

Fernunterstützung und Zusammenarbeit mit 3D Punktwolken

Masterarbeit
von

Kai Westerkamp

An der Fakultät für Informatik
Institut für Anthropomatik und Robotik
Fraunhofer IOSB (IAD)

Erstgutachter:	Prof. Dr.-Ing. Rainer Stiefelhagen
Zweitgutachter:	Prof. Dr.-Ing. habil. Jurgen Beyerer
Betreuender Mitarbeiter:	M.Sc. Adrian Hoppe
Zweiter betreuender Mitarbeiter:	M.Sc. Sebastian Maier

Bearbeitungszeit: 01.06.2017 – 30.11.2017

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Zusammenfassung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Inhaltsverzeichnis

1. Einleitung	1
1.1. VR	1
1.1.1. vive	1
2. Stand der Technik	3
2.1. Section	3
2.1.1. VR AR Assistance	3
2.1.2. 3D Scan	3
2.2. Virtual Reality (VR)	3
2.2.1. Ungenauigkeiten im Lighthouse Tracking	3
3. 3D Aufnahmen mit Punktwolken	5
3.1. Frames aufnehmen und bereinigen	6
3.1.1. Aufnahme und Glättung	6
3.2. Zusammenfügen von Frames	7
3.2.1. Kalibrierung Kinect zu Vive	7
3.3. Ergebnisse	9
4. Speichern der Punktwolke mit 3D Tiles	11
4.1. 3D Tiles	11
4.1.1. glTF	11
4.1.2. Tileset und Tiles	12
4.2. Implementierung der 3D Tiles	14
5. Visualisierung	15
5.1. UE4 Rendering System	15
5.2. 3D Tiles laden und vorbereiten	15
5.3. Rendering	16
5.4. Ergebnisse	17
6. HoloLens	19
6.1. Unreal Engine 4 und HoloLens	19
6.2. HoloLens Implementierung	19
6.3. Kalibrierung	19
6.3.1. Kalibrierungsfehler	20
6.4. Ergebnisse	21
7. Evaluation	23
7.1. Versuchsaufbau	23
7.1.1. Punktwolken	24
7.2. Versuchsablauf	24
7.3. Statistische Verfahren	25
7.4. Probanden und Teams	26

7.5.	Ergebnisse	27
7.5.1.	Vorbereitung des Experten	27
7.5.2.	Kommunikation des gesuchten Steins	30
7.5.2.1.	Videostream	30
7.5.2.2.	VR/AR Strahl	30
7.5.2.3.	Sprachliche Kommunikation	30
7.5.3.	Fehleranzahl	32
7.5.3.1.	Timings	32
7.5.4.	NASA TLX und UEQ	34
7.5.5.	Unabhängigkeit	38
7.5.6.	Allgemeines	38
8.	Fazit und Ausblick	41
A.	3D Tile JSON	43
B.	Evaluations Fragebogen	44
C.	Tabellen	44
D.	Alternative Plots	44
	Literaturverzeichnis	49

1. Einleitung

(TODO)

ToDo

1.1. VR

1.1.1. vive

2. Stand der Technik

related

2.1. Section

paper

2.1.1. VR AR Assistance

[OES⁺15] Virtuelle Proxys

2.1.2. 3D Scan

<https://www.microsoft.com/de-de/store/p/3d-scan/9nblggh68pmc>

<http://www.kscan3d.com/>

(<https://steamcommunity.com/app/507090/discussions/0/144513248276793628/>)

2.2. Virtual Reality (VR)

2.2.1. Ungenauigkeiten im Lighthouse Tracking

Ein großes Problem sind Ungenauigkeiten im Lighthouse Tracking.

Noise in der Ruhelage 0.3mm [lig] RMS (Rooted mean squared, quadratisches Mittel)
1.9mm [lig]

mein Jitter bild

Im Paper [NLL17] wurden signifikante Fehler nach Tracking Abbrüchen festgestellt. bsi zu 150cm

2m = 1,98m

Tracker tracking (bilder

VR Definition

imersion reale Welt ausblenden

Geräte, handy cave, hmd vive,oculus

Lighthous Trackign system <https://dl.acm.org/citation.cfm?id=2996341> funktionsweise
GENUAIGKEIT eigen messugnen längenuntreue

AR

Assistancec Papers

Poitrn cloud generatiion

Technische Grundlagen Unereal Unity HTC Vive Vive Tracker Wireless Kit Hololens Ki-
nect

3. 3D Aufnahmen mit Punktwolken

Damit ein Experte assistierend zugeschaltet werden kann benötigt dieser Informationen zu dem problematischen Objekt. Diese Daten sollen anschließend in einer VR Umgebung dem Experten präsentiert werden, deshalb sollte die Aufnahme ein 3D Scan sein. Ziel der Arbeit war es eine Lösung für das Aufnehmen der benötigten 3D Daten zu finden die einfach Bedienbar ist und ohne Aufwendige Nachbearbeitung oder Berechnungen auskommt. Deshalb wurde sich für die Aufnahme und Visualisierung von 3D Punktwolken entschieden.

Als Sensor wurde die Kinect ausgewählt. Diese ist günstig, portabel und wird in der Forschung häufig als 3D Sensor eingesetzt. Die Kinect als Sensor bietet die Möglichkeit aus einzelnen Aufnahmen bestehend aus einem Farbbild und einem Tiefenbild eine Punktwolke aus der Perspektive der Kinect zu errechnen. Eine Aufnahme (ein Frame) beinhaltet aber nur alle Informationen die aus der Perspektive der Kamera sichtbar sind. Das beinhaltet zum einen die Flächen der nächsten Oberfläche. Dahinterliegende Geometrie wird verdeckt und ist aus einer Perspektive nicht sichtbar (siehe 3.1 (a)). Außerdem sind an Kanten meist nicht genügend Informationen in der Aufnahme enthalten sodass seitliche Flächen richtig in Punkte konvertiert werden können. In vielen Fällen führt das zu falschen und nicht existierenden Flächen der errechneten Punktwolke (siehe 3.1 (b)). Diese Informationen aus einer Aufnahme reichen für eine die Visualisierung in VR für den Experten nicht aus. Das Objekt sollte von allen Seiten gescannt werden damit der Experte frei entscheiden kann von welcher Seite er das Objekt bzw. die Punktwolke betrachtet.

Um eine Punktwolke zu erhalten die das gesamte Objekt abdeckt, werden mehrere Aufnahmen aus unterschiedlichen Perspektiven gemacht. Diese Aufnahmen müssen aber anschließend richtig zu einer großen Punktwolke zusammengefügt werden. Eine Möglichkeit 2 Aufnahmen zusammenzufügen ist über die Transformation zwischen den Kamerapositionen. Mit dieser Transformation lässt sich eine Punktwolke aus einem Frame in das Koordinatensystem einer anderen Aufnahme transformieren. Um diese Transformation zu errechnen gibt es einige Ansätze die aber meist rechenaufwändig und fehleranfällig sind. (ref **realted work**) In dieser Arbeit wurde die Kinect mit dem Lighthouse Tracking der HTC Vive verbunden. Das Trackingsystem liefert eine globale Position des Kinect Sensor und ermöglicht damit eine einfache Berechnung der relativen Transformation zwischen 2 Kameraperspektiven. Durch ein globales Tracking zur Aufnahmezeit entfallen nachträgliche Berechnungen um aus den einzelnen Aufnahmen die Kamerapositionen zu errechnen. Das spart Zeit und Rechenleistung und bietet somit eine einfache und schnelle Möglichkeit eine 3D Aufnahme von einem Objekt zu erstellen.

ToDo

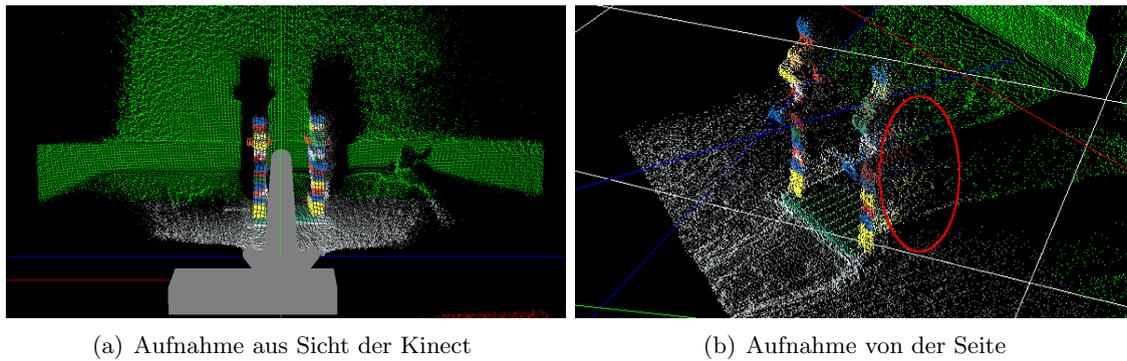


Abbildung 3.1.: Aufnahme der Kinect aus verschiedenen Perspektiven. In Bild (a) ist zu erkennen wie die Türme den Hintergrund verdecken. In Bild (b) sind falsche Punkte zu sehen, die durch die Rekonstruktion aus einem 2D Bild entstehen.

3.1. Frames aufnehmen und bereinigen

Im ersten Schritt wird ein Frame mit der Kinect aufgenommen. Die Rohdaten der Kinect sind verrauscht und deshalb wird das Tiefenbild geglättet. Anschließend wird das Tiefen- und Farbbild in 3D Punkte umgewandelt und unerwünschte Punkte verworfen.

3.1.1. Aufnahme und Glättung

Zum Aufnehmen einer Punktwolke aus einem Frame wird das Kinect SDK verwendet. Sowohl das Tiefenbild als auch das Farbbild kann man aus der API erhalten. Anschließend wird das Tiefenbild geglättet, um glattere Oberflächen in der Punktwolke zu erhalten. Hierfür braucht man einen Filter, der zwar die Flächen glättet, aber gleichzeitig die Objektkanten erhält. Ein Bilateral Filter erzielt den gewünschten Effekt, ist aber relativ rechenaufwändig. Im Paper [MCS14] wird hierfür ein Filter vorgestellt, den auch in dieser Ausarbeitung verwendet wurde. Hierbei wird zunächst das Bild mit einem Gauß-Filter geglättet. Dieser ist nicht kantenerhaltend. Deshalb wird das geglättete Bild anschließend mit dem Original verglichen. Bei zu starker Abweichung vom Original wird der Wert des Pixels auf das Original zurückgesetzt.

Nach der Glättung des Tiefenbildes wird dieses in eine Punktwolke umgewandelt. Hierfür wurde ebenfalls das Microsoft Kinect SDK verwendet, das alle benötigten Methoden bereitstellt.

Nach der Umwandlung werden noch weitere Punkte verworfen: Zunächst werden alle Punkte ohne zuordnungsfähige Farbe verworfen, um eine schönere Aufnahme zu erhalten. Der Farbsensor ist nicht an der gleichen Stelle der Kinect und deshalb kann es vorkommen, dass die Tiefenkamera Geometrie aufnimmt, die aus Sicht der Farbkamera verdenkt ist. Außerdem hat der Tiefensensor eine andere Auflösung und ein anderes Seitenverhältnis als die Farbkamera, sodass es am oberen und unteren Rand zu nicht farbigen Punkten kommt. Auch alle Punkte, die zu nah oder zu weit vom Sensor entfernt sind, werden nicht weiter betrachtet. Je weiter das Objekt entfernt ist, desto ungenauer werden die Aufnahmen. Im folgenden wurde ein Mindestabstand von 30cm und ein Maximalabstand von 90cm verwendet.

Als letztes werden alle Flächen, deren Oberflächennormale zu weit von dem Kameravektor abweicht (siehe Abb. 3.1 b). Diese Flächen entstehen durch die Umwandlung des 2D Tiefenbildes in eine 3D Punktwolke. Die benötigten Informationen fehlen an dieser Stelle und Punkte werden auf die Fläche zwischen Oberflächenobjekt und Hintergrund gesetzt.

Diese Ebene stimmt nicht mit der wirklichen Oberfläche überein und diese falschen Punkte müssen entfernt werden. Hierfür wird die Oberflächennormale verwendet. Die Normale wird aus dem Tiefenbild geschätzt.

$$\begin{aligned} dzX Axis &= depthImageAt[x + 1, y] - depthImageAt[x - 1, y] \\ dzY Axis &= depthImageAt[x, y + 1] - depthImageAt[x, y - 1] \\ Normale[x, y] &= Normalize(-dzX Axis, -dzY Axis, 1.0) \end{aligned} \quad (3.1)$$

Mit dem Skalarprodukt lässt sich der Winkel zwischen dem Kameravektor $(0, 0, 1)$ und der Normalen ausrechnen. Ein maximaler Winkel von 65° hat in den Tests ein gutes Ergebnis geliefert.

3.2. Zusammenfügen von Frames

Für das Zusammenfügen der einzelnen Frames werden die Punktwolke und 3 Transformationen benötigt. Die Frames der Kinect und die daraus resultierenden Punktwolken haben ihren Ursprung im Tiefensensor. Die Transformation *transformControllerToKinect* zwischen dem Koordinatensystem des Controllers und der Kinect wurde bestimmt und die globale Transformation des Controllers *transformController* ist in der OpenVR API abfragbar. Die Transformation der lokalen Punktwolke in ein globale ist mit diesen beiden Transformationen möglich.

$$globalPosition = transformController * transformControllerToKinect * localPosition \quad (3.2)$$

(kürzer Namen?)

ToDo

3.2.1. Kalibrierung Kinect zu Vive

Eine Transformation ist die zwischen dem Koordinatensystem der Kinect und dem des Vive Controllers.

Ist zum Beispiel die Transformation entlang der X Achse der Kinect verschoben, so verstärkt sich der Fehler, wenn man das Objekt von der anderen Seite, also um 180° gedreht aufnimmt (siehe Abb. 3.2). Der Fehler im lokalen Koordinatensystem wird in das globale transformiert und ist in dem Fall dann in genau entgegengesetzter Richtung.

In der offiziellen Dokumentation der Kinect ist beschrieben, dass der Ursprung von Punktwolken in dem Tiefensensor liegt (siehe [Kina]). Leider fehlt die exakte Positionsangabe im Gehäuse. Im Bild 3.3 aus dem chinesischen Microsoft Forum ist eine von Benutzern vermessene schematische Darstellung der Kinect abgebildet. Der Tiefensensor liegt hinter der kleineren runden Öffnung. Die exakte Position ließ sich ohne die Kinect zu zerlegen nicht ermitteln, da auch Fertigungsungenauigkeiten die genaue Position beeinflussen können. Für die Implementation wurde angenommen, dass er sich mittig hinter der Öffnung befindet. Eine digitale Kalibrierung gestaltet sich schwierig, da das Lighthouse Tracking des Controllers zusätzlich einige Ungenauigkeiten mit sich bringt. Der Ursprung des Controllers lässt sich aus den Modellen von Steam VR auslesen (siehe Abb.3.4 (c)) Dieser liegt geschickt für VR Anwendungen da die Z-Achse in der Hand liegt. Aber das für das Tracking von Objekten liegt der Ursprung ungeschickt, da er im inneren des Controllers liegt und es keine ebene Auflagefläche parallel zu den Achsen gibt. Bei der Implementation stand noch kein Vive Tracker zur Verfügung. Die Tracker sind extra Sensoren die für das Tracken von Objekten mit dem Lighthouse System entwickelt wurden. Bei diesen ist der Ursprung in der Schraube und parallel zur Auflagefläche (siehe Abb. 3.5). Für die Arbeit wurde der Controller so nah wie möglich an dem Tiefensensor, also direkt darüber angebracht (siehe Abb. 3.4). Als Hilfe wurde ein 3D gedruckter Zylinder verwendet der in den Ring des Controllers passt.

