Wahrnehmung der virtuellen Präsenz durch RGB-D-Videostreams in Augmented Reality

BACHELORARBEIT

für die Prüfung zum

BACHELOR OF SCIENCE

des Studienganges

Informationstechnik

an der Dualen Hochschule Baden-Württemberg Mannheim

von

Anna-Lena Ehmer

Abgabe: 07.09.2021

Bearbeitungszeitraum	14.06.2021 - 06.09.2021
Matrikelnummer, Kurs	5007696, TINF18IT1
Ausbildungsfirma:	Deutsches Zentrum für Luft- und Raumfahrt e.V.
Betreuer der Ausbildungsfirma	Frau Anna Bahnmüller
Gutachter der Dualen Hochschule	Herr Prof. Dr. Konstantin Bayreuther

Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit mit dem Thema Wahrnehmung der virtuellen Präsenz durch RGB-D-Videostreams in Augmented Reality, selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.
Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Braunschweig, den 4. September 2021

Genderhinweis: Aus Gründen der besseren Lesbarkeit wird auf eine geschlechtsneutrale Differenzierung verzichtet. Entsprechende Begriffe gelten im Sinne der Gleichbehandlung grundsätzlich für alle Geschlechter. Die verkürzte Sprachform beinhaltet keine Wertung.

Kurzfassung

Die Zusammenarbeit zwischen mehreren Personen, welche sich an verschiedenen Orten befinden, bildet in der heutigen Welt eine grundlegende Notwendigkeit. Durch sie wird es ermöglicht, dass Experten zu Problemen befragt werden können, ohne dass diese weite Anreisewege auf sich nehmen oder Objekte transportiert werden müssen.
Zu den ursprünglichen Videokonferenzen, welche mittels einer Webcam durchgeführt werden, können mittlerweile auch AR und VR-basierte Systeme verwendet werden, so dass sich die verschiedenen Parteien im digitalen Raum gegenüber stehen und miteinander interagieren können. Dabei werden häufig Avatare von Personen eingesetzt, welche das Aussehen der Person nachahmen und so ein persönliches Gefühl vermitteln sollen. Zudem können diese Avatare sich wie eine Person bewegen, so dass sie Erklärungen mit Gestiken unterstützen. Alternativ zu diesen Avataren können aber auch Punktwolken verwendet werden, welche durch Daten von Tiefenkameras erschaffen werden. Diese Punktwolken bietet den Vorteil, dass sie immer dem aktuellen Bild einer Person entsprechen und zusätzlich auch Objekte aufgenommen werden können, ohne dass neue Avatare erzeugt werden müssen.
Im Rahmen dieser Arbeit wird die Azure Kinect DK als Tiefenkamera verwendet, von welcher über ein Netzwerk eine RGB-D-Videoübertragung zu einer Microsoft HoloLens 2 übertragen wird. Diese stellt die Daten als Punktwolke in Augmented Reality (AR) dar. Dabei soll die Wahrnehmung eines solchen Streams durch Personen beurteilt werden. Im Rahmen dieser Frage wird eine Nutzerstudie erstellt, durchgeführt und ausgewertet. Hierbei werden die Einflussfaktoren Auflösung, Latenz und Bildwiederholrate genauer betrachtet.

Abstract

Collaboration between several people located in different places is a fundamental necessity nowadays. It enables experts to be consulted on problems without having to travel long distances or transport objects.
In addition to the original video conferences, which are conducted using a webcam, AR and VR-based systems can now be used so that the various parties can face each other in digital space and interact with each other. This often involves the use of avatars of people, which are designed to mimic the appearance of the person and thus convey a personal feeling. In addition, these avatars can move like a person so that they support explanations with gestures.
As an alternative to these avatars, however, point clouds can also be used, which are created using data from depth cameras. These point clouds offer the advantage that they always correspond to the current image of a person and, in addition, objects can be included without the need to create new avatars.
In the context of this work, the Azure Kinect DK is used as a depth camera, from which an RGB-D video transmission is transferred to a Microsoft HoloLens 2 via a network. The HoloLens 2 displays the data as a point cloud in AR. Thereby, the perception of such a stream by persons is to be assessed. In the context of this question, a user study is created, carried out and evaluated. Here, the influencing factors resolution, latency and frame rate are examined in more detail.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Abkürzungsverzeichnis

AR	Augmented Reality
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V.
FPS	Frames per Second
HMD	Head Mounted Display
RGB	Rot-Grün-Blau
SC	Institut für Softwaretechnologie
SUS	System-Usability-Scale
ToF	Time-of-Flight
VR	Virtual Reality

1. Einleitung

Am Institut für Softwaretechnologie (SC) des Deutschen Zentrums für Luft- und Raumfahrt e.V. (DLR) wird an innovativen Software-Engineering-Technologien geforscht. Dabei erfolgt eine intensive Zusammenarbeit mit anderen Instituten, so dass auch diese von den neusten Erkenntnissen profitieren können.[1]
Innerhalb der Gruppe 3D - Visualisierung wird vorwiegend mit AR und Virtual Reality (VR) Technologien gearbeitet. Dabei werden Applikationen erstellt, welche entweder für die Arbeit einer einzelnen Person oder auch für die Zusammenarbeit von mehreren Personen gedacht sind.
So wurden Projekte verwirklicht, welche beispielsweise Simulationen für die Raumfahrt auf AR - Brillen ermöglichen[2] oder an aktuellen weltweiten Herausforderungen, wie der digitalen Zusammenarbeit in der Flugzeugwartung.[3]
Im Rahmen dieser Projekte, sowie der wachsenden Globalisierung, wird die Zusammenarbeit von Personen an verschiedenen Orten immer bedeutsamer. Dabei steht nicht nur die akustische Verbindung der Personen im Sinne eines Telefonats zwischen diesen im Vordergrund, sondern oftmals auch eine visuelle Verbindung. Ein Zuwachs der digitalen Zusammenarbeit konnte dabei schon seit längerem beobachtet werden. Allerdings wurde dies durch die Coronapandemie noch einmal beschleunigt.[4]
Um während einer visuellen Verbindung die Position der zweiten Person einschätzen zu können, sollte eine perspektivische oder besser noch 3D-Ansicht von dieser verfügbar sein. Somit kann bsw. "der Blick über die Schulter" digital umgesetzt werden.
Für die digitale Darstellung von Personen werden oftmals Avatare verwendet, welche die Position der zweiten Person verdeutlichen. Dabei ist es notwendig, ein 3D-Modell einer Person zu erstellen, welches an Gelenken beweglich ist. Dabei muss das Modell sehr komplex erstellt werden, um Bewegungen verdeutlichen zu können.
Ein neuartiger Ansatz hierbei ist der Einsatz von Punktwolken, welche mithilfe der Informationen einer Tiefenkamera erstellt werden. Durch die Veröffentlichung der Azure Kinect DK Kamera[5] im März 2020, einer günstigeren Lösung einer guten Tiefenkamera, ergeben sich hierbei neue Möglichkeiten. Dabei soll im Rahmen dieser Arbeit die Frage gestellt werden, wie Menschen einen RGB-D Videostream wahrnehmen. Hierfür wird eine Nutzerstudie mit einer Applikation durchgeführt, welche das Bild einer Azure Kinect DK auf der Microsoft HoloLens 2 darstellt.[6] Bei der HoloLens 2 handelt es sich um eine 2019 veröffentlichte kabellose AR-Brille.

[1] vgl. DLR21.
[2] vgl. Muh+21.
[3] vgl. Utz+19.
[4] vgl. eV21, S.46.
[5] Mic20.
[6] Doc20b.

2. Problemstellung

2.1. Zielsetzung

Im Rahmen dieser Bachelorarbeit wird die Wahrnehmung von RGB-D Videostreams untersucht. Um eine Aussage zu ermöglichen, entsteht eine Applikation mit der Spiel-Engine Unity[7] für die AR-Brille Microsoft HoloLens 2[8], welche für eine Nutzerstudie verwendet wird. Dabei ist angedacht, zu ermöglichen, Echtzeitdaten einer Azure Kinect[9] darzustellen. Dabei ist geplant, die Daten als RGB-D-Videostream aufzufassen. Somit sind neben den normalen Farbwerten auch die Tiefendaten vorhanden, welche auf die Microsoft HoloLens 2 weiter zu leiten sind. Es soll eine Punktwolke erstellt werden, welche ihre Position und Farbgebung anhand des Streams zugewiesen bekommt.
Das Ziel der fertigen Applikation ist es, einer Person, welche die Microsoft HoloLens 2 trägt, die 3D-Ansicht eines Objekts als Hologramm zu ermöglichen. Dabei entspricht dieses Hologramm dem Bild, welches eine Azure Kinect DK, im folgenden Kinect genannt, aufnimmt. Alternativ können ein Objekt und eine zweite Person gefilmt und diese Daten an die erste Person übertragen werden. Dabei ist für sie deutlich erkennbar, wie die erste Person und das Objekt zueinander ausgerichtet sind.
Diese Anwendung wird so ausgebaut, dass sie für eine Nutzerstudie verwendet werden kann. Ziel dieser Nutzerstudie ist es, eine Aussage darüber treffen zu können, welche Faktoren dieser Übertragung für den Nutzer den stärksten Einfluss haben. Besonders von Interesse sind die Auflösung und die Bildwiederholrate (FPS) sowie die aus diesen beiden Parametern resultierende Latenz. Hierfür wird ein Testszenario entwickelt, welches mittels eines Fragebogens ausgewertet werden kann.

Die Arbeit gliedert sich in die folgenden Meilensteine, welche in aufsteigender Reihenfolge bearbeitet werden:

1. Darstellung der Daten der Microsoft Kinect als Punktwolke auf der Microsoft HoloLens 2

2. Konzeption und Erstellung einer Szene für eine Nutzerstudie

3. Durchführung der Nutzerstudie und Auswertung der Ergebnisse dieser Arbeit

[7] Uni21.
[8] Doc20b.
[9] vgl. Mic20.

2.2. Vorgehen

Um die im vorherigen Abschnitt genannten Meilensteine zu erreichen, werden folgende Aufgaben in der genannten Reihenfolge bearbeitet. Dabei ist das folgende Vorgehen geplant:

1. Einarbeitung in die theoretischen Grundlagen
    a) Allgemeine Definition von AR und VR
    b) Grundlagen Microsoft HoloLens 2
    c) Grundlagen Tiefenkamera
    d) Grundlagen aktueller Forschungsstand
2. Analyse der Anforderungen
    a) Anforderungen an die Anwendung
    b) Anforderungen an die Nutzerstudie
3. Erstellung eines Konzepts für die Darstellung der Punktwolke auf der Microsoft HoloLens 2
    a) Aufbau der Applikation
    b) Netzwerksverbindung
    c) Applikation Sender
    d) Applikation Empfänger
    e) Konzept Nutzerstudie
4. Umsetzung der Konzepte
    a) Netzwerk
    b) Host
    c) Client
    d) Nutzerstudie
5. Durchführung der Nutzerstudie
6. Auswertung der Nutzerstudie

3. Grundlagen

In diesem Kapitel werden die Grundlagen hinsichtlich AR vorgestellt. Dabei wird auf die Begriffsdefinition als auch auf die Abgrenzung zu VR eingegangen.
Anschließend erfolgt die Vorstellung der AR-Brille Microsoft Hololens 2, welche im folgenden als HoloLens bezeichnet wird, mit deren technischen Daten. Weiterhin wird die weitere wichtige Hardwarekomponente, die Kinect sowie die Funktionsweise einer Tiefenkamera präsentiert.
Abschließend wird der aktuelle Stand im Bereich der Darstellung von Punktwolken im Vergleich zu Avataren dargestellt.

3.1. VR/AR

Die Begriffe der virtuellen und erweiterten Realität können unter der Bezeichnung "gemischte Realität" (mixed reality) zusammengefasst werden. Dieser werden alle Darstellungsformen zugeordnet, welche weder nur aus der Realität noch aus einer digitalen Darstellung bestehen (siehe Abbildung 1).[10]

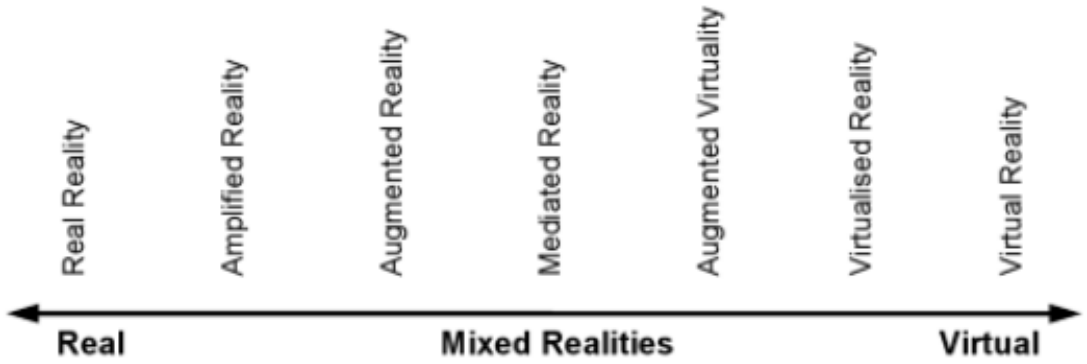

Abbildung 1: Darstellung des Spektrums von Realitäten[11]

Die einzelnen "Realitäten"lassen sich wiederum auf VR und AR aufteilen. Sie unterscheiden sich darin, ob die reale Welt um digitale Inhalte ergänzt wird (AR), diese die reale Welt überlagern, einzelne Teile der realen Welt in die digitale übernommen werden oder man sich komplett in einer digitalen Welt (VR) befindet. Die Grenzen verlaufen dabei fließend mit zahlreichen Mischformen. So zeigen etwa einige VR-Brillen die reale Welt auf ihrem Display an. Somit wird auch die reale Welt mit digitalem Inhalt überlagert, allerdings muss die reale Welt nicht original dargestellt werden. Daher ist die Einordnung einer solchen Technik schwer genau zu definieren.

[10] Xia08, vgl. S. 3f.
[11] Xia08, vgl. S. 5f.

Dieses Spektrum ist in Abbildung 2 dargestellt.

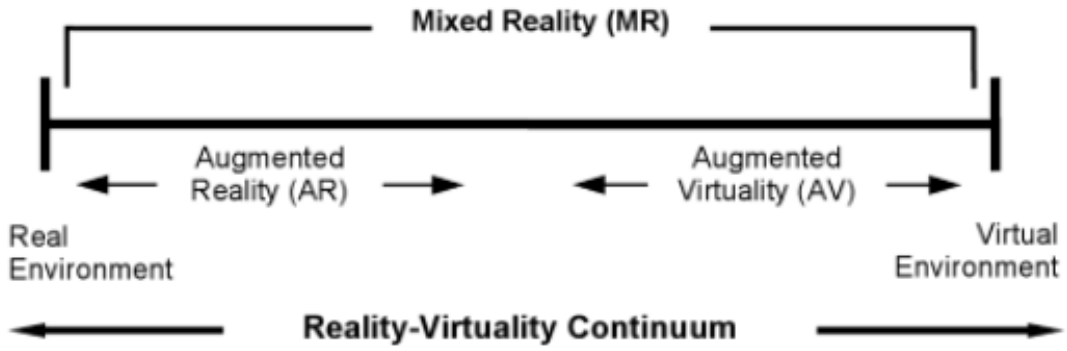

Abbildung 2: Darstellung des Spektrums von AR zu VR [12]

In dieser Arbeit wird AR-Technologie verwendet. Diese kann durch verschiedene Geräte einem Nutzer zugänglich gemacht werden. Dabei werden heutzutage zumeist Head Mounted Displays (HMD) verwendet. Ein HMD beschreibt dabei ein Gerät, welches eine brillenartige Konstruktion besitzt und auf dem Kopf des Nutzers getragen wird. Dabei blickt dieser auf durchsichtige Scheiben, auf welche mittels verschiedener Techniken Hologramme gebracht werden können. Wie in Abbildung 3 gibt es diese als Peripheriegeräte (links im Bild) oder als eigenständige holografische Computer (rechts im Bild).[13]

Abbildung 3: Verschiedene Arten von AR-Brillen[14]

[12] Xia08, vgl. S. 4f.
[13] Abh17, vgl. S. 10.
[14] Bas21.

Dabei werden immer mehr eigenständige Geräte bevorzugt, da diese dank modernster Technik immer mehr Rechenleistung ermöglichen.
Kabel-gebundene Geräte sind vor allem dann im Einsatz, wenn besonders aufwendige Rechentechnik benötigt wird. Zudem sind sie selbst meist leichter und schmaler gebaut und bieten so für eine lange Tragedauer einen höheren Komfort.
Andererseits ist die Verbindung ein einschränkender Faktor, da so eine Bewegung nur im Rahmen des verbindenden Kabels möglich ist. Geräte ohne Verbindung sind von ihrer Leistungsfähigkeit eingeschränkt, da sämtliche Rechentechnik in der Brille verbaut sein muss. Somit ist allein aus platz- und gewichtstechnischer Sicht eine Grenze gegeben.
Jedes dieser Geräte besitzt ein Display, auf welches mittels verschiedener Techniken ein Bild erzeugt wird. Zudem sind Sensoren verbaut, welche die Umgebung und Bewegung des HMD erfassen.

3.2. HoloLens

Die Microsoft HoloLens 2 ist eine im November 2019 erschienene AR-Brille. Hierbei handelt es sich um eine brillenartige Konstruktion, welche Hologramme in das Sichtfeld des Nutzers projiziert.[15] Somit erscheint es dem Nutzer so, als würden sich die 3D-Objekte in der realen Welt befinden. Diese Art des Einfügens von virtuellen Bildern in die reale Welt wird als Augmented Reality bezeichnet.
Das Besondere an der HoloLens, sowohl bei HoloLens 1 (erste Generation aus dem Jahr 2016)[16] als auch der HoloLens 2 (zweite Generation) ist, dass sie als eigenständiges Gerät verwendet werden kann und keine dauerhafte Verbindung zu einem Computer benötigt.[17] Dies ist möglich, da sich die Recheneinheit und der Speicher am Kopfband befinden.[18] Der Aufbau der HoloLens 2 ist in Abbildung 4 dargestellt.
Zudem sind seit der neuen Generation Augen- und Finger-Tracking gegeben, welche eine einfachere Bedienung ermöglichen sollen. Ebenso ermöglichen die erweiterten, integrierten Steuergesten eine umfangreichere Bedienung der virtuellen Objekte wie z.B. das natürliche Betätigen eines Schalters durch Hand-Drück-Bewegungen oder das Auswählen eines Objekts durch Anblicken von diesem.
Durch eine große Anzahl an Sensoren, etwa normale Kameras und eine Infrarotkamera, einen Tiefensensor sowie Beschleunigungsmesser, Gyroskop und Magnetometer wird die Umgebung umfangreich aufgezeichnet und ausgewertet.

[15] vgl. Doc20b.
[16] vgl. Doc21.
[17] vgl. A J17, p.10.
[18] vgl. Doc20c.

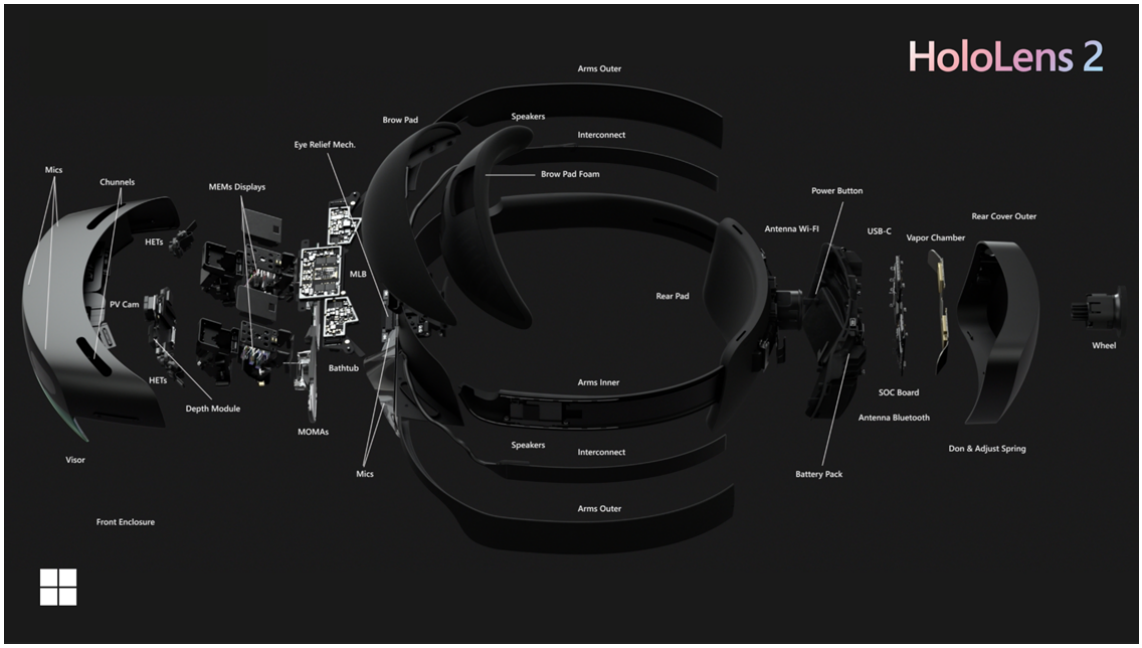

Abbildung 4: Explosionsdarstellung der HoloLens 2 [19]

So ist es möglich, mit realen Gegenständen im Raum zu interagieren und diese mit virtuellen Inhalten anzureichern sowie in die argumentierte Realität einzubeziehen. Zudem sind mehrere Mikrofone und Lautsprecher entlang des Kopfbandes verbaut, um einen möglichst realitätsnahen Klang zu ermöglichen.[20]
Für die Entwicklung von Anwendungen für die HoloLens 2 wird die Spiele-Engine Unity3D oder Unreal empfohlen.[21] Im Folgenden soll die Applikationserstellung mit Unity3D näher betrachtet werden. Die Wahl auf Unity3D ist damit zu begründen, dass mit dieser Engine zuvor schon gearbeitet wurde und bekannte Lösungen verwendet werden können.
Die Spielengine Unity3D ermöglicht es, Applikationen für verschiedene Plattformen zu erstellen. Dabei werden diese im Editor der Engine erarbeitet. Diese ist Anhand von Szenen organisiert, welche in der Applikation dann automatisch durchlaufen werden. Dabei wird beispielsweise durch das "Mixed-Reality-Toolkit"der Import von bereits bestehenden Objekten oder Funktionalitäten ermöglicht. Diese können mittels Programmcode in C# verändert werden. Zudem ist es möglich, den Ablauf von Events zu bestimmen.
Sobald das Projekt in Unity fertiggestellt wurde, kann es gebaut werden. Um es

[19] Doc20b.
[20] vgl. Doc20c.
[21] vgl. Doc20a.

beispielsweise auf der HoloLens zu verwenden, wird es mittels Visual Studio 2019 auf diese übertragen und kann so mit einem Debug Log gestartet werden.

3.3. Tiefenkamera

Tiefenkameras werden dazu verwendet, zu ermitteln, in welcher Entfernung sich ein Objekt insbesondere von der Kamera befindet. Vorreiter bei dieser Erforschung war insbesondere die Spielindustrie, da aus der Erfassung der Spieler ein neues Spielerlebnis generiert werden konnte.[22]Es gibt verschiedene Arten von Tiefenkameras, welche im Folgenden kurz erklärt werden.
Tiefenkameras werden anhand der Verfahren unterschieden, welche sie zur Entfernungsmessung verwenden. Dabei sind hauptsächlich geometrische Berechnungen oder die Messung der Laufzeit eines Lichtstrahls Grundlage für die Entfernungsberechnung.
Erst genannte Methode lässt sich im einfachsten Fall mithilfe eines Laserpointers, eines Detektors und eines Spiegels erklären. Der Laserpointer wird auf einen beliebig weit entfernten Spiegel gerichtet. Dabei sollte der Winkel zwischen Laserstrahl und Spiegel mehr als 90 Grad betragen und bekannt sein. Der Spiegel reflektiert den Laserpunkt und stahlt ihn zurück. Mithilfe eines Detektors kann nun erkannt werden, wie weit oberhalb des Austrittspunkts des Lasers sich der reflektierte Lichtpunkt befindet. Mithilfe von trigonometrischen Berechnungen kann so bestimmt werden, wie weit entfernt sich der Spiegel befinden.

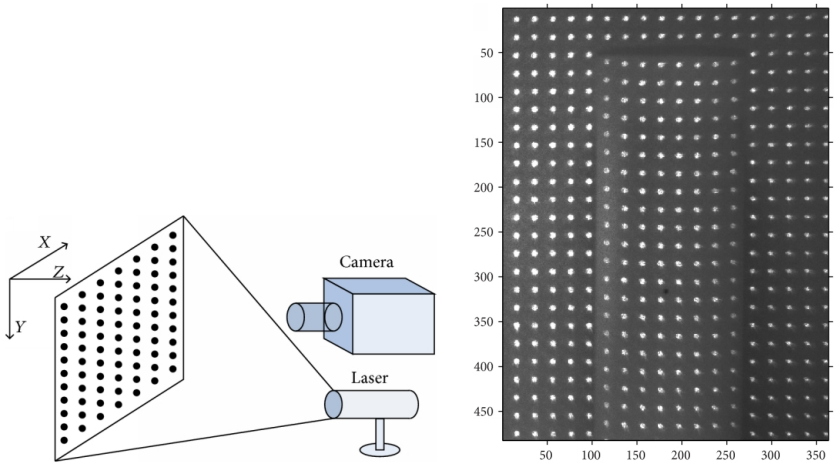

Abbildung 5: Links: Messaufbau eines auf eine senkrechte Wand projiziertes Infrarotmuster. Rechts: Entfernungsmessung anhand eines Infrarotmusters auf einem Gegenstand. [23]

[22]Alb+20, vgl. S.1.
[23]Ton14.

Eine Weiterentwicklung dieses Prinzips, lässt sich mittels eines Infrarotsensors und einer Infrarotlampe umsetzen. Dabei sendet die Lampe mithilfe von Laserstahlen ein vordefiniertes Muster, welches zumeist aus Quadraten besteht, aus. Dieses Muster wird mit einem Infrarotsensor aufgezeichnet. Ein solcher Aufbau ist in Abbildung 5 links dargestellt. Sofern das Muster an eine genau Senkrecht zur Kamera stehende Wand gerichtet ist, ergibt sich ein symmetrisches Muster. Wird es hingegen, wie in Abbildung 5 rechts, auf einen Gegenstand geworfen, verschiebt sich dieses Muster. Anhand der Verschiebung der Punkte zueinander kann bestimmt werden, in welchem Abstand sich diese von dem Sensor befinden. Hierfür ist es notwendig, den Abstand zwischen Sender und Sensor zu kennen. Dieses Prinzip wurde beispielsweise bei der Kinect v1 verwendet, um es Spielern zu ermöglichen, ohne Controller mit Spielinhalten zu interagieren.[24]
Ein weiteres häufig verwendetes Verfahren basiert auf der Laufzeitmessung von Licht. Diese wurde anfangs bei Entfernungsmessern, etwa zum gleichmäßigen Ausmessen von Räumen, verwendet. Dabei wird ein Lichtimpuls abgesendet und anhand des Zeitpunktes des Eintreffens der Reflexion bestimmt, wie lange der Impuls benötigte. Anhand dieser Daten und der Information, dass Licht die gleiche Zeit zum Objekt und wieder zurück erfordert, kann berechnet werden, wie weit das Objekt entfernt ist. Dieser Aufbau sowie die Berechnung sind in Abbildung 6 dargestellt. Dabei ist d der Abstand zwischen Objekt und Sender/Sensor, t die benötigte Zeit und c die Lichtgeschwindigkeit.

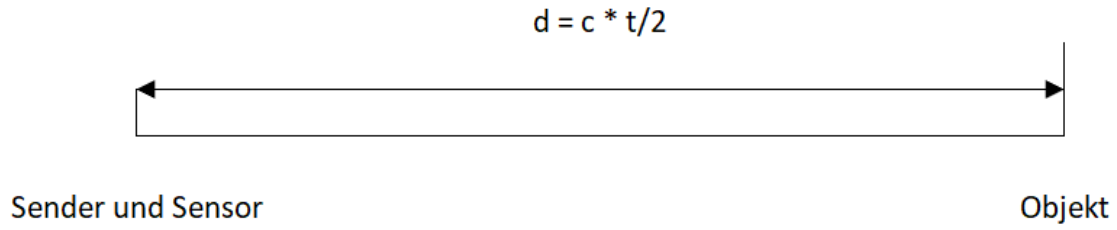

Abbildung 6: Berechnung der Entfernung eines Objektes mittels Laufzeitverfahrens

Eine darauf aufbauende, relativ neue Methode sind Time-of-Flight (ToF) Kameras. Diese tragen diesen Namen, da auch ihre Messwerte auf der Dauer der Lichtimpulse beruhen. Dabei wird mithilfe von pulsmodulierten Wellen ein Bereich abgetastet. Aufgrund der Phasenverschiebung kann aus den reflektierten Daten berechnet werden, wie weit sich ein Objekt von der Kamera entfernt befindet. Aufgrund der

[24] Alb+20, vgl. S. 2.

Lichtgeschwindigkeit ist es möglich, sehr schnell aktuelle Bilder zu erhalten.[25] Ebenso lässt sich mithilfe dieses Prinzips eine Fläche erfassen, so dass nicht jeder Punkt einzeln bearbeitet werden muss. Ein Beispiel für dieses Prinzip ist die Azure Kinect DK. Dabei handelt es sich um eine Weiterentwicklung der Azure Kinect, welche 2019 veröffentlicht wurde. Diese weißt eine deutlich höhere Genauigkeit auf, als viele andere kommerzielle Modelle[26].

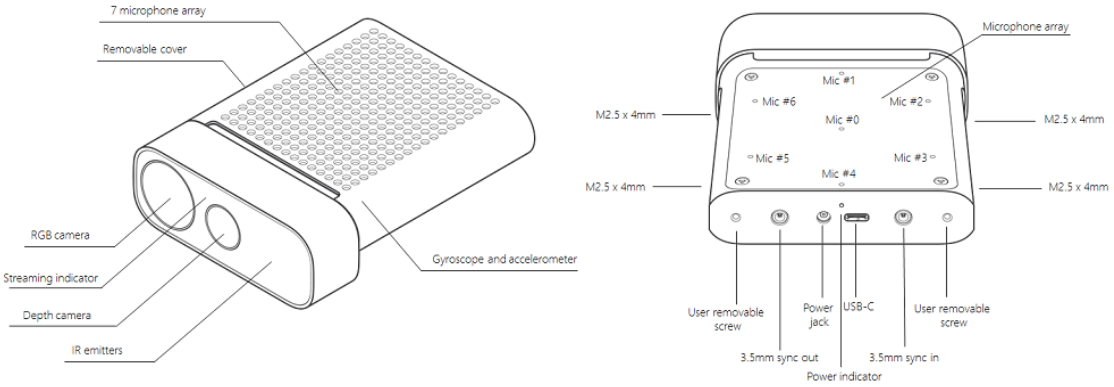

Abbildung 7: Aufbau der Kinect[27]

Die Kinect ist in Abbildung 7 dargestellt und verfügt neben der Tiefenkamera über eine normale Farbkamera (RGB camera) sowie ein Mikrophone Array, welches Geräusche wahrnimmt. Mittels Gyroskop (Gyroscope) und Beschleunigungssensor (accelerometer) kann jederzeit die aktuelle Position der Kamera abgefragt werden. Jede Art der zuvor beschriebenen Kameras gibt ihre Daten als Stream aus. Da hier zusätzlich zu den üblichen Rot-Grün-Blau (RGB) - Daten, welche den Farbwert der Pixel darstellen, auch Tiefendaten gesendet werden, spricht man von einem RGB - D Stream. Dabei symbolisiert D die Tiefe (Depth).

3.4. Aktueller Forschungsstand

Die Projektion einer virtuellen Abbildung eines Menschen auf AR-Geräten ist schon seit langer Zeit ein zentraler Forschungsschwerpunkt. Dabei ist der zugrundeliegende Sinn der, dass eine Person möglichst realitätsnah projiziert wird. Die anfängliche Entwicklung fand im Kino/Theaterumfeld statt, um Personen oder Wesen abbilden zu können, welche real nicht vorkommen, z.B. Aliens oder ausgedachte Figuren. Da anfangs hierfür Kostüme verwendet wurden, waren durch die Anatomie des Men-

[25] Alb+20, vgl. S.2.
[26] vgl. J B20.
[27] Doc.

schen hier Grenzen gesetzt. So ist zu begründen, dass beispielsweise die meisten Aliens in alter Filmen eine menschenähnliche Gestalt haben. Ab 1977 wurden in Kinoproduktionen CGI (Computer Generated Image) - Effekte verwendet. Die erste bekannte Verwendung fand dabei in Star Wars statt.[28] Anfangs handelte es sich dabei um reine animierte Aufnahmen, welche allerdings mit dem aufkommen von Motion Capture erweitert wurden. Dabei werden "nicht menschliche" Figuren so erzeugt, dass Personen in speziellen Anzügen aufgenommen wurden, welche mit reflektierenden Kugeln versehen wurden (siehe Abbildung 8 links).
Anhand der Bewegung der Punkte zueinander konnten die Bewegungen der Person nachgebildet werden. Hierfür ist ein bewegliches digitales Modell einer Person notwendig, welches sich anhand der Punktbewegungen steuern lässt. Ein solches Modell ist in Abbildung 8 rechts zu sehen.
Dabei wurde anfangs nur eine geringe Anzahl an Punkten verwendet, da die Animation der Figur mit steigender Anzahl an Gelenken komplizierter und rechenaufwendiger wird, da Positionen von Körperteilen in Abhängigkeit von diesen Punkten berechnet werden müssen.

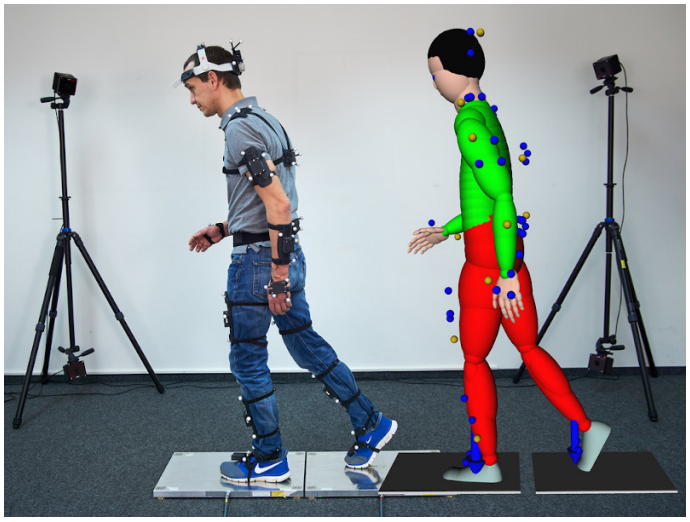

Abbildung 8: Darstellung von Motion Capture[29]

Um Menschen auch in AR als Hologramme darstellen zu können, wurden ebenfalls Avatare erstellt, welche der darzustellenden Person möglichst ähnlich sehen sollen. Dabei ist je nach Art der Anwendung entweder der ganze Mensch, Teile eines Menschen (z.B. dessen Oberkörper) oder eine völlig andere Figur dargestellt.
Um die Bewegungen einer Person abzubilden, kann diese entweder ferngesteuert wer-

[28] vgl. Edi.
[29] Tra.

den oder es müssen die Bewegungen einer Person erfasst werden. Ein Beispiel für eine solche Umsetzung ist das Tablet iPad Pro[30], welches das Gesicht einer Person auf eine auswählbare digitale Figur übertragen kann (siehe Abbildung 9).

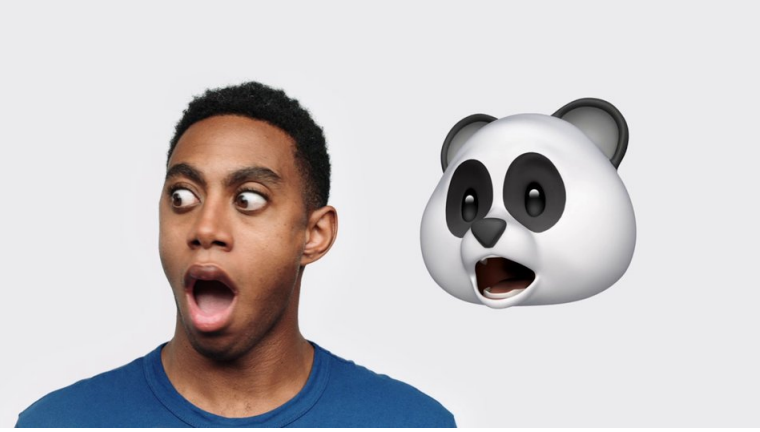

Abbildung 9: Darstellung des Gesichtsausdrucks einer Person auf einem Avatar[31]

[30]vgl. App.
[31]Kra.

Dabei wird das Gesicht der Person mit einer Tiefenkamera erfasst, die mittels Künstlicher Intelligenz nach Punkten sucht, welche auf den Avatar übertragen werden können. Diese Bewegungen werden anhand von Mustern dann auf dem Avatar dargestellt.
Das hauptsächliche Problem bei dieser Art der Darstellung ist allerdings weiterhin, dass um einen möglichst detailreichen Avatar erstellen zu können, dieser sehr aufwendig gegliedert werden muss. Nur mithilfe einer Großzahl an Gelenken und Punkten ist es möglich, sehr nah an das originale Bild zu gelangen. Zusätzlich ist anzumerken, dass für eine sehr detaillierte Darstellen einer Person ein eigener Avatar erstellt werden muss.[32]
In neuen Forschungsarbeiten wird daher an der Idee gearbeitet, statt eines Avatars eine Punktwolke zu verwenden, welche den Nutzer repräsentiert. Diese Punktwolke wird mithilfe einer Tiefenkamera erstellt, welche die Person dauerhaft abtastet. Hierfür wurden schon Applikationen entwickelt, welche das Bild einer Tiefenkamera bsw. auf einen Laptop übertragen.[33] Aufgrund der fehlenden Geschwindigkeit der Tiefenkameras sowie ihres hohen Kostenfaktors wurde bisher aber hauptsächlich mit Avataren gearbeitet.[34]
Mit dem Aufkommen der Azure Kinect, einer bezahlbaren ToF-Kamera ist es möglich geworden, die benötigte Menge an Bildern in der erforderlichen Zeit zu erhalten.
Dabei wurden unter anderem Systeme erstellt, welche es ermöglichen, einen gesamten Raum mit einer Azure Kinect aufzunehmen und diesen dann als 3D-Modell zu betrachten.
Auch wurden schon Umsetzungen für den Medizinbereich erstellt, in welchem beispielsweise das Laufverhalten von Personen betrachtet wird.
Für die HoloLens 1 wurden ebenfalls Anwendungen entwickelt, welche das Bild der Azure Kinect auf eine solche bringen können.[35] Das hauptsächliche Augenmerk lag dabei auf der Übertragung zu einem Monitor.
Im Rahmen dieser Arbeit wird untersucht, wie ein Videostream einer solchen Punktwolke in AR wahrgenommen wird, um so eine weitere zielgerichtete Entwicklung einer solchen Technologie zu ermöglichen.

[32] vgl. PLH18.
[33] Yos.
[34] vgl. S. 5 Brü11.
[35] vgl. Jun.

4. Anforderungen

Um die Aufgabe vollständig zu lösen, wird zuerst erfasst, welche Anforderungen an die Anwendungen gestellt werden. Dabei werden die einzelnen Anwendungen nun nacheinander beschrieben.

4.1. Darstellung der Azure Kinect Daten auf der HoloLens 2

Zuerst wird eine Anwendung zur allgemeinen Verwendung erstellt, an welche die folgenden Anforderungen gestellt werden:

- Die Anwendung ist zur Bearbeitung der Forschungsfrage dafür gedacht, dass Personen an verschiedenen Orten miteinander interagieren können. Somit ergibt sich:
  - Es darf keine Notwendigkeit einer physikalischen Verbindung von Kinect und HoloLens bestehen.
  - Die Daten sollen von der Kinect auf die HoloLens übertragen werden.
  - Für die Übertragung ist eine Netzwerkverbindung zwischen HoloLens und Kinect zu verwenden.
- Die Anwendung muss Bilddaten einer Kamera übertragen. Somit ergibt sich, die Kinect einen ausgewählten Bereich im Raum aufnimmt.
- Die Anwendung hat die Daten als Punktwolke darzustellen.
- Aufgrund des Einsatzgebiets im DLR und somit im Forschungsbetrieb, ist es notwendig, dass die Daten nicht über Fremdservices versendet werden. Darauf folgt, dass in dieser Übertagung keine fremder Server oder Cloud-Anbieter beteiligt sein dürfen.
- Um die Anwendung für den Nutzer ansprechend zu gestalten, sollte diese möglichst ohne Latenz und mit hoher Auflösung arbeiten. Somit ergibt sich:
  - Die Daten sollen möglichst echtzeitfähig auf die HoloLens 2 übertragen werden.
  - Die Daten sollen dabei für den Nutzer in einem ansprechenden Format dargestellt werden.

4.2. Darstellung der Daten der Azure Kinect für eine Nutzerstudie

Um die zuvor erstellte Anwendung zu testen, soll diese für eine Nutzerstudie ausgebaut werden. Dabei soll eine Applikation erstellt werden, welche es ermöglicht, verschiedene Verbindungen zu vergleichen. Dabei sind die folgenden Anforderungen gestellt:

- Um einen Vergleich zu ermöglichen, müssen verschiedenen Versionen gegeben sein:
  - Die Übertragung soll durch die Bildauflösung, FPS und Latenz bestimmbar sein.
  - Die Unterschiede müssen in einem für den Nutzer erkennbaren Maß liegen.
  - Dem Nutzer müssen die verschiedenen Szenarien nacheinander dargestellt werden.
- Um verschiedene Streams vergleichen zu können, muss sich in diesen ein ähnlicher Inhalt befinden, welcher allerdings nicht identisch ist:
  - Es muss ein Inhalt für die Streams gefunden werden, welcher leicht variiert werden kann.
  - Der Inhalt sollte allerdings nicht von dem eigentlichen Ziel ablenken.
  - Die verschiedenen Einflussfaktoren sollten dabei gut sichtbar sein.
- Es sollte möglich sein, die Nutzerstudie einfach durchführen zu können:
  - Die Anwendung muss so aufgebaut sein, dass der Nutzer sie ohne längere Einweisung verwenden kann.
- Um die Studie durchführen zu können werden Teilnehmer benötigt:
  - Das Stattfinden der Nutzerstudie muss bekannt gegeben werden.
  - Es müssen alle rechtlichen Vorgaben im Vorfeld organisiert werden.
- Die Ergebnisse der Nutzerstudie müssen erfasst und ausgewertet werden:
  - Ein Fragebogen muss erstellt werden.
  - Eine Auswertung des Fragebogens muss erfolgen.

5. Konzeption

Resultierend aus den in Kapitel 4 genannten Anforderungen wird zuerst für die eigentliche Applikation ein Konzept erarbeitet, welches im darauffolgenden Kapitel 6 umgesetzt wird.

5.1. Aufbau

Da die Applikation die Daten von der Azure Kinect zur Microsoft HoloLens 2 transportieren soll, ist es notwendig, dass diese abgefragt, verarbeitet, übertragen und dargestellt werden. Dieser Ablauf ist in Abbildung 10 zu sehen.

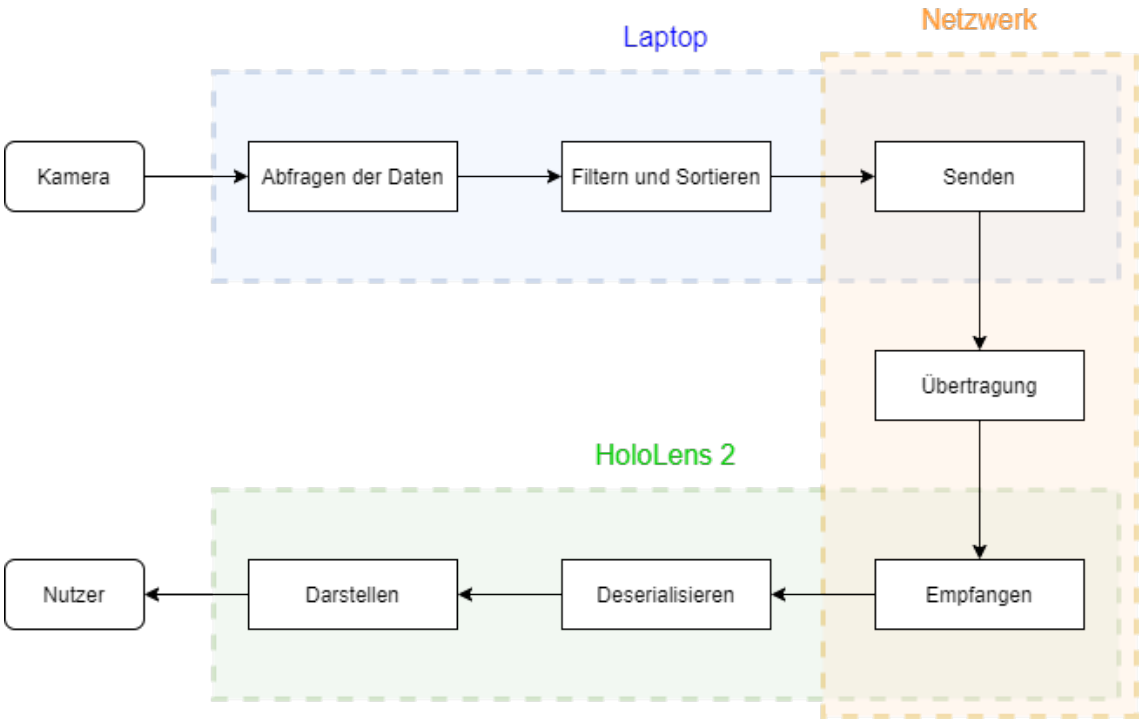

Abbildung 10: Aufbau der Anwendung

Da anhand der Anforderungen keine physikalische Verbindung zwischen Kamera und HoloLens bestehen darf, ist hierfür eine Netzwerkübertragung notwendig. Um diese zu realisieren, muss die Kamera mit einem netzwerkfähigen Gerät verbunden werden. Die Begründung ist dabei, dass die HoloLens selbst nicht über Netzwerk kommunizieren kann.
Hierfür wird ein Laptop verwendet, welcher über das Datenkabel der Kamera mit dieser verbunden ist. Dabei wird für den Laptop eine Applikation benötigt, welche die Daten der Kamera abfragt, verarbeitet und dann über ein Netzwerk weitervermittelt.

Auf der Seite der HoloLens 2 ist ebenfalls eine Applikation notwendig, welche die Daten empfangen kann und diese dann auf dem Display der HoloLens darstellt.
Aus diesem Aufbau ergibt sich, dass ein Netzwerksystem zwischen den beiden Applikationen erstellt werden muss. Ein Konzept für dieses Netzwerk sowie für die Applikation der Kamera als auch der HoloLens 2 im Folgenden vorgestellt.

5.2. Netzwerk

Um Daten zu transportieren ist es notwendig, eine Verbindung zwischen HoloLens und Laptop zu schaffen. Da die HoloLens keinen LAN-Anschluss besitzt, muss diese über WLAN realisiert werden. Dabei kann allerdings der Laptop mittels LAN mit einem Router verbunden werden, da so eine geringere Verzögerung zu erwarten ist. Dieser Aufbau ist in Abbildung 11 dargestellt:

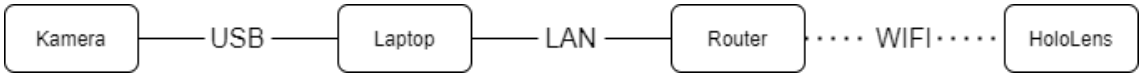

Abbildung 11: Aufbau des Netzwerks

Der Laptop übermittelt dabei die Daten der Kamera an die Brille.
Hierfür ist eigentlich nur eine unidirektionale Verbindung notwendig, da keine Daten zurückgesendet werden müssen. Sollten Verluste bei der Übertragung auftreten, ist es nicht nötig diese Information zu senden, da zu diesem Zeitpunkt die Informationen bereits veraltet wären.
Allerdings ist es für den Verbindungsaufbau notwendig, um beiden Geräten zu ermöglichen, sicherzustellen, dass eine Verbindung besteht. Somit sollte eine bidirektionale Verbindung geschaffen werden, welche das Empfangen und Senden von beiden Seiten aus ermöglicht. Dieser Datenverkehr ist in Abbildung 12 dargestellt, wobei grün der Datenverkehr und orange der Verbindungsaufbau eingezeichnet sind.

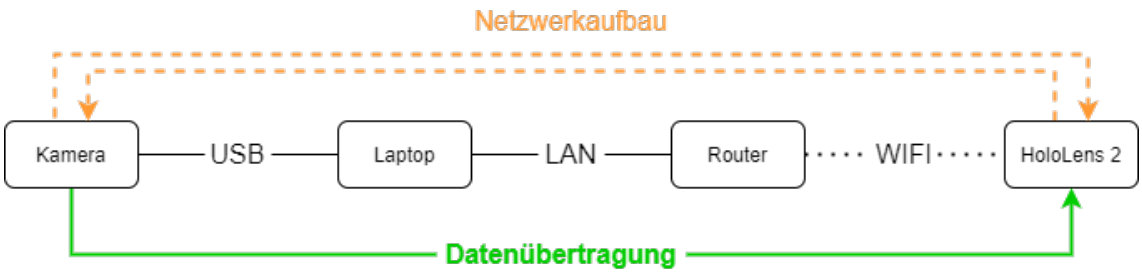

Abbildung 12: Konzept der Netzwerkkommunikation

Als passende Netzwerkstruktur wird das Server-Client Modell verwendet. Dies ist damit zu begründen, dass ein Broadcasting nicht gewünscht ist, da sich noch andere Geräte im Netzwerk befinden könnten.
Da im späteren Verlauf es aber beispielsweise möglich sein soll, mehrere Kameras oder AR-Brillen parallel zu verwenden, erscheint es sinnvoll, diese dann über eine zentrale Stelle ansprechen zu können. Dabei wurde die Entscheidung, ob Brille oder Laptop vorerst als Host agieren soll, anhand der folgenden Punkte getroffen.

- Die Leistungsfähigkeit der Brille ist geringer als die des Laptops
- Die unternehmensbezogenen Einstellungen des Laptops ermöglichen keinen Betrieb als Server
- Die Verwendung von mehreren Kameras, um eine größere Fläche zu erfassen, scheint wahrscheinlicher, als auf mehrere Brillen gleichzeitig mit einem Bild versorgen zu wollen.

Somit wurde, wie in Abbildung 13 zu sehen, entschieden, die Brille als Host und den Laptop/die Kamera als Client einzusetzen.

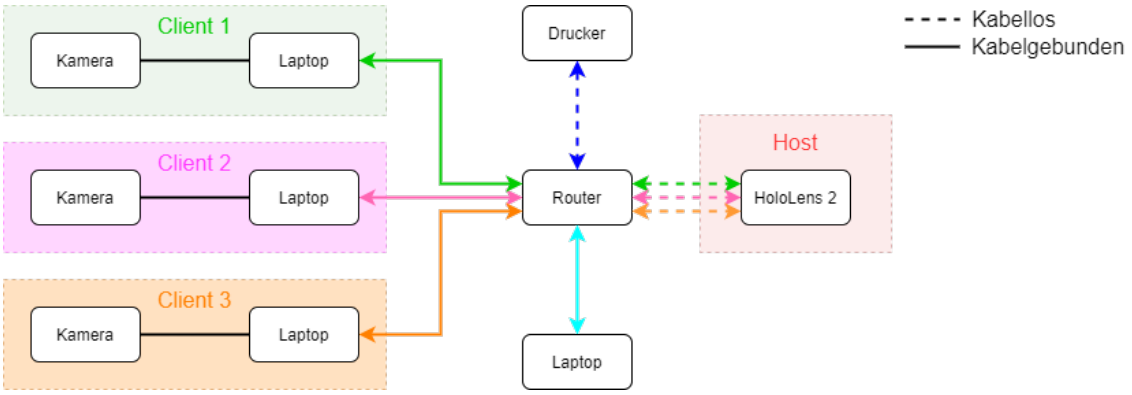

Abbildung 13: Abbildung des gesamten Netzwerks

Es ist möglich mehrere Clients mit einer HoloLens zu verbinden und somit mehrere Bilder parallel anzuzeigen. Zudem können sich weitere Geräte im System befinden, welche möglicherweise die gleiche Software verwenden. Außerdem bietet es sich so an, dass die Kabelverbindung zwischen Laptop und Router bei Bedarf durch eine VPN Verbindung ersetzt werken kann. Somit sind auch Übertragungen über sehr weite Entfernungen gesichert möglich.

5.3. Applikation für den Laptop

Die Applikation auf der Seite der Kamera ist ein Client im Netzwerksystem zwischen HoloLens und Kamera (siehe vorheriger Abschnitt). Dieser soll die Daten der Kamera erfassen und an die HoloLens weiterleiten. Es muss zuerst eine Verbindung zum Host hergestellt werden.
Dann gliedert sich die Arbeit des Hosts in drei Schritte, welche dauerhaft wiederholt werden. Zuerst müssen die Daten abgerufen werden. Dabei sind jeweils die gewünschten Eigenschaften der Kamera auszuwählen.
Danach müssen die Daten gefiltert und serialisiert werden. Dafür sind die Kriterien des Sortierens festzulegen. Dabei sind insbesondere die Anzahl an Punkten und deren Auflösung zu beachten. Hier ist insbesondere die Tiefeninformation von Bedeutung, da Punkte ab einer gewissen Entfernung zur Kamera nur noch wenige Informationen liefern bzw. nicht mehr benötigt werden.

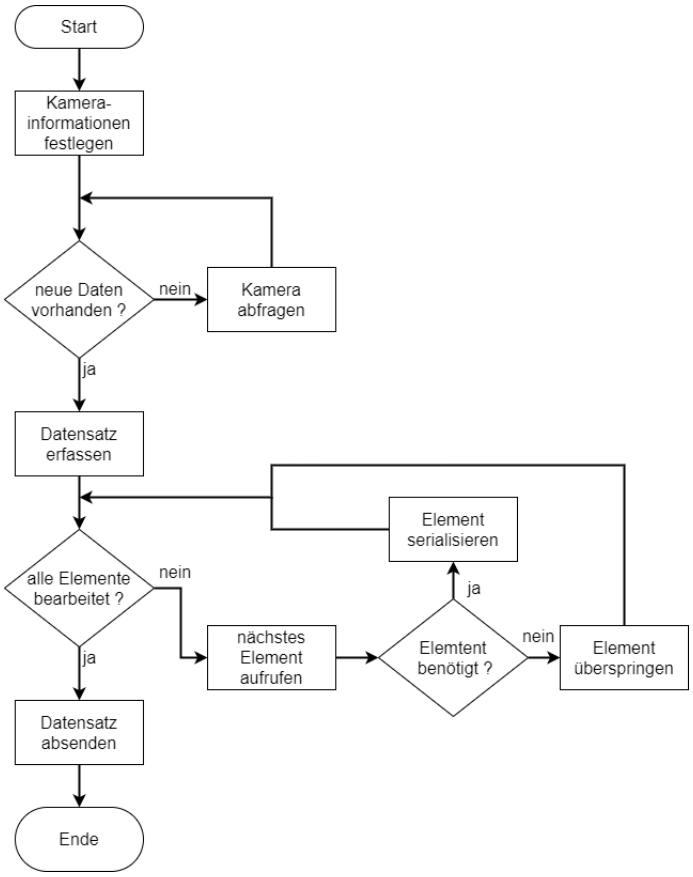

Abbildung 14: Ablaufplan der Client Datenerfassung

Alle gewonnenen Daten müssen für den Datentransport vorbereitet, also serialisiert, werden. Hierfür muss eine geeignete Möglichkeit gefunden werden. Dieser Ablauf ohne Netzwerkkomponenten ist in Abbildung 14 als Ablaufdiagramm dargestellt.

Dieses Serialisieren muss in Abstimmung mit der Netzwerkkomponente erfolgen, da sich hier eine Schnittstelle befindet. Die Daten müssen in eine Form gebracht werden, mit welche die Netzwerkkomponente arbeiten kann.

5.4. Applikation für die HoloLens 2

Da die HoloLens 2, wie in Unterabschnitt 5.2 beschrieben, als Host dienen soll, ist es wichtig, dass sie Datenverbindungen zu verschiedenen anderen Client aufbauen kann. Dabei müssen für diese Arbeit allerdings noch keine parallelen Funktionsweisen der Clients gegeben sein. Somit wird die Applikation so konzipiert, dass die HoloLens jeweils die Daten des aktuell verbundenen Clients darstellt. Diese müssen empfangen, deserialisiert und dargestellt werden. Erst dann kann der Nutzer sie betrachten. Dieser Ablauf ist ebenfalls als Ablaufdiagramm in Abbildung 15 dargestellt. Auch hier wurden die Netzwerkkomponenten entfernt.

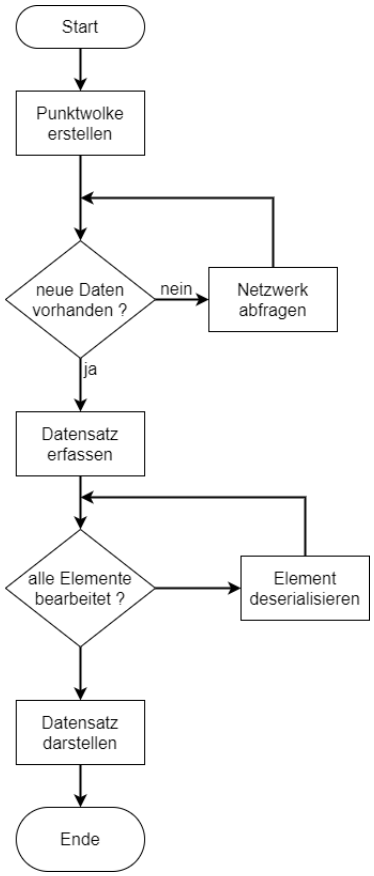

Abbildung 15: Ablaufplan des Hosts

Anhand von Abbildung 15 ist ebenfalls ersichtlich, was bei der Verbindung mit einem Host anfangs geschieht. Hier muss zuerst die Punktwolke als Struktur generiert

werden, an welche im späteren Verlauf die Daten übertragen werden. Dabei wird immer die gesamte Punktwolke aktualisiert und nicht etwa Teile von dieser. Die Begründung dieser Entscheidung besteht darin, dass eine Neuberechnung der gesamten Wolke für jeden Punkt für die HoloLens zu rechenaufwendig wäre. Zudem könnten die verschiedenen Pixel einer Darstellung verschiedene Zeitstempel haben. So wäre es möglich, dass bsw. Objekte doppelt erscheinen oder gänzlich verschwinden.

5.5. Nutzerstudie

Um die Anwendung mit Nutzern testen zu können, soll mit dieser eine Nutzerstudie stattfinden. Dabei soll allerdings nicht nur geklärt werden, wie die Nutzer mit dieser speziellen Version der Software arbeiten können, sondern insbesondere auf welche Punkte die Nutzer besonderen Wert legen.
Hierfür ist es notwendig, dass den Testpersonen verschiedene Möglichkeiten dargestellt werden, so dass sich diese einen Eindruck von den verschiedenen Ansätzen machen können. Nur so ist ein aussagekräftiger Vergleich möglich, da sonst die Auswahl eines Szenarios die Entscheidung grundlegend beeinflussen würde.
Dabei sollen die beiden Einstellungen Latenz und Auflösung betrachtet werden. Auflösung beschreibt hierbei, wie viele Bildpunkte auf dem Display der HoloLens dargestellt werden. Die Latenz definiert den Zeitraum, welcher zwischen Aufnahme einer Bildes und Darstellung von diesem liegt. Hierbei kommt zusätzlich der Faktor Bildwiederholrate, kurz FPS (frames-per-second), zum tragen, da dieser beschreibt, wie viele Bilder in einer Sekunde dargestellt werden können.
Um die beiden Szenarien zu vergleichen zu können, muss jeweils ein Videostream erstellt werden, welcher auf eine der beiden Eigenschaften fokussiert ist. Hierbei wird die hauptsächlich entstandene Version in beide Richtungen abgeändert. Der Videostream soll dabei sowohl Auflösung als auch Latenz in beiden Fällen deutlich zeigen. Zudem sollen die Unterschiede der Szenarien erkenntlich sein.
Es ist notwendig, dass in beiden Videos eine ähnliche Handlung gezeigt wird, welche allerdings einfach abgewandelt werden kann. Hierfür muss ein einfach zu erstellendes Beispiel gefunden werden, welche mit geringen, aber sichtbaren Unterschieden aufgeführt werden kann. Zudem muss sich das Beispiel beliebig oft wiederholen lassen.
Die Auswertung soll dann mithilfe eines Fragebogens erfolgen, welcher den Nutzern jeweils nach dem Betrachten eines Streams vorgelegt wird. Der Ablauf für die jeweiligen Nutzer ist in Abbildung 16 dargestellt.

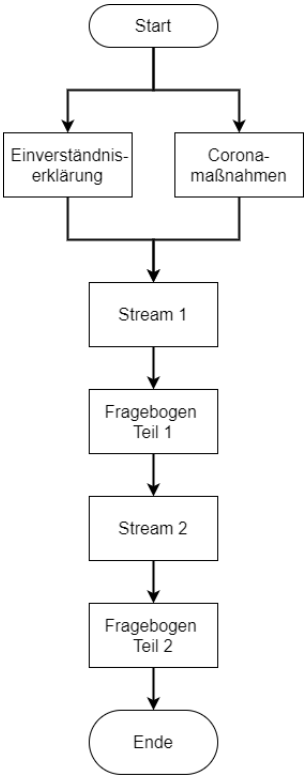

Abbildung 16: Konzeptioneller Ablauf der Nutzerstudie

Hierbei ist auch ersichtlich, dass vor der eigentlichen Studie eine Einwilligung der Nutzer eingeholt und zudem den aktuellen Coronamaßnahmen entsprochen werden muss. Diese sind dabei anhand der aktuellen Situation einzuschätzen.
Aus organisatorischer Sicht ist es zudem notwendig, die Räumlichkeiten zu bestimmen, die beteiligten Personen einzuladen und mit diesen Termine zu vereinbaren.

6. Umsetzung der Applikation

Im folgenden Kapitel wird beschrieben, wie die einzelnen Konzepte für die Applikation umgesetzt worden sind. Dabei wird zuerst die Entwicklung des Netzwerks als Grundlage des Projekts erläutert. Anschließend wird die verwendete Datenstruktur und folgend die Erstellung von Client und Host beschrieben. Dabei dient zum Teil das Github-Projekt von Takashi Yoshinaga[36] als Orientierung, wobei teilweise Dateien und Codestücke übernommen wurden. Diese sind entsprechend im Text gekennzeichnet.

6.1. Umsetzung des Netzwerk

Im Rahmen der Arbeit von SC am DLR wurde ein Unity-Projekt erstellt, welches die grundlegenden Bausteine für einen Netzwerkverkehr beinhaltet. Dabei sind sowohl Versionen mit Host-Client als auch Broadcast Topologie vorhanden. Hierfür wird auf Grundlage von ZeroMQ ein Netzwerk aufgebaut. ZeroMQ ist eine Bibliothek, welche zum verbinden von Netzwerk Sockets auf TCP Ebene verwendet werden kann[37]. Dabei ist durch das Projekt ein Netzwerkmanager gegeben, welcher zur Verwaltung des Netzwerkverkehrs verwendet wird. In diesem kann zudem angegeben werden, ob es sich bei einem Socket um einen Client oder Server handelt. Sofern ein Client gewünscht ist, kann hier die IP-Adresse des zu verbindenden Hosts ausgewählt werden. Dabei wird über die TCP-Ports 25001 und 25002 kommuniziert. Dabei wird 25001 beim Host und 25002 bei Client zum Senden genutzt.
Mittels der Dateien MessageSender.cs und MessageReceiver.cs des SC-Projekts ist es möglich, empfangene Daten zu verarbeiten und eigene Informationen zu senden. Dabei besitzen sowohl Client als auch Host beide Dateien. Der Client sendet seine Nachrichten immer an den Host und dieser seine Nachrichten an alle Clients.
Entsprechend werden mittels der Dateien MessageSender.cs und MessageReciever.cs Schnittstellen bereitgestellt, an welche der Sender bzw. Empfänger angesetzt werden kann. Dabei besitzen beide sowohl Sende- als auch Empfangsfunktionalität. Zum Senden der Daten verwendet man die Funktion:

```
CustomNetworkManager.Instance.SendBroadcastMessage(topic,payload)
```

Als *topic* kann dabei ein beliebiger String übersendet werden, so dass bereits anhand von dieser beim Empfänger ein möglicher Inhalt sichtbar ist. Der *payload* beinhaltete die zu übersendenden Daten, welche als String oder ein Byte Array vorliegen müssen. Entsprechend muss das Datenformat für die Applikation definiert werden.

[36] Yos.
[37] vgl. Zer.

6.2. Datenformat

Um das Bild von der Kamera zur HoloLens zu transportieren, müssen die Daten serialisiert werden. Das bedeutet, dass die Daten in eine Kette von Werten verändert werden müssen. Dies ist notwendig, da die Daten nur nacheinander gesendet werden können. Dafür müssen die einzelnen Punkte hintereinander aufgereiht werden.

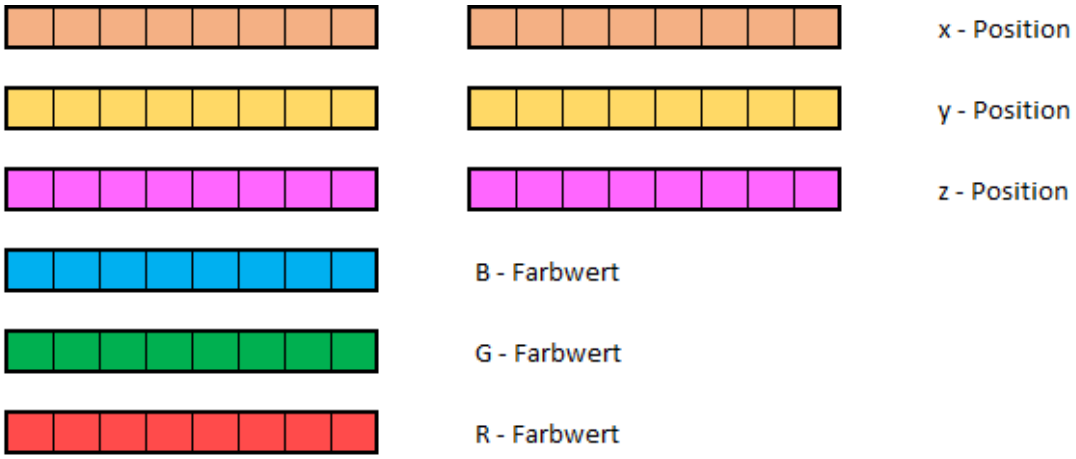

Abbildung 17: Datenformat eines einzelnen Punktes

Da die Koordinaten in x, y und z Wert angegeben werden und die Farben in RGB, sind pro Punkt 6 Werte notwendig.
Dabei werden die Farben in Werten zwischen 0 und 255 je Farbkanal ermittelt. Somit genügt für die Farben je ein Byte.
Die Entfernungen sind jeweils als Short Wert gegeben, welcher die Position in Millimetern darstellt. Da es sich bei Short Werten um jeweils 2 Byte handelt, werden für die Position 6 Byte benötigt. Die Darstellung dieses Aufbaus eines Punkts ist in Abbildung 17 zu sehen und besteht aus 9 Byte.
Um alle Punkte transportieren zu können, werden diese Abschnitte lückenlos hintereinander gelegt. Dabei ergibt sich die in Abbildung 18 dargestellte Struktur, welche die gleichen Farben verwendet wie Abbildung 17.

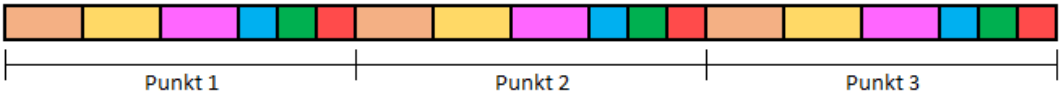

Abbildung 18: Aufbau eines Datenpakets

6.3. Sender

Für die Erstellung des Senders wird die in Unterabschnitt 6.1 erstellte Applikation des Clients verwendet und weiterentwickelt. Dabei wird die Azure-Kinect-SDK genutzt, welche kostenfrei auf dem Github von Microsoft[38] heruntergeladen und installiert werden kann.
Mithilfe von dieser ist es möglich, die Daten der Kinect abzugreifen. Dabei wird die Abfolge innerhalb der Datei MessageSender definiert, da die Verbindung zur Kamera erst hergestellt wird, sobald das Netzwerk gestartet wurde.

```
1  // Variable for handling Kinect
2  Device kinect;
3
4  // Number of point
5  int number;
6
7  // Array for bytes to send data
8  byte[] sendbyte;
9
10 // Class used to safe coordinate transformation(e.g.Color-to-depth
       , depth-to-xyz, etc.)
11 Transformation transformation;
```

Quelltext 1: Client - globale Variablen in MessageSender.cs

Zuerst wird die Kamera als Variable gespeichert (siehe Quelltext 1). Zudem wird die maximale Anzahl der Punkte und ein Array definiert. Diese wird später zum Speichern der Daten verwendet. Zudem wird die Übertragung der Position der Kamera in der Variable *transformation* gespeichert.
Danach wird eine Funktion Start() erstellt, welche beim ersten Aufruf des Skripts ausgeführt wird (siehe Quelltext 2). Diese ruft die Funktion InitDevice() auf. Zudem wird ein Task t erzeugt, welche ausschließlich die Funktion LoopDevice() enthält.

```
1  void Start() {
2   // init device
3   InitDevice();
4
5   // Loop to get data
6   Task t = LoopDevice();
7  }
```

Quelltext 2: Client - Start Funktion

[38] Mica.

Zeile 3 der Funktion InitDevice() (Quelltext 3) stellt hierbei die Verbindung mit der ersten gefundenen Azure Kinect her. Sollten mehrere Kameras angeschlossen sein, so kann durch den Übergabeparameter bestimmt werden, welche verwendet werden soll.

```
1  private void InitDevice() {
2   // Connect the Kinect (always first found one)
3   kinect = Device.Open(0);
4
5   // Set up Settings
6   kinect.StartCameras(new DeviceConfiguration {
7    ColorFormat = ImageFormat.ColorBGRA32,
8    ColorResolution = ColorResolution.R720p,
9    DepthMode = DepthMode.NFOV_Unbinned,
10   SynchronizedImagesOnly = true,
11   CameraFPS = FPS.FPS30
12  });
13
14  // Access to coordinate transformation information
15  transformation = kinect.GetCalibration().CreateTransformation();
16
17  // Allocation for coordinates and colors
18  int width = kinect.GetCalibration().DepthCameraCalibration.
        ResolutionWidth;
19  int height = kinect.GetCalibration().DepthCameraCalibration.
        ResolutionHeight;
20  number = width * height;
21
22  // Allocate for sendstream
23  sendbyte = new byte[number * 9];
```

Quelltext 3: Client - Init der Kamera

In Zeile 6 von Quelltext 3 werden die gewünschten Eigenschaften der Kamera definiert. Hierbei sind insbesondere das Farbformat (Zeile 7), die Auflösung (Zeile 8) und die FPS-Zahl (Zeile 11) für diese Arbeit von Bedeutung. Dabei wurde die Auflösung von 720p verwendet, dies entspricht 368640 Punkten. Theoretisch wären für die RGB-Kamera Auflösungen bis zu 4K möglich, allerdings sind Tiefenaufnahmen nur mit maximal 1K Auflösung herstellbar.
Da die RGB-Daten allerdings mit den Tiefendaten zusammengerechnet werden, ergibt sich nicht die klassische Rechteckige Form eines Bildes. Stattdessen sieht ein Bild mit den angegeben Werten wie in Abbildung 19 aus.
Dabei sind in dieser Aufnahme die verschiedenen Tiefenwerte in verschiedenen Far-

ben dargestellt. Je weiter entfernt ein Objekt ist, desto weiter rot gefärbt ist es. Die nächstliegenden Objekte sind blau. Die Form des Bildes (8-eckig statt rechteckig) ist nur auf die Tiefenkamera zurück zu führen.

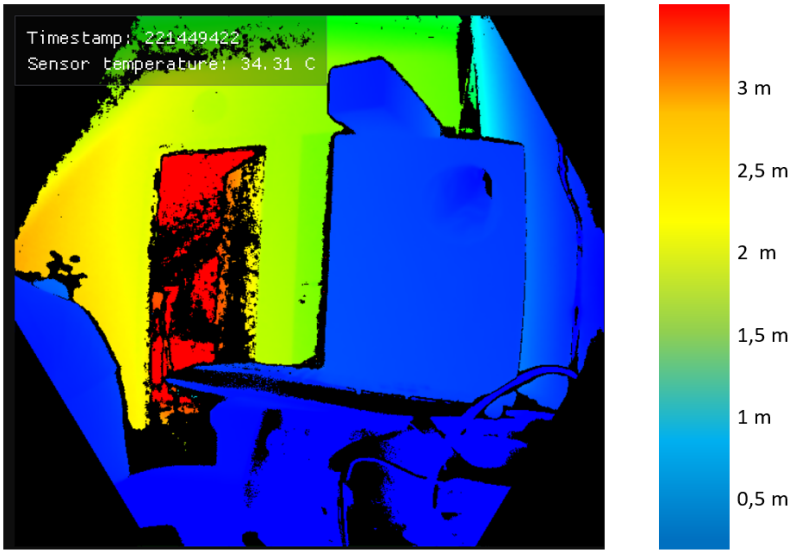

Abbildung 19: Aufnahme der Tiefenkamera

In Zeile 15 in Quelltext 3 wird die Transformation der Kamera gespeichert. Diese wird benötigt, um später die Daten der RGB-Kamera mit den Daten der Tiefenkamera zu verknüpfen. Nur so kann erreicht werden, dass dem richtigen Farbpixel die zugehörige Position zugeordnet wird.
Zeile 20 berechnet die maximale Anzahl an Punkten, welche benötigt werden können. Dabei wird in Zeile 18 und 19 jeweils die Pixelbreite des aktuellen Bildes erfasst. Dabei wird die Breite und Höhe der Bilder der Tiefenkamera verwendet, da diese durch ihre besondere Form der einschränkende Faktor sind.
Der letzte Arbeitsschritt der Funktion InitDevice() besteht in der Erstellung eines Byte-Arrays. Diese wird für das Versenden der Daten benötigt. Da wie in Unterabschnitt 6.2 beschrieben 9 Byte für jeden Punkt notwendig sind, ist eine Länge von 9 mal der Anzahl an maximal möglichen Punkten notwendig.
Nach der Erstellung aller Variablen beim ersten Ausführen wird in der Funktion LoopDevice() definiert, was in dem zuvor erstellten Task t stattfindet. Dabei gliedert sich dieser Task in vier Bestandteile: das Abfragen des Bildes, das Erfassen der Bilddaten, das Serialisieren der Daten sowie dem Versenden der Daten.
Dabei beinhaltet der Task eine anhaltende Schleife (Zeile 2 in Quelltext 4). Aufgrund der verschiedenen Dauer einer Verarbeitung auf der HoloLens und dem Laptop ergibt sich ein Problem, welches sich in einer steigenden Latenz äußert.

```
1 private async Task LoopDevice() {
2  while (true) {
3   using (Capture capture = await Task.Run(() => kinect.GetCapture()
        ).ConfigureAwait(true)) {
4    now = DateTime.Now;
5    if (((now - old).Milliseconds + (now - old).Seconds * 1000)
         >200) { //only send every 200ms
6     old = DateTime.Now;
```

Quelltext 4: Abfragen der Kamera

Da die Kamera deutlich schneller Daten erfasst und diese auch verarbeitet werden können, als die HoloLens darstellen kann, ergibt sich eine Zeitdifferenz. Bei andauernder Verwendung der Applikation kommt es zu einer steigenden Verzögerung zwischen Aufnahme und Darstellung. Dieser Zusammenhang ist in Abbildung 20 dargestellt.

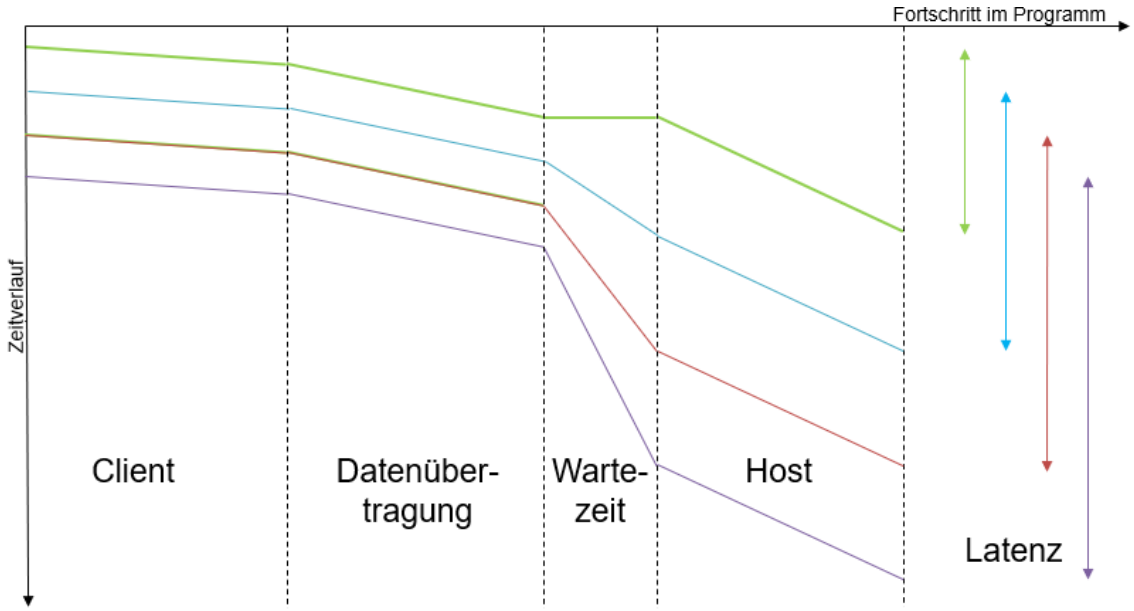

Abbildung 20: Zeitverzug durch verschieden lange Bearbeitungszeiten bei Host und Client

Dabei wird mithilfe der parallelen Linien im Bereich des Clients und Hosts angedeutet, dass diese Arbeitsschritte jeweils die gleiche Zeit benötigen. Da die Dauer einer Verarbeitungsphase beim Host allerdings länger benötigt, als beim Client, summiert sich eine Wartezeit auf. Dabei ist bereits anhand der 4 dargestellten Durchläufe sichtbar, dass so sehr schnell große Verzögerungen erreicht werden können. Die darauf resultierende Latenz ist anhand der Länge der Doppelpfeile sichtbar. Um diesem

Effekt entgegen zu wirken, wurde eine Abfrage in den Client eingebaut, welche neue Daten erst nach einer bestimmten Zeitspanne versenden lässt. Damit hierbei keine "alten" Daten versendet werden, werden zwischenzeitlich erfasste Daten verworfen und nur aktuelle Daten gesendet.
Diese Anfrage ist in Quelltext 4 in Zeile 5 zu sehen. Dabei wird für jeden Durchlauf die aktuelle Zeit erfasst und der Zeitpunkt des letzten Sendens zwischengespeichert. Die Ausgabe der vollständigen Daten nimmt etwa 200ms in Anspruch.
Hierbei ist ein zusätzlicher Effekt zu beachten. Sollte das neue Bild nach 199ms vorhanden sein, wird dieses verworfen. Somit werden Bilder mit mindesten 200ms Abstand versendet. Entsprechend der Einstellung der FPS der Kamera kann es sich in einem solchen Fall aber um eine deutlich höhere Wartezeit handeln.
Sofern ein Datensatz zur Verarbeitung freigegeben wurde, wird dieser als nächstes abgerufen und zwischengespeichert. Dieser Bestandteil ist in Quelltext 5 dargestellt und wurde von Takshi Yoshinaga[39] übernommen.

```
1 //Getting color information
2 Image colorImage = transformation.ColorImageToDepthCamera(capture);
3 BGRA[] colorArray = colorImage.GetPixels<BGRA>().ToArray();
4
5 //Getting vertices of point cloud
6 Image xyzImage = transformation.DepthImageToPointCloud(capture.
      Depth);
7 Short3[] xyzArray = xyzImage.GetPixels<Short3>().ToArray();
```

Quelltext 5: Erfassen des Bildes

Dabei wird jeweils das Tiefen- und Farbbild als Image gespeichert und die Bilder bereits auf eine gemeinsame Größe gebracht (Zeile 2 und 4). Das Tiefenbild wird in eine Punktwolke verwandelt, sprich alle Positionswerte werden aufgereiht.
Danach wird aus den Bildern jeweils ein Array an Werten erzeugt. Hierfür ist für die Farbwerte ein eigener Datentyp vorhanden (BGRA)[40]. Dieser beinhaltet neben RGB auch die Transparenz (Alpha-Wert), welche jeweils als 8 Bit gespeichert werden. Das Bild wird mithilfe einer ToArray() Funktion direkt in den Datentypen übertragen (Zeile 3).
Die Positionsdaten bestehen jeweils aus x,y und z Wert, welche als Short gegeben sind. Hierfür bietet sich eine Speicherung in einem Short3[41] an, wobei dieser Datentyp durch die Azure Kinect SDK importiert wird[42]. Dabei wird auch hier das Bild

[39] Yos.
[40] vgl. Unib.
[41] vgl. Micb.
[42] Mica.

mithilfe einer ToArray() Funktion transformiert (Zeile 7), sodass auf die einzelnen Werte zugegriffen werden kann.
Da diese Datenstrukturen nicht mithilfe der in Unterabschnitt 6.1 beschriebenen Funktion versendet werden können, ist es notwendig, diese in ein Byte-Array umzuwandeln. Da die Daten wie im Datenformat in Unterabschnitt 6.2 dargestellt, abgelegt werden sollen, ist es notwendig, diese systematisch zu verarbeiten. Dieser Prozess ist in Quelltext 6 sichtbar.

```
1  int j = 0;//only send unsed points
2  for (int i = 0; i < number; i++){
3   if (xyzArray[i].Z < 1500) { //cgeck if background
4    sendbyte[j * 9 + 4] = (byte)(xyzArray[i].Z & 255);
5    sendbyte[j * 9 + 5] = (byte)(xyzArray[i].Z >> 8);
6
7    if(xyzArray[i].X < 0) { //check if negativ
8     sendbyte[j * 9] = (byte)((xyzArray[i].X * -1) & 255);
9     sendbyte[j * 9 + 1] = (byte)(((xyzArray[i].X * -1)  >> 8) + 128)
          ;
10   }
11   else {
12    sendbyte[j * 9] = (byte)(xyzArray[i].X & 255);
13    sendbyte[j * 9 + 1] = (byte)(xyzArray[i].X >> 8);
14   }
15
16   if((-xyzArray[i].Y) < 0) { //check if negative
17    sendbyte[j * 9 + 2] = (byte)((-xyzArray[i].Y * -1) & 255);
18    sendbyte[j * 9 + 3] = (byte)(((-xyzArray[i].Y * -1) >> 8) + 128)
          ;
19   }
20   else {
21    sendbyte[j * 9 + 2] = (byte)(-xyzArray[i].Y & 255);
22    sendbyte[j * 9 + 3] = (byte)(-xyzArray[i].Y >> 8);
23   }
24
25   //serialize colors
26   sendbyte[j * 9 + 6] = colorArray[i].B;
27   sendbyte[j * 9 + 7] = colorArray[i].G;
28   sendbyte[j * 9 + 8] = colorArray[i].R;
29   j++;
30  }
31 }
```

Quelltext 6: Serialisieren des Bilds

Hierfür wird eine for-Schleife verwendet, welche ihren Endpunkt mittels der Varia-

blen *i* feststellt. Diese Variable beschreibt die aktuelle Position im originalen Datensatz. Da die Anzahl an Punkten der Variable *number* entspricht, kann ein Vergleich mit dieser als Abbruchbedingung verwendet werden.
Zudem wird die Variable *j* verwendet, welche die aktuelle Position im verarbeiteten Array beschreibt. Dabei handelt es sich um das zuvor definierte Byte-Array *sendbyte*, welches eine Länge von 9 mal Anzahl der Punkte besitzt. In dieses Array werden die Punkte nach dem selbst definierten Datenformat eingetragen.
Hierfür wird zuerst der z-Wert überprüft. Dies ist darauf zurück zu führen, dass nur Punkte mit einem maximalen Abstand von 1,5m versendet werden sollen. Sofern ein Punkt diese Bedingung nicht erfüllt, kann er aussortiert werden. In diesem Fall wird nur der Zähler, welcher die Position im originalen Bild beschreibt, inkrementiert, der Zähler im Array nicht. Somit sind nur die später benötigten Daten ohne Leerstellen in *sendbyte* vorhanden.
Sofern die Bedingung erfüllt wurde, wird zuerst der z -Wert behandelt. Dies ist damit zu begründen, dass bei diesem, anders als bei x und y keine negativen Werte auftreten können, welche gesondert behandelt werden müssen.
Dabei ergibt sich allerdings das Problem, dass Unity die Umwandlung von Short, in welchem der z - Wert vorliegt, zu Byte nur für Werte kleiner 255 automatisch erledigen kann. Sofern Werte größer 255 gewandelt werden, werden nur die niedrigstwertigen 8 Bit betrachtet.
Hierfür wurde eine eigene Umsetzung gefunden, welche mittels binärer Operatoren arbeitet. Dies ist in Zeile 4 und 5 zu sehen. Dabei werden zuerst die hinteren 8 Bit bearbeitet (Zeile 4). Um diese von der gesamten Zahl zu lösen, wird ein binäres *und* mit der Zahl 255 durchgeführt. Somit werden die niedrigstwertigen 8 Bit übernommen. Die daraus resultierende Zahl kann direkt in eine Byte umgewandelt werden, da sie maximal 255 (2 hoch 8 minus 1) betragen kann.
Die vorderen 8 Byte werden durch einen Bitshift um 8 Stellen gewonnen, sodass diese dann auch problemlos umgewandelt werden können. Hierbei besteht keine Problematik darin, dass diese Speicherbereiche noch alte Daten enthalten können, da etwaige Leerstellen mit 0 aufgefüllt werden.
Um den gewollten Aufbau der Daten (siehe Abbildung 18) zu erreichen, müssen die Daten hintereinander gespeichert werden. Dabei bietet sich die Schreibweise i * 9 + x an, wobei i der Anzahl der bisherigen Punkte und x der Position im aktuellen Abschnitt einspricht. Somit werden die Daten von z in *sendbyte* an die Position j * 9 + 4 bzw. j * 9 + 5 geschrieben (Zeile 4 und 5).
Anschließend werden x und y bearbeitet. Dabei ist zu beachten, dass hier sowohl positive als auch negative Werte möglich sind, da der Punkt 0,0 in der Mitte des Bildes liegt. Da allerdings eine Umsetzung von negativen Wert in Byte wieder nicht

mit der standardisierten Methode ToByte() möglich ist, muss hier zwischen positiven und negativen Werten unterschieden werden. Die hierfür entwickelte Art des Serialisierens wird im folgenden anhand der x-Werte erläutert. Mit dem y-Wert wird genauso verfahren.
Mittels einer Abfrage wird erkannt, ob es sich um einen positiven oder negativen Wert handelt (Zeile 7). Sollte es sich um einen positiven Wert handeln, wird der gleiche Mechanismus verwendet wie zuvor beim z - Wert. Dabei befinden sich die x - Werte an den Positionen 0 und 1 im späteren Datenpaket.
Sollte es sich um einen negativen Wert handeln, wird diese durch eine Multiplikation mit -1 umgewandelt und dann ebenfalls byteweise gespeichert. Um beim Deserialisieren eine Information darüber zu erhalten, ob es sich um einen negativen Wert gehandelt hat, wird mittels eines einzelnen Bytes diese Information übertragen. Da die Kinect nur einen begrenzten Aufnahmeradius hat, kann sichergestellt werden, dass keine Zahlenwerte größer 1000 auftreten können. Somit kann gefahrlos die erste Stelle des höherwertigen Bytes verwendet werden. Um dort eine 1 einzufügen, wird 128 zum höherwertigen Byte addiert, nachdem dieses mit 8 Bitshifts gewonnen wurde (Zeile 9).
Nachdem so die vorderen 6 Byte eines Punktes befüllt sind, müssen noch die Farbkanäle eingefügt werden. Dabei können die Werte allerdings direkt übertragen werden, da jeder Kanal genau ein Byte beinhaltet. Dieser Schritt wird in Quelltext 6 in Zeile 26 bis 28 durchgeführt. Dabei können die Farbkanäle dank das RGBA Datentyps direkt abgerufen werden.
Somit wurden alle Daten in das sendbyte - Array geschrieben. Dieses kann maximal 368.640 Datensätze beinhalten. Zumeist sind es allerdings weniger Datensätze, da Punkte im Hintergrund entfernt wurden. Somit wäre es falsch, dass gesamte Konstrukt zu versenden, da sonst unnötige Daten mit gesendet werden. Deshalb wird im nächsten Schritt eine Übertragung in ein exakt passendes Array vorgenommen (siehe Quelltext 7) und dieses versendet.

```
1     //create stream
2     byte[] send = new byte[(j + 1) * 9];
3     for (int i = 0; i < (j + 1) * 9; i++) {
4      send[i] = sendbyte[i];
5     }
6
7     //send Stream
8     CustomNetworkManager.Instance.SendBroadcastMessage("", send);
9    }
```

Quelltext 7: Versenden des Bilds

Um die benötigte Länge zu ermitteln, kann die Variable j verwendet werden. Da hierbei auch die 0 einen Datensatz widerspiegelt, muss ein Array mit (j + 1) * 9 Stellen erzeugt werden (Zeile 2).
Danach können die Daten übertragen werden, indem jeder Wert kopiert wird. Dabei bildet der Vergleich zwischen der Zählvariable i und j die Abbruchbedingung. Da j die Anzahl an Datensätzen und nicht Byte widerspiegelt, muss hier zuerst mittels (j + 1) * 9 die Byteanzahl berechnet werden. Da allerdings wieder die 0 als Position mit betrachtet werden muss, erfolgt der Vergleich mittels kleiner als.
Sobald alle Daten gepackt wurden, kann dieses Byte-Array mittels der vom Netzwerk bereitgestellten Funktion SendBroadcastMessage(" ", send) versendet werden (Zeile 9).
Durch die anhaltende Wiederholung dieses Ablaufs, werden fortlaufend die aktuellen Bilder gesendet.

6.4. Empfänger

Für den Empfänger wird die in Unterabschnitt 6.1 erstellte Applikation für den Host verwendet. Dieser erhält die in Quelltext 7 versendeten Nachrichten und muss diese verarbeiten. Es wird zuerst eine Punktwolke erzeugt, welche später die Daten darstellt. Dabei wurden die Elemente von dem Beispielprojekt[43] übernommen. Es handelt sich um einen Aufbau aus einem Mesh, welches die Punktwolke darstellt. Den einzelnen Punkten des Meshs werden Werte zugewiesen, welche durch die jeweils eintreffenden Werte ersetzt werden.
Das Mesh sowie der verwendete Shader wurden von[44] übernommen. Die Verbindung dieser Struktur mit den eintreffenden Daten wurde in MessageReciever.cs durchgeführt.

```
1 // Max. Number of points
2  int number;
3
4  //Mesh for pointcloud
5  Mesh mesh;
6
7  //Arrays of coordinates and colors for each point in PointCloud
8  Vector3[] vertices;
9  Color32[] colors;
```

Quelltext 8: Host globale Variablen

[43] Yos.
[44] Yos.

Dieser beinhaltet einige globale Variablen, welche im folgenden Verlauf häufig verwendet werden. Sie werden wie in Quelltext 8 zu sehen definiert. Dabei beschreibt *number* die Anzahl der darzustellenden Punkte. Das Mesh ist die Punktwolke, welche dargestellt wird (Zeile 5). Die beiden Arrays dienen zum Speichern der Werte der einzelnen Punkte (Zeile 8 und 9).
Wie bereits beim Unterabschnitt 6.3 ist ebenfalls eine Start() Funktion gegeben, welche einmalig ausgeführt wird. Diese ist in Quelltext 9 zu sehen.

```
1 void Start() {
2  number = 368640; //Resolution of 720p
3  InitMesh();
4  CustomNetworkManager.Instance.OnMessageReceivedEvent +=
       OnMessageReceived;
5 }
```

Quelltext 9: Host - Start Funktion

Dabei wird zuerst die Variable *number* auf 368.640 Punkte gesetzt (Zeile 1). Somit wird zum aktuellen Zeitpunkt festgelegt, dass immer die volle Anzahl an Punkten dargestellt wird. Hier wäre es zu einem späteren Zeitpunkt möglich, die Variable anhand der aktuellen Punktzahl zu setzen.
Danach wird eine InitMesh() Funktion aufgerufen, welche alle weiteren Bestandteile initialisiert. Diese ist in Quelltext 10 und Quelltext 11 dargestellt.
Danach wird das OnMessageRecievedEvent mit der Funktion OnMessageRecieved() verknüpft (Zeile 4).

```
1  private void InitMesh() {
2   //Instantiate mesh
3   mesh = new Mesh();
4   mesh.indexFormat = UnityEngine.Rendering.IndexFormat.UInt32;
5
6   //Allocation of vertex and color storage space for the total
        number of pixels in the depth image
7   vertices = new Vector3[number];
8   colors = new Color32[number];
9   int[] indices = new int[number];
10
11  //Initialization of index list
12  for (int i = 0; i < number; i++) {
13   indices[i] = i;
14  }
```

Quelltext 10: Host - InitMesh

Innerhalb der InitMesh() Funktion (siehe Quelltext 10) wird zuerst eine Variable Mesh erstellt (Zeile 3), für welche anschließend das Format festgelegt wird (UInt32 , Zeile 4). Damit wird sich für UInt32 entschieden, da ein UInt16 nur 65535 Punkte erlauben würde[45].
Danach werden die globalen Variablen initialisiert. Da diese je Punkt nur ein Feld benötigen, kann hier number als Größenangabe verwendet werden (Zeile 7 und 8). Ebenfalls wird das int Array indices definiert, welches eine Liste aller Zahlen von 0 bis number - 1 enthält. Diese wird in Zeile 9 bis 14 erzeugt. Der in Quelltext 10 und Quelltext 11 gezeigte Ablauf ist in Abwandlung vom Beispielprojekt von Takashi Yoshinaga[46] entstanden.

```
1  //Allocate a list of point coordinates and colors to be drawn to
       mesh
2  mesh.vertices = vertices;
3  mesh.colors32 = colors;
4  mesh.SetIndices(indices, MeshTopology.Points, 0);
5
6  //get mesh
7  gameObject.GetComponent<MeshFilter>().mesh = mesh;
8 }
```

Quelltext 11: Verbinden des Mesh

Nach der Erstellung der Arrays werden diese mit den entsprechenden Gegenstücken des Mesh verknüpft. Zudem wird dem Mesh eine Topology, nämlich Points, um eine Punktwolke zu erzeugen, zugewiesen. Diese werden nach den *indices* benannt (Zeile 2 bis 4). Anschließend wird das Mesh-Objekt mit der Mesh-Componente, welche aus dem Projekt[47] übernommen wurde, verknüpft (Zeile 7).
Danach wird die Funktion OnMessageRecieved() definiert. Dabei wird theoretisch die UserID des Senders und den Inhalt als Byte-Array übertragen. Dieses Array muss nun deserialisiert und entsprechend den Punkten zugeordnet werden. Dieser Abschnitt ist in Quelltext 12 beschrieben.
Dabei werden zuerst alle Punkte in payload betrachtet. Dafür werden diese mithilfe einer for-Schleife durchlaufen. Zuerst wird überprüft, ob das erste Bytepaar ein negativer oder positiver Wert ist. Hierfür wird das zweite Byte um 7 Stellen nach rechts gebitshiftet, so dass ein negativer Wert durch eine eins und ein positiver Wert durch eine 0 dargestellt werden würde (Zeile 6). Anhand dieser Entscheidung wird der Wert rücktransformiert. Ist der Wert negativ, werden hierfür beide Werte einzeln

[45] vgl. Unia.
[46] Yos.
[47] Yos.

behandelt. Das vordere, niedrig wertige Byte wird mit -1 multipliziert. Vom höher wertige Byte werden zuerst 128 abgezogen, so dass die anmerkung als negative Zahl verschwindet. Dann wird der Wert um 8 gebitshiftet, so dass er seine ursprüngliche Wertigkeit zurück erhält. Anschließend wird der Wert mit -1 und 0.001 multipliziert. Dies ist damit zu begründen, dass die Kamerawerte in mm sind, Unity zur Anzeige allerdings in Meter rechnet. Anhand dieses Systems werden die ersten beiden Werte (x und y) deserialisiert (Zeile 6 bis 12 und 14 bis 21).
Der z Wert kann keine negative Zahl enthalten, somit muss hier keine Unterscheidung vorgenommen werden (Zeile 24).
Die Deserialisierung verläuft ansonsten exakt gleich. Nach diesem Schritt sind alle Werte aus der Datenübertragung in den Arrays abgespeichert.

```
1  private void OnMessageReceived(long userId, byte[] payload) {
2   //set every points values
3   for (int i = 0; i < payload.Length/9; i++) {
4    //set position
5
6    if ((payload[i * 9 + 1] >> 7) == 1) {
7     //Restore negativ number and scale it
8     vertices[i].x = (float)(((payload[i * 9] * -1) + (((payload[i *
          9 + 1] - 128) << 8) * -1)) * 0.001f);
9    }
10   else {
11    vertices[i].x = (float)((payload[i * 9] + (payload[i * 9 + 1] <<
           8)) * 0.001f); //Restore Number and scale it
12   }
13
14   if ((payload[i * 9 + 3] >> 7) == 1) {
15    //Restore negativ Number and scale it
16    vertices[i].y = (float)(((payload[i * 9 + 2] * -1) + (((payload[
          i * 9 + 3] - 128) << 8) * -1)) * 0.001f);
17   }
18   else {
19    //Restore Number and scale it
20    vertices[i].y = (float)((payload[i * 9 + 2] + (payload[i * 9 +
          3] << 8)) * 0.001f);
21   }
22
23   //Restore Number and scale it
24   vertices[i].z = (float)((payload[i * 9 + 4] + (payload[i * 9 + 5]
          << 8)) * 0.001f);
```

Quelltext 12: Deserialisieren der Daten

Neben den Positionsdaten müssen ebenfalls die Farbwerte erfasst werden. Dieser Schritt ist in Quelltext 13 dargestellt.
Dabei wird für jeden Wert der jeweilig entsprechende Wert aus dem übertragenen Daten erfasst. Zusätzlich wird der Alpha-Wert überall auf 255 gesetzt, was bedeutet, dass keinerlei Transparenz vorliegt.

```
 //set values
 colors[i].b = payload[i * 9 + 6];
 colors[i].g = payload[i * 9 + 7];
 colors[i].r = payload[i * 9 + 8];
 colors[i].a = 255;
}
```

Quelltext 13: Deserialisieren der Farben

Da die Datenarray theoretisch 368640 Werte beinhalten können, ist es notwendig, die leeren Werte zu überschreiben, so dass keine Artefakte vom vorherigen Bild bestehen bleiben. Dies wird wie in Quelltext 14 dargestellt erledigt. Dabei werden die Werte alle auf 0 gesetzt. Die Punkte befinden sich somit direkt im Nutzer und werden von der HoloLens nicht dargestellt/berechnet. Durch die Transparenz von 0 wäre ein solcher Punkt auch beim Bewegen des Blickfelds nicht sichtbar.

```
//override all old data
for (int i = payload.Length/9; i < number; i++) {
 vertices[i].x = 0;
 vertices[i].y = 0;
 vertices[i].z = 0
 colors[i].b = 0;
 colors[i].r = 0;
 colors[i].g = 0;
 colors[i].a = 0;
}
```

Quelltext 14: Löschen von alten Daten

Sobald alle Werte eingetragen wurden, werden diese wieder in die entsprechenden Stellen des Mesh übertragen. Danach kann eine Neuberechnung und Darstellung des Mesh erfolgen. Dieser Schritt ist in Quelltext 15 dargestellt. Mit diesem Abschnitt endet die Darstellung der Daten.
Dabei wird immer das gesamte Bild aktualisiert und nicht etwa einzelne Elemente, um zu verhindern, dass alte Bilddaten und neue gemischt werden. So wäre es sonst

möglich, dass Objekte doppelt erscheinen oder gänzlich verschwunden wären.

```
1  //set mesh values
2  mesh.vertices = vertices;
3  mesh.colors32 = colors;
4  mesh.RecalculateBounds();
5 }
```

Quelltext 15: Darstellen des Bildes auf der HoloLens

7. Umsetzung der Nutzerstudie

Ausgehend von der Applikation, wurde eine Nutzerstudie durchgeführt. Entsprechend der Vorüberlegungen aus Unterabschnitt 5.5 wurde hierfür ein Beispiel gewählt, welches einfach abgewandelt werden kann. Zudem ist ein Aufbau notwendig, welcher es ermöglicht, möglichst deutlich den Unterschied verschiedener Auflösungen und Latenzen zu veranschaulichen.
Wie in den Anforderungen beschrieben, ist es hierfür jeweils erforderlich, verschiedene Werte zu demonstrieren. Deshalb werden zwei Szenarien erstellt, welche für jede Eigenschaft das Gegenteil des Anderen beinhalten. So soll ein Stream mit hoher Bildauflösung und hoher Latenz (Szenario 1) und ein Stream mit geringerer Auflösung aber auch geringerer Latenz (Szenario 2) entstehen.

Latenz
Die Latenz beschreibt die Zeit, welche zwischen der Entstehung des Bildes und seiner Darstellung auf der HoloLens 2 vergeht. Um die Latenz darstellen zu können, wird eine Referenzgröße benötigt, durch welche die Testperson einen Vergleichswert ohne Latenz geliefert bekommt. Da es sich bei der Anwendung um einen Videostream handelt, bietet es sich an, hierbei den Ton zu verwenden, welcher nicht über die Netzwerkverbindung übertragen werden muss. Dabei ist es notwendig, dass sich die Testperson und die Tonquelle in der Nähe voneinander aufhalten, so dass der Abstand zwischen Ton und Bild genau der Latenz des Bildes entspricht.

Bildauflösung
Die Auflösung beschreibt in diesem Fall, wie viele Punkte für die Darstellung eines Objekts verwendet werden. Somit bedeutet eine niedrigere Auflösung, dass nicht das Originalbild mit einer niedrigeren Auflösung im herkömmlichen Sinne aufgenommen wurde, sondern dass nicht alle Punkte auf die HoloLens übertragen werden. Als Auflösung der Kamera werden weiterhin 720p verwendet. Dies entspricht der HD-Auflösung und wird wie schon in der Ursprungsversion der Applikation verwendet.
Der Unterschied auf der HoloLens soll durch kleine Bestandteile des Streams verdeutlicht werden, so dass diese in den verschiedenen Szenarien unterschiedlich wahrgenommen werden können. Entsprechend ist es notwendig Elemente in den Stream einzubauen, welche sehr fein gegliedert sind bzw. mit schmalen Markierungen ausgestattet sind.

7.1. Inhalt des Streams

Um die Elemente der Auflösung und Latenz abzudecken, wurde der Aufbau eines Objekts auf verschiedenen kleineren Objekten als Inhalt gewählt. Hierfür wurden sechs Objekte aus Pappe/Papier entworfen, welche in Abbildung 21 abgebildet sind.

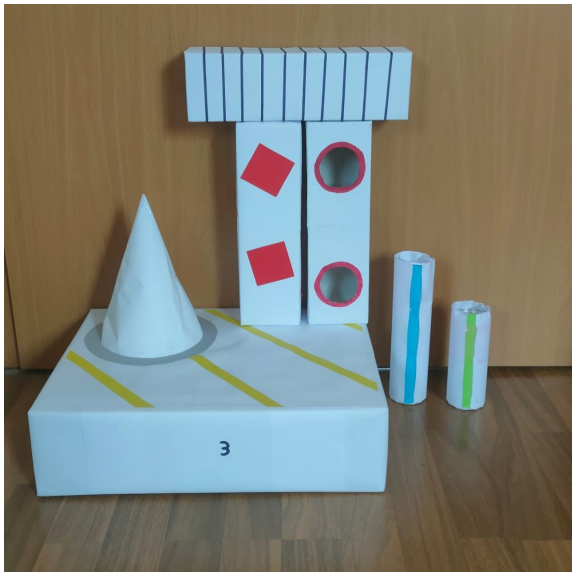

Abbildung 21: Konstruierte Objekte für die Nutzerstudie

Dabei sind die verschiedenen und unterschiedlichen Objekte zwischen 40cm und 5cm groß und lassen sich beliebig aufeinander platzieren. Sie sind alle hohl und bestehen zumeist aus einem Pappkarton, welcher mit weißem Papier umhüllt wurde. Danach wurden einzelne Details auf den Boxen angebracht, welche für die Auffassung der Auflösung entscheidend sind.

Im Folgenden werden die Boxen kurz beschrieben.

- **große Box** (40cm x 30cm x 10cm)
  - gelbe, 1,5cm breite Streifen auf der Oberseite
  - Kegel mit grauen Kreis umrandet
  - die Zahlen von eins bis vier auf den Seiten abgebildet, wobei die eins etwa vier mal so groß ist wie die vier
- **kleine Box mit schwarzen Linien** (28cm x 8cm x 7cm)
  - umlaufenden schwarze, 0,3cm breite Linien
  - keine Linien auf den Stirnseiten

- **kleine Box mit roten Vierecken** (28cm x 8cm x 7cm)
  - rote, 5,5cm große Quadrate auf den 4 Seitenflächen
  - keine Quadrate auf den Stirnseiten
- **kleine Box mit Löchern** (28cm x 8cm x 7cm)
  - 2 Löcher mit 5,5cm Durchmesser
  - beide Löcher sind gleich weit vom Rand entfernt (3cm)
  - die Löcher sind mit einem 0,5cm breitem roten Strich umrandet
- **kleiner Zylinder** (Durchmesser 5cm x 15cm)
  - Zylinder oben und unten geschlossen
  - Zylinder passt in die Löcher der kleinen Box mit Löchern
  - außerhalb der kleinen Box sind noch 8cm sichtbar
  - grüner, 1cm breiter Streifen an der Seite
- **großer Zylinder** (Durchmesser 5cm x 21cm)
  - Zylinder oben und unten geschlossen
  - Zylinder passt in die Löcher der kleinen Box mit Löchern
  - außerhalb der kleinen Box sind noch 16cm sichtbar
  - blauer, 1cm breiter Streifen an der Seite

Aus diesen Bauteilen werden Objekte gebaut, wobei die bauende Person beschreibt, welche Handgriffe sie gerade durchführt. So kann erkannt werden, wie die Bewegungen einer Person dargestellt werden können. Dabei werden durch Schlagworte wie "Ich hebe **jetzt** die kleine Box an." oder Lageangaben "Die Boxen liegen **exakt an der Tischkante**." dem Nutzer zusätzliche Möglichkeiten gegeben, genau auf bestimmte Faktoren zu achten.
Die Darstellung der Latenz zeigt sich dabei in der Verzögerung zwischen dem gesprochenen Wort und der Ansicht. Beispielsweise wird gesagt, dass nun eine Box angehoben wird, die Hololens übermittelt das Bild aber erst einige Millisekunden später.

7.2. Aufbau für die Nutzerstudie

Um zu verhindern, dass die Testperson den Demonstrierenden sieht, muss ein Aufbau gegeben sein, welcher dies verhindert.

Hierfür wurde eine Trennwand verwendet, welche sich zwischen Testperson und darstellender Person befindet. Zudem dient diese Trennwand als Hintergrund der Hologramme, so dass der Nutzer diese gut erkennen kann. Der Nutzer nimmt hierbei an einem Tisch mit etwa 3 Meter Abstand zur Trennwand Platz, sodass seine Blickrichtung zur Trennwand gerichtete ist (siehe Abbildung 22).

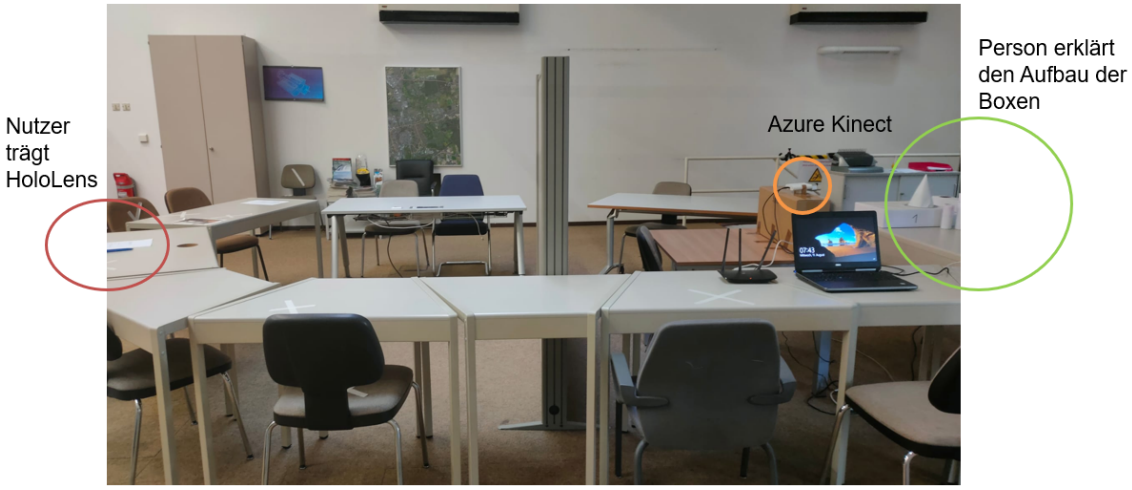

Abbildung 22: Aufbau für die Nutzerstudie

Auf der rechten Seite der Trennwand steht ein Tisch, auf welchem die Azure Kinect Tiefenkamera abgestellt ist. Unter dieser befindet sich eine Erhöhung, sodass sie die Person mit den Boxen von vorn und nicht von unten filmt. Dabei entspricht die Höhe der Kamera etwa der Höhe der Augen einer sitzenden Position. Dies führt dazu, dass die Person mit der HoloLens das Bild direkt in der richtigen Perspektive dargestellt bekommt. kann.
In etwa einem halben Meter Abstand zu diesem Tisch mit der Azure Kinect ist ein weiterer Tisch, auf welchem die Boxen aufgebaut und abgelegt werden.
Der Nutzer trägt während der Studie die HoloLens. Der Laptop und der verwendete Router stehen rechts von der Kamera auf einem Tisch, so dass die Kamera mit dem Laptop und der Laptop mit dem Router verbunden ist.

7.3. Vorbereitung der Studie

Um die Studie ordnungsgemäß durchführen zu können, mussten einige Vorbereitungen getroffen werden. Hierbei waren insbesondere die andauernde Coronapandemie sowie die hauptsächliche Arbeit im Homeoffice zu beachten, da die Teilnehmer nur vor Ort an der Studie teilnehmen konnten.

Sicherheitsbestimmungen

Um die Gefahr einer Ansteckung zu minimieren, war die Einhaltung von strengen Hygienemaßnahmen, wie das Lüften, Desinfizieren von Kontaktflächen sowie das Testen jedes Probanden Voraussetzung zur Durchführung der Nutzerstudie.

Teilnehmereinladung

Um möglichst viele Teilnehmer zu erreichen, wurden diese eine Woche vor Start der Nutzerstudie eingeladen. Dabei wurde mittels der Software "Terminplaner 4.1"[48] eine Terminauswahl ermöglicht.

Einwilligung zur Nutzerstudie

Um die Nutzerstudie rechtmäßig durchführen zu können, muss die Einwilligung jeder Person eingeholt werden. Zudem wird darauf hingewiesen, welche Daten erfasst werden und es wird ermöglicht, dieser Erfassung zu widersprechen.

Ablauf

Jeder neu eintreffende Nutzer wird zuerst zur Studie begrüßt und ihm anschließend erläutert, welche Aufgaben ihn erwarten werden. Danach wird ihm das Formular zum Einwilligung zur Nutzerstudie/Datenschutz vorgelegt. Anschließend wird der Selbsttest durch den Nutzer durchgeführt und nach der entsprechenden Wartezeit das Ergebnis auf dem dafür vorgesehen Formular notiert. Alle diese Formulare sind am Anhang A dieser Arbeit zu sehen.

Zeit	Inhalt	Dauer
5 min	Begrüßung und Test	20
10 min		
15 min		
20 min		
25 min	Szenario 1	5
30 min	Fragebogen Szenario 1	5
35 min	Szenario 2	5
40 min	Fragebogen Szenario 2 und Vergleich	10
45 min		

Abbildung 23: Ablaufplan für eine einzelne Testperson

[48] https://terminplaner.dfn.de/

Sofern ein negatives Testergebnis vorliegt, darf der Nutzer zu diesem Zeitpunkt die Maske absetzen und die HoloLens aufsetzen. Auf dieser startet er den Host des Programmes durch Auswählen des Programms. Vom Laptop der durchführenden Person kann nun das entsprechende Szenario gestartet werden. Dabei werden den Nutzern abwechselnd die verschiedenen Szenarien vorgespielt, so dass einige Testpersonen zuerst Szenario 1 und einige Szenario 2 sehen. So soll verhindert werden, dass die Reihenfolge eine Auswirkung auf die Ergebnisse beinhaltet.
Anschließend füllt die Testperson den ersten Teil eines Fragebogens zu diesem Stream aus. Die Zeitplanung für diese Schritte ist jeweils in Abbildung 23 dargestellt.
Nach dem Ausfüllen des Zeitplans wird der Testperson ein zweiter Stream gezeigt. Danach wird die zweite Hälfte des Fragebogens sowie ein Vergleich abgefragt. Anschließend ist die Studie für diese Testperson beendet.

7.4. Fragebogen

Um die Daten der Personen zu erfassen, wird ein Fragebogen erstellt. Da dieser einen Vergleich zwischen zwei Streams bieten soll, werden zu beiden Streams die gleichen Fragen gestellt. Zusätzlich wird eine Auswertung erstellt, in welcher die Testperson die beiden Streams vergleicht. Zudem wird sie hier gefragt, welche Eigenschaften des Streams sie als bedeuten erachtet.

Ursprung des Fragebogens
Da die Studie eine Aussage über die Verwendbarkeit des Systems geben soll, dient der System-Usability-Scale (SUS) als Grundlage. Diese Art, ein System zu bewerten gelingt so, dass den Testpersonen zuvor 10 festgelegte Aussagen vorgelegt werden, welche diese mittels einer Skala von 1 bis 5 bewerten können. Dabei beschreibt 5, dass die Testperson der Aussage völlig zustimmt, während 1 ein völliges widersprechen bedeutet. Diese wird als "Likert Skala" bezeichnet.[49] Da sich dieser Fragebogen allerdings für Websites oder interaktive Systeme eignet, ist er nur bedingt für diese Nutzerstudie verwendbar. Entsprechend werden die unpassenden Fragen durch solche Fragen verändert/ersetzt, welche den Inhalt der Arbeit betrachten. Dadurch sind die Ergebnisse natürlich nicht mehr eindeutig mit dem eigentlichen SUS-Score vergleichbar, allerdings kann dessen Auswertung weiterhin verwendet werden. Zudem ist das Anpassen des Fragebogens an das eigene System erlaubt.[50] Der Standardfragebogen ist dabei im Anhang B zu finden.

[49] vgl. Tho.
[50] vgl. Tho.

Eigener Fragebogen

Im Folgenden werden die Fragen für diese Studie vorgestellt. Dabei wird für jede Frage erläutert, welche Erkenntnisse aus dieser gewonnen werden sollen. Es ist zu beachten, dass mehr Fragen gestellt wurden, als im originalen SUS vorhanden sind. Es werden für die SUS-Auswertung nur ein Teil der Fragen verwendet. Zusätzlich zu den aufgelisteten Fragen werden für statistische Zwecke Demografische Daten, wie z.B. Alter und Geschlecht, erhoben.

Die ersten drei Fragen beschreiben die Ausgangssituation des Nutzers und geben Informationen darüber, wie vertraut er mit den verwendeten Technologien ist.

1. Do you have experience with AR ? (much experience / no experience)
Mithilfe dieser Frage soll erfasst werden, ob die Testperson bereits AR-Technologien verwendet hat. Dies beeinflusst das Ergebnis der Umfrage insofern, dass ein häufiger Nutzer zusätzliche Vergleichswerte besitzt. Somit sind seine Antworten nicht nur durch die Studie selbst beeinflusst.

2. Do you often work with AR ? (daily / never)
Diese Frage dient dazu abzugrenzen, ob eine Person produktiv mit AR einer Arbeit nachgeht oder diese eher für Spiele bzw. im Freizeitbereich verwendet. Dies beeinflusst die Umfrage insofern, dass bsw. eine Person die bereits mit AR arbeitet eine andere Ansicht auf einen möglichen Einsatz des neuen Systems in ihrem Umfeld hat.

3. Do you often use videostreams ? (daily / never)
Dabei sind neben AR-Streams auch jegliche Arten von Videostreams gemeint, wie etwa die Nutzung von Streamingportalen. Mithilfe dieser Frage soll erfasst werden, ob Personen mit der Problematik der herabgesetzten Videoqualität oder möglichen Versatz von Bild und Ton häufiger in Kontakt treten. Diese Erfahrungen beeinflussen die Meinung insofern, welche Eigenschaften des Streams bedeutsamer sind als andere.

Die folgenden Fragen werden für jeden Stream beantwortet.

4. I needed to learn more before I could get going with the system. (strongly disagree / strongly agree)
Die Antwort auf diese Frage soll zeigen, wie intuitiv das System verwendet werden kann. Ebenfalls zeigt sich hier, ob mögliche Hilfsanweisung integriert werden sollten, um den Nutzern eine bessere Anleitung zu geben.

5. I found the stream very consistent. (strongly disagree / strongly agree)
Die Antwort auf diese Frage ermöglicht eine Aussage darüber, ob für den Nutzer genügend FPS vorhanden waren, damit eine Bewegung flüssig ablaufen konnte. Anhand dieser Angaben soll abgeschätzt werden, wie stark der Mensch den Unterschied wahrnehmen kann und ab wann eine Bewegung als flüssig gewertet wird.

6. I thought there was too much delay between voice and picture. (strongly disagree / strongly agree)
Anhand dieser Frage soll festgestellt werden, wie die Testpersonen auf die Latenz im jeweiligen Stream reagieren. Zudem kann durch die Unterschiede zwischen den Bewertungen festgestellt werden, wie weit sich die Meinung zwischen den beiden Streams verändert. Somit sind hier sowohl die eigentliche Wertung als auch die Differenz der beiden Wertungen für die Ergebnisse relevant.

7. I could see all details clearly. (strongly disagree / strongly agree)
Die Ergebnisse dieser Frage lassen eine Aussage über die Wahrnehmung der Bildauflösung zu. Dabei wird zusätzlich sichtbar, in wie weit sich ein herabsetzen der Auflösung auf den Betrachter auswirkt. Durch die Veränderung der Wertung zwischen den beiden Streams kann dieser Wert bestimmt werden.

8. I could follow all explanations very well. (strongly disagree / strongly agree)
Durch die Ergebnisse dieser Frage soll es möglich sein zu erkennen, ob der Nutzer den Inhalt des Streams verstehen konnte. Dies könnte beispielsweise durch nicht erkennen von Objekten oder zu starke Verzögerung verhindert werden. Zudem zeigt sich hier auch die Nützlichkeit der Technik im allgemeinen.

9. I think I could follow better without video. (strongly disagree / strongly agree)
Mithilfe dieser Frage soll beobachtet werden, ob die Testpersonen den Stream als eine Hilfe empfinden. Dabei ist gemeint, ob man beispielsweise durch das Videosignal oder eine etwaige Verzögerung so abgelenkt wurde, dass man schlechter folgen konnte, als es ohne das Video der Fall gewesen wäre.

10. I think that I would like to use the system frequently. (strongly disagree / strongly agree)
Diese Frage liefert Ergebnisse zu einer der zentralen Fragen dieser Arbeit. Dabei soll geklärt werden, ob der Nutzer selbst das System als eine positive Ergänzung in seinem Arbeitsalltag ansieht und es dort entsprechend gern häufiger verwenden würde. Dabei ist allerdings zu beachten, dass eine Ablehnung nicht unbedingt dem System

gelten muss, sondern auch durch den Aufgabenbereich der Testperson bedingt sein könnte. Somit steht diese Frage im Bezug zu den Grundangaben aus Frage eins und zwei.

11. I felt very confident using the system. (strongly disagree / strongly agree)
Hierbei soll geklärt werden, wie die Nutzer ihre Arbeit mit dem System einschätzen. Dabei ist insbesondere gefragt, wie zufrieden/zuversichtlich sich dabei waren, da dies mit darüber entscheidet, ob man ein System häufiger verwenden würde.

12. How insecure, discouraged, irritated, stressed and annoyed were you? (very low / very high)
Das Ergebnis dieser Frage sagt aus, wie sehr sich die Testperson durch das System gestört gefühlt hat. Dabei sollen sowohl Irritationen durch bsw. Ungenauigkeiten oder Unsicherheiten durch Verzögerungen einbezogen werden.

13. How mentally demanding was it to follow the explanation? (very low / very high)
Die mentale Anstrengung, welche benötigt wird, um den Erklärungen zu folgen wird mithilfe dieser Frage abgefragt. Damit soll eingeschätzt werden, ob die Verwendung dieses Systems anstrengender ist als die Verwendung eines anderen Systems. Ebenso kann so abgeschätzt werden, wie "einfach" die Verwendung des Systems war.

14. How hurried or rushed was the pace of the explanation? (very low / very high)
Mithilfe dieser Frage soll eingeschätzt werden, wie eilig die Erklärung vorgetragen wurde bzw. als wie eilig dies empfunden wurde. Somit gibt sie eher eine direkte Einschätzung zur Nutzerstudie als zur Aufgabenbestellung selbst.

15. How much latency do you think was given ? (in milliseconds or seconds)
Mithilfe dieser Frage soll abgeschätzt werden, wie die Testpersonen die Latenz in den Videos einschätzen. Dies soll einen Rückschluss darauf zulassen, wie sehr sich die Wahrnehmung des Menschen von der real gemessenen Zeit unterscheidet. Dabei wird insbesondere darauf geachtet, wie sehr der Unterschied zwischen den beiden Streams wahrgenommen wird, um zu erkennen wie sich die Wahrnehmung im Vergleich zur Realität verhält.

16. Would you want to have something described that way? (yes / no / Don't now)
Hierbei soll ein allgemeiner Überblick erfolgen, wie das System angenommen wird. Dabei ist es besonders wichtig zu erfahren, ob die Testpersonen es gern für ähnliche

Verwendungszwecke einsetzen würden. Durch diese Einschränkung wird ausgeschlossen, dass Personen das System ablehnen, da es in ihrem Arbeitsgebiet nicht benötigt wird.

Zusätzlich wird nach der Betrachtung beider Streams ein Vergleich durchgeführt, welcher aus den folgenden Fragen besteht.

1. Which system would you prefer to use? (first stream / second stream / other / none)
Mithilfe dieser Frage soll ermittelt werden, welchen Stream die Personen bevorzugen. Hierfür werden allerdings außerhalb der beiden Streams ebenfalls die Antwortmöglichkeiten "other" und "none" geboten, so dass auch eine völlige Ablehnung oder der Wunsch nach einem anderen Stream ausgedrückt werden kann. Um eine genauere Aussage zu erhalten, wird nach der Frage darum gebeten, die Entscheidung zu begründen.

2. What is more important for you? (high resolution / no latency)
Diese Frage stellt den Testpersonen eine der Zentralfragen dieser Arbeit. Dabei wird sich direkt auf die Person bezogen, also was der explizite Tester für sich selbst als wichtig erachtet. Auch hier wird um eine Begründung der Entscheidung gefragt.

3. What do you think is more important for the system in general? (high resolution / no latency)
Abweichend von Frage 2 des Vergleichs ist hier eine Information darüber gefordert, was die Testperson allgemein für den wichtigeren Punkt hält. Dieser kann natürlich der persönlichen Meinung entsprechen, muss es allerdings nicht. Hierbei wird auch um eine Erklärung gebeten, so dass vor allem bei verschiedenen Angaben bei Frage 2 und 3 eine Erklärung vorhanden ist.

4. Further comments:
Hier wurde zusätzlicher Platz gelassen, so dass die Testperson zusätzliche Gedanken oder Informationen vermerken kann.

7.5. Erstellen der Szenarios

Für die Nutzerstudie wurden zwei Szenarios angefertigt, welche sich durch ihre Stream-Eigenschaften unterscheiden.
Aufgrund des in Unterabschnitt 6.1 erstellten Netzwerks ist es hierbei möglich, die beiden Szenarien als einzelne Unity-Projekte zu erstellen, welche nacheinander mit dem gleichen Host verbunden werden können. So ist es für die Testperson nicht notwendig, während der Studie eine weitere Applikation zu starten.
Hierfür werden ausgehend von der Applikation zwei neue Projekte generiert. Die Veränderungen an diesen werden in den folgenden beiden Abschnitten beschrieben.

Szenario 1

Szenario 1 zeichnet sich dadurch aus, dass eine möglichst hohe Auflösung angeboten wird. Hierfür wird die vollständige Punktezahl (368640 Punkte) bei 720p (HD)-Auflösung übertragen, wobei die in Abschnitt 6 beschriebene Verbesserung übernommen wird, dass nur Punkte mit maximal 1,5m Tiefe übertragen werden.
Zusätzlich wird der Wert, welcher definiert, nach minimal wie vielen Sekunden ein neues Bild verarbeitet und versendet werden darf, angepasst, um eine möglichst große FPS-Zahl zu erreichen.

```
1 now = DateTime.Now;
2  if (((now - old).Milliseconds + (now - old).Seconds * 1000) >200)
       {
3    old = DateTime.Now;
```

Quelltext 16: FPS-Festlegung Szenario 1

Hierfür wird mit jedem Durchlauf der aktuelle Zeitstempel in der Variable *now* gespeichert. Um die möglichst große FPS-Zahl zu erreichen, wurde der Wert in der if-Bedingung in Zeile 2 im Quelltext 16 verändert. Dabei bedeutet 200 in diesem Fall, dass zwischen dem Zeitstempel des letzten gesendeten Bildes und des aktuellen Bildes mindestens 200ms liegen müssen. Diese Zahl wurde anhand der Verarbeitungszeiten der HoloLens und des Clients bestimmt. Hierbei wurde das gleiche Verfahren verwendet, wie in Abbildung 6.3 beschrieben. Die einzelnen Werte der Zeitschritte werden in Tabelle 8.1 genauer betrachtet. Ebenfalls wie in Abbildung 6.3 beschrieben sind auch bei diesem Client mehr als 200ms zwischen den Bildern möglich. Dies geschieht, falls ