei ist es äquivalent, wenn eine Geste vor dem Oberkörper ausgeführt wird oder nahe am Boden, da die restliche Körperstellung vom System ignoriert wird.

Tabelle 4.1.: Übersicht aller Verfügbaren Gesten;
Handanzahl: Ist die Geste mit einer oder zwei Händen ausführbar

Geste	Handanzahl	Quelle (sofern nicht selbst entwickelt)
Zeigen	1	
$=$	2	
$\neq$	2	
$>$	2	im Gespräch mit Abrami, G.
$<$	2	im Gespräch mit Abrami, G.
$\geq$	2	
$\leq$	2	
$\wedge$	1	im Gespräch mit Abrami, Giuseppe
$\neg$	1	
$\vee$	2	
$\rightarrow$	1	
Link	1	Mehler, Lücking und Abrami (2014)
Contact	2	Mehler, Lücking und Abrami (2014)
Part-whole	1	Mehler, Lücking und Abrami (2014)

4.1.2. Zeigen

Die Geste Zeigen beruht darauf, dass der Zeige- und Mittelfinger ausgestreckt sind. Damit ist es möglich einen Zeigestrahl zu erzeugen. Da das „Zeigen“, bei dem lediglich der aus-

gestreckte Zeigefinger verwendet wird, teilweise auch bei anderen Gesten (4.1.7, 4.1.9) zum Einsatz kommt, musste diese Geste abgewandelt werden, um Mehrdeutigkeit zu vermeiden (siehe Abbildung 4.1).

Abbildung 4.1.: Darstellung vom Zeigen

4.1.3. Gleich

Damit ein Gleich erkannt wird, muss der Nutzer beide geöffneten Handflächen übereinander halten, sodass die Hände die Form eines Gleichheitszeichens (=) annehmen. Dabei bildet jede Hand einen Strich vom Gleichheitszeichen (siehe Abbildung 4.2).

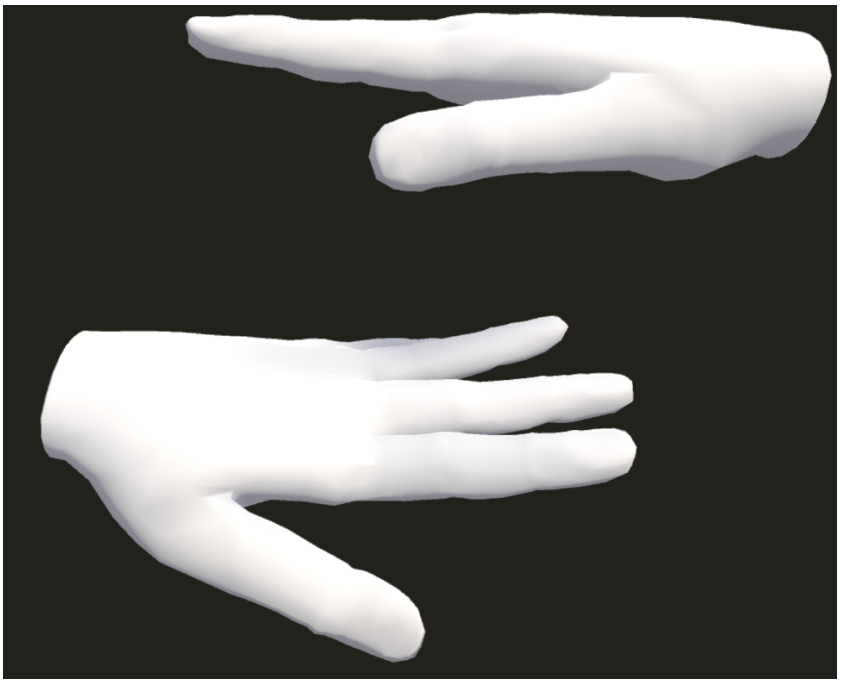
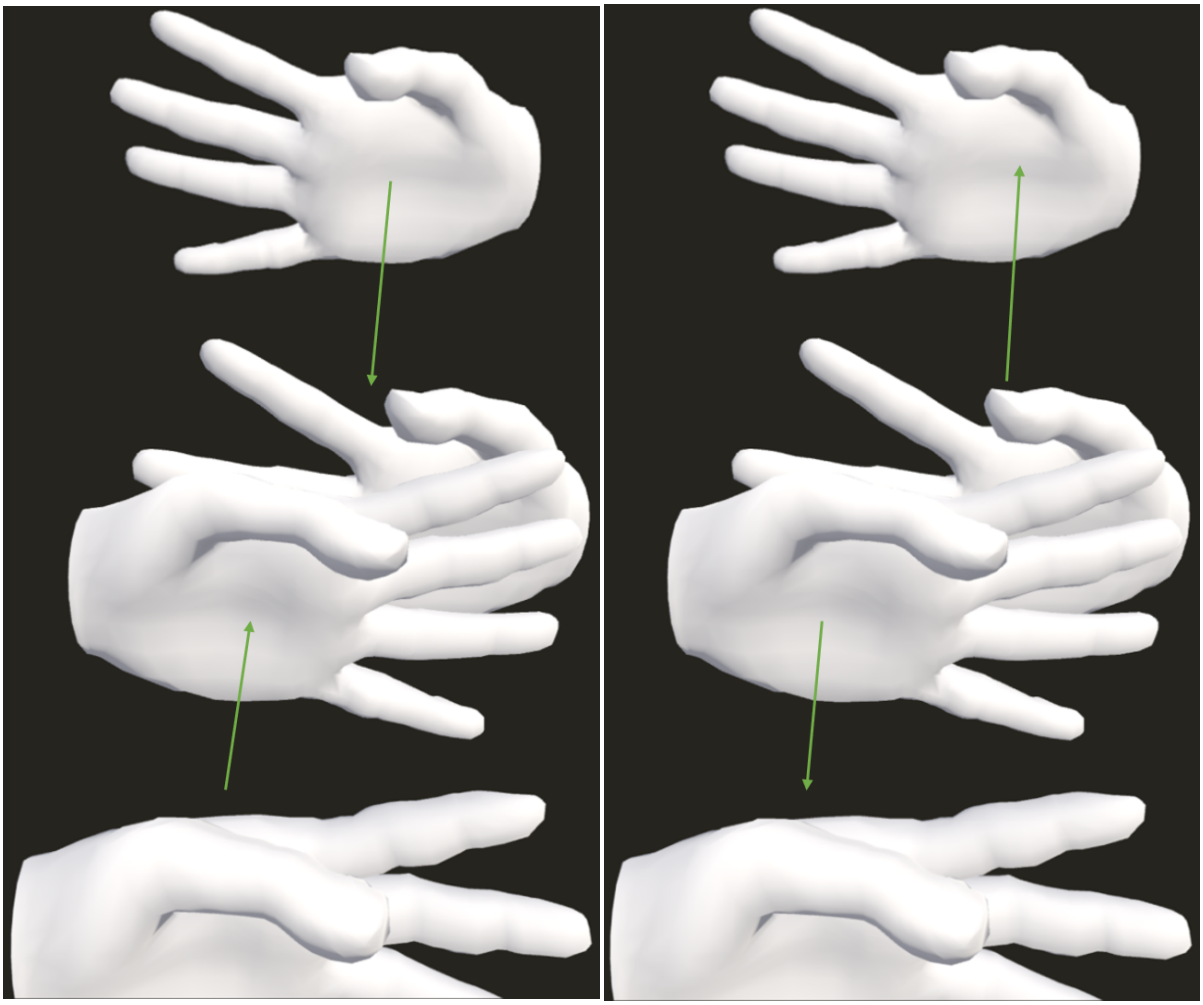


Abbildung 4.2.: Darstellung einer Gleich-Geste

4.1.4. Kleiner und größer

Das Kleiner-Zeichen (>) läuft zusammen. Nach dem selben Prinzip kann man die Geste konstruieren. Man hält beide Hände vor dem Oberkörper ausgestreckt. Bei einer Kleiner-Geste bewegt man beide Hände zusammen, während man sie bei einer Größer-Geste auseinander bewegt. Wichtig dabei ist, dass sich beide Hände bewegen, da sie von den $\leq$ und $\geq$ (4.1.5) unterschieden werden müssen (siehe Abbildung 4.3).



(a) Darstellung von <

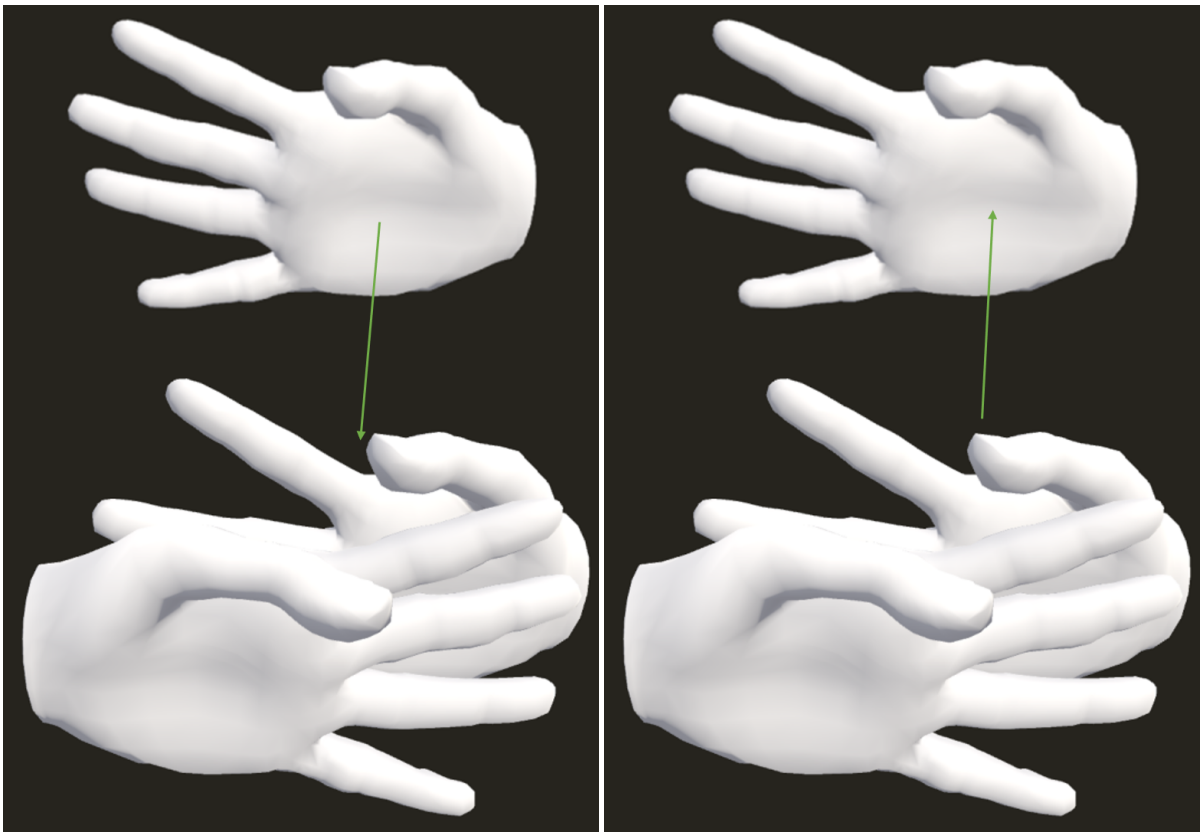
(b) Darstellung von >

Abbildung 4.3.: Kleiner und Größer Gesten

4.1.5. Kleiner und größer gleich

Wie bereits in 4.1.1 gesagt wurde, sollen ähnliche Aussagen ebenfalls durch ähnliche Gesten dargestellt werden. Dementsprechend werden die Kleinergleich und Größergleich-Gesten an < und > angelehnt. Während sich bei „Kleiner“ (4.1.4) beide Hände bewegen, ist die linke Hand hier fest und nur die rechte Hand bewegt sich. Dabei gilt wieder das selbe

Prinzip. Bewegt sich die Hand nach außen, wird es größer ansonsten kleiner (siehe Abbildung 4.4).



(a) Darstellung von $\leq$ (b) Darstellung von $\geq$

Abbildung 4.4.: Kleiner gleich und Größer gleich Gesten

4.1.6. Ungleich

Diese Geste ist als liegendes T konzipiert und soll ausdrücken, dass zwei Objekte nicht identisch sind. Sie ist an das „!= “ aus den Programmiersprachen angelehnt und man muss mit den Fingerspitzen der einen Hand die Handfläche der anderen berühren (siehe Abbildung 4.5).

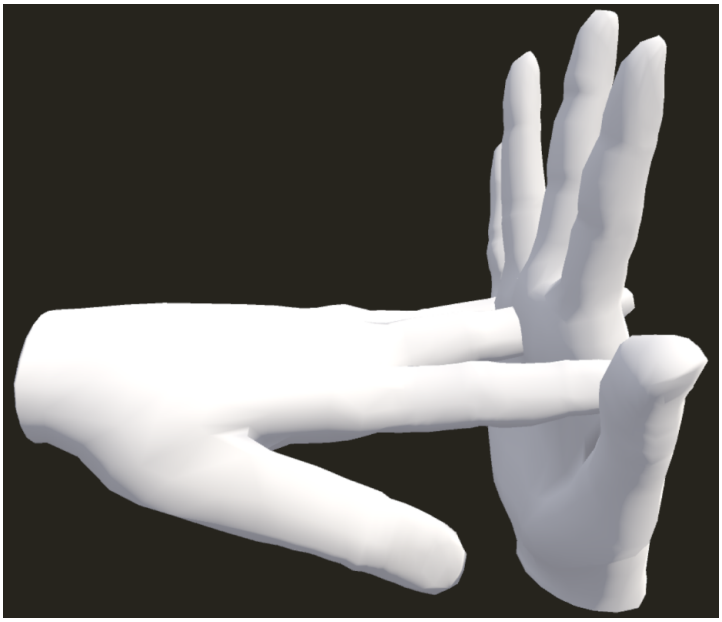


Abbildung 4.5.: Darstellung einer Ungleich-Geste

4.1.7. Logisches Und

Damit das System ein Und erkennt, muss man nur den Zeigefinger ausstrecken. Jedes Objekt, das man verknüpfen möchte, wird, durch darauf zeigen, zu dieser Liste hinzugefügt. Dabei besteht noch eine große Ähnlichkeit zum Zeigen (4.1.2). Der grundlegende Unterschied liegt in der Bewegung zwischen dem Zeigen auf zwei Objekte. Nachdem man auf ein Objekt gezeigt hat, bewegt man den Unterarm wieder zurück zum Kopf, sodass man senkrecht nach oben zeigt. Von dort aus bewegt man den Arm nun in die Richtung des nächsten Objekts (siehe Abbildung 4.6).

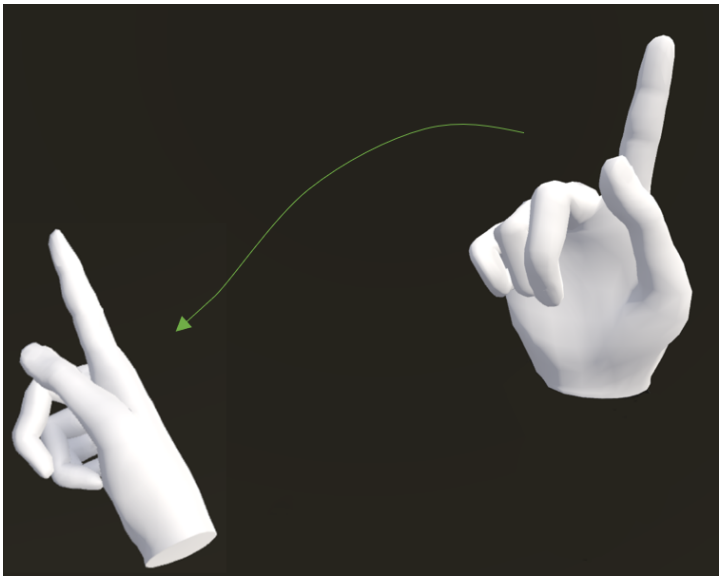


Abbildung 4.6.: Darstellung eines Unds

4.1.8. Negation

Wie bereits in 4.1.1 ausgeführt, soll es eine *Nicht*-Geste geben. Die Geste ist eine ausgestreckte waagerechte Hand vor dem Oberkörper des Nutzers (siehe Abbildung 4.7).

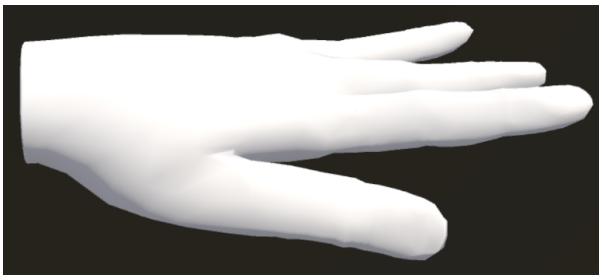


Abbildung 4.7.: Darstellung einer Negation

4.1.9. Logisches Oder

Die Geste orientiert sich am Und (4.1.7). Mit der rechten Hand wird die selbe Bewegung, wie beim Und ausgeführt. Die linke Hand wird vor dem Brustkorb waagerecht gehalten und beruht auf der Negation (4.1.8). Damit soll ein „nicht Und“ symbolisiert werden (siehe Abbildung 4.8).

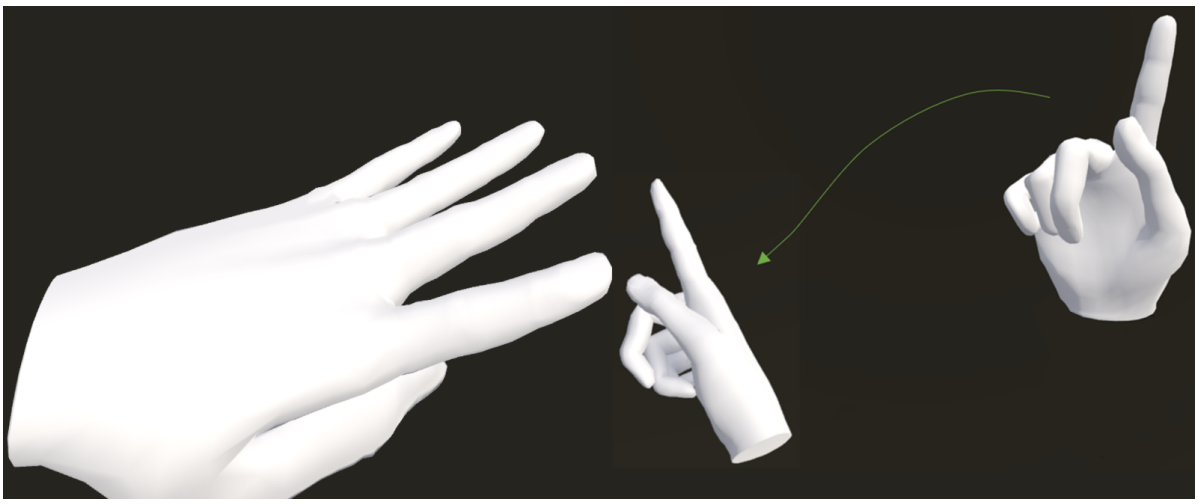


Abbildung 4.8.: Darstellung eines Oders

4.1.10. Implikation

Um eine Folgerung darzustellen, bewegt man die rechte Hand von links nach rechts bzw. die linke von rechts nach links. Dabei ist die Hand geöffnet und das Handgelenk wird leicht gedreht (siehe Abbildung 4.9).

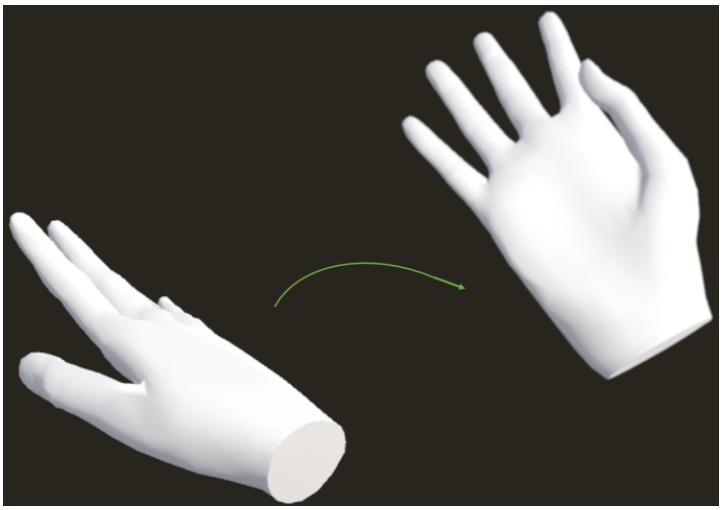


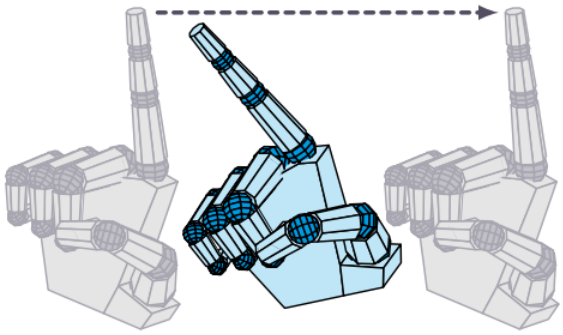
Abbildung 4.9.: Darstellung der Implikation

4.1.11. Link Geste

Um Objekte zu verknüpfen, kann man diese Geste nutzen. Anders als bei anderen Gesten setzt man hiermit nur eine Referenz zwischen Objekten, ohne eine bestimmte Klassifizierung vorzunehmen. Dabei handelt es sich um die allgemeinste Art der Darstellung einer Verknüpfung. Sie wurde in der Veröffentlichung von Mehler, Lücking und Abrami (2014, S.

7, Fig.15) entworfen. Man streckt den Zeigefinger aus und bewegt ihn kurz von links nach rechts (siehe Abbildung 4.10).

(a) Schematische Darstellung (Mehler, Lücking und Abrami 2014, S. 7, Fig.14)



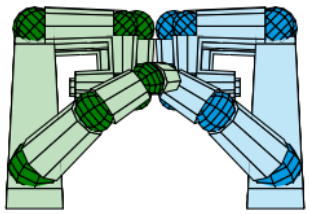
(b) Handdarstellung der Geste (Mehler, Lücking und Abrami 2014, S. 7, Fig.15)

Abbildung 4.10.: Darstellung der Link Geste

4.1.12. Contact Geste

Um Events zu verknüpfen, kann man die Contact Geste von Mehler, Lücking und Abrami (2014, S. 8, Fig. 27) verwenden. Ein Event könnte beispielsweise eine Eheschließung sein. Dafür sollen beide Personen miteinander verknüpft werden. Wie sich bereits aus dem Namen schließen lässt, beruht die Geste auf Kontakt. Beide zur Faust geballten Hände berühren sich (siehe Abbildung 4.11).

(a) Schematische Darstellung (Mehler, Lücking und Abrami 2014, S. 8, Fig. 26)



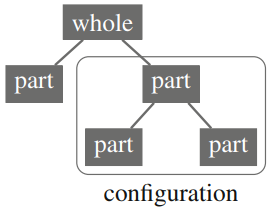
(b) Handdarstellung der Geste (Mehler, Lücking und Abrami 2014, S. 8, Fig. 27)

Abbildung 4.11.: Darstellung der Contact Geste

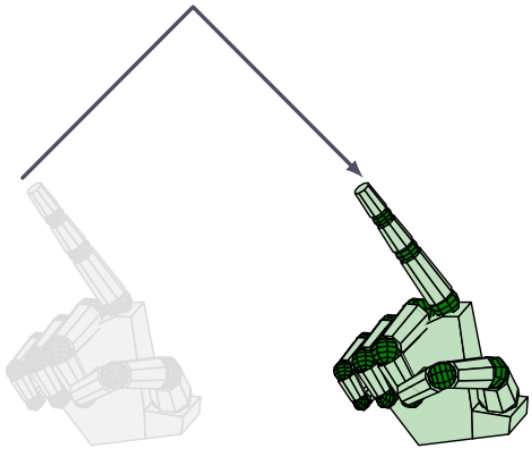
4.1.13. Part-Whole Geste

Diese Geste stammt ebenfalls aus der Veröffentlichung Mehler, Lücking und Abrami (2014, S. 7, Fig.13) und drückt eine Referenz zu etwas Übergeordneten aus (Bsp. ein Apfel ist Obst). Man versucht eine Baumstruktur nachzuahmen (siehe Abbildung 4.12). Die Hand bewegt

man nur mit dem Handgelenk von unten links nach oben rechts und von dort schräg nach rechts unten.



(a) Schematische Darstellung (Mehler, Lücking und Abrami 2014, S. 7, Fig. 12)



(b) Handdarstellung der Geste (Mehler, Lücking und Abrami 2014, S. 7, Fig. 13)

Abbildung 4.12.: Darstellung der PART-WHOLE Geste

4.2. Handrepräsentation

Die erste wichtige Neuerung, im Vergleich zum später genauer erläuterten alten Ansatz (5.3), ist die Trennung zwischen Geste und genutzter Hardware. Dafür muss als Erstes eine Handrepräsentation (siehe Abbildung 4.13) geschaffen werden, die einerseits so genau ist, dass man damit eine Geste definieren kann und die andererseits von vielen verschiedenen Hardwarearten erkannt wird.

Welche Werte definieren eine Hand?

Wenn man sich diese Frage stellt, wird man schnell zu dem Ergebnis gelangen, dass man überprüfen muss, ob ein Finger ausgestreckt ist oder nicht. Bei einem ausgestreckten Finger sind die einzelnen Fingerknochen annähernd auf einer Linie und der Knickwinkel am entsprechenden Gelenk ist nahe $0°$. Dies lässt sich gut überprüfen und gibt verlässliche Informationen darüber, ob ein Finger ausgestreckt ist. Da hierbei nur Winkel betrachtet werden, gibt es bei dem Ansatz nicht die Möglichkeit verschiedene Arten von gestreckten Fingern zu erkennen. Es kann sein, dass ein Finger ausgestreckt ist, aber nicht in Verlängerung zur Handfläche steht, sondern im $90°$ Winkel abgeknickt ist. Daher ist die Idee zwei Vektoren pro Finger zu überwachen[1]. Der erste Vektor fängt am Handgelenk an und zeigt von dort aus die Fingerrichtung an, während der zweite erst am Fingergrundgelenk beginnt und die von dort resultierende Richtung speichert. Damit werden die Fingerstellungen gut überwacht.

[1] Im Gespräch mit Abrami, G.

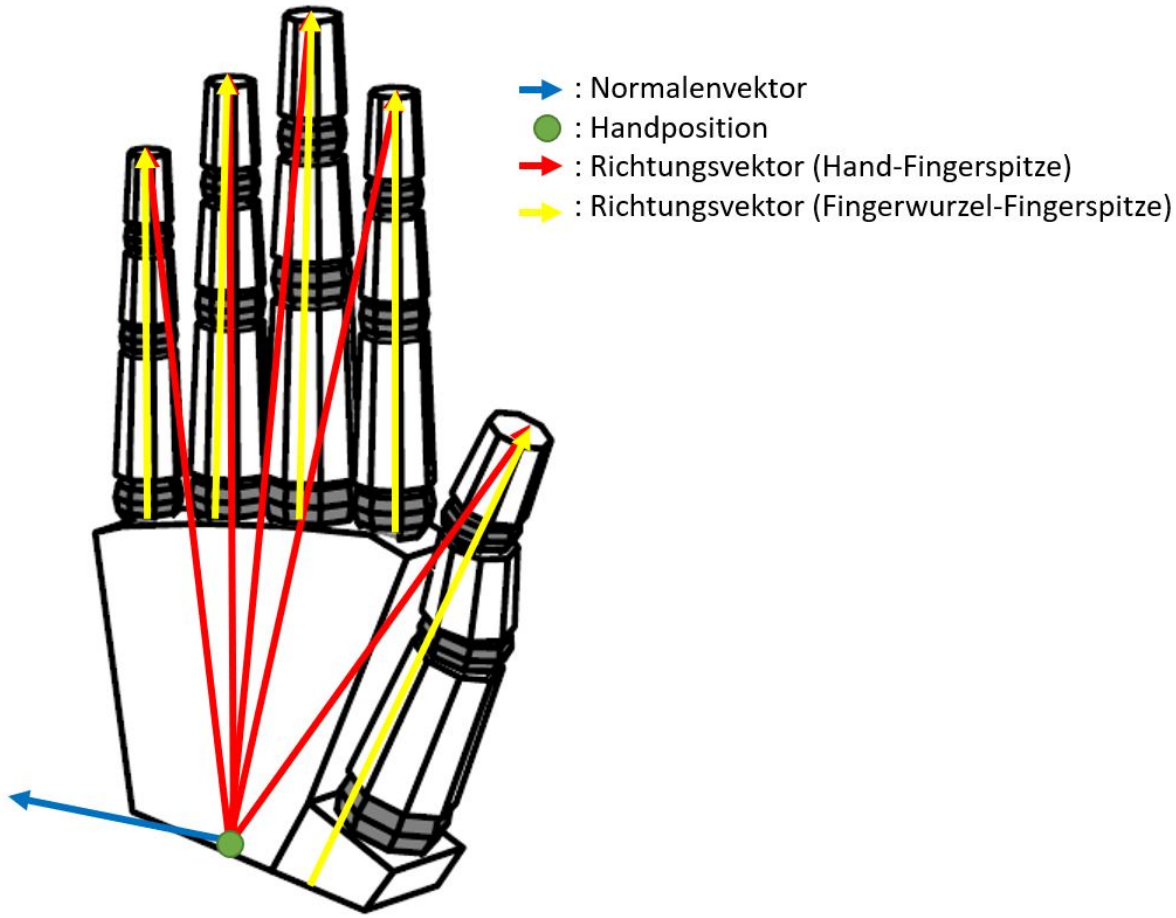


Abbildung 4.13.: Allgemeine Handdarstellung

Allerdings gibt es bisher noch keine Informationen über die Position einer Hand oder deren Rotation im Raum. Dies sind die anderen Parameter, die eine Hand ausmachen. Die Position stellt die globale Position der Hand dar und steht nicht in Relation zur anderen Hand oder zum Nutzer. Dies führt zu einer allgemeineren Darstellung, die man je nach Anwendungsfall immer noch einschränken kann. Aus dem gleichen Grund wie die Position ist die Drehung der Hand ebenfalls global dargestellt. Eine Drehung lässt sich gut durch einen Normalenvektor einer Ebene darstellen. Da die Handfläche nichts anderes als eine Ebene ist, muss man lediglich den Normalenvektor ausrechnen. Durch die Position und die Rotation hat man die Stellung der Hand für die Gesten hinreichend gut dargestellt und jegliche Weiterverarbeitung geschieht auf Basis dieser Daten.

5. Technische Umsetzung

5.1. Überblick

Für die Technische Umsetzung wurden zwei verschiedene Ansätze entwickelt, wobei der zweite Ansatz, der ein Neuronales Netz beinhaltet, den ersten ersetzt hat. Beide Ansätze werden im Folgenden genauer erläutert. Obwohl beide Implementierungen unterschiedlich funktionieren, gibt es trotzdem Gemeinsamkeiten, die im Anschluss näher betrachtet werden.

Das Gesamtsystem zur Erkennung (siehe Abbildung 5.1) basiert auf drei einzelnen Modulen, die über eine Schnittstelle miteinander kommunizieren. Im ersten Modul, das für die Gesten zuständig ist, wird überprüft, ob eine bestimmte Handstellung vom Nutzer eine Geste darstellt. Falls dies der Fall ist, wird dies an die vermittelnde Stelle, den Gesture Controller weitergegeben. Dieser dient zur organisierten Weitergabe der Informationen an die entsprechende ausführende Einheit. Dort wird beschrieben, was beim Eintreten der Geste geschehen soll. Sobald eine Geste erkannt wird, wird die entsprechende Aktion ausgeführt.

Durch diese Aufteilung ist gewährleistet, dass Geste und Bedeutung stets getrennt sind. Dies ist wichtig, da in unterschiedlichen Kulturkreisen und Anwendungsbereichen eine Geste nicht immer die selbe Bedeutung aufweist. Ein klassisches Beispiel dafür ist der nach oben ausgestreckte Daumen. Im normalen Sprachgebrauch in Deutschland wird diese Geste mit „OK“ verbunden. Befindet man sich allerdings beim Tauchen wird die selbe Geste nicht mehr mit „OK“, sondern mit „Auftauchen“ verbunden (Recreational Scuba Training Council 2005, S. 5). Dieses Beispiel veranschaulicht stellvertretend für viele andere Fälle, dass es wichtig ist, Geste und Bedeutung getrennt zu betrachten.

5.2. Ansatz Gestenerkennung

Die Unterschiede zwischen beiden Ansätzen beziehen sich hauptsächlich auf das erste Glied der Kette, also die Gestenerkennung. Der zweite Ansatz, der anschließend implementiert wurde, nutzt ein Neuronales Netz zur Erkennung der Gesten[1]. Parallel zum Neuronalen Netz wurde die Schnittstelle für den Input der Handdaten abstrakt gestaltet, sodass das ganze Interface unabhängig von der genutzten Hardware benutzt werden kann, solange diese in der Lage ist, die Fingerrichtungen, die Lage der Hand im Raum und die Handposition zu erkennen und wiederzugeben.

[1] Idee von Abrami, G.

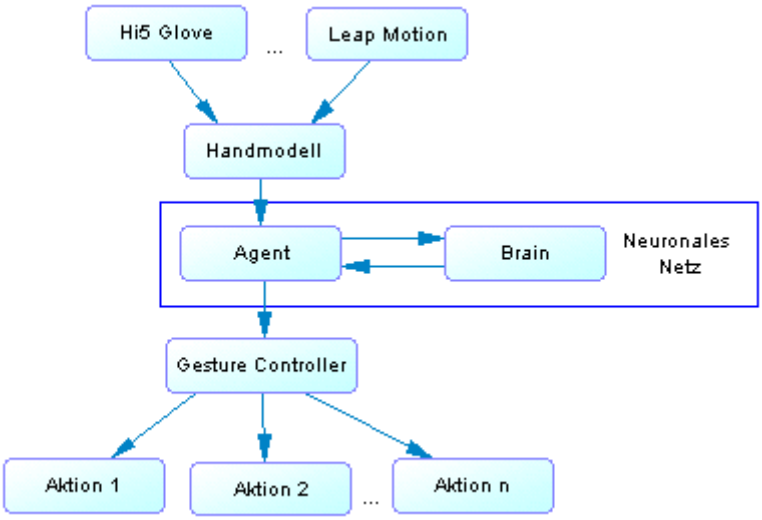


Abbildung 5.1.: Use case Diagramm der Gestenerkennung

5.2.1. Neural Network Aufbau

Das hier genutzte Netz ist eine von Unity selbst entwickelte AI, die das ganze Neuronale Netz beinhaltet. Es besteht aus drei Komponenten (siehe Abbildung 5.2). Der AGENT dient hauptsächlich als Schnittstelle und ausführende Instanz des Netzes. Jeder Agent liest die vorher definierten Handdaten (4.2) ein und übermittelt diese an das BRAIN. Darin ist das Neuronale Netz eingebettet.
Sobald eine Entscheidung gefällt wurde, wird diese wieder an die Agenten übermittelt und von ihnen entsprechend weiterverarbeitet. Die dritte Komponente ist die ACADEMY. Diese soll die Umwelt überwachen und koordiniert alle Agenten. Da hier die Umwelt nicht nach x Schritten zurückgesetzt werden muss, wird die ACADEMY nicht tiefergehend genutzt. Jeder Agent, der Gesten erkennen soll, muss sich selbst verwalten, da er für sich seine Regeln aufstellt und überprüft.

5.2.2. Arbeitsweise - eine Iteration

Wie bereits beschrieben (5.1), wurde eine strikte Trennung der einzelnen Ebenen und Module durchgeführt. Dies ermöglicht eine Generalisierung und vereinfacht die Erweiterung des Systems um weitere Gesten. Da in jedem Schritt viele unterschiedliche Verarbeitungen stattfinden, gibt es hier einen Überblick, wie diese genau zusammenhängen und sich gegenseitig beeinflussen.

1. Sammlung der unverarbeiteten Daten der Handschuhe
2. Überführung der Daten in die abstrakte Handdarstellung
3. Agent leitet diese Daten an das Netz weiter
4. Das Gehirn wertet aus welche Geste eingetreten ist

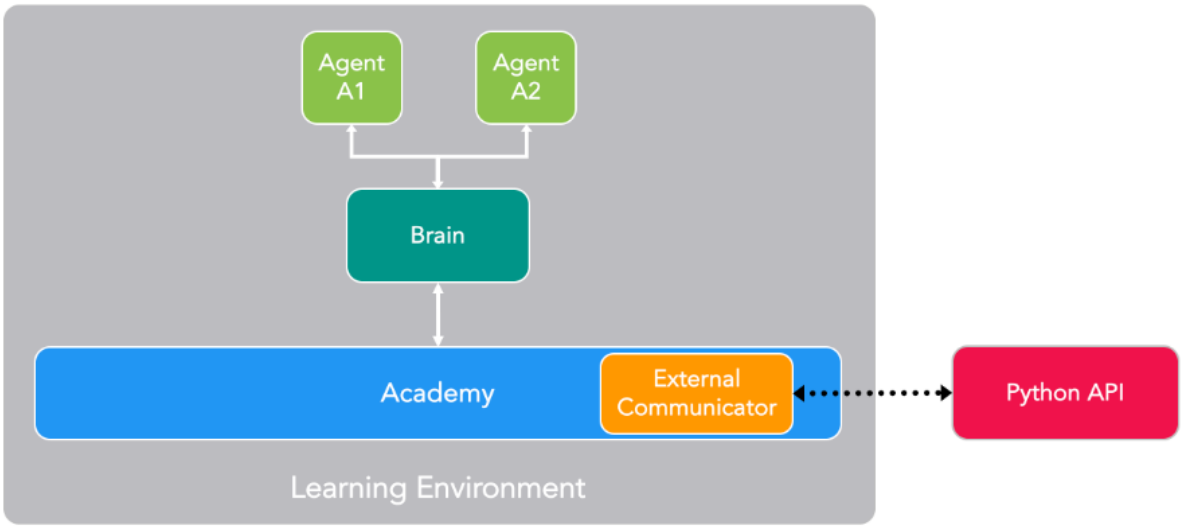


Abbildung 5.2.: Schematischer Aufbau des genutzten Neural Network (Unity Technologies 2018b)

5. Agent übergibt diese Geste an den Gesture Controller
6. Aktion wird ausgeführt

In jedem Schritt ist es wichtig, die aktuellen Daten der Hände zu bekommen. Da diese abhängig vom genutzten Controller anders vorliegen können, müssen sie als Erstes in die allgemeine Abbildung der Hand transformiert werden (4.2, Abbildung 4.13). Wenn die bereinigten Daten vorliegen, müssen diese an das Neuronale Netz übermittelt werden, damit die richtige Verarbeitung anfangen kann. Der Agent dient als Schnittstelle nach außen zur Umwelt (5.2.1). Er sammelt alle Daten, die genutzt werden sollen und übergibt sie an das Netz. Dieses entscheidet daraufhin, welche Geste gewählt wurde und gibt diese Entscheidung an den Agenten weiter. Die gewählte Geste wird an die nächste Schnittstelle zwischen Netz und Aktion der Geste weitergereicht, den Gesture Controller. Die Klasse, die die entsprechende Aktion verwaltet, wird die Daten sammeln, die sie zur Ausführung der Aktion braucht, sobald die Geste gewählt wurde. Daraufhin wird wieder mit dem ersten Schritt fortgefahren und der nächste Iterationsschritt beginnt.

5.2.3. Agenten - Inputverwaltung

Mehrfach wurde bereits erwähnt, dass der Agent die zentrale Schnittstelle für das Netz ist. Nun sind die Daten für das Netz von zentraler Bedeutung für die Qualität der Entscheidung.

Anfangs wurden als Input nur die Vektoren, die die Hand darstellen, übergeben. Wichtig dabei ist das Normalisieren der Vektoren, damit die Werte zwischen -1 und 1 liegen und besser verarbeitet werden können (Unity Technologies 2018a, vgl.). Da das Neuronale Netz pro Neuron im Input Layer nur eine Zahl verwalten kann, wird ein Vektor auf drei Neuronen aufgespalten. Dabei wurden insgesamt 67 Werte an das Netz übergeben. Dadurch konnten

bereits fünf Inputs eingespart werden, da man nicht die Handpositionen übernimmt, sondern nur die Distanz zwischen diesen Vektoren. Es ist nur relevant, wie die Hände zueinander stehen, aber nicht wo sich diese im Raum befinden. Trotzdem bleibt dieses erste Modell sehr komplex. Dadurch dass die Daten richtungsabhängig sind, braucht man sehr viele Daten aus verschiedenen Richtungen. Ansonsten werden die Gesten aus manchen Stellungen schlechter erkannt. Bemerkbar macht sich das, wenn man zu wenig verschiedene Positionen trainiert hat.

Zudem ist ein Modell nicht, um einzelne Gesten, erweiterbar. Sobald eine Geste antrainiert wird, stagniert das Netz bei der Belohnung, wenn es eine neue zweite Geste lernen soll. Deswegen muss man immer das komplette Gestenalphabet trainieren.
Der große Nachteil an diesem System ist die fehlende Vorverarbeitung der Daten und die hohe Anzahl an Inputs, da dadurch das Training sehr lange dauert. Außerdem ist die Eingabe der Fingerrichtungen hinderlich, da die Geste dadurch richtungsabhängig ist. Das Netz kann dann nicht wirklich überprüfen welche Eingaben wichtiger sind, da sich jeder Vektor bei den Bewegungen verändern wird und keine Generalisierung möglich ist bzw. sehr lange dauert.

Inputs in das RNN:

- Abstand der Hände (1 Input)
- $5\cdot$ Vektor zwischen Fingerspitze und Handgelenk (15 Inputs pro Hand)
- $5\cdot$ Vektor zwischen Fingerspitze und Fingeranfang (15 Inputs pro Hand)
- Normalenvektor für rechte und linke Hand (6 Inputs)

Daraufhin wurde ein stark vereinfachter Ansatz entwickelt. Da man bei diesem mehr Vorarbeiten verwendet, werden insgesamt nur noch 17 Inputs benötigt. Der grundlegende Unterschied liegt darin, dass nur noch ein boolescher Wert pro Finger eingelesen wird und nicht zwei Vektoren. Der Boolean gibt an, ob ein Finger ausgestreckt ist oder nicht. Wie in 4.2 beschrieben, gelingt dies über den Winkel am Gelenk. Der Vorteil hierbei ist, dass diese Vorverarbeitung genau ist und gute Erkennungsraten liefert. Da man hierbei nur einen Freiheitsgrad hat - den Winkel - ist diese Verarbeitung nicht richtungsabhängig und kann im Gegensatz zum vektorenbasierten Ansatz (5.3) leicht generalisiert werden, da der Grad an Komplexität wesentlich geringer ist. Die Vereinfachung besteht jedoch nicht nur in der Senkung der Inputanzahl, sondern ebenfalls in den möglichen Eingabewerten, die übergeben werden. Die Vektoren können Werte zwischen $[-1, 1]$ annehmen. Der Boolean unterscheidet hingegen nur zwischen *true* und *false*. Das Netz kann wesentlich schneller Gesten lernen, da es nur überprüft welche Geste welche ausgestreckten Finger benötigt. Allerdings fällt hierbei die Möglichkeit weg, verschiedene Arten von ausgestreckten Fingern zu erkennen (4.2), da man nicht mehr überprüfen kann, wie dieser Finger im Raum aussieht. Das System, das man nutzen sollte, hängt folglich vom Alphabet an Gesten ab, die man verwendet. Sollte sich keine Geste im Alphabet befinden, welche auf verschiedenen ausgestreckten Fingern beruht, kann man den vereinfachten Ansatz nutzen. Ansonsten muss der komplexe Ansatz genutzt werden.

5.2.4. Agenten - Outputverwaltung

Während es für den Input bedeutend schwieriger ist, ein optimales System zu entwickeln, ist dies beim Output einfacher, da im wesentlichen die Ausgabe vom Neuronalen Netz in eine Geste konvertiert werden muss. Dies kann auf zwei Arten geschehen. Bei der ersten Möglichkeit (siehe Algorithmus 1), gibt das Netz ein Array an Gewichtung aus. Um die Geste zu finden, für die sich das Netz entscheiden würde, muss man nur nach der höchste Gewichtung suchen. Wird eine neue Gewichtung gefunden, aktualisiert man das Maximum und die Geste. Die Zuweisung zur Geste erfolgt hierbei über den Index der Gewichtung im Array. Die letzte Stelle im Array zeigt an, ob die Geste an der linken oder rechten Hand erkannt wurde. Man kann zusätzlich einen Schwellenwert übergeben, bei dem die Geste als nicht erkannt gewertet wird. Dies hat den Hintergrund, dass das RNN immer eine Entscheidung ausgibt, auch wenn eventuell gar keine Geste ausgeführt wurde. Falls mehrere Gesten eine ähnliche Gewichtung haben, wird eher keine Geste gewählt als eine falsche Entscheidung getroffen. Damit soll verhindert werden, dass das Neuronale Netz nur eine Geste rät, weil die Daten zu keiner Geste passen. Der Schwellenwert ist aber beim Training nicht relevant, da beim Training die Gewichtung angepasst werden sollen und dort eine Veränderung des Outputs vor der Vergabe der Belohnung (2.3.3) hinderlich wäre.
Der zweite Ansatz (siehe Algorithmus 2), den man wählen kann, ist die direkte Ausgabe der Geste. Hierbei gibt das RNN nur genau einen Wert aus, da es im Diskreten Modus arbeitet. Dieser kann allerdings nicht direkt übergeben werden, weil hierbei nur die Geste angegeben wird, ohne die Information, ob sich das Netz für Rechts oder Links entschieden hat. Um dies zu lösen, wird die Anzahl an möglichen Werten bzw. Gesten verdoppelt. Dabei dient die erste Hälfte für Gesten, die die linke Hand betreffen und die zweite Hälfte dient zur Erkennung von Gesten der Rechten Hand.
Als Beispiel dient die AND-Geste. Diese hätte den Index 2 von insgesamt 4 Gesten. Wird die Geste für die linke Hand erkannt, gibt sie eine 2 aus. Bei der Erkennung der Geste an der rechten Hand würde 6 ausgegeben werden.

Bei diesem Ansatz kann kein Schwellenwert genutzt werden, da man nur genau einen Wert zurückbekommt und keine Gewichtung. Es muss eine andere Lösung gefunden werden, die die Entscheidung nachbearbeitet und Fehler reduziert. Der genutzte Algorithmus überprüft, dass innerhalb der x letzten Entscheidungen die Geste $\frac{3}{4}x$-mal auftreten muss, damit sie weitergeleitet wird. Damit wird eine konstante Ausgabe erzeugt und Flackern vermieden. Es hat sich gezeigt, dass ein x von 10 gute Ergebnisse erzielt, da man dann einen größeren Zeitraum betrachtet und die Fehlertoleranz nicht zu gering ist.

5.3. Verworfener Ansatz

Der folgende Ansatz wurde als erstes verfolgt, allerdings wurde er vor der endgültigen Fertigstellung durch die Implementierung eines Neuronalen Netzes ersetzt.

Die Idee dieses Ansatzes besteht darin, dass es eine Heuristik gibt, die das Aussehen einer Geste in den Grundzügen definiert und daraufhin erkennt, ob der Nutzer eine Geste ausge-

Input :

- Array A of floates
- optional: threshold

Output :

- $gesture$, Gesturetype
- $isLeft$, bool which indicates if the Gesture was done with the left or right hand

```
maximum;
gesture = None;
for i = 0; i < A.length; i++ do
    if A[i] > maxvalue then
        maxvalue = A[i]; //If there is a higher weight the gesture will be updated
        gesture = i
    end
end
if A[n − 1] < 0 then
    isLeft = true; //Linke Hand wurde erkannt
end
else
    isLeft = false;
end
if maxvalue < Schwellenwert then
    gesture = None //value of the gesture was to low and gesture will be classified as
      not recognized
end
return (gesture, isLeft)
```

Algorithmus 1 : Algorithmus zur Erkennung von Gesten mittels Gewichtung (5.2.4)

Input :

- Array A of one float
- int $quantity$; number of possible gestures

Output :

- $gesture$, Gesturetype
- $isLeft$, bool which indicates if the Gesture was done with the left or right hand

```
valid = false;
maximum;
gesture = None;
if A[0] < quantity then
    gesture = A[0];
    isLeft = true
end
else
    gesture = A[0] mod quantity;
    isLeft = false;
end
return (gesture, isLeft)
```

Algorithmus 2 : Algorithmus zur Erkennung von Gesten bei nur einem Output (5.2.4)

führt hat oder nicht.
Im Praktikum „Stolperwege“ von Mehler, Abrami u. a. wurden bereits zwei verschiedene Vorgehensweisen zur Implementierung dieser Heuristiken evaluiert. Hierbei konnte festgestellt werden, dass es einfacher und genauer ist, wenn es eine Startpose gibt und ein Objekt aufgerufen wird, sobald diese Haltung eingenommen wird. Daraufhin muss der Nutzer mit dem Objekt interagieren. Der Vorteil ist die Führung des Nutzers und eine klare Grenzen zwischen unterschiedlichen Gesten (Mehler, Abrami u. a. 2018, vgl.).

Der zweite Ansatz aus dem genannten Praktikum, welcher positiv bewertet wurde, bildet hierbei ebenfalls die Grundlage. Im Algorithmus (3) sieht man, dass als erstes eine Startpose definiert werden muss. Wenn man sich als Beispiel die Gleich-Geste (4.1.3) anschaut, wäre die Startpose zwei offene Handflächen. Dies ist natürlich nicht die einzige Bedingung, da aktuell die Hände nicht miteinander in Relation stehen, wie es bei einem Gleich sein soll. Deswegen muss man weitere Bedingungen in die Startpose definieren, beispielsweise, dass die Hände innerhalb eines gewissen Bereichs horizontal sein müssen. Dieser Bereich wird hauptsächlich durch Tests ermittelt und angepasst. Aber nicht alle Merkmale müssen direkt am Anfang zwangsläufig eingeschränkt und abgefragt werden. Beim Beispiel von der Gleich-Geste wäre dies der Abstand zwischen den Händen. Dieser ist für die Geste nur bedingt relevant.
Sobald alle Bedingungen der Startpose (Algorithmus 3, Z. 3) erfüllt sind, wird das dazugehörige Objekt geladen. Dann müssen die tiefergehenden Bedingungen, wie der Abstand zwischen den Händen erfüllt werden. In dieser Phase werden auch die Bereiche der Bedingungen der Startpose eingeengt, da der Nutzer seine Hände der Form des Objekts anpassen muss. Da der Nutzer eine visuelle Referenz hat, kann er dies leichter, als wenn er kein Feedback bekommt. Die Evaluierung dazu erfolgte in Mehler, Abrami u. a. (2018, S. 97) beim Vergleich der Ansätze mit und ohne Objekt als Feedback.

Im Grunde lassen sich mehrere Nachteile bei diesem Ansatz erkennen. Erstens benötigt man zwangsläufig eine Startpose. Diese sollte sich von denen der anderen Gesten unterscheiden, damit genau ein Objekt geladen wird und der Nutzer nicht plötzlich mehrere Objekte sieht. Auch bei wenigen Gesten kann das schon zum Problem werden, falls diese ähnlich aussehen. Dann müssen mehr Bedingungen in der Startpose abfragt werden und man würde das Objekt eventuell nicht mehr benötigen. Damit folgt man dann wieder dem Vektoren basierten Ansatz von Mehler, Abrami u. a. (2018, S. 96).
Ein weiteres Problem, welches zeitgleich mit der Einführung des Neuronalen Netz gelöst wurde, ist die Trennung zwischen Eingabegerät und Geste. Die Daten auf denen der Algorithmus (3) arbeitet, stammen direkt von den genutzten Handschuhen. Dies setzt natürlich voraus, dass jeder, der dieses Programm nutzen möchte, die selben Handschuhe braucht. Dieser Ansatz wäre folglich nicht optimal gewesen und hätte in jedem Fall überarbeitet werden müssen.

Input : S, Handdata from Hi5 Gloves
Output :

- $valid$, bool which is true if the gesture was recognized
- $isLeft$, bool which indicates if the Gesture was done with the left or right hand

```
valid = false;
GestureObject = ObjectOfGesture;
if (Startpose_Attributes(S)) then
    Load(GestureObject)
end
if (GestureObject.RestrictionValid) then
    valid = true;
    Set(isLeft); //true/false depending if the gesturewas done with the left hand or not
end
return (valid, isLeft)
```

Algorithmus 3 : Zugrundeliegender Algorithmus für die Gesten aus dem ersten Ansatz (5.3)

6. Evaluation

6.1. Modellerzeugung

Nachdem man das System fertig implementiert hat, muss es noch trainiert werden, damit das RNN gezielt die Gesten erkennt. Würde dies nicht gemacht werden, dann würde das neuronale Netz lediglich raten. Damit man ein gutes Modell bekommt, müssen möglichst viele verschiedene Testpersonen die Gesten vormachen bzw. die Gesten müssen oft wiederholt werden. Dies ist wichtig, da jedes Mal die Geste leicht anders ausgeführt wird. Dies bezieht sich vor allem auf die Vektoren. Da sie ca. fünf Nachkommastellen haben, wird die Bewegung für den Computer nie gleich ausgeführt werden. Zwar ist die Geste grundlegend definiert, allerdings gibt es bestimmte Parameter die relevanter sind als andere. Bei der Gleich-Geste (4.1.3) sind die beiden Richtungsvektoren von jedem Finger relevant. Damit wird ein ausgestreckter Finger dargestellt. Weiterhin ist relevant, dass beide Hände ungefähr horizontal ausgerichtet sind. Allerdings ist der Abstand zwischen den Händen nicht zwangsläufig ein Kriterium, das ein Gleich von einer anderen Geste unterscheidet. Der Abstand ist also in dem Beispiel ein Parameter, der durch das Neuronale Netz als nicht wichtig eingestuft werden soll. Um das Modell entsprechend zu erzeugen, muss man die Geste in mehreren Abständen vormachen, damit diese generalisiert wird und das Gewicht für den Abstand klein und das für die Fingerrichtungen groß wird.

6.1.1. Imitation Learning

Der erste Ansatz war rein auf das *Imitation Learning* (2.3.4) beschränkt. Hierbei wurde ein Algorithmus zum Trainieren (2) genutzt. Nach einer Trainingszeit von ca. 20 bis 25 Minuten für eine Geste, hat sich gezeigt, dass sich das System leicht verbessert hat, aber trotzdem noch viel zu oft rät. Der große Nachteil am reinen *Imitation Learning* ist die Zeit, die aufgewendet werden muss, bis akzeptable Ergebnisse vorliegen. Da man in Echtzeit trainieren muss, dauert es wesentlich länger, bis das RNN viele Zyklen durchgegangen ist und so viele Daten vorliegen, dass ein gutes Modell entsteht. Der Vorteil besteht aber darin, dass man vergleichsweise zu künstlichen Daten (6.1.2) viel mehr verschiedene Daten einliest und das RNN besser generalisiert werden kann.

6.1.2. Künstliche Testdaten

Da das Lernen in Echtzeit sehr lange dauert (6.1.1), wurde das System so überarbeitet, dass die Möglichkeit besteht, Daten der Hände aufzuzeichnen und zu einem späteren Zeitpunkt ins Netz zu laden. Damit hat man die Möglichkeit das Netz zum Trainingszeitpunkt schneller als in Echtzeit laufen zu lassen. Ein weiterer Vorteil besteht darin, dass man schneller verschiedene Modelle berechnen lassen kann, da die Daten der Hände wiederverwendet werden können. Dadurch ist das System leicht erweiter- und änderbar.

Insgesamt wurden Gesten von fünf unterschiedlichen Personen aufgezeichnet. Nachdem diese vorlagen, wurde daraus ein Modell erstellt, mit dem es möglich ist, die Gesten zu erkennen und schlussendlich mittels Gesten zu Schreiben.

6.2. Programmevaluation

6.2.1. Übersicht

Die Evaluation soll zeigen, wie gut die Erkennung der Gesten mit den Handschuhen ist. Zum einen soll getestet werden, ob die Handschuhe zur Gestenerkennung genutzt werden können. Zum anderen soll das System zur Erkennung der Gesten, das Neuronale Netz, überprüft werden. Allerdings sind das nicht die beiden einzigen Faktoren, die das Gesamtsystem beeinflussen und sich ändern lassen. Die Gesten sind ebenfalls ein zentraler Bestandteil des System, weshalb diese genauso evaluiert werden müssen. Hierbei steht vor allem der Aspekt der Nutzerfreundlichkeit und Verständlichkeit im Vordergrund.

6.2.2. Aufbau

Zur Validierung wurde eine Szene geschaffen, die es ermöglicht, alle Komponenten zu überprüfen. Die Idee hierbei liegt darin, dass der Nutzer einen Satz angezeigt bekommt, den er mit Hilfe der Gesten nachstellt. Dafür gibt es zwei verschiedene Schwierigkeitsstufen. In der ersten müssen Beziehungen zwischen zwei Objekten hergestellt werden. Ein Beispielsatz lautet: „Der Sessel ist größer als der Stuhl“. Hierbei muss der Nutzer erst auf den Sessel zeigen, danach die Größer-Geste (4.1.4) ausführen und als letztes auf den Stuhl zeigen. In der zweiten Stufe werden drei verschiedene Objekte[1] mit zwei Gesten verknüpft. Da der Fokus mehr auf der Evaluation von den unterschiedlichen Schwierigkeitsstufen liegt, werden nicht alle Gesten betrachtet, sondern nur acht von 14 Stück.

Zur Evaluation steht der Nutzer in einem virtuellen Raum, in dem alle Objekte in zwei Reihen vor ihm aufgestellt sind. Oberhalb der Objekte sind drei bzw. fünf Eingabefelder, die es ihm ermöglichen Eingaben zu tätigen. In der Mitte der Blickrichtung des Nutzers befindet sich ein „Fokuspunkt“, dargestellt als kleiner Punkt. Sobald er ihn auf ein Eingabefeld richtet, kann er Eingaben tätigen. Dies ist nötig, da das Neuronale Netz sich in jedem Schritt für eine Geste entscheidet und bei einfachen Handbewegungen ungewollt Gesten erkannt werden können. Da immer zwei Objekte durch die Gesten verknüpft werden, sind die Felder so konzipiert, dass immer abwechselnd ein Objekt und eine Geste eingegeben wird. Des weiteren sieht der Nutzer vor sich eine Texttafel (siehe Abbildung 6.1), in welcher der Satz, den er nachbilden soll, steht. Rechts und links neben der Anzeige für den Satz gibt es jeweils einen Pfeil, mit dem man durch die Sätze blättern kann. Damit hat der Nutzer die Möglichkeit einen Satz erneut zu bearbeiten, bzw. zu überspringen. Ganz rechts befindet sich ein Button zum Evaluieren. Nur über diesen Button lässt sich der nachgebildete Satz mit dem vorgegebenen vergleichen. Die automatische Überprüfung wurde abgeschaltet, damit der Nutzer den Satz abschließend überprüfen kann. Dadurch sollen Fehler bei der Eingabe minimiert werden.

[1] Nach der Idee von Abrami, G.

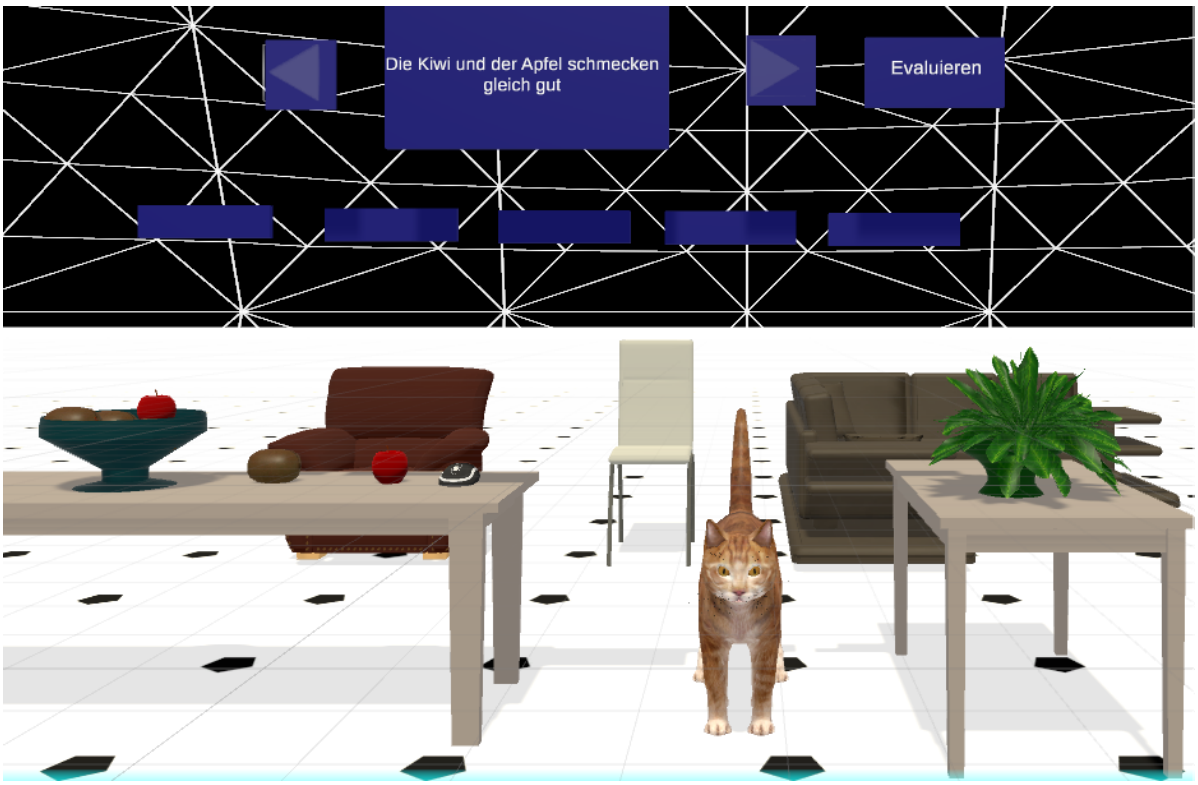


Abbildung 6.1.: Raum zur Evaluation aus der Sicht des Nutzers

6.2.3. Beispielhafter Durchgang

Die Evaluation besteht aus fünf verschiedenen Sätzen, davon sind drei einfachere und zwei komplexe Sätze zu bearbeiten. Sobald der Nutzer startet, wird die Gesamtzeit gemessen. Außerdem wird die Bearbeitungszeit für jeden einzelnen Satz gestoppt. Der Nutzer bekommt nun einen Satz angezeigt und muss nacheinander die Eingabefelder mit Daten befüllen. Beim Satz „Der Sessel ist größer als der Stuhl“ (siehe Abbildung 6.2) muss als erstes auf den Sessel gezeigt und dann in das erste Feld geschaut werden. Dann wird die Eingabe übernommen. Sobald er aus dem Feld raus schaut, wird die Eingabe gesichert und kann erst durch erneutes hineinschauen überarbeitet werden. Analog funktioniert das für das dritte Feld mit dem Objekt „Stuhl“. Nun fehlt nur noch die Eingabe der Geste in das mittlere Feld. Diese wird, genau wie bei den Eingabefeldern der Objekte, nur gespeichert, wenn man in das Feld schaut und dann die Geste macht. In diesem Fall (siehe Abbildung 6.2) wird eine Größer-Geste erwartet. Möchte der Nutzer nun seine Eingabe überprüfen, muss er auf Evaluieren zeigen. Er bekommt für jedes Feld ein eigenes Feedback. Dabei färbt sich das Feld grün, wenn die Eingabe richtig ist und rot, sobald ein falscher Wert erkannt wurde. Danach wird automatisch der nächste Satz angezeigt.

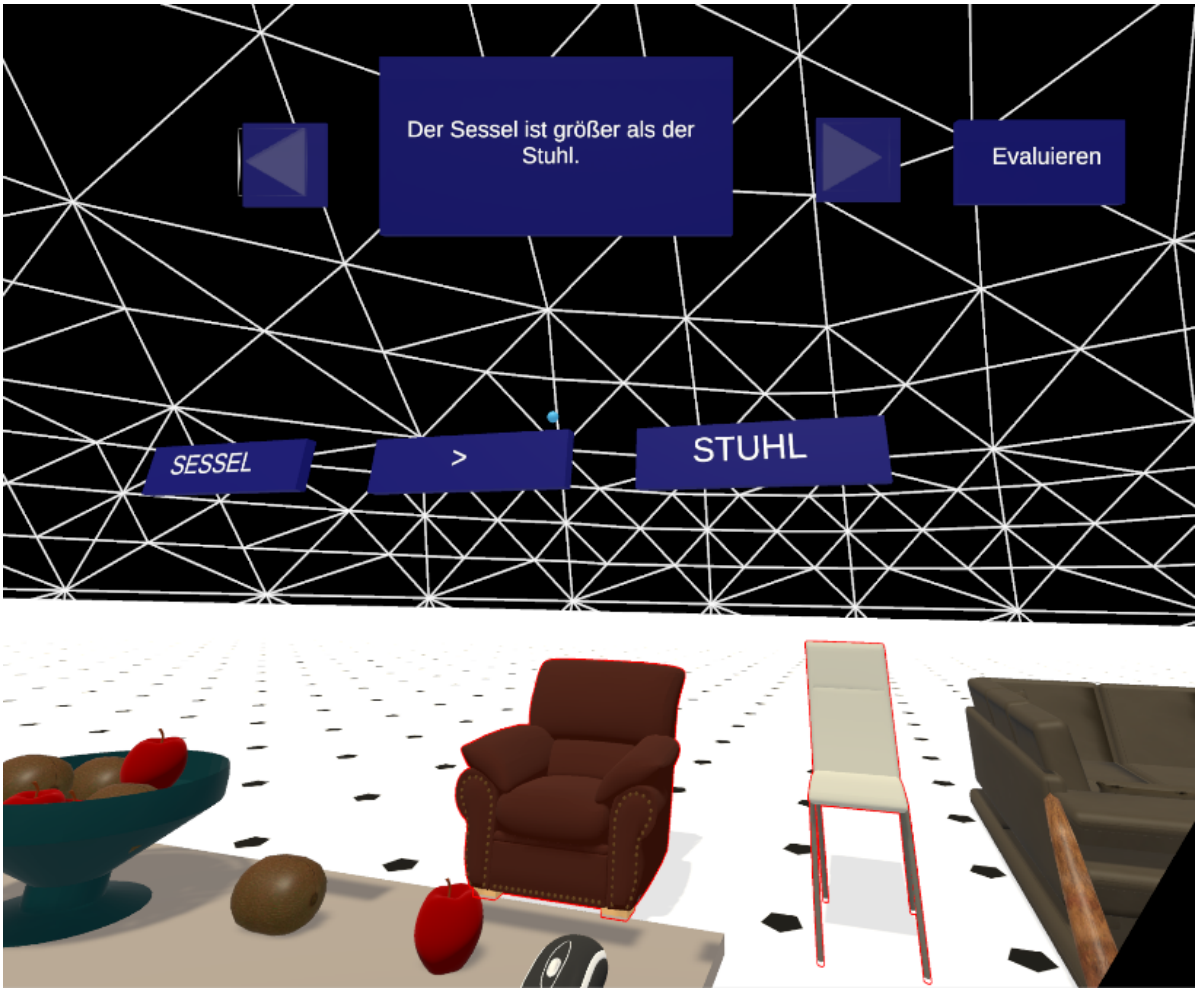


Abbildung 6.2.: Darstellung eines Satzes aus drei Bestandteilen

Abbildung 6.3.: Darstellung eines Satzes aus fünf Bestandteilen

6.2.4. Ergebnisse

An der Evaluation haben insgesamt 12 Testpersonen teilgenommen. Zwar sind dadurch die Ergebnisse statistisch nicht signifikant, es lassen sich jedoch Tendenzen erkennen, ob es Sätze gibt, die leichter darzustellen sind als andere. Gemessen wurden drei verschiedene Parameter. Diese sind die Zeit pro Satz und die Überprüfung, ob der Nutzer die geforderten Gesten/Objekte eingegeben hat.

Insgesamt wurden von den 92 bearbeiteten Sätzen 70% korrekt wiedergegeben. Allerdings gibt es zwei Sätze, die in weniger als der Hälfte der Fälle erkannt wurden („Die Kiwi und der Apfel schmecken gleich gut." und „Der Apfel und die Kiwi sind verschieden.") Hierbei wurde eine Gleich- (4.1.3) und eine Ungleich-Geste (4.1.6) verlangt. Wenn man sich die Erkennungsraten des ersten Satzes genauer anschaut (siehe Abbildung 6.5), sieht man, dass die einzelnen Erkennungsraten von Objekten und Gesten mit 40% höher liegen, als die Gesamterkennungsrate (20%). Dies liegt daran, dass der Satz nur in einem von fünf Fällen richtig bearbeitet wurde und in zwei weiteren Fällen jeweils nur eine Kategorie richtig war. Dadurch fällt das Gesamtergebnis schlechter als die Einzelergebnisse aus.
Wenn man nur die Gestenerkennung betrachtet, stellt man fest, dass in 77,68% der Fälle die Geste korrekt erkannt wurde. Der Median liegt etwas höher bei 81,82%. Nun ist es natürlich wichtig zu wissen, ob es Gesten gibt, die tendenziell schlechter erkannt werden als andere. Schaut man sich die Sätze mit den drei schlechtesten Erkennungsraten an, stellt man fest, dass der Satz mit der Gleich-Geste vorkommt. Bei den beiden anderen Sätzen wurde die Ungleich-Geste schlecht erkannt. Dass die Gesten schlecht erkannt werden, kann verschiedene Gründe haben. Einerseits muss man sich die Frage stellen, ob der Nutzer die Geste richtig ausgeführt hat oder zu weit von der geforderten Geste abgewichen ist. Interessanterweise sind gerade zwei Gesten schlecht erkannt worden, die statisch sind. Ein weiterer Grund dafür kann das Modell des Neuronalen Netzes sein. Am wahrscheinlichsten liegt das Problem in den Testdaten. Da diese Gesten statisch sind, gibt es weniger unterschiedliche Daten, die man an das Netz übergibt. Wenn eine Person eine statische Geste zehn mal ausführen soll, wird sie dies immer gleich machen. Eine Geste, bei der man sich bewegt, wird aber immer unterschiedlich sein bzw. die Unterschiede sind wesentlich größer. Daher bräuchte man bei statischen Gesten wesentlich mehr Daten von verschiedenen Personen als bei Bewegungsgesten.
Die Gesten, die am besten erkannt wurden, sind die Größer- bzw. Kleiner-Geste (4.1.4), sowie die Part-whole-Geste (4.1.13). Die Erkenungsraten liegen bei mehr als 92%.

Das Auswählen von Objekten hat sehr gut funktioniert. Die Durchschnittliche Erkennungsrate liegt bei 81,82%. Nur zwei Konstellationen haben Probleme bereitet. Zum einen waren das die Sätze, in der der Apfel und die Kiwi vorkamen, zum anderen die beiden Sätze in denen die Pflanze zum Gesamtsystem des Tisches gehört (Pflanze Part-whole Tisch). Bei der ersten Konstellation kann das Problem darin liegen, dass die Kiwi, als Objekt, zu nah an den anderen Gegenständen angeordnet wurde. Bei der Pflanze und dem Tisch liegt das Problem wahrscheinlich an der Interpretation des Satzes. Die Part-whole-Geste ist so konzipiert, dass man von einer Teilmenge auf ein Ganzes schließt. Folglich muss die Teilmenge immer

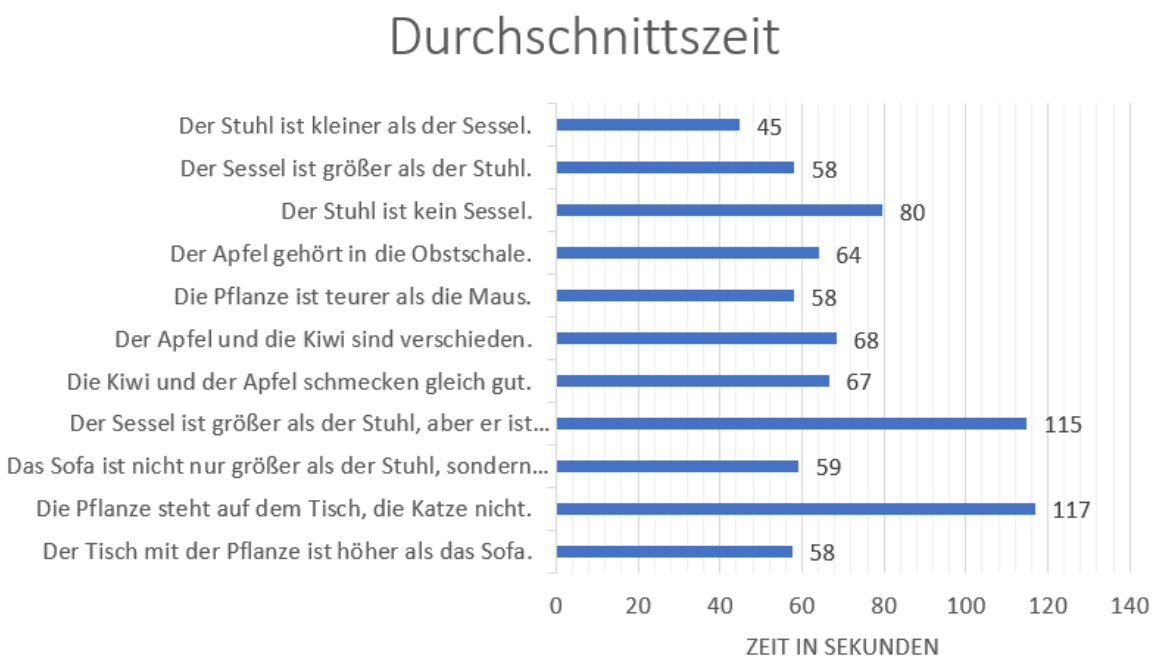


Abbildung 6.4.: Durchschnittliche Zeit, die der Nutzer gebraucht hat

links von der Geste stehen. Beim Satz „Der Tisch mit der Pflanze […]“ wird der Tisch zu erst genannt und dementsprechend wurde der Tisch links von der Geste eingesetzt und nicht rechts.

Wenn man die Zeit betrachtet (siehe Abbildung 6.4), die die Nutzer durchschnittlich für einen Satz gebraucht haben, sieht man erwartungsgemäß, dass kurze Sätze schneller dargestellt werden können. Allerdings hängt die Geschwindigkeit bei den komplexen Sätzen auch stark vom darzustellenden Satz ab. Der schnellste komplexe Satz wurde in durchschnittlich 57 Sekunden bearbeitet. Für zwei andere Sätze wurden ca. 115 Sekunden benötigt. Bei den einfachen Sätzen ist dieser Unterschied nicht so gravierend. Der große Zeitunterschied bei den komplexen Sätzen kann mehrere Ursachen haben. Einerseits hat die Anordnung der Gegenstände bei drei Objekten einen größeren Einfluss. Wenn man sich von links nach rechts wenden muss, um alle Objekte auszuwählen, braucht man natürlich länger, als bei Gegenständen, die nebeneinander liegen. Außerdem wurde bereits angesprochen, dass manche Gesten besser erkannt werden als andere. Auch dies kann einen zeitlichen Einfluss haben.

Der letzte Teil der Evaluation bezieht sich auf den Fragebogen (B.2), den die Teilnehmer ausfüllen sollten. Dabei wurden vier verschiedene Fragen gestellt. Diese mussten dann auf einer Skala von eins bis sieben bewertet werden, wobei eins bedeutete, dass man der Frage bzw. Aussage nicht zustimmte. Insgesamt wurde das System als gut bewertet (siehe Abbildung 6.6). Die erste Frage „The gestural writing tool’s capabilities meet my requirements.“ wurde durchschnittlich mit 5,92 bewertet und zeigt, dass die Nutzer sich ein System mit den

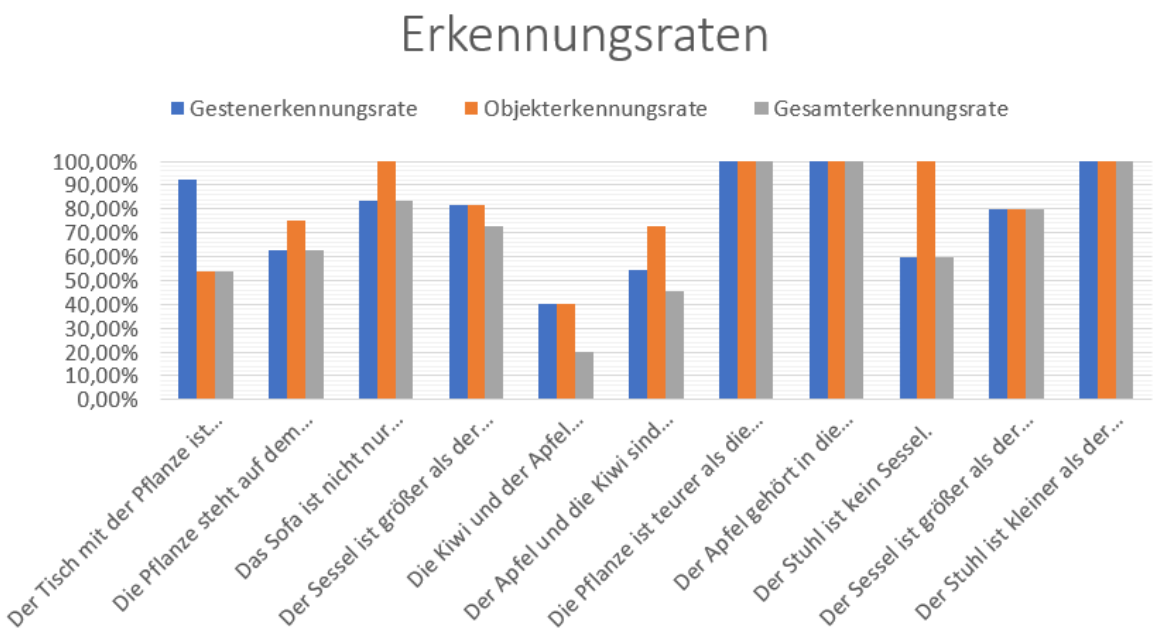


Abbildung 6.5.: Auswertung der Erkennungsraten der einzelnen Sätze

entsprechenden Eigenschaften und Möglichkeiten vorgestellt haben. Des weiteren sind sie mit der Nutzbarkeit zufrieden und haben das Level an Frustration eher niedrig eingeschätzt. Auch die Frage, ob das System einfach zu Nutzen ist, wurde überwiegend positiv bewertet und hat einen Durchschnittswert von 5,08 von 7 Punkten. Die Antworten der ersten drei Fragen sind ungefähr wie eine Glockenkurve verteilt. Diese Beobachtung lässt sich nicht auf die letzte Frage übertragen. Die Antworten auf die Frage, ob man zu viel Zeit zum Schreiben mittels Gesten aufwenden muss, teilt sich in zwei Gruppen. Die erste mit acht Teilnehmern, stimmt der Aussage nicht zu. Sie denken, dass sie nicht zu lange brauchen. Die zweite Häufung an Antworten, diese vier gehören zur zweiten Gruppe, liegt bei fünf. Es gibt folglich eine Gruppe, die das System gut nutzen kann und eine zweite, die es zeitlich betrachtet, nicht bevorzugt Gesten als Eingabe zu verwenden. Dies kann verschiedene Ursachen haben. Einerseits braucht es eine gewisse Zeit, bis man die Gesten verinnerlicht hat und man diese, ohne groß über die Bewegung nachzudenken, macht. Dies ist vergleichbar mit dem Lernen vom 10-Finger Schreiben mit der Tastatur. Anfangs kostet es noch viel Zeit, aber danach kennt man alle Positionen der Tasten. Hierfür reicht natürlich die Kürze der Evaluation nicht aus, bis man alles gelernt hat. Gerade auch in diesem Fall vergleicht man wahrscheinlich das System mit einer Tastatur, die aktuell auf jeden Fall noch schneller ist. Ein weiterer Grund der Gruppe an schlechten Bewertungen kann durch das Modell zustande kommen. Da bereits von zwei Gesten gesprochen wurde (Gleich- 4.1.3 und Ungleich-Geste 4.1.6), die bisher schlechter erkannt werden, ist das Modell ebenfalls ein Grund. Wenn eine Geste nicht gut erkannt wird, muss sie öfters vom Nutzer gemacht werden, bis sie erkannt wird. Folglich braucht man länger, um einen Satz zu schreiben.

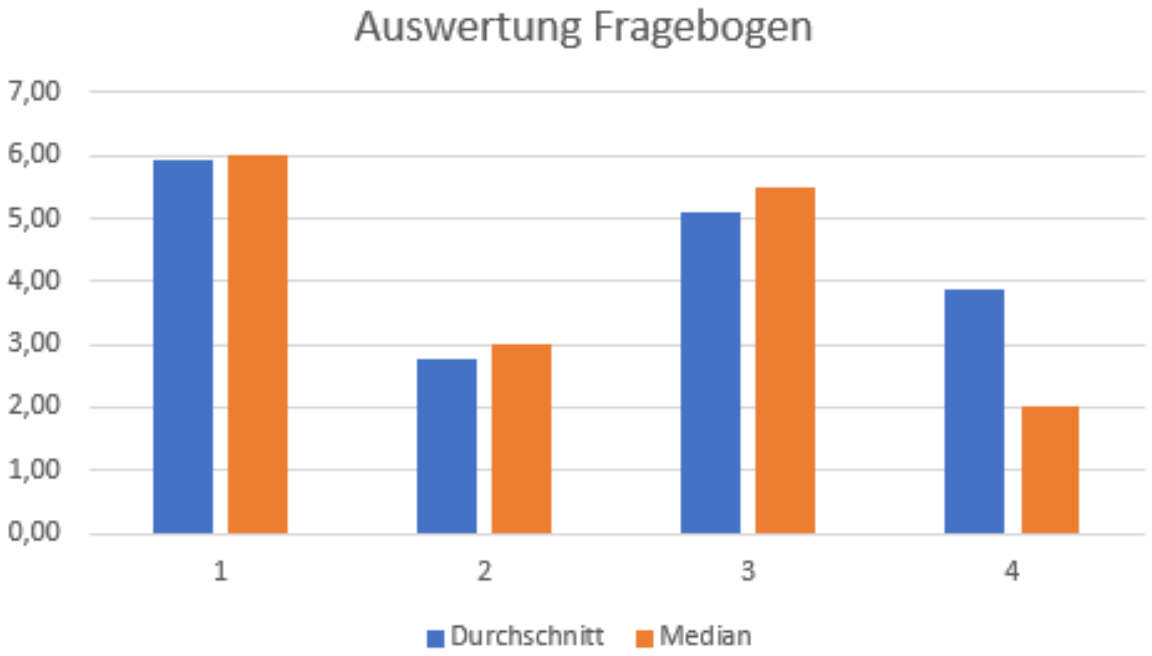


Abbildung 6.6.: Auswertung des Fragebogens

7. Ausblick

7.1. Fazit

Damit die Aufgabenstellung (1.1) gelöst werden kann, wurde eine andere Herangehensweise in der VR gewählt als vorher im Praktikum (Mehler, Abrami u. a. 2018). Die Nutzung eines Neuronalen Netzes bringt viele Vorteile mit sich, wenn man ihn mit dem vorher genutzten Interface vergleicht (5.3). Für den Nutzer ist es einfacher, weitere Gesten hinzuzufügen oder Gesten zu ändern, als zuvor. Des weiteren spart sich der Entwickler Arbeit, da die Klassifizierung einer Geste nicht mehr durch Werte festgelegt bzw. ermittelt werden muss, sondern das RNN diese Aufgabe übernimmt. Trotzdem gibt ein Neuronales Netz nicht immer auf Anhieb die richtige Entscheidung. Wichtig ist, dass eine Funktion implementiert wurde, die die Ausgaben gefiltert hat. Dadurch wurde die Anzahl an Fehlentscheidungen reduziert (5.2.4). In den Tests hat sich bestätigt, dass dadurch eine Erkennungsrate der Gesten von 78% erreicht werden kann.

7.2. Offene Probleme

Ein bisher ungelöstes Problem besteht darin, zu erkennen, wann eine Geste aufhört. Das Neuronale Netz wird immer eine Antwort ausgeben und man wird dadurch in jedem Schritt eine Geste erhalten. Wenn diese Problematik gelöst wäre, wäre eine wesentlich genauere Vorhersage der Geste möglich.

7.3. Mögliche Weiterentwicklungen

- *Hinzufügen weiterer Hardware*
  Aktuell unterstützt das System nur die Hi5 Handschuhe von Noitom International Inc. (Version 2018). Es könnten jedoch weitere Eingabemöglichkeiten, die die Hände erkennen, eingebunden werden. Dies könnten einerseits weitere Handschuhe sein, aber andererseits auch VR-Eingaben, die auf anderen Techniken beruhen. Ein Beispiel wäre die LeapMotion (LeapMotion Inc. Version 2017), welche auf Infrarotsensoren basiert.

- *Erhöhung der Anzahl an Gesten*
  Außerdem könnten neue Gesten hinzugefügt werden (A.1), durch die das Spektrum an Darstellungs- und Ausdrucksmöglichkeiten erweitert werden könnte.Dann könnte ein neues Modell mit dem RNN erstellt und genutzt werden.

- *Unterteilung der Anwendungsbereiche*
  Möchte man einen gänzlich anderen Bereich an Wörtern nutzen, kann man den Gesten

neue Aktionen und Ausgaben zuweisen (siehe dazu das Beispiel Tauchen, 5.1). Entweder man fügt eine neue Geste hinzu oder man ändert nur die Ausgabe einer Geste. Man könnte neben den unterschiedlichen Modellen ebenfalls ein Interface entwickeln, das es leichter macht, zwischen diesen Einstellungen zu wechseln.

A. Entwicklerinformationen

Im Folgenden wird kurz die Struktur erklärt und gezeigt, welche Schritte nötig sind, damit man eine neue Geste hinzufügen kann. Möchte man neue Hardware hinzufügen, muss das entsprechende Skript von *HandDefinitions* erben. Dort wird das Handmodell definiert. Die Evaluationsszene zum Schreiben findet man unter dem Namen *Evaluation*. Es gibt Einstellungen, die man anpassen kann. Ersten kann man im *RecognitionBrain* unter *Graph Model* das vorhandene Modell ändern. Dadurch kann es nötig werden, dass man im Brain die *Space Size* ändern muss, da dieses eine andere Zahl an Inputs benötigt (5.2.3). Der *Agent* muss daraufhin auf die selbe Anzahl an Inputs eingestellt werden. Dies ist mit dem Auswahlkästchen *Complex Model* möglich. Ist es angewählt, werden 67 Vektoren erwartet, ansonsten 17.

A.1. Hinzufügen einer neuen Geste

Die Grundlage besteht darin, im *Enum GestureHelper.GestureType* den neuen Typ anzulegen. Dies ist nur ein Name und definiert die Geste. Im Normalfall möchte man künstliche Testdaten nutzen und dementsprechend aufnehmen. Dies ist mit der Szene *GestureScene_NN* in Unity möglich.
Das Skript *Hi5 Definitions* verwaltet alles. Über den Unity Editor kann man es aktivieren. Man hat die Möglichkeit eine *PersonenID* einzugeben, damit die Daten eindeutig zugeordnet werden können. Des weiteren muss man einige Einstellungen anpassen. Darunter fällt die Geste, die man aufnehmen möchte. Falls es eine einhändige Geste ist, muss jeweils *LeftHand* angeklickt werden, je nachdem ob die Geste links oder rechts ausgeführt wird. Schlussendlich setzt man den Haken bei *Record Gesture*, damit die Daten aufgezeichnet werden. Es gibt den Button *Start Recording*. Dieser dient dazu, alte Daten, die nach dem letzten Speichern einer Geste aufgezeichnet wurden, zu verwerfen. Mittels *Save Gesture* werden alle Daten seit dem letzten Speichern bzw. Reset in einer Datei gespeichert.

Um ein Modell zu erstellen, müssen die Daten eingelesen werden. Für neue Gesten wird die Datei *ArtificialRecognition* angepasst. In der Start-Funktion wird ein weiterer Fall hinzugefügt, mit der die Daten in eine Liste gepackt werden. Außerdem wird ein weiteres *case* in der Funktion *ListToGesture* eingefügt, damit die Daten der Liste als Daten für das Netz zur Verfügung stehen.
Die letzte Änderung wird im Skript *EvaluationHelper* vorgenommen. Die Funktion *StringToGesture* übernimmt die Konvertierung einer eingelesenen Geste aus einer Textdatei. Hier wird ein weiterer Fall für die entsprechende Geste erstellt. Des weiteren muss man die Konvertierung für den Output vornehmen. Dazu fügt man einen weiteren Fall in der Methode *GestureMappingInit* ein, damit der Nutzer das entsprechende Zeichen sieht.
Nun ist man bereit für das Training des Neuronalen Netztes. Dafür muss in der Unity Szene das entsprechende Brain ausgewählt werden. Am besten eignet sich das *AIBrain* mit den

Agenten 8 bis 16. Man startet das Training mit dem Befehl *learn.py - - train - - run-id=*name**. Weiterführende Informationen finden sich bei Unity Technologies (2018a).

B. Evaluation

B.1. Evaluierte Sätze

Sätze mit zwei Objekten:

- Der Stuhl ist kleiner als der Sessel.

  Kleiner 4.1.4

- Die Pflanze ist teurer als die Maus.

  Größer 4.1.4

- Der Apfel gehört in die Obstschale.

  Part-whole 4.1.13

- Der Sessel ist größer als der Stuhl.

  Größer 4.1.4

- Der Stuhl ist kein Sessel.

  Ungleich 4.1.6

- Der Apfel und die Kiwi sind verschieden.

  Ungleich 4.1.6

- Die Kiwi und der Apfel schmecken gleich gut.

  Gleich 4.1.3

Sätze mit drei Objekten:

- Das Sofa ist nicht nur größer als der Stuhl, sondern auch größer als der Sessel

  Größer 4.1.4

  Größer 4.1.4

- Der Sessel ist größer als der Stuhl, aber er ist kleiner als das Sofa.

  Größer 4.1.4

  Kleiner 4.1.4

- Die Pflanze steht auf dem Tisch, die Katze nicht.

  Part-whole 4.1.13

  Negation 4.1.8

- Der Tisch mit der Pflanze ist höher als das Sofa.

  Part-whole 4.1.13

  Größer 4.1.4

B.2. Fragebogen

Vincent Kühn
E-Mail: kuehn-vincent@t-online.de

Nutzerstudie zum Schreiben mithilfe von Gesten

UMUX-Fragebogen

VP ___

Zur Auswertung des Gestenmodells wird die *Usability Metric for User Experience* (UMUX, Finstad 2010) verwendet. UMUX umfasst die folgenden vier Fragen, die Sie bitte per Ankreuzen jeweils eines Feldes beantworten.

1. The gestural writing tool's capabilities meet my requirements.

1	2	3	4	5	6	7
Strongly Disagree						Strongly Agree

2. Using the gestural writing tool is a frustrating experience.

1	2	3	4	5	6	7
Strongly Disagree						Strongly Agree

3. The gestural writing tool is easy to use.

1	2	3	4	5	6	7
Strongly Disagree						Strongly Agree

[bitte wenden]

4. I have to spend too much time writting sentences with the gestural writing tool.

1	2	3	4	5	6	7
Strongly Disagree						Strongly Agree

Literatur

Finstad, Kraig (2010). „The Usability Metric for User Experience“. In: *Interacting with Computers 22.5*, S. 323–327. DOI: http://dx.doi.org/10.1016/j.intcom.2010.04.004.

Literatur

Badelt, R. (2013). *Gesture-Based Computing. Potentiale und Auswirkungen auf Lehr- und Lernszenarien durch Verwendung von Gestensteuerung.* Master of Arts. URL: https://learninglab.uni-due.de/system/files/pruefungsarbeiten/gesturebasedcomputing_badelt_2013.pdf.

Fu-Chuang, C. (1990). *Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control.*

Ekman, P. und Friesen W. (1969). „The repertoire of nonverbal behavior: Categories, origins, usage, and coding". In: *Semiotica 1.*

Finstad, Kraig (2010). „The Usability Metric for User Experience". In: *Interacting with Computers 22.5*, S. 323–327. DOI: http://dx.doi.org/10.1016/j.intcom.2010.04.004.

Ghosh, D. und S. Ari (2011). *A Static Hand Gesture Recognition Algorithm Using K-Mean Based Radial Basis Function Neural Network.*

Graves, A. (2012). *Supervised sequence labelling with recurrent neural networks.* Bd. 385. Springer.

LeapMotion Inc. (Version 2017). *Leap Motion Manual.* URL: https://www.leapmotion.com/.

McNeill, D. (1992). *Hand and mind: what gestures reveal about thought.*

Mehler, A., G. Abrami, A. Bender und V. Kühn (2018). *Stolperwege: Eine App zur Realisierung einer Public History of the Holocaust.* Goethe Universität Frankfurt. URL: http://stolperwege.hucompute.org.

Mehler, A., A. Lücking und G. Abrami (2014). „WikiNect: Image Schemata as a Basis of Gestural Writing for Kinetic Museum Wikis". In: *Universal Access in the Information Society*, S. 1–17. ISSN: 1615-5289. DOI: 10.1007/s10209-014-0386-8.

Microsoft Corporation (Version 2018). *Kinect Website.* URL: https://developer.microsoft.com/de-de/windows/kinect.

Murakami, K. und H. Taguchi (1991). *Gesture Recognition using Recurrent Neural Networks.*

Noitom International Inc. (Version 2018). *Hi5 Glove Manual.* URL: https://hi5vrglove.com/.

Recreational Scuba Training Council (2005). *Common Hand Signals for Recreational Scuba Diving.* URL: http://wrstc.com/downloads/12%20-%20Common%20Hand%20Signals.pdf.

Rey, G. und F. Beck (2018). *Neuronales Netz.* deutsch. URL: http://www.neuronalesnetz.de.

Riedmiller, M. und H. Braun (1993). *A Direct Adaptive Method for Faster Backpropagation Learning The RPROP Algorith.*

Risse, T. (2018). *Einführung in Neuronale Netze.* deutsch. Hochschule Bremen. URL: http://www.weblearn.hs-bremen.de/risse/RST/SS97/Synapse/Einfuehrung/Einfuehr.htm.

Unity Technologies (2018a). *Unity ML-Agents Toolkit Documentation.* URL: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Readme.md.

– (2018b). *Unity ML-Agents Toolkit Documentation.* URL: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md.

Weissmann, J. und R. Salomon (1999). *Gesture Recognition for Virtual Reality Applications Using Data Gloves and Neural Networks.*

Zhu, W. u. a., Hrsg. (2016). *Co-Occurrence Feature Learning for Skeleton Based Action Recognition Using Regularized Deep LSTM Networks.*