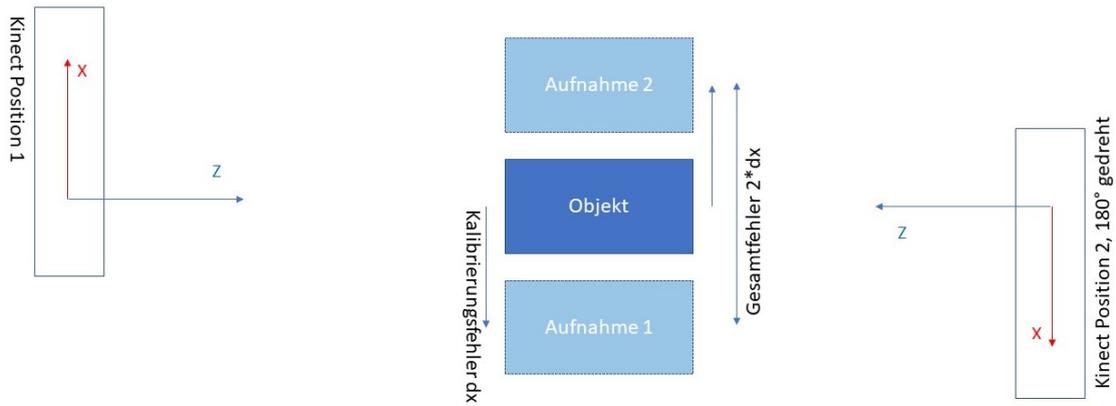


Abbildung 3.2.: Effekt einer falschen Kalibrierung dx auf die endgültige Punktwolke. Aufnahme 1 und 2 sind von lokalen Koordinaten in Welt Koordinaten transformiert

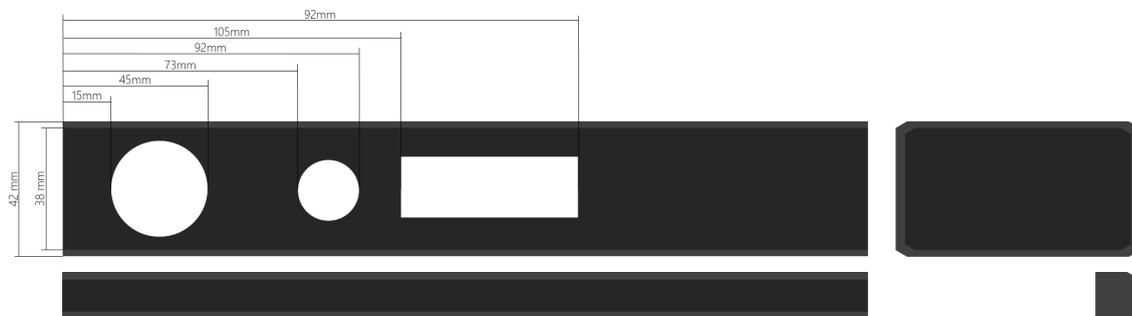


Abbildung 3.3.: Abmessungen der Kinect. Der Tiefensensor liegt in der kleinen runden Öffnung. Quelle:[Kinb]



(a) Befestigung der Kinect

(b) Kinect mit Controller

(c) Relative Position in 3D

Abbildung 3.4.: Befestigung des Controllers an der Kinect. Die Mitte des Controllers ist direkt über dem Tiefensensor. In Bild a) ist die 3D gedruckte Halterung zusehen. Der Controller wird auf den Zylinder gesteckt. In c) ist die Virtuelle Repräsentation. Koordinatenkreuze zeigen den jeweiligen Ursprung des Geräts ($[x,y,z]$ Achse= $[rot,grün,blau]$)

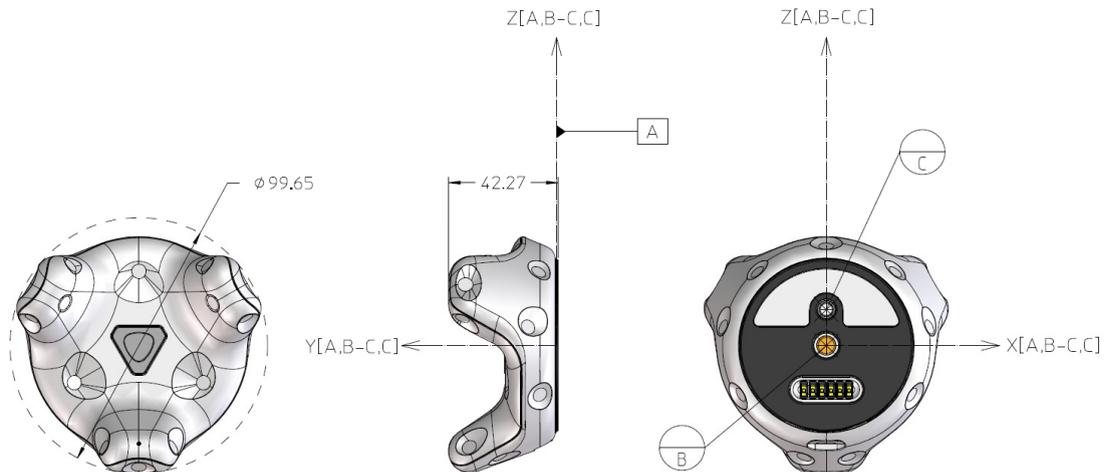


Abbildung 3.5.: Ursprung des Vive Trackers aus der offiziellen Dokumentation. Er ist in der Befestigung schraube und parallel zu der Auflagefläche.

3.3. Ergebnisse

Mit dem vorgestellten Verfahren lässt sich einfach und schnell eine Punktwolke erstellen. Die Ungenauigkeiten des Trackings und Fehler in der Kalibrierung führen aber zu sichtbaren Fehlern in der endgültigen Punktwolke. **(Bild)** Zwischen 2 Aufnahmen und den daraus resultierenden Punktwolken ist ein Versatz bis zu 2-3 cm sichtbar. In einer 3D Umgebung insbesondere in VR ist das eine zu große Ungenauigkeit. Durch die Ungenauigkeiten des Trackings verändert sich die Genauigkeit und damit der Versatz der Punktwolken ständig. Vergleicht man mit einem 2m Zollstock die reale Distanz mit der reaktiven Distanz in VR, so erhält man in VR eine Länge von 1,98 bis 2 m Die Distanz ist hierbei abhängig von der Orientierung zu den Basisstationen und der aktuellen Kalibrierung des Lighthouse Tracking Systems. Dieses Problem erschwert es, die Kalibrierung zwischen Controller und Kinect zu überprüfen.

ToDo

Im Gegensatz zur Translation war die Messung der Rotation sehr genau und erzeugte keine Sichtbaren Fehler beim Zusammenfügen der unterschiedlichen Punktwolken.

4. Speichern der Punktwolke mit 3D Tiles

In diesem Kapitel wird ein grober Überblick über die Struktur und die Komponenten des GL Transmission Formats und der 3D Tiles gegeben. Diese wurden verwendet, um die Punktwolken zu speichern.

4.1. 3D Tiles

3D Tiles [3DT] ist eine neue offene Spezifikation für das Streamen von massiven, heterogenen, geospatialen 3D Datensätzen. Die 3D Tiles können genutzt werden, um Gelände, Gebäude, Bäume und Punktwolken zu streamen und bieten Features wie Level of Detail (LOD). LOD war bei der Auswahl des Datenformats ein wichtiger Faktor, da dies bei großen Punktwolken ein signifikantes Performancevorteil bringen könnte. Bei der Implementierung wurden LOD verfahren nicht verwendet, da die Darstellung der gesamten Punktwolke möglich ist.

4.1.1. glTF

Das GL Transmission Format (glTF [GLT]) ist ein Format zum effizienten Übertragen von 3D Szenen für Gl Api's wie WebGL, OpenGL ES und OpenGL. glTF dient als effizientes, einheitliches und erweiterbares Format zur Übertragung und Laden von komplexen 3D Daten. Dieses wird in den 3D Tiles verwendet um komplexe Geometrie wie Gebäude zu übertragen. In dieser Arbeit kamen aber nur Punktwolken zum Einsatz. Im Vergleich zu aktuellen Standards wie COLADA ist glTF optimiert, schnell übertragen und kann schnell in eine Applikation geladen werden. In einer JSON formatierten Datei (.gltf) wird eine komplette Szene samt Szenengraf, Materialien und deren zugehörigen Shadern, Kamerapositionen, Animationen und Skinning Informationen übertragen. Dabei kann auf externe Dateien verwiesen werden. Diese sind zum Beispiel Binärdaten oder Bilder, die für das einfache und effiziente Übertragen von Geometrie, Texturen oder den nötigen GLSL Shadern genutzt werden.

Die .gltf Datei ist JSON formatiert und bildet den Kern jedes glTF Modells. In ihr werden alle grundlegenden Informationen wie zum Beispiel die Baumstruktur des Szenengrafen und die Materialien gespeichert (siehe Abb. 4.1.1). Eine Szene bildet hierbei den Startpunkt für die zu rendernde Geometrie. Szenen bestehen aus Knoten (Nodes), die beliebig viele Knoten als Kinder haben können. Jeder Knoten kann eine Transformation im lokalen Raum definieren, bestehend aus einer Translation, einer Rotation und einer Skalierung.

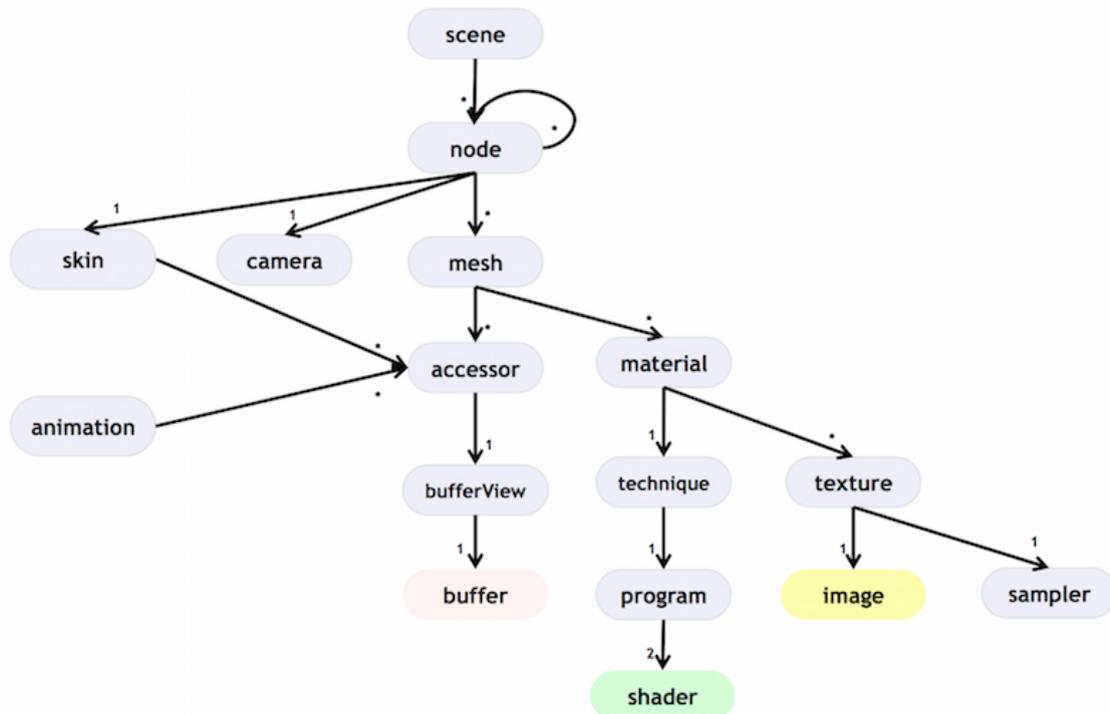


Abbildung 4.1.: Struktur einer glTF Szene.

Jeder Knoten kann eine Mesh und damit die eigentliche Geometrie referenzieren. Diese Geometrie wird in Buffern als Binärdaten gespeichert. Diese sind entweder als Base64 String direkt im JSON oder als zusätzliche Binärdatei gespeichert. Auf einen Buffer wird mit einem Accessor und einer BufferView zugegriffen. In diesen ist spezifiziert, in welchem Format die Daten vorliegen (z.B. ein Array aus 2D Vektoren (VEC2) aus UNSIGNED SHORT). Alle Datenformate entsprechen Formaten, die in OpenGL vorliegen, sodass die Daten ohne Konvertierung in OpenGL Vertex Array Objekts (VAO), bzw. Vertex Buffer Objekts (VBO) umgewandelt werden können.

Jedes Mesh kann auf ein Material referenzieren. Materialien bestehen aus Materialparametern, Texturen und Techniken. Techniken bestehen hauptsächlich aus einem GLSL Shader Programm, das ebenfalls im glTF mitgeliefert wird. Außerdem wird spezifiziert, wie die VAO und VBO aus dem Mesh bei dem Rendervorgang an den Shader gebunden werden müssen.

Ein weiteres Feature von glTF Dateien ist die Möglichkeit, Animationen und Skinning Informationen zu übertragen.

Buffer sind die eigentlichen Daten in einem Binären Block. Diese können entweder als externe Datei (.bin) oder als BASE64 encodierter String in der JSON Datei angefügt werden. Die Hauptaufgabe der Buffer ist es, große Mengen an Daten wie die Geometrie effizient zu übertragen.

4.1.2. Tileset und Tiles

Als Basis der 3D Tiles wird JSON formatiertes Tileset verwendet, das auf die eigentlichen Daten in Tiles verweist. Das Tileset hat eine baumartige Struktur aus Tiles und deren Metadaten. Jedes Tile hat hierbei ein 3D Volumen, das den geografischen Bereich

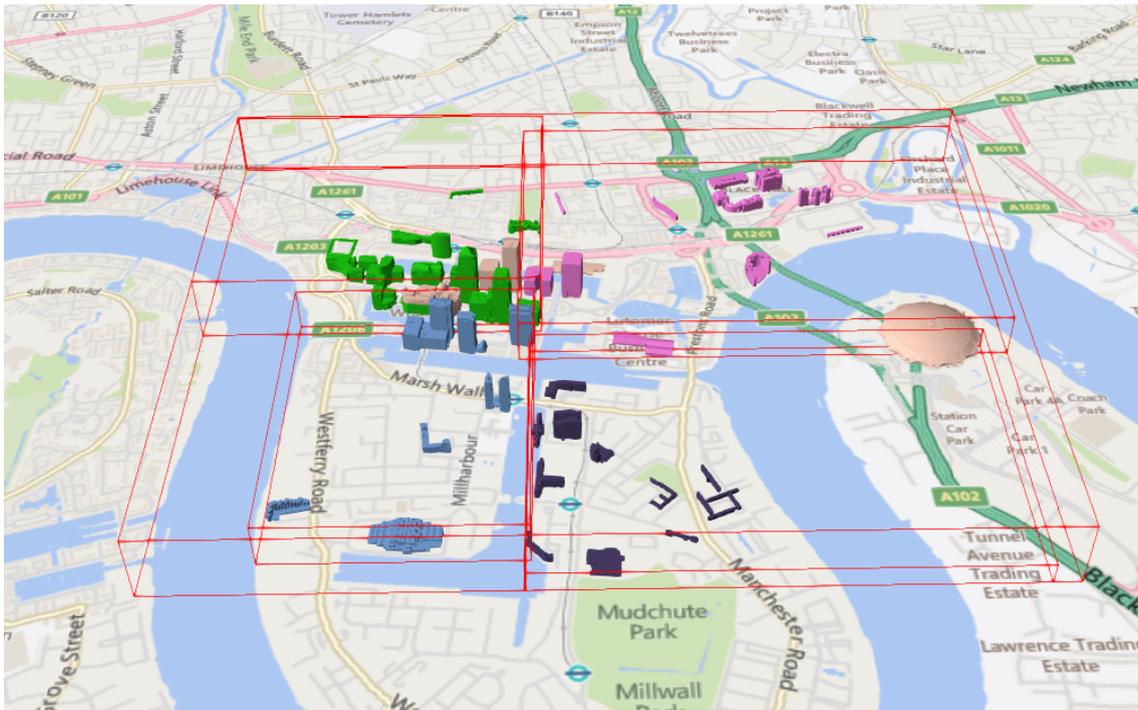


Abbildung 4.2.: Ein Tile mit 4 Kindern. Die 4 Kinder fügen die Gebäude hinzu und liegen im Volumen des Elterntiles. Als Datenstruktur liegt ein nicht uniformer Quadtree vor.

beschreibt und einen geometrischen Fehler zur Echtheit. Außerdem können Kinder und deren Transformationen zu dem Elterntile angegeben werden. Alle Kinder liegen hierbei in dem Volumen des Elternknotens und können mit verschiedenen Datenstrukturen, wie K-D Bäumen Quadtrees oder ähnlichem die Region genauer spezifizieren (siehe Bild 4.1.2). Hierbei können die Kinder das Elterntile ersetzen (replace, z.B. ein genaueres Mesh) oder das bestehende Tile ergänzen (refine, zusätzliche Gebäude oder Details). Die eigentlichen Daten der Tiles sind durch eine URL verlinkt und können dynamisch nachgeladen werden.

Tiles können in unterschiedlichen Formaten sein, zum Beispiel:

Batched3D Model 3D Daten, die im GL Transmission Format (glTF 4.1.1) übertragen werden. Zusätzlich können pro Modell Metadaten zum Visualisieren enthalten sein.

Instanced3D Model Tileformat für Instancing. Die Geometrie wird als glTF übertragen und zusätzlich eine Liste aus Positionen an denen die Objekte instanziiert werden sollen. Das kann zum Beispiel für Bäume genutzt werden.

Point Cloud Format um Punktwolken zu übertragen. Das Tileformat enthält einen kleinen Header mit allgemeinen Metadaten. Danach folgt ein JSON String, in dem steht, welche Daten wie in dem Binärteil vorliegen. Außerdem ist enthalten, ob Daten wie Position und Farbe dabei sind und wie diese gespeichert sind. Die eigentlichen Daten werden als Binärdaten übertragen und können so ohne Parsen direkt in den Speicher und Grafikspeicher geladen werden.

Composite Tileformat zum gleichzeitigen Übertragen mehrerer einzelner Tileformate in einem. Es lässt sich zum Beispiel ein Batched3D Modell für Gebäude mit Instanced3D Modell für Bäume verbinden und als ein Tile übertragen.

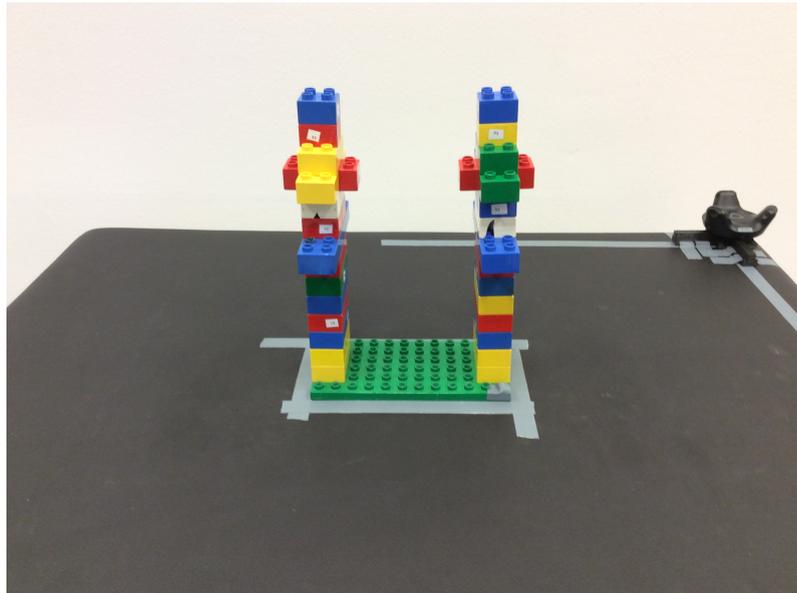


Abbildung 4.3.: Objekt für die Aufnahme. Hinten rechts ist der Tracker fest platziert der als Ursprung für die 3D Punktwolken verwendet wird.

4.2. Implementierung der 3D Tiles

Für das Speichern der Punktwolke wurden keine LOD Verfahren angewendet. In der Praxis hat sich gezeigt, dass die Wolken klein genug sind, sodass sie als ganzes effizient gerendert werden konnten. Sollte man größere Punktwolken, z.B. von einem ganzen Raum erstellen, könnte das Performancevorteile beim Visualisieren bringen. Das verwendete Tileset ist statisch und sehr einfach gehalten (siehe Anhang A) Es beinhaltet ein Tile, das auf die Punktwolke referenziert. Es ist nicht transformiert und hat ein statisches Boundigvolume eine 5m große Kugel.

Die eigentlichen Daten werden in einem Point Cloud Tile abgespeichert. Die Positionsdaten der einzelnen Punkte werden als Array aus float abgespeichert. Dabei bilden 3 floats immer die x,y,und z Koordinaten eines Punktes. Zusätzlich wird einen Array an Farbdaten gespeichert. Pro Punkt wird jeweils ein Byte pro RGB gespeichert.

Um das Kalibrieren zwischen der Echtwelt zu vereinfachen, wurde beim Aufnehmen ein Vive Tracker in der Welt platziert und als Ursprung verwendet (siehe Abbildung 4.2). Alle Punkte wurden vor dem Schreiben der Datei in das lokale Koordinatensystem des Trackers transformiert und können beim Visualisieren erneut an dem Tracker orientiert werden.

$$\text{exportPosition} = \text{TrackerPosition}^{-1} * \text{globalPosition} \quad (4.1)$$

5. Visualisierung

In diesem Kapitel wird die Visualisierung der Punktwolke in der Unreal Engine 4 [UE4b] erklärt. Die Unreal Engine ist eine mächtige Game Engine, die unter anderem Support für verschiedene Virtual Reality Systeme bietet. Durch die Verwendung einer Game Engine muss kein eigener performanter Renderer geschrieben werden, der den Anforderungen für Virtual Reality entspricht.

5.1. UE4 Rendering System

Die Unreal Engine verwendet ein Renderer System, das auf DirectX aufgebaut ist. Die gesamte Rendering ist abstrahiert und aus der Engine hat man keinen direkten Zugriff auf die Grafikkarte und die Shader. Das Unreal Engine Materialsystem ist der vorgesehene Weg, um Shader zu implementieren. Im Stil der UE4 Blueprints lassen sich damit grafisch Materialien definieren, die von der Engine in zugehörige Shader umgewandelt werden.

5.2. 3D Tiles laden und vorbereiten

Die Punktwolken können als Array direkt geladen werden, jedoch lässt die Unreal Engine nicht zu, diese auch direkt als 1D Buffer auf die Grafikkarte zu laden und zu verwenden. Als Alternative wurde der Array in eine quadratische 2D Textur umgewandelt. Jeder Pixel der Textur besteht aus 3 Werten, jeweils einen für die 3 Farbkanäle Rot, Grün und Blau. Ein Punkt im 3D Raum besteht ebenfalls aus 3 Werten für die Achsen X, Y und Z. Um die Daten auf die Grafikkarte zu laden, codieren wir die Positionen in den Farbkanälen. Farbwerte sind im Wertebereich $[0 - 1]$ und werden in unterschiedlichen Datenformaten gespeichert. Für die Implementierung wurde das HDR Datenformat der Unreal Engine verwendet. Dabei wird jeder Farbkanal in einer 16 Bit Integer codiert und ergibt damit pro Farbkanal 2^{16} diskrete Farbwerte. Das bedeutet auch für eine Punktwolke, die wir als Farbtextur codieren, haben wir eine maximale Auflösung pro Achse.

Um die Daten in eine Textur zu überführen, müssen wir diese vorbereiten. Zunächst wird das Minimum und die Größe entlang jeder Achse bestimmt. Mit diesen Daten lässt sich

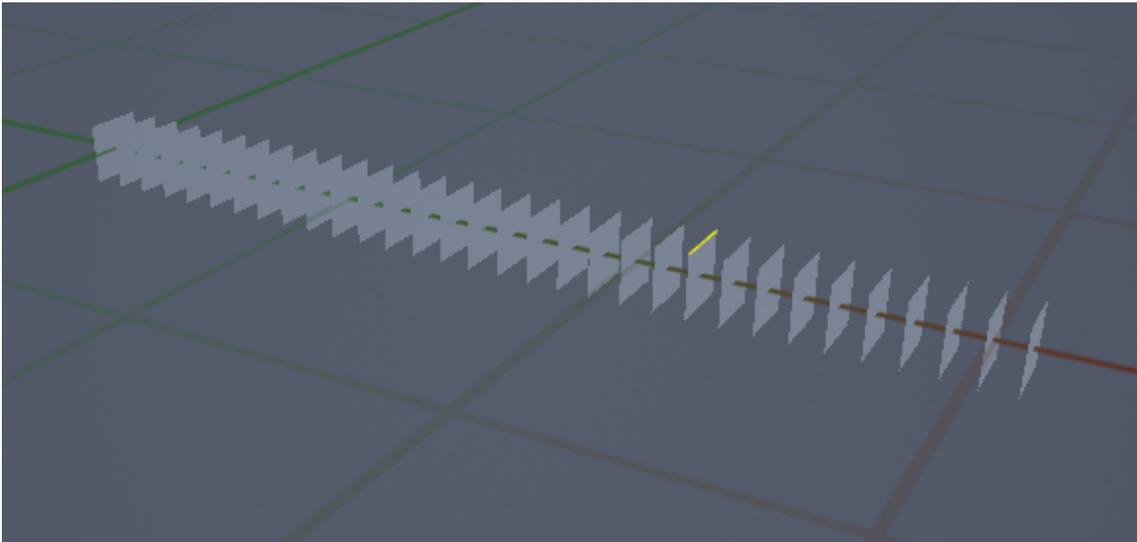


Abbildung 5.1.: Ein Quadchain mit 32 Quads

die gesamte Punktwolke in den Einheitswürfel transformieren.

$$\begin{aligned}
 \forall p \in \text{Punktwolke} \\
 \min &= \text{Minimum}(p) \\
 \max &= \text{Maximum}(p) \\
 \text{size} &= \max - \min \\
 p_{\text{Einheitswürfel}} &= \frac{p - \min}{\text{size}}
 \end{aligned}
 \tag{5.1}$$

Die so codierten Punkte lassen sich in einer Textur abspeichern und im Shader wird diese Transformation umgekehrt.

Das Aufbereiten der Farbtextur ist einfach. Die Farben sind im 3D Tile als Byte große Integer gespeichert und können direkt in eine Unreal Textur konvertiert werden.

Die erzeugten Texturen sind immer quadratisch aber die Anzahl der Punkte ist nicht vorgegeben. Für die Implementierung wurde immer die quadratische Textur bestimmt, die genug Platz für alle Punkte bietet und die nicht verwendeten Punkte haben in der Farbtextur einen α Wert (4. Farbkanal, Transparenz) von 0 sonst 1.

5.3. Rendering

In der Unreal Engine ist das Rendern von Punktwolken nicht vorgesehen und deshalb gibt es keinen Punkt Modus wie in OpenGL. Als Basis für den Renderingprozess wurde deshalb eine Quadchain verwendet (Siehe Abbildung ??) Diese Quadchain besteht aus 2^{20} einzelnen Quadraten entlang der Z Achse. Jedes Quad ist dabei an einer ganzzahligen Z Position und die Ecken sind bei 1 und -1.

Beim eigentlichen Rendervorgang kann im Shader, also in UE4 Material jedes Quads an eine Position eines Punktes geschoben werden. Hierbei wird die X-Position des Quads als Index in die Punktwolke verwendet. Aus der Größe der quadratischen Textur und dem Index können die Texturkoordinaten (uv) in der 2D Textur berechnet werden. **(Formel?)**

ToDo

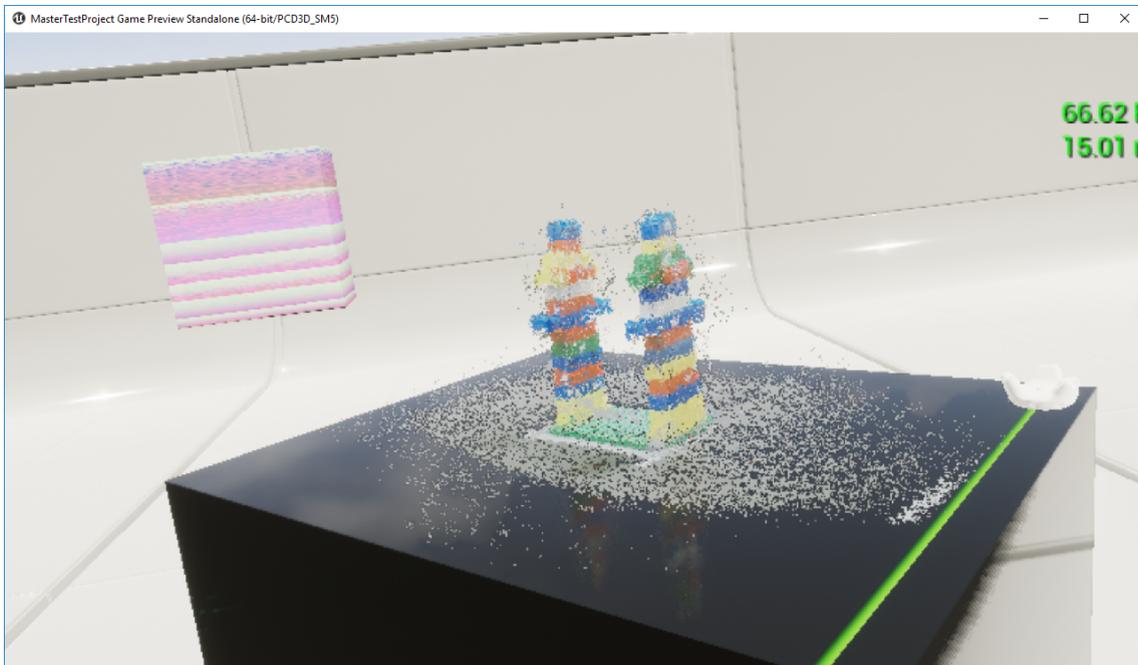
Mit den Texturkoordinaten kann die Position und die Farbe des Punktes ausgelesen werden. Bevor die Position verwendet werden kann, muss diese aus dem Einheitswürfel in die wirklichen Koordinaten zurück transformiert werden. Die in der Vorberitung errechnete Größe und das Minimum können verwendet werden um die Gleichung 5.1 umzukehren. Anschließend fehlt nur noch die Translation und Rotation der Punktwolke in der Welt. Auch hier lässt die Unreal Engine nicht zu, das direkt eine Matrix als Parameter für das Material übergeben werden kann, aber durch eine Position und eine Rotation lässt sich die Transformation im Material ausrechnen. Die berechnete Endposition wird als WorldPositionOffset an die Grafikkarte weitergegeben, die damit den Punkt an die richtige Weltposition setzt.

Für die Sichtbarkeit wird der Quad zu dem Betrachter gedreht und in der Größe skaliert. Zusätzlich wird das Quad noch durch eine runde Maske zu einem kleinen Kreis umgewandelt und mit der entsprechenden Farbe texturiert.

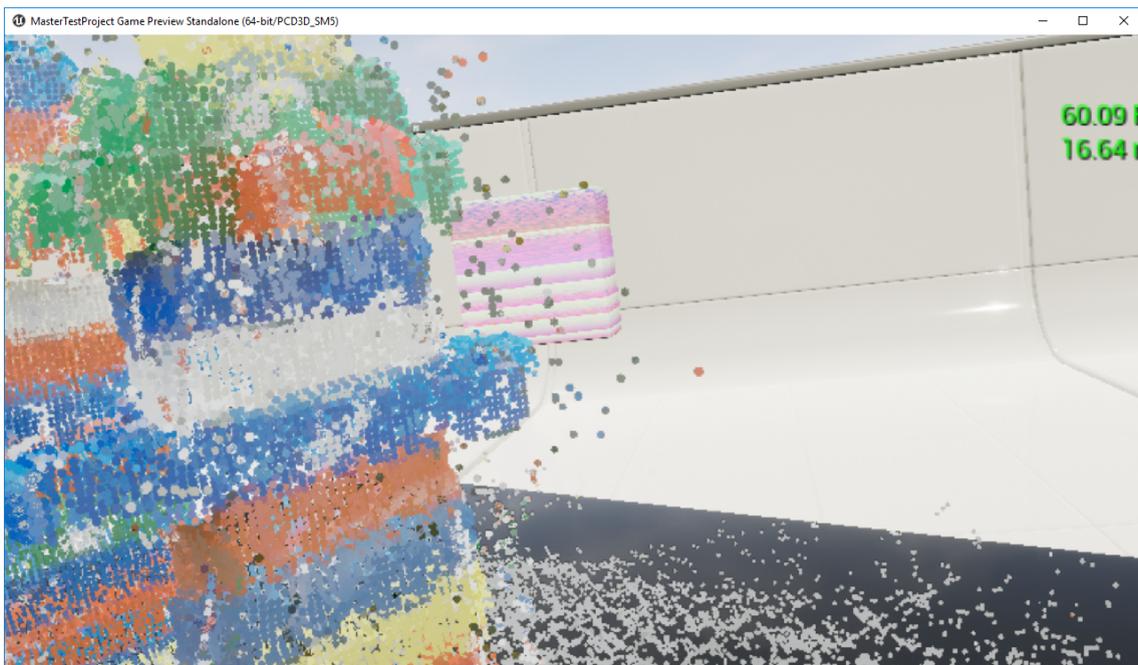
Die Quadchain hat eine feste Länge und die Punktwolke hat eine veränderbare Größe. Deshalb werden größere Punktwolken in 2 aufgeteilt. Bei kleineren Punktwolken werden die nicht verwendeten Quadrate verworfen.

5.4. Ergebnisse

Die resultierenden Punktwolken funktionieren für den Anwendungsfall und die in dieser Arbeit verwendeten relativ kleinen Punktwolken sind in VR visualisierbar. Außer dem Deaktivieren von Schattenberechnungen wurden keine Performance Optimierungen vorgenommen. Die Beispielwerte wurden mit einem i7 6700 und eine GTX 1070 aufgenommen. Eine genaue Analyse ist nicht erfolgt. Eine Punktwolke mit 1596685 einzelnen Punkten erreicht ca 60fps (siehe Bild 5.4. Hierbei ist zu bedenken, das die ganze Quadchain von 2^{20} Punkten gerendert wird und überflüssige Geometrie erst im Shader/Material verworfen wird. Bei 4 einzelnen Punktwolken mit 443175 Punkten (110369, 115991, 110377, 106438; 4 Instanzen der Quadchain) erreicht dieses vorgehen ca 44 fps. Für eine große Punktwolke mit 1797690 Punkten gesplittet in 2 Instanzen sind es ebenfalls 44 fps.



(a) Gesamte Punktwolke



(b) Nahaufnahme

Abbildung 5.2.: Punktwolke mit 1596685 Punkten. Die Textur im Hintergrund ist die zugehörige Positionstextur, Rechts oben ist FPS und die Renderzeit in ms zu sehen

6. HoloLens

Als Interaktion zwischen dem Experten in einer VR Umgebung und dem lokalen User wird die Microsoft Augmented Reality Brille HoloLens verwendet. Die HoloLens bietet die Möglichkeit, Hologramme in die echte Welt zu projizieren. Als Interaktionsmöglichkeit zwischen VR und AR wurde eine einfache Ausgewählt. In VR wurde an einen Controller ein Laserbeam mit fester Länge befestigt. Bei synchronisierten Welten zwischen VR und AR sollte dieser Strahl auch in der HoloLens an dem Controller hängen.

6.1. Unreal Engine 4 und HoloLens

Es gibt keinen offiziellen Support der HoloLens für die Unreal Engine 4. Aktuell wird das Einwickeln für die neue AR Plattform nur in Unity unterstützt. Microsoft hat auf Github einen Fork der Unreal Engine `UWP` in dem Support für die Universal Windows Plattform (UWP) enthalten ist. UWP bietet eine allgemeine Plattform für Apps, die auf allen Windows basierten Systemen funktionieren. In diesem Repository befindet sich auch ein Development Branch, der die HoloLens Unterstützung in Unreal integriert. Für die Implementierung wurde dieser Development Branch getestet. Nach dem Bauen der aktuellen Version wurde ein HoloLens Template [UE4a] in die Engine geladen. Jedoch gab es Probleme, das Projekt zu packen, sodass es auf die HoloLens geladen werden kann. Deshalb wurde für die AR Implementierung Unity verwendet.

6.2. HoloLens Implementierung

Die HoloLens Applikation in Unity ist einfach gehalten und die notwendige Logik wurde in der VR Applikation in Unreal umgesetzt. Zunächst setzt man in der HoloLens einen Weltanker in den Raum. Ein Weltanker dient als besonders wichtiger Punkt und das System soll diesen Punkt immer tracken und konstant halten. Die Unreal Engine sendet jedem Frame einen JSON formatierten String per UDP an die HoloLens in dem Start- und Endpunkt des aktuellen Lasers enthalten sind. Diese werden anschließend in der HoloLens relativ zum Welt Anker visualisiert.

6.3. Kalibrierung

Die Kalibrierung erfolgt anhand des Weltankers und eines Vive Trackers. Der Weltanker orientiert sich an dem erkannten Boden, sodass die y Achse senkrecht nach oben zeigt. Um

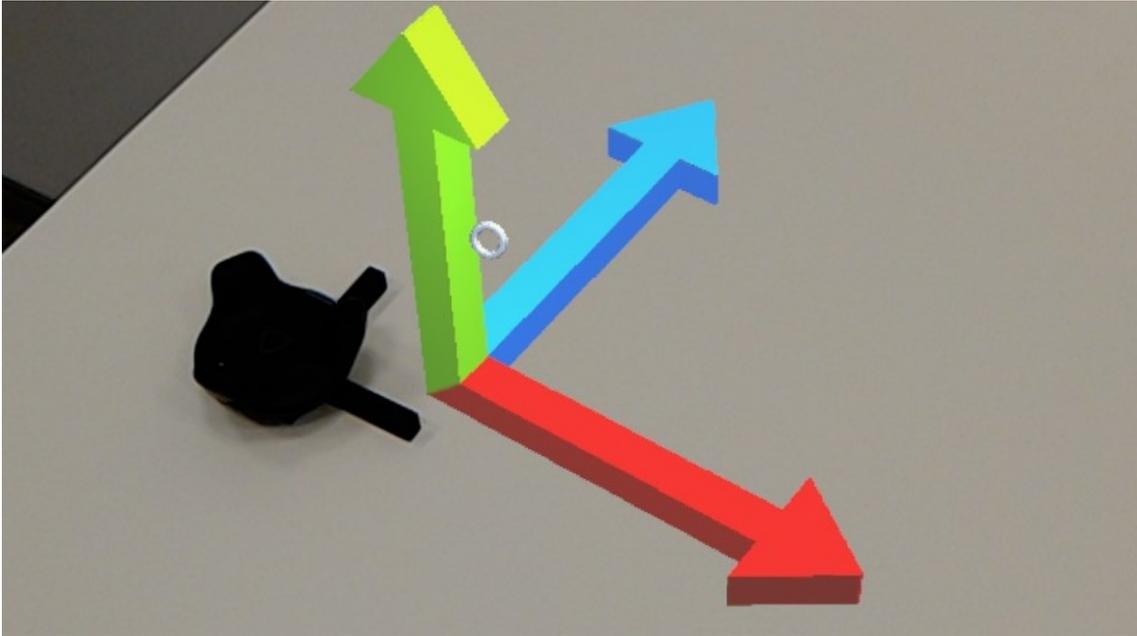


Abbildung 6.1.: Mixed Reality Capture der HoloLens. Zu sehen ist das Koordinatenkreuz und der Vive Tracker. Am Tracker ist die 3D gedruckte Hilfe zu sehen.

die beiden Welten zu synchronisieren, wird ein Vive Tracker so platziert, dass die beiden Koordinatensysteme übereinstimmen (siehe Abb. 6.3) Hierfür wurde an dem Tracker eine 3D gedruckte Hilfe angebracht, um die aktuelle Rotation besser erkennen zu können. Durch einen Tastendruck auf einen Controller wird die aktuelle Rotation und Position des Trackers in der Unreal Engine gespeichert. Mit dieser Transformation kann der Strahl der VR Umgebung in das lokale Koordinatensystem des Weltankers transformiert werden.

6.3.1. Kalibrierungsfehler

Diese Art der Kalibrierung ist einfach, aber mit der aktuellen Hardware an mehreren Stellen fehleranfällig. Das erste Problem ist die menschliche Ungenauigkeit. Der Tracker muss exakt an die richtige Position mit der richtigen Rotation gelegt werden. Dabei ist das Hologramm direkt über dem Tracker, verdeckt diesen und erschwert damit das exakte Positionieren. Insbesondere die Rotation ist bei der Kalibrierung ein Problem. Eine kleiner Rotationsfehler wirkt sich weiter entfernt vom Ursprung stark auf die Kalibrierung aus. Zum Beispiel bringt ein Fehler von 1° bei 2m Distanz eine Verschiebung von 3,4cm ($distance(rotate(1^\circ, (2, 10))(2, 0))$). **(check distance)**

ToDo

Zu den menschlichen Fehlern kommen Ungenauigkeiten vom Vive Tracking und dem HoloLens Tracking. Zum einen verschiebt sich der HoloLens Ursprung manchmal leicht, wenn er nicht direkt angeschaut wird. Dies ist bedingt durch das Inside-out Tracking der HoloLens. Durch erneutes Ansehen der Ursprungsumgebung wird dieser meistens wiedererkannt und wird zurück auf die ursprüngliche Position gesetzt. Aber es konnten sehr vereinzelt dauerhafte Verschiebungen beobachtet werden. Im Extremfall waren ca. 10cm nach unten zu beobachten. Ein weiterer Faktor sind die Ungenauigkeiten im Vive Tracking. Problematisch sind hierbei insbesondere die Längenunterschiede zwischen Echtwelt und Virtueller Welt gewesen. In dem verwendeten Setup wurde ein 2m Zollstock mit einem Vive Controller vermessen. Dabei war die in VR gemessene Distanz 1,98m. Dieser Fehler wirkt sich direkt auf das Zusammenspiel der Vive und HoloLens aus. Wird der Controller

2m vom Ursprung weg bewegt, dann bewegt sich der Beam nur um 1,98 in VR und damit entfernt sich auch der Beam in der AR Visualisierung von seiner eigentlichen Position. In der Implementierung wurde versucht das auszugleichen, indem alle Werte, die per UDP versendet werden, mit dem Faktor 1,025 skaliert wurden. Der Wert wurde experimentell bestimmt.

(Bidler)

ToDo

6.4. Ergebnisse

Die Kalibrierung funktioniert, ist aber nicht schnell und fehlerfrei umsetzbar. Bis die Kalibrierung vollständig stimmt, muss der Prozess teilweise ein paar Mal wiederholt und getestet werden. Nahe an dem Weltanker erreicht das umgesetzte Tool den gewünschten Effekt, den Laser an den Vive Controller zu hängen. Entfernt man sich von diesem Punkt, wird das Tracking immer schlechter und die Fehler werden sichtbarer. Für die Evaluation wurde die VR Umgebung verschoben, um getrennte Ort zu simulieren. Damit der Fehler möglichst klein bleibt und keinen großen Einfluss auf die Evaluation hat wurde die Verschiebung so klein wie möglich gehalten (1.5m).

7. Evaluation

Das grundlegende Szenario, das evaluiert wird, spielt zwischen einem lokalen Nutzer (häufig auch Techniker) und einem Experten der remote zugeschaltet werden soll. Der lokale Nutzer hat ein Hardwareproblem und der Experte das Fachwissen, um das Problem zu lösen. Der allgemeine Ablauf bei so einem Szenario ist, dass der lokale Nutzer zunächst Daten aufnimmt und dem Experten zur Vorbereitung sendet. Anschließend lösen die beiden gemeinsam das Problem. Als Vorbereitung kann der lokale Nutzer eine Punktwolke aufnehmen und diese dem Experten senden. Dieser kann sich die Punktwolke in einer VR Umgebung anschauen und mit seinem Controller und dem daran befestigten Laser auf die Punktwolke zeigen. Der lokale Nutzer bekommt in seiner AR Brille den Laser an der zugehörigen realen Position visualisiert. So kann der Experte auf die Punktwolkenrepräsentation des Objekts zeigen und der lokale Nutzer sieht diese Zeigegeste am echten Objekt.

Als Referenzszenario wurde ein Videostream gewählt. Als Vorbereitung sendet der lokale Nutzer Bilder an den Experten und bei der Zusammenarbeit stand ein Videostream zur Verfügung.

7.1. Versuchsaufbau

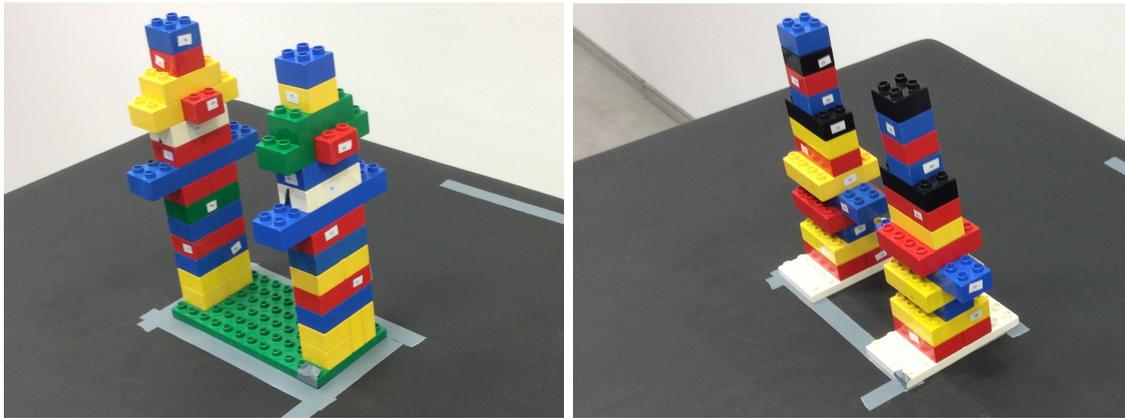
Das Hardwareproblem wurde in der Evaluation mit Duplosteinen simuliert. Aus den Steinen wurde insgesamt 2 Turmpaare aus 2 relativ ähnlichen Türmen gebaut (Siehe Abb. 7.1).

(Todo Bilder von beiden Turmpaaren auf dem Tisch) In den Türmen wurden verschiedene Farben benutzt, sodass jeder Turm insgesamt 13 farbige Ebenen hat. Jede Farbe wurde mit einer eindeutigen kleinen Beschriftung versehen, bestehend aus einem Buchstaben und einer Zahl. Diese Beschriftung kann dazu verwendet werden, um die Korrektheit bei einem Durchlauf des Experiments zu überprüfen. Des Weiterem wurde darauf geachtet, dass in jedem Turmpaar eine Farbsequenz von 4 aneinander grenzenden Farben eindeutig ist. Hierbei sollten sich die Türme aber möglichst ähnlich sein, um die Aufgabe zu erschweren.

ToDo

Auf einem fahrbaren Tisch wurden für beide Turmpaare Markierungen angebracht, damit diese immer an der gleichen Position auf dem Tisch stehen.

Für das VR Szenario wurde zusätzlich ein Vive Tracker auf dem Tisch platziert. Damit ist es möglich, das Objekt auch in der virtuellen Welt zu tracken und richtig zu positionieren. Hierfür wurde zunächst beim Aufnehmen der Punktwolke diese in das lokale



(a) Turmpaar 1

(b) Turmpaar 2

Abbildung 7.1.: Die beiden Duplotürme, die in der Evaluation verwendet wurden. Die Markierung auf dem Tisch hilft bei der exakten Positionierung.

Koordinatensystem des Trackers transformiert, und beim Visualisieren die aktuelle Transformation des Trackers hinzugefügt. Damit bei der Durchführung die echte Welt nicht komplett mit der virtuellen synchronisiert ist, wird die virtuelle Welt um einen konstanten Vektor verschoben. Stehen lokaler Nutzer und Experte an demselben Objekt, könnte der lokale Nutzer zum Beispiel aus der Handbewegung Rückschlüsse ziehen, die das Ergebnis verfälschen. Diese Verschiebung wird vor dem Senden der Daten an die HoloLens wieder heraus gerechnet. In der Evaluation wurde dafür eine Verschiebung um 1,5m entlang der negativen X-Achse des Trackers gewählt. Diese Distanz wurde möglichst klein gehalten, um Trackingungenauigkeiten nicht zu verstärken.

Für das Videoszenario wurde die Handy-App IP Webcam genutzt. Diese stellt den Videostream einer Handykamera als Webstream zur Verfügung. Der Experte kann diesen dann am PC anschauen.

7.1.1. Punktwolken

Für die Evaluation wurde auf das Aufnehmen von Punktwolken durch die Probanden verzichtet. Die Methode, die Kinect mit dem Lighthouse Tracking zu verbinden, liefert zu ungenaue Wolken. Deshalb wurden nur statische Punktwolken verwendet, die vorher aufgenommen wurden und per Hand nachbearbeitet wurden. Diese Limitierung führt dazu, dass die Türme nicht umgebaut werden können und nur statisch betrachtet wurden. Für das Videoszenario wurden im voraus Bilder aufgenommen, die dem Experten bei der Vorbereitung zur Verfügung stehen.

7.2. Versuchsablauf

Als erstes muss der Experte das nötige Vorwissen erhalten. In einem echten Szenario hat der Experte bereits alles benötigte Wissen im Vorfeld erlangt. Bei der Evaluation müssen aber alle Probanden in beiden Szenarien das gleiche Wissen erhalten. Das Vorwissen wurde durch die eindeutigen Farbsequenzen simuliert.

Ein Test in beiden Szenarien besteht aus 15 Durchläufen der gleichen Aufgabenstellung. Bei einem Durchlauf erhält der Experte eine Aufgabe. Diese Aufgabe ist eine eine Farbsequenz von oben nach unten von 4 aufeinanderfolgenden Farben. Am Anfang oder Ende der Farbsequenz ist ein zusätzlicher gesuchter Stein markiert. Beispiel: Aufgabe aus dem Turmpaar 1: *XXX, Blau, Rot, Grn, Blau*. Im ersten Schritt sucht der Experte damit den

mit XXX markierten Stein. Im VR Szenario kann er dieses direkt in der Punktwolke, die der lokale Nutzer vorher aufgenommen hat. Für das Nicht VR Szenario stehen dem Experten dafür 6 Bilder aus verschiedenen Perspektiven zur Vorbereitung zur Verfügung.

Nachdem der Experte den gesuchten Stein und dessen Farbe gefunden hat, soll er diesen dem lokalen Nutzer beschreiben. In beiden Szenarios dürfen die beiden Probanden miteinander sprechen, aber keinesfalls die ursprüngliche Farbsequenz verraten. Erlaubt ist damit unter anderem die Farbe des gesuchten Steins oder auch die Position im Turm mitzuteilen.

Im VR Szenario wird dem Experten zusätzlich der Beam angeschaltet, mit dem er auf die entsprechende Stelle zeigen kann. Im Video Szenario wird der Videostream angeschaltet, sodass dieser als Interaktionsmöglichkeit zur Verfügung steht.

Nachdem der lokale Nutzer den gesuchten Stein erkannt hat, liest dieser die Beschriftung vor.

Der Experte bestimmt selbst, wann er zum nächsten Aufgabenteil voranschreitet. Nachdem er aus dem Vorbereitungsdaten den Stein erkannt hat, drückt er eine Taste (Controller Trigger/ Enter) und bekommt damit Zugriff auf den Laserbeam bzw. den Videostream. Wurde das Label vorgelesen kann er erneut mit der gleichen Taste zur nächsten Aufgabe gelangen. Bei jedem Tastendruck wird die aktuelle Uhrzeit gespeichert. Damit können die Zeiten errechnet werden können.

Der Gesamttablauf der Evaluation erfolge immer im ähnlichen Ablauf. Vor dem Test wurden der allgemeine Fragebogen ausgefüllt, das grundlegende Szenario erklärt und eine Beispielaufgabe an einem separaten Turm erklärt.

Anschließend wurden die beiden Szenarien evaluiert. Ein Team startet hierbei entweder mit dem Video Szenario oder dem VR Szenario und einem Turmpaar. Beim 2. Szenario wurden dann Rollen getauscht und das andere Turmpaar genutzt, um die Ergebnisse nicht zu verfälschen. Außerdem wurde darauf geachtet, dass beide Turmpaare in VR und Video verwendet wurden.

Vor dem VR Szenario wurde kurz die Kalibrierung überprüft. Der Experte sollte auf die Turmspitzen zeigen und der lokale Nutzer sollte kurz überprüfen, ob der Laserstrahl auch in der HoloLens den Turm an der gleichen Stelle schneidet. War die Versatz zu groß, wurde neu kalibriert.

Nach jedem Szenario wurden die Fragebögen zu dem Test ausgefüllt. Hierbei wurden eigene Fragen, NASA-TLX und der User Experience Questionnaire (UEQ) verwendet.

Abschließend gab es noch einen weiteren Fragebogen mit einer allgemeine Frage und freien Kommentaren.

Um anfängliche Probleme und den Lerneffekt in der Evaluation nicht mit einzubeziehen, wurden bei jedem Testlauf die ersten 5 Aufgaben verworfen. Das heißt, es wurden nur die letzten 10 Durchläufe der jeweiligen Szenarien ausgewertet und auf Fehler und Zeitunterschiede untersucht.

7.3. Statistische Verfahren

Für die statistische Auswertung der Nutzerstudie sind einige Verfahren notwendig. Die Ergebnisse bestehen aus den Antworten der Fragebögen und den automatisierten Zeitmessungen. Die Fragebögen sind meist ordinal skalierte Daten auf einer Skala von -3 bis 3 bei dem jeder Wert nur einmal vorkommen darf. Zusätzlich war es den Probanden möglich, Freitextanmerkungen zu machen. Diese werden bei Relevanz an der entsprechenden Stelle erwähnt.

Für die ordinal skalierten Daten wird der Median und das 1. und 3. Quartil verwendet. Diese werden in Box-Whisker Plots dargestellt. Hierbei ist die Kennzeichnung wie folgt:

Minimum und Maximum Whisker

1.&3. Quartil Die Box

Median gepunktete Linie

Mittelwert Raute

Für metrisch skalierte Daten wird als Maß das arithmetische Mittel, im folgenden Mittelwert genannt, verwendet. Zusätzlich wird in den Schaubildern die Standardabweichung angegeben.

Um Zusammenhänge in den Daten zu finden und zu analysieren, wurden Signifikanzanalysen der Daten durchgeführt. Hierfür wird eine Gegenhypothese, auch Nullhypothese genannt, aufgestellt, welche ausdrückt, dass kein Zusammenhang besteht. Eine berechnete Testgröße (p-Wert) gibt Aufschluss darüber, wie wahrscheinlich die Nullhypothese zutrifft. Liegt der p-Wert unter einem Signifikanzniveau, kann die Nullhypothese verworfen werden. Für das Signifikanzniveau wird meist ein Wert von 5% verwendet [LM] Für die Evaluation wurde deshalb ebenfalls 5% gewählt.

In dieser Arbeit wurde der t-Test für die Signifikanzanalyse verwendet. Dieser bietet Verfahren für abhängige (gepaarte) und unabhängige Stichproben. Abhängige Stichproben liegen vor, wenn Messwiederholung vorliegt, zum Beispiel, falls die Messwerte von der gleichen Person stammen oder bei natürlichen Paaren, d.h. die Messwerte stammen von unterschiedlichen Personen, die zusammengehören. In dieser Evaluation liegt eine Abhängigkeit zwischen dem Experten und dem lokalen Benutzer vor. Unabhängige Stichproben sind zum Beispiel der Vergleich der beiden Experten zwischen dem VR und Video Szenario. Die Rollen werden zwischen den Tests gewechselt und damit nimmt jeder Teilnehmer jede Rolle nur einmal an.

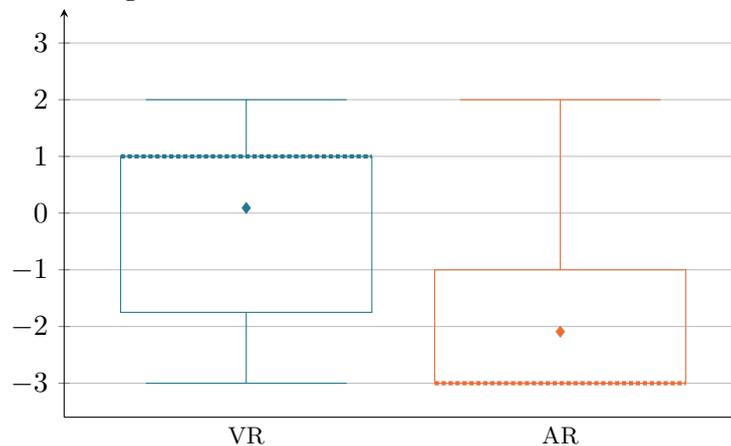
In dieser Arbeit wird die Nullhypothese verwendet, dass sich die 2 Stichproben nicht unterscheiden. Das ist keine gerichtete Hypothese, deshalb wurde der zweiseitige t-Test verwendet.

7.4. Probanden und Teams

Insgesamt wurden die Evaluation mit 13 Teams mit jeweils 2 Personen durchgeführt. Bei den Versuchen 2 und 5 gab es technische Probleme mit dem VR /AR Setup (große tracking Ungenauigkeiten und Abbrüche). Deshalb wurden diese Testläufe komplett aus den Daten gestrichen. Für die folgende Evaluation werden nur die Daten der verbleibenden 11 Teams mit insgesamt 22 Personen betrachtet. An der Studie nahmen 19 männlich und 3 weibliche Probanden teil. Für die Evaluation wird nur die männliche Sprachform verwendet. Alle Aussagen beziehen sich aber auf beiden Geschlechter. In der Altersgruppe bis 20 Jahren waren 2 Teilnehmer, von 20 bis 30 Jahren 17 Teilnehmer und in der Altersgruppe von 30 bis 40 Probanden. 6 der Probanden nutzten eine Sehhilfe und 2 gaben an, an einer Rot-Grün Schwäche zu leiden. Aus den freien Antworten geht hervor, dass eine Rot-Grün Schwäche in dem Versuchsaufbau keine Einschränkungen mit sich brachte. Die rote und grüne Farbe der Duplosteine sei kräftig und unterschiedlich genug, sodass diese trotzdem erkennbar waren.

Jeder Proband wurde nach Erfahrung zu Virtual Reality und Augmented Reality und die dabei verwendeten Systeme gefragt (siehe Box Plot 7.2). Dabei ist zu erkennen, dass viele Personen schon ausgeprägte Erfahrungen gesammelt haben. Der Median der Befragten lag bei 1 wobei -3 für keine Erfahrung und 3 für sehr viel Erfahrung steht. Verwendete Systeme

Abbildung 7.2.: Erfahrung mit VR und AR, -3 ist keine Erfahrung, +3 ist sehr viel Erfahrung



waren hierbei Vive (13), Playstation VR (6), Oculus Rift (6) und verschiedenen Systeme, die ein Handy nutzen (8; Cardboard, Daydream, Gear VR, etc.). Augmented Reality ist hingegen noch nicht bei der Allgemeinheit angekommen. 14 der 22 Probanden gaben an, noch keine AR Erfahrung zu haben (Median -3). Die restlichen gaben Erfahrungen mit der HoloLens und mit Pokemon Go an.

(Plot y beschriftung Erfahrung)

ToDo

7.5. Ergebnisse

7.5.1. Vorbereitung des Experten

Zunächst betrachten wir die Vorbereitung des Experten. In beiden Szenarien bekam der Experte ein Farbsequenz und sollte den gesuchten Stein in den Turmpaaren erkennen. Im VR Szenario hatte er dazu die Punktwolke gesehen und im Video Szenario standen dem Experte 6 Bilder aus unterschiedlichen Perspektiven zur Verfügung. Im Fragebogen wurde gefragt, wie einfach es sei, den Stein zu finden (Plot 7.3). Hierbei gaben die Experten im Video Szenario an, dass das Auffinden des Steins sehr einfach sei (Median 3), wobei der VR Experte dieses als signifikant (**check**) schwerer einstufte (Median 1). Diese Einstufung ging zusätzlich auch aus den freien Kommentaren hervor. 6 der Experten fanden die Punktwolke "schwer sichtbar", "pixelig" und "ungenau" und haben vermutlich deshalb das Finden als schwieriger eingestuft. Während des Versuchs wurde auch die Zeit für die Vorbereitung gemessen. In der Abbildung 7.4 sind die Zeiten eingezeichnet. Bei 4 Gruppen war der Video Experte schneller als der VR Experte, bei Gruppe 1 auch signifikant schneller. In den restlichen 7 Gruppen war der VR Experte schneller bei den Gruppen 10 und 13 auch hier signifikant. Im Durchschnitt über alle Versuche hat der VR Experte 9,9 Sekunden und der Video Experte 11 Sekunden gebraucht. Damit ist das Steinfinden im VR Szenario im Durchschnitt 1,1s schneller, aber es konnte keine statistische Signifikanz festgestellt werden. (**Check**) Bemerkenswert ist, dass 5 von 6 Probanden in der Expertenrolle, die die Punktwolke kritisiert haben, im VR Szenario schneller waren als ihr Partner als Experte im Video Szenario. Ein Aspekt auf dem in diesem Test nicht eingegangen wurde, sind die persönlichen Qualifikationen der jeweilige Experte. Dieser könnte Einfluss auf das Messergebnis haben.

ToDo

ToDo

Abbildung 7.3.: Wie einfach war es, den beschriebenen Stein zu finden?

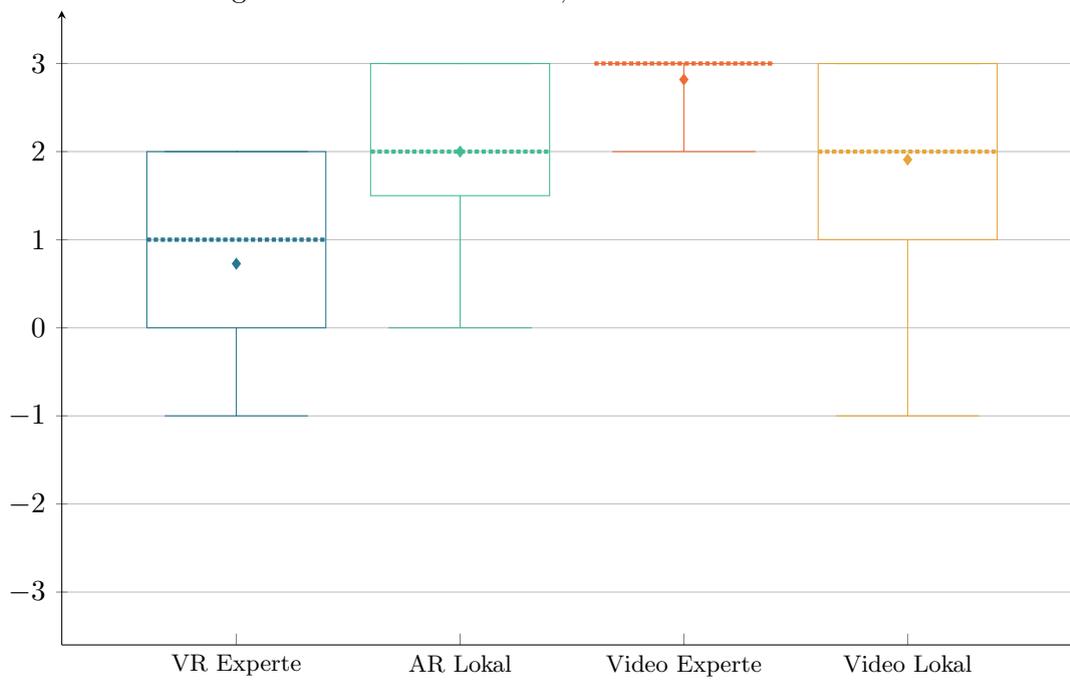
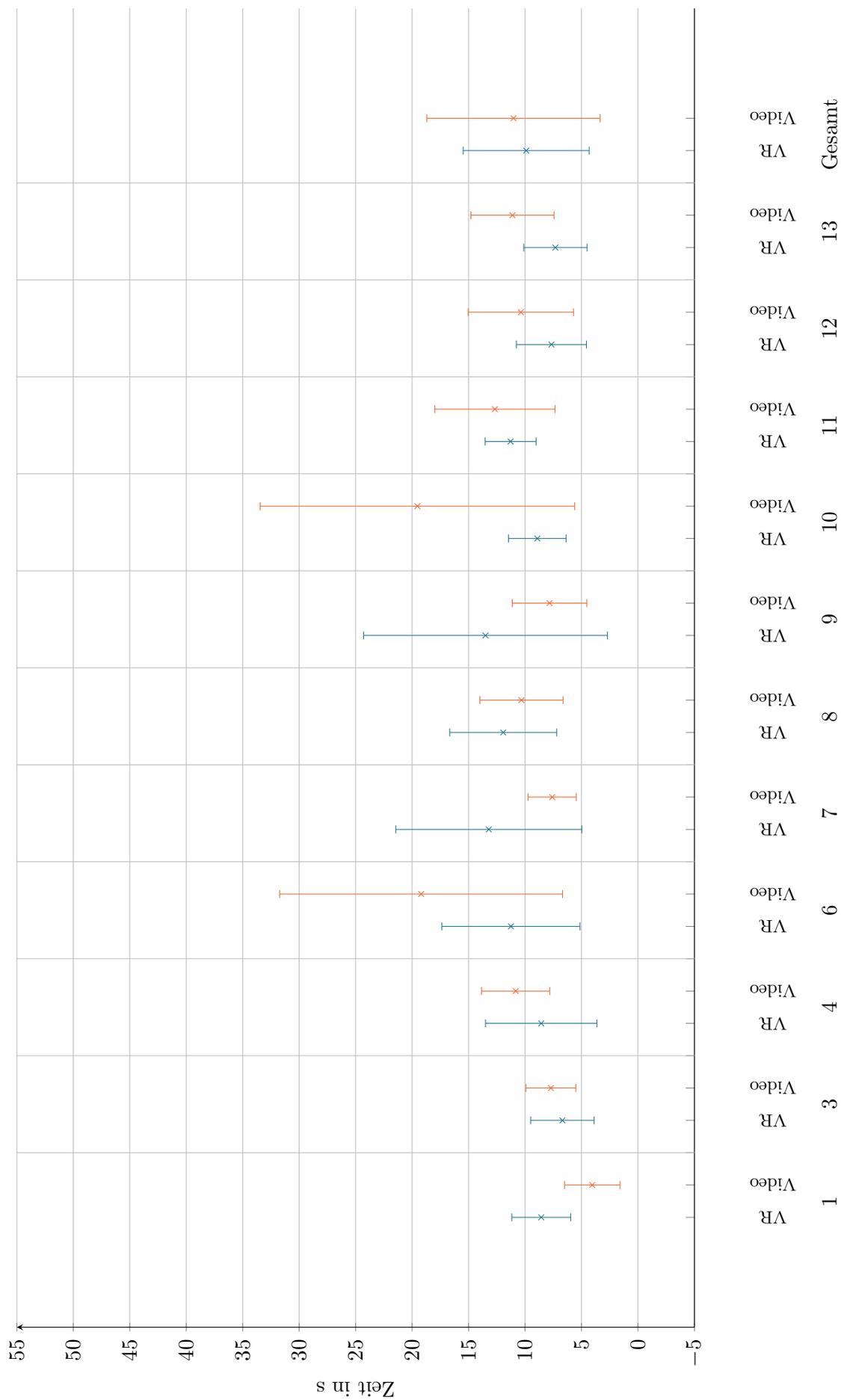


Abbildung 7.4.: Vorbereitungszeit des Experten



7.5.2. Kommunikation des gesuchten Steins

Die nächste Aufgabe des Experten war es, den gesuchten Stein an die lokalen Nutzer zu kommunizieren. Im VR Szenario stand hierfür der Laser Strahl zur Verfügung, und im Video Szenario ein Livestream.

7.5.2.1. Videostream

Der Videostream wurde teilweise als verwirrend wahrgenommen und dann von dem Experten fast vollständig ignoriert. Die meisten Teams haben sich bei den ersten Versuchsdurchläufen direkt oder indirekt auf eine gemeinsame Bezeichnung der Türme geeinigt. Häufig wurde links und rechts verwendet, aber auch unterscheidende Merkmale in den Türmen, wie der Turm mit der blauen Spitze oder der Turm mit der Deutschlandflagge. Links und rechts funktionieren insbesondere dann sehr gut, wenn der Nutzer nach jedem vorgelesenen Label an seine Ausgangsposition zurückkehrt. Ansonsten ist diese Bezeichnung teilweise verwirrend für den Experten. Ändert sich die Ansicht des lokalen Nutzers ständig, können sich auch die Bezeichner für jeweiligen Türme ändern. Außerdem wackelt die Kamera stark, wenn sich der lokale Nutzer bewegt, zum Beispiel, um die Beschriftung vorzulesen. Deshalb haben einige Teams nach dem initialen Einigen über die Bezeichnungen der Türme, den Videostream nicht für die Lösung der Aufgabenstellung verwendet.

bei anderen Teams wurde der Videostream zum direkten Feedback benutzt. Der lokale Nutzer hat zum Beispiel mit einem Finger auf einen Turm oder Stein gezeigt und nachgefragt, ob er das richtig verstanden hat. Diese Info kann der Experte im Video sehen und die Kontrollfrage direkt beantworten.

7.5.2.2. VR/AR Strahl

Bei funktionierendem Tracking hat der Beam gute Ergebnisse geliefert. Einige Teams konnten durch Zeigen und zusätzliches Sagen der Farbe den Stein eindeutig beschreiben. Damit ist der Beam zumindest eine gute Grundorientierung für den lokalen Nutzer. Ein großes Problem mit dem tracking waren kleine konstante Verschiebungen in eine globale Richtung (Kalibrierfehler, bzw. Längenuntreue). Bei kleinen Verschiebungen wurde dieses als störend empfunden, aber wenn bekannt ist, wie die Verschiebung ist, dann kann diese im Kopf ausgeglichen werden.

Ein weiteres Problem, das in den freien Texten genannt wurde, waren zitterige Hände. Diese kleinen Bewegungen werden mitübertragen und können dem lokalen Nutzer das Erkennen erschweren.

7.5.2.3. Sprachliche Kommunikation

In beiden Szenarien war es erlaubt, frei zu reden und zu kommunizieren. Es war lediglich verboten, die Aufgabenstellung des Experten zu sagen. Im anschließenden Fragebogen wurde gefragt, wie wichtig die sprachliche Kommunikation für die jeweilige Person gewesen ist 7.5. Zwischen den lokalen Nutzern ist in AR und Video hierbei ein signifikanter Unterschied feststellbar. Die Person in AR findet die Sprache zwar wichtig (Median 2), jedoch unwichtiger als der lokale Nutzer im Videoszenario. Zwischen den Experten ließ sich kein signifikanter Unterschied feststellen, jedoch ist der Durchschnitt im VR Szenario geringer.

Außerdem wurden die Probanden gefragt, wie einfach es war, den beschriebenen Stein zu finden (siehe Abbildung 7.3). Zwischen den lokalen Nutzern ist kein Unterschied feststellbar. Das heißt, die sprachliche Kommunikation war unwichtiger, aber das eigentliche Auffinden wurde in beiden Szenarien gleich leicht bewertet.

Abbildung 7.5.: Wie wichtig war die sprachliche Kommunikation?

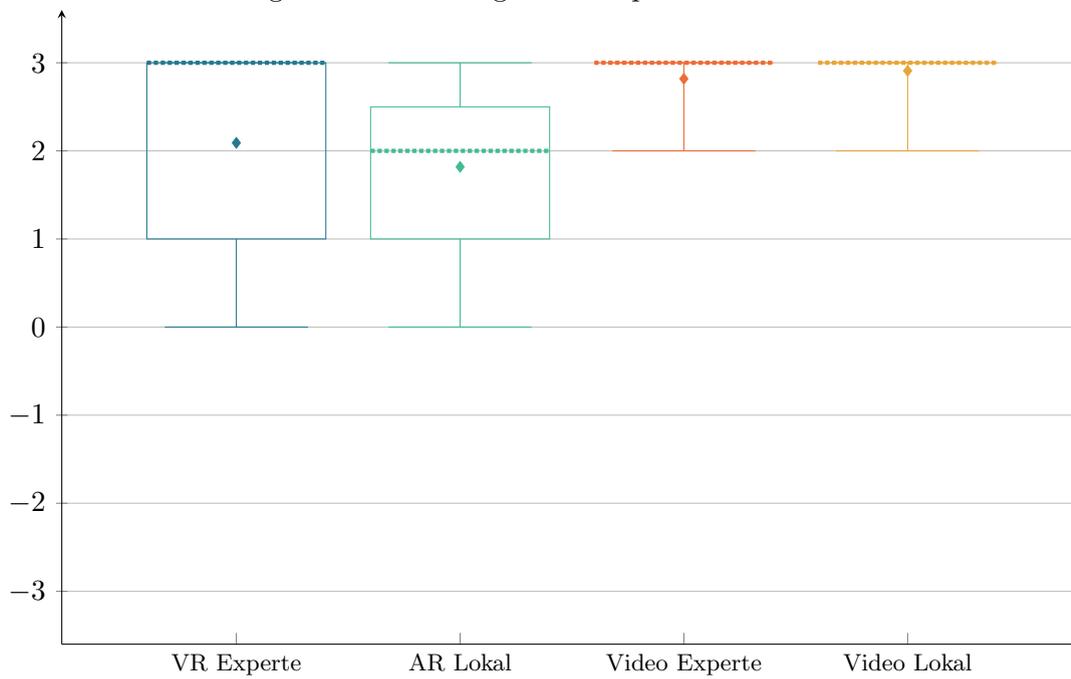
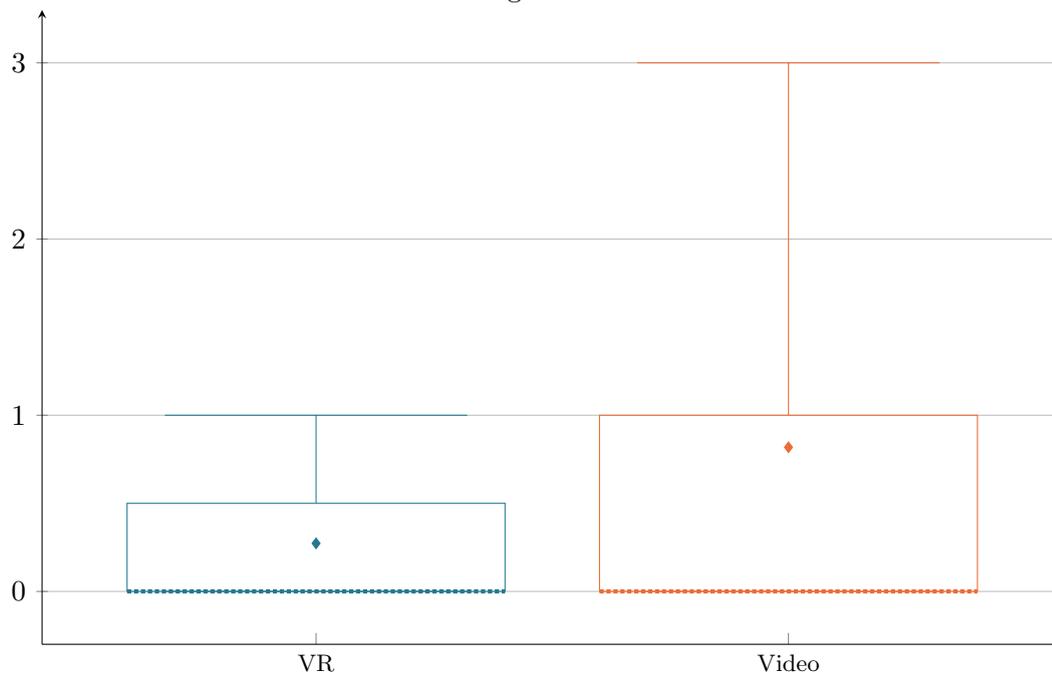


Abbildung 7.6.: Fehleranzahl



7.5.3. Fehleranzahl

Wie in der Statistik 7.6 zu sehen ist, wurden kaum Fehler in den letzten 10 Durchläufen gemacht (VR: 8 von 11 Teams fehlerfrei, Video: 6 von 11 Teams fehlerfrei) In den nicht gewerteten ersten 5 Trainingsdurchläufen waren mehr Fehler zu finden. Im Video Szenario wurde 2 mal 3 Fehler gemacht, was auf eine größere Fehleranfälligkeit hinweist.

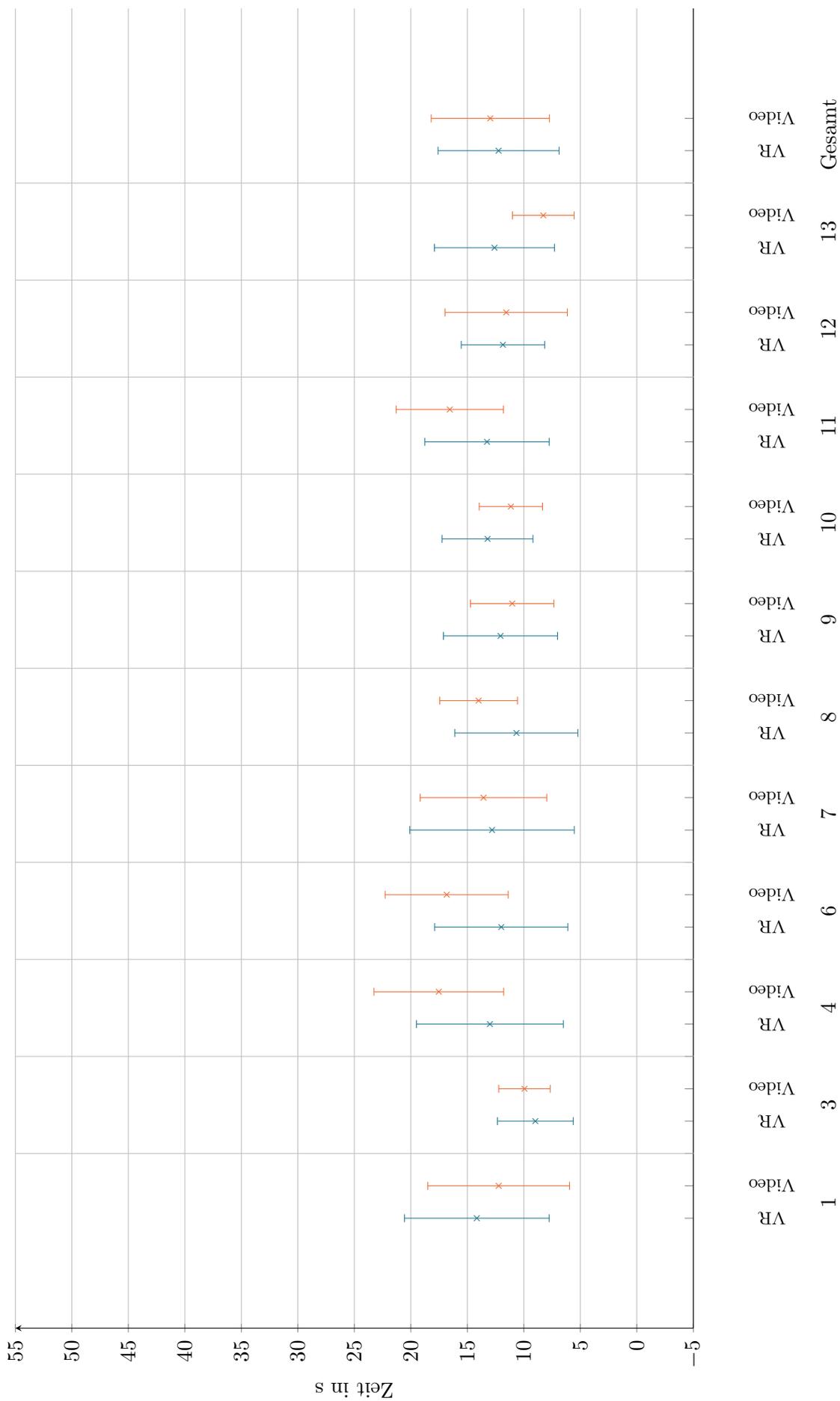
In beiden Szenarien sind allgemeine Fehler bei dem Versuch häufig beim Experten passiert. Zum Beispiel wurde der letzte Stein der Sequenz beschreiben und nicht der eigentlich gesuchte. Sprachliche Ungenauigkeiten haben auch zu Fehlern geführt. “Der zweite schwarze Stein” und “der zweite Stein, der Schwarze” klingen ähnlich, bedeuten aber unterschiedliche Steine.

Eine zusätzliche Fehlerquelle im VR Szenario ist ein globaler Versatz des Strahls. Ist zum Beispiel der Strahl um 1 bis 2 Steine verschoben, dann zeigt dieser in der AR Umgebung andere Steine an, als der Experte in seiner VR Umgebung sieht.

Im Video Szenario war das Verwechseln von rechts und links ein häufiger Fehler. Teilweise wurden die Steine richtig beschreiben, aber dann die Beschriftung auf dem anderen Turm vorgelesen. Der zweite Fall, der zu einer links rechts Verwechslung geführt hat war, dass der Experte eine andere Ansicht in den Bildern gewählt hat, als der lokale Nutzer in seinem Videostream. Passt der Experte nicht auf, dann beschreibt er rechts und links anhand des Bildes, aber der lokale Nutzer steht auf der falschen Seite.

7.5.3.1. Timings

Abbildung 7.7.: Kommunikationszeit



Während der Kommunikation zwischen dem Experten und dem lokalen Nutzer wurde ebenfalls die Zeit gestoppt. Die Messung beginnt, sobald der Experte anfängt zu kommunizieren und endet sobald die Beschriftung vorgelesen wurde. In Abbildung 7.7 sind die Zeiten abgebildet. Wie bei der Vorbereitung gab es 6 Gruppen, die in VR schneller waren und 5, die in dem Video Szenario schneller waren. Signifikant schneller war nur Gruppe 13 im Videoszenario. **(überprüfen)** Insgesamt war das VR Szenario mit 12s Durchschnittszeit 700ms schneller. Ein großer Faktor, warum bei dieser Messung keine Unterscheide festgestellt werden können, ist, dass der lokale Nutzer, nachdem er den Stein erkannt hat, noch die zugehörige Beschriftung finden musste. Dieser Vorgang hat je nach Positionierung der Beschriftung länger gedauert als die eigentliche Kommunikation.

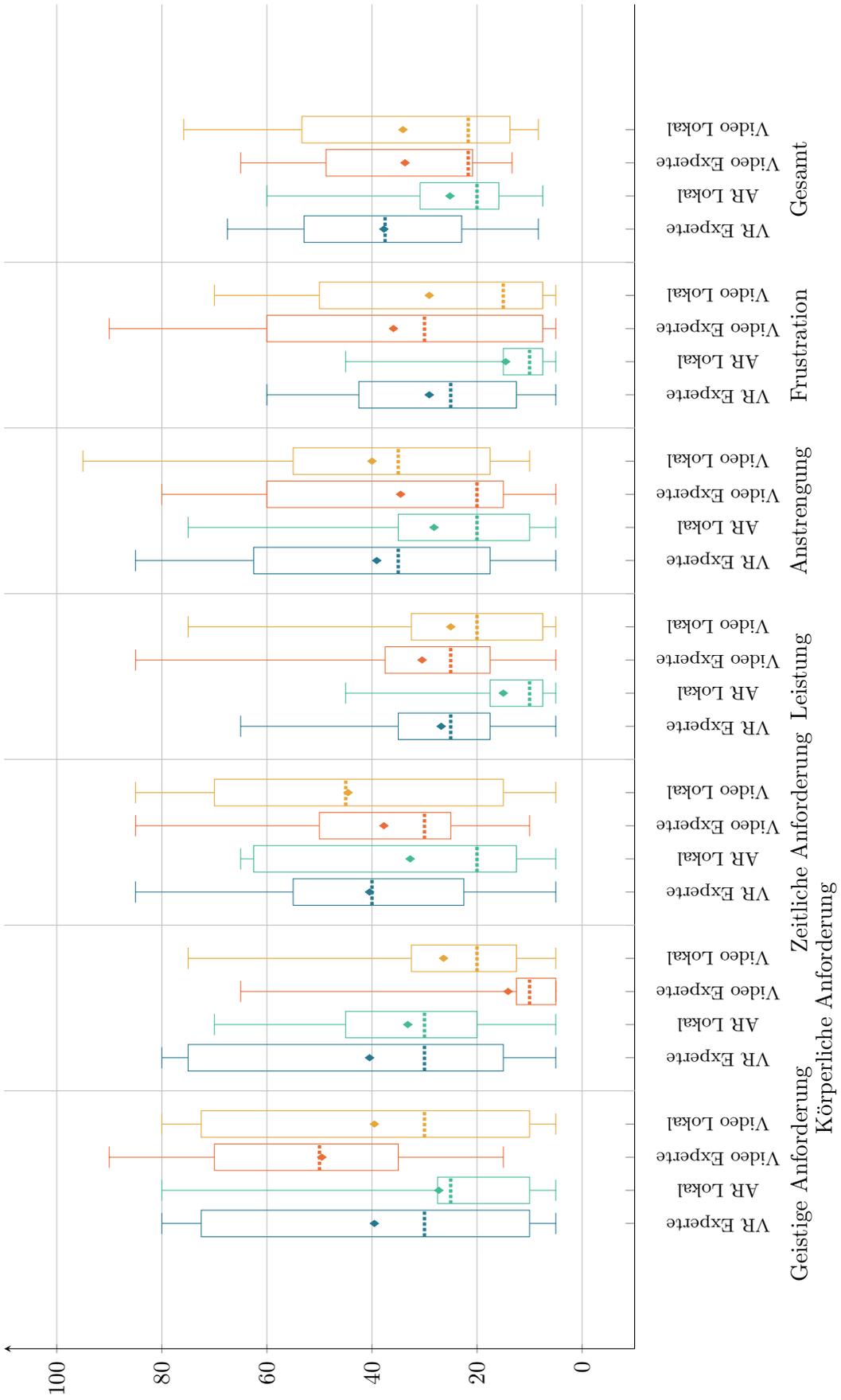
ToDo

7.5.4. NASA TLX und UEQ

ToDo

(NASA: legende statt vr und ar drunterzuschrieben, zusätzliche Achsenbeschriftung mit gering und hoch)

Abbildung 7.8.: NASA-TLX, Die Skala geht von gering (0) bis hoch (100). Bei Leistung von gut (0) bis schlecht (100)

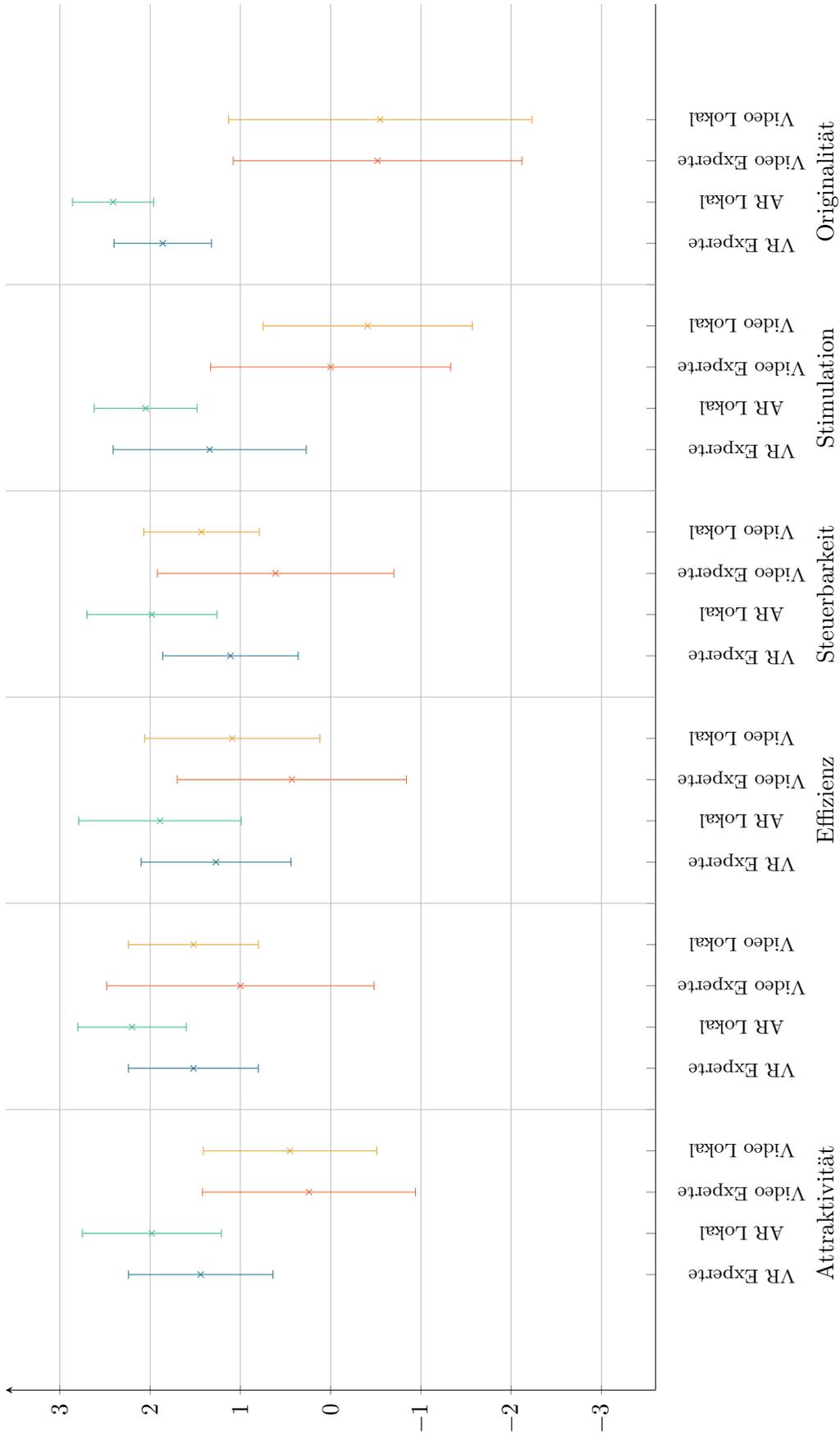


In der Evaluierung wurden 2 standardisierte Tests verwendet: NASA-TLX und der User Experience Questionnaire (UEQ).

Beim NASA TLX [Har06] werden 6 Kategorien auf einer Skala von 0 bis 100 in 5er Schritten bewertet. Auf die anschließende Gewichtung der Kategorien wurde verzichtet. Die Ergebnisse wurden als Box-Whisker Plot 7.9 dargestellt. Zunächst hat der Video Experte eine signifikant geringere körperliche Anforderung angegeben als der VR Experte. Dies ist nicht verwunderlich, da der Video Experte am PC sitzt, während der VR Experte steht und sich frei in der virtuellen Welt bewegen konnte. Vergleicht man den VR Experten mit dem lokalen Nutzer in AR, dann gibt es bei Leistung und Frustration einen signifikanten Unterschied. Der AR Benutzer hat die eigene Leistung am besten bewertet und gleichzeitig die geringste Frustration angegeben. Zusätzlich ist in Augmented Reality im Durchschnitt die geringste geistige Anforderung zu sehen.

Zusammenfassend lässt sich also sagen, dass das System für den lokalen Benutzer mit der HoloLens eine Verbesserung darstellt, obwohl der Strahl tracking-bedingt nicht immer ganz korrekt war und diese Fehler vom lokalen Nutzer ausgeglichen werden mussten.

Abbildung 7.9.: Ergebnisse des User Experience Questionnaire



ToDo (cite UEQ) Der User Experience Questionnaire (UEQ) ist dazu gedacht, die Nutzererfahrung zu messen. Der Nutzer beantwortet hierzu 26 Gegensatzpaare von Eigenschaften auf einer Skala von -3 bis 3. Die 26 Paare werden dann 6 Skalen zugeordnet und ein darüber ein Mittelwert gebildet. In dem Grafen sind die Werte der einzelnen Personen zu sehen. Hier gibt es einige signifikante Unterscheide zwischen dem VR Szenario und dem Video Szenario zu erkennen. Die Attraktivität, Stimulation und Originalität schneidet beim VR Szenario Szenario wesentlich besser ab, wenn man die Experten und die lokalen Nutzer miteinander vergleicht.

Auch zwischen den Experten und dem zugehörigen Technikern gibt es Unterscheide. Im VR Test unterscheiden sich die Werte bei Durchschaubarkeit, Steuerbarkeit Stimulation, und Originalität. Der lokale Nutzer bewertet diese besser als der Partner in der VR Umgebung.

Im Video g ist nur die Steuerbarkeit von dem Techniker besser bewertet worden.

Zusammenfassend ist hier wie beim Nasa TLX zu sehen, dass der lokale Nutzer in der AR Umgebung am meisten von einem VR/AR Setup profitiert. Das System wird generell als attraktiver und origineller bewertet als ein herkömmlicher Videostream.

7.5.5. Unabhängigkeit

Nachdem die Teams beide Szenarien durchgeführt haben, wurde eine abschließende Frage gestellt. Es wurde gefragt, ob es in dem VR Szenario von Vorteil war, von der andern Person unabhängig zu sein. Im Videostream ist der Experte an die Perspektive und die Bewegungen der Kamera, also den lokalen Nutzer selber gebunden.

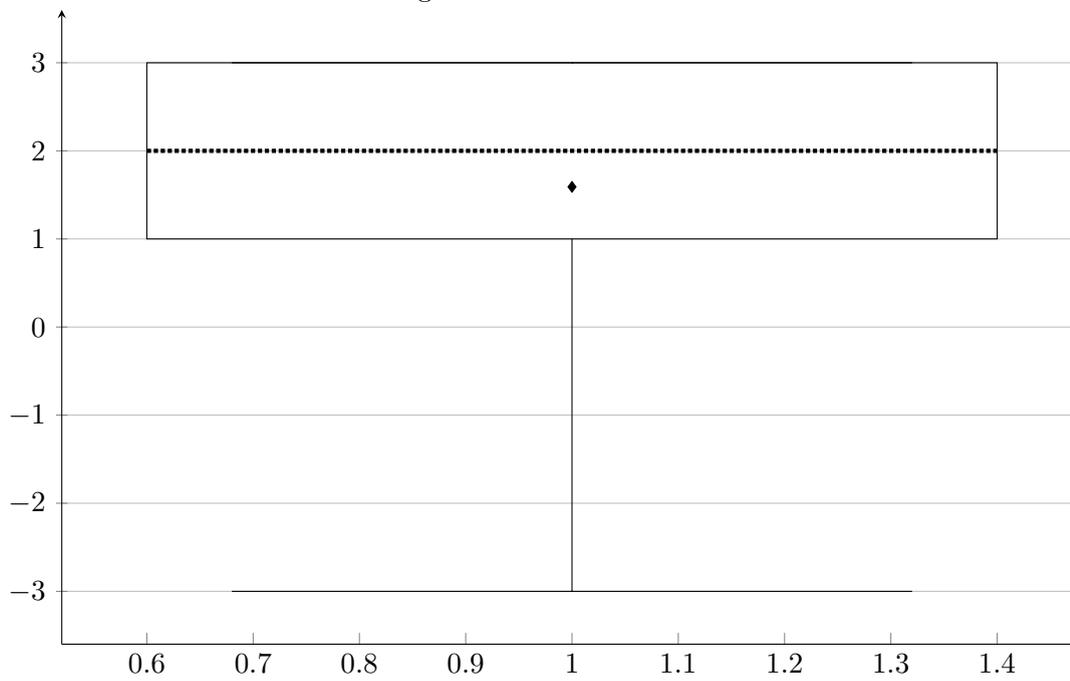
In dem Plot 7.10 kann man erkennen, dass die meisten Nutzer diese Unabhängigkeit als Vorteil gesehen haben. Die Kamera bewegt sich und zeigt nicht immer den richtigen Bildausschnitt.

Aber es gibt auch Bewertungen, die diese Unabhängigkeit als sehr schlecht eingestuft haben. Dieses hat 2 in den Bemerkungen genannte Gründe. Zum einen wird der Strahl in der HoloLens nicht mit der Geometrie geschnitten und endet damit am ersten Schnittpunkt, sondern geht durch das Objekt hindurch. Stehen Experte und lokaler Nutzer auf unterschiedlichen Seiten des Objekts und der Strahl geht schräg durch das Modell, so zeigt er auf der Austrittsseite einen falschen Schnittpunkt. Dieser kann zu Verwirrungen führen. Ein zweiter Grund ist, dass wir die gleiche Perspektive gewöhnt sind. Wenn wir entlang des Strahls schauen, ist es einfach er zu erkennen, wo er aktuell schneidet. Steht man senkrecht dazu, ist es schwerer zu erkennen, wo die genauen Schnittpunkte sind.

7.5.6. Allgemeines

HoloLens kliense FOV schwer schlechte Auflösung

Abbildung 7.10.: War es im VR/AR Szenario von Vorteil, von der Perspektive und Bewegung der anderen Person unabhängiger zu sein und nicht an die Ansicht aus des Videos gebunden zu sein?



8. Fazit und Ausblick

In dieser Arbeit wurde eine Möglichkeit vorgestellt, um das Zusammenarbeiten eines lokalen Benutzers mit einem nicht anwesenden Experten an einem Objekt zu realisieren. Verwendet wurde dabei die Vive, die Kinect und die HoloLens. Hierzu wurde zunächst eine Methode vorgestellt, mit einer Kinect und dem Lighthouse Tracking der Vive einen Punktwolken-Scan des Objektes anzufertigen. Der Experte und der lokale Nutzer können dann gemeinsam mit einer Zeigegeste an demselben Objekt arbeiten. Der Experte sieht die 3D Punktwolke und den Beam, wobei der lokale Nutzer den Beam in einer AR Umgebung am echten Objekt eingeblendet bekommt.

Dieses Verfahren und die noch relativ neue Hardware bringt einige Probleme mit, jedoch zeigt die Nutzerstudie, dass ein Zusammenspiel in VR und AR attraktiver, im Durchschnitt schneller und weniger fehleranfällig ist.

Das vorgestellte Aufnehmen der Punktwolke mit Kinect und Lighthouse Tracking bietet eine gute und schnelle Grundlage für einen 3D Scan, ist aber zum direkten Verwenden noch zu ungenau. Durch Verbesserung der Kalibrierung des Controllers zu Kinect und das Tracking des Controllers könnte dieses Verfahren eine ideale Grundlage für schnelle Aufnahmen bieten. Bei der Kalibrierung zueinander könnte man einen anderen Sensor verwenden, bei dem die Position im Gehäuse besser dokumentiert ist. Außerdem könnte ein Vive Tracker anstelle des Controllers bessere Ergebnisse bringen, da der Ursprung des Trackers geschickter liegt. In praktischen Anwendungen sollten die Punktwolkenaufnahmen nachbearbeitet werden. Die einzelnen Teile der Punktwolke haben vergleichsweise kleine Verschiebungen und besitzen häufig gemeinsame Flächen. An diesen könnte im Nachhinein die exakte Punktwolke berechnet werden.

Ein weiterer Kritikpunkt aus der Evaluation sind die zu ungenauen Punktwolken und das normale Nutzer es nicht gewöhnt sind mit Punktwolken zu arbeiten. Generiert man ein Mesh aus der Wolke, dann könnte dieses eine verbesserte Darstellung ergeben.

Ein weiterer Nachteil an den in dieser Arbeit verwendeten Punktwolken ist, dass diese statisch sind. Verändert der lokale Nutzer den betrachteten Bereich, dann erhält der Experte darüber keine Infos. Ist die Aufnahmetechnik performant genug, könnte man dem lokalen Nutzer regelmäßig neue Aufnahmen machen lassen. Alternativ positioniert man mehrere Kinects aus verschiedenen Richtungen im Raum, sodass die Area of Interest von allen benötigten Richtungen gescannt wird. Diese Daten könnten live in die VR Umgebung gestreamt und visualisiert werden. Mit einer live Punktwolke kann das Objekt in der Eva-

luation auch verändert werden. Problematisch könnte die Verdeckung durch den lokalen Nutzer sein.

Die Kalibrierung von Vive zu HoloLens und die Ungenauigkeiten im Tracking bringen weiter. Hier gibt es an einigen Stellen Verbesserungspotenzial. Die einfache Lösung wäre, die HoloLens selber mit einem Tracker zu versehen und man erhält einen dauerhaft sich selber aktualisierenden gemeinsamen Punkt. Damit eliminiert man die Ungenauigkeit aus dem HoloLens Tracking. Das Problem, dass das Lighthouse Tracking nicht längengetreu ist, wird dabei nicht gelöst. Eine weitere Möglichkeit wäre es, Vive Tracker mit Markern zu versehen. Diese können dann mit der HoloLens erkannt werden und in dem Koordinatensystem der HoloLens platziert werden. Damit umgeht man den menschlichen Kalibrierfehler und wenn man mehrere Tracker mit Markern verwendet, erhält man die Längen in der HoloLens und der Vive. Mit dieser Information kann man die exakte Längenverzerrung ausrechnen und beseitigen.

Das verwendete Evaluationszenario war relativ einfach, sehr abhängig vom Tracking, der Kalibrierung und den jeweiligen Probanden. Bei weiteren Untersuchungen könnten komplexere Szenarien evaluiert werden, um die Vorteile der VR Umgebung klarer herauszuarbeiten. Um Ungenauigkeiten im Tracking auszugleichen, könnte man größere Objekte verwenden als Duplos. Eine Möglichkeit wäre es, einen ca. 1m großen Würfel auf jeder Seite mit farbigen Schachbrettmustern zu versehen. Bei großen Feldern sollten Trackingungenauigkeiten ein kleineres Problem darstellen und in einem vergleichbaren Videostream wäre die Orientierung noch schwieriger. Jedoch muss sich bei dem Szenario überlegt werden, wie das Vorwissen dem Experten vermittelt wird. In VR ist das relativ einfach möglich durch Highlights oder eingeblendete Pfeile, im Referenzszenario ist dieses schwieriger.

Beim Zusammenspiel im VR und AR könnten mehr Interaktionstechniken evaluiert werden. Bisher hat der lokale Nutzer keine Möglichkeit, dem Experten etwas zu zeigen. Die einfachste Lösung hierfür wäre, dem lokalen Nutzer selbst einen Controller zu geben mit dem er auf das Objekt zeigen kann. Alternativ könnte auf die Klick-Geste der HoloLens zurückgegriffen werden. Klickt der lokale Nutzer in der Welt etwas an, so wird die Linie zwischen Kopf und der Klickposition an den Experten übertragen und dort visualisiert.

Eine weitere hilfreiche Ergänzung könnte das Visualisieren von Avataren in VR und AR sein. Allein die Visualisierung der aktuellen Kopfposition (Headsets) des Partners könnte darüber Aufschluss geben, was dieser gerade betrachtet.

Eine weitere Interaktionsmöglichkeit wäre das Platzieren von 3D Objekten bzw. Hologrammen in der Welt. Der Experte hätte damit z.B. die Möglichkeit, Referenzobjekte direkt darzustellen. Bei animierten Hologrammen könnten so direkt Montageschritte visualisiert werden. (cite **Virtual Proxy**)

ToDo

A. 3D Tile JSON

In diesem Anhang finden Sie das verwendete Tileset der 3D Tiles

```
1 {
2   "asset": {
3     "version": "0.0"
4   },
5   "geometricError": 0,
6   "root": {
7     "boundingVolume": {
8       "sphere": [
9         0,
10        0,
11        0,
12        5
13      ]
14    },
15    "geometricError": 0,
16    "refine": "add",
17    "children": [
18      {
19        "transform": [
20          1.0, 0.0, 0.0, 0.0,
21          0.0, 1.0, 0.0, 0.0,
22          0.0, 0.0, 1.0, 0.0,
23          0.0, 0.0, 0.0, 1.0
24        ],
25        "boundingVolume": {
26          "sphere": [
27            0,
28            0,
29            0,
30            5
31          ]
32        },
33        "geometricError": 0,
34        "content": {
35          "url": "example.pnts"
36        }
37      }
38    ]
39  }
40 }
```


Abbildung B.1.: Der Anfangsfragebogen

Abbildung B.2.: Der Fragebogen nach jedem Versuchsdurchlauf

Abbildung B.3.: Der Abschlussfragebogen

Abbildung D.4.: Vorbereitungszeit des Experten als Boxplot

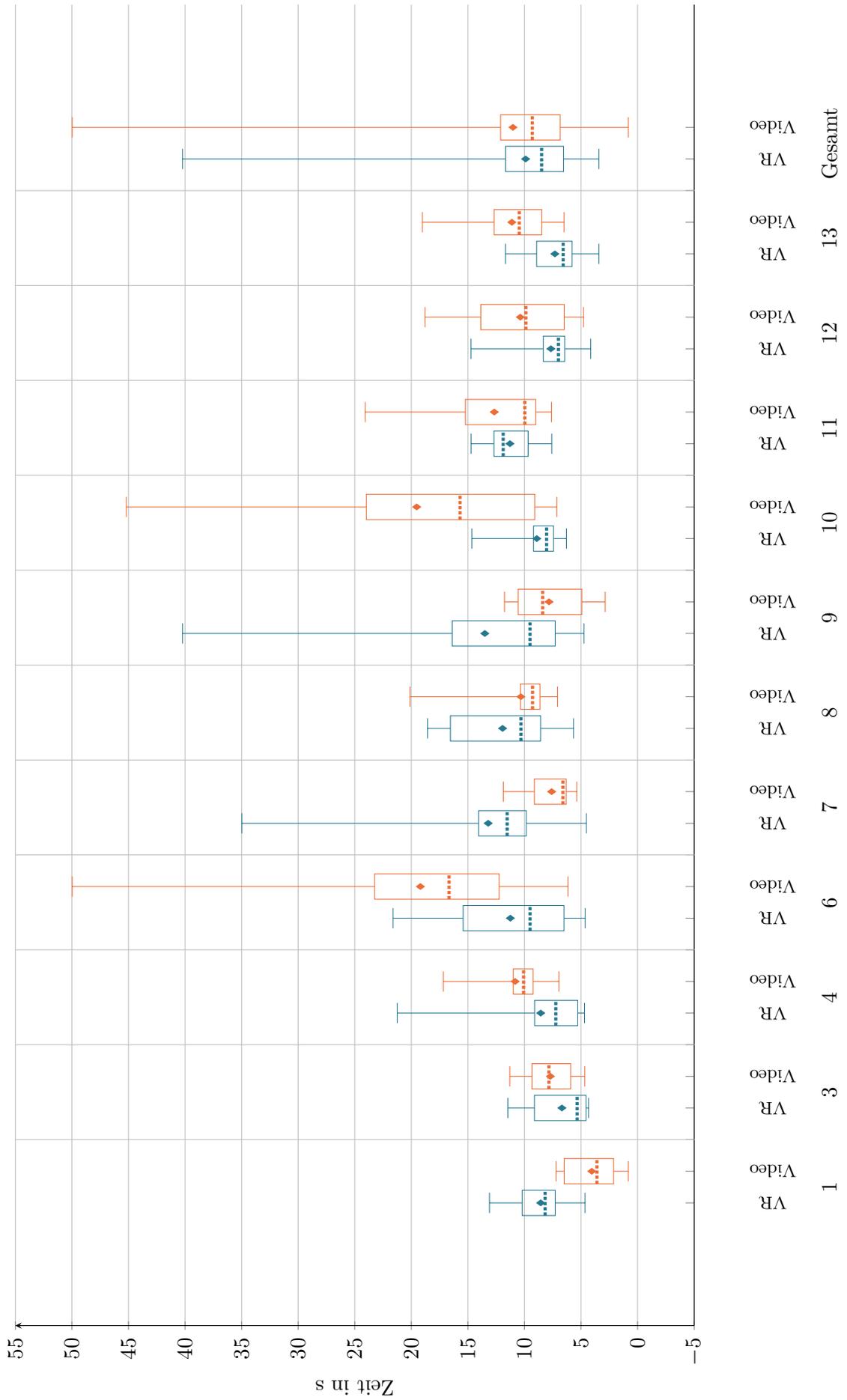
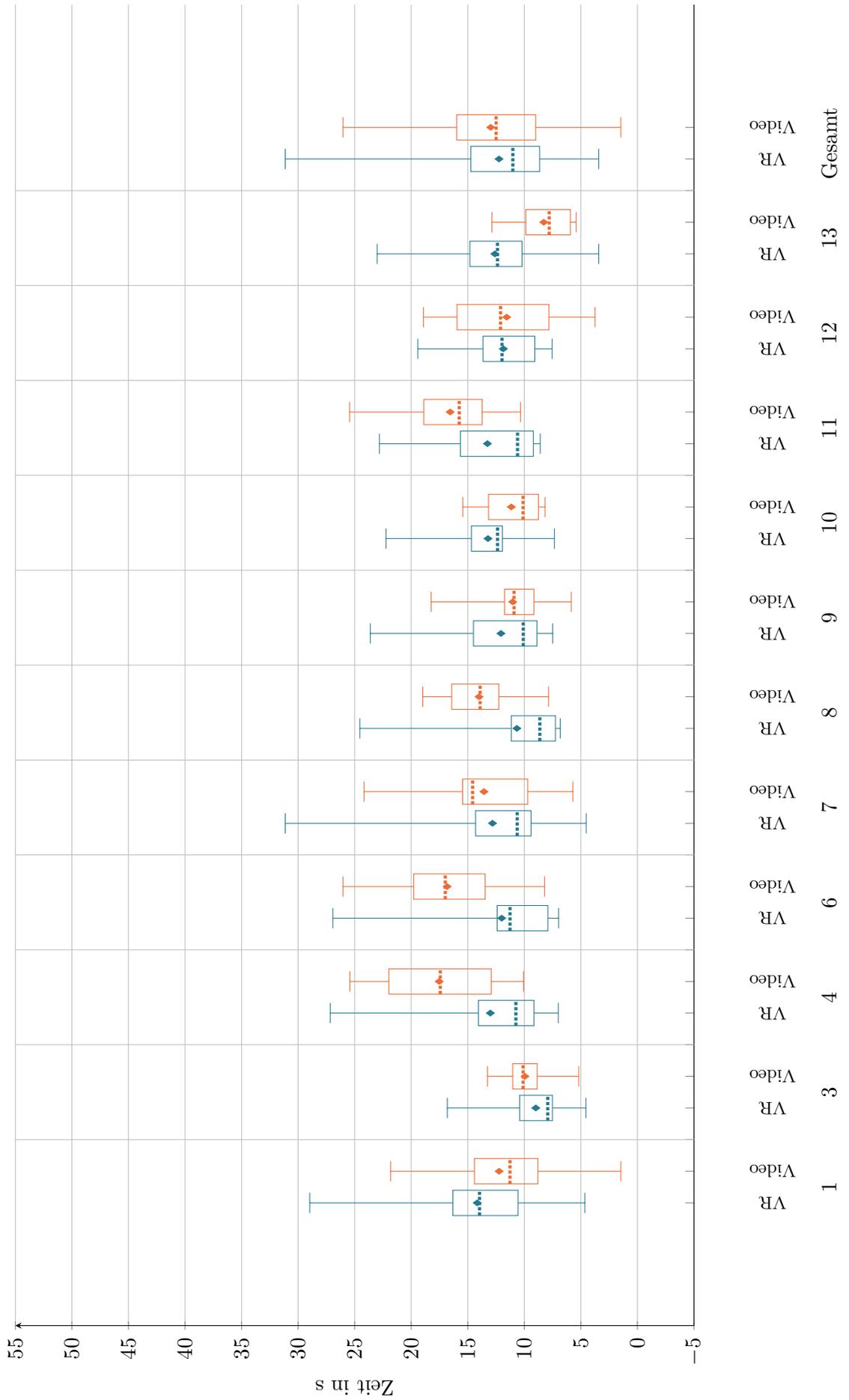


Abbildung D.5.: Kommunikationszeit



Literaturverzeichnis

- [3DT] *3d tiles spezifikation.* <https://github.com/AnalyticalGraphicsInc/3d-tiles>. Accessed: 2017-09-13.
- [GLT] *GlTF spezifikation.* <https://github.com/KhronosGroup/glTF>. Accessed: 2017-09-13.
- [Har06] Sandra G. Hart: *Nasa-task load index (nasa-tlx); 20 years later.* Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 50(9):904–908, 2006. <https://doi.org/10.1177/154193120605000909>.
- [Kina] *Kinect dokumentation koordinatensysteme.* <https://msdn.microsoft.com/de-de/library/dn785530.aspx>. Accessed: 2017-11-02.
- [Kinb] *Kinect tiefensensor position.* <https://social.msdn.microsoft.com/Forums/sqlserver/ja-JP/05a6d2b3-9096-4236-b77a-691c5f047066/kinect-for-windows-v2-?forum=windowsgeneraldevelopmentissuesja>. Accessed: 2017-11-02.
- [lig] *Lighthouse tracking examined.* <http://doc-ok.org/?p=1478>. Accessed: 2017-11-02.
- [LM] Wolfgang Ludwig-Mayerhofer: *Signifikanztests (so) kurz (wie möglich) erklärt.* https://www.uni-siegen.de/phil/sozialwissenschaften/soziologie/mitarbeiter/ludwig-mayerhofer/statistik/statistik_downloads/signifikanztests.pdf.
- [MCS14] Manuel Martin, Florian van de Camp, and Rainer Stiefelhagen: *Real time head model creation and head pose estimation on consumer depth cameras.* In *Proceedings of the 2014 2Nd International Conference on 3D Vision - Volume 01, 3DV '14*, pages 641–648, Washington, DC, USA, 2014. IEEE Computer Society, ISBN 978-1-4799-7000-1. <http://dx.doi.org/10.1109/3DV.2014.54>.
- [NLL17] Diederick Christian Niehorster, Li Li, and Markus Lappe: *The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research.* In *i-Perception*, 2017.
- [OES⁺15] Ohan Oda, Carmine Elvezio, Mengü Sukan, Steven Feiner, and Barbara Tversky: *Virtual replicas for remote assistance in virtual and augmented reality.* In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, UIST '15*, pages 405–415, New York, NY, USA, 2015. ACM, ISBN 978-1-4503-3779-3. <http://doi.acm.org/10.1145/2807442.2807497>.
- [UE4a] *Unreal engien hololens template.* <https://github.com/ProteusVR/Hololens><https://www.youtube.com/watch?v=KxvAm2qNJOQ&feature=youtu.be>. Accessed: 2017-11-02.
- [UE4b] *Unreal engine 4.* <https://www.unrealengine.com/en-US/blog>. Accessed: 2017-11-02.

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, xx.xx.xx

.....

(Max Mustermann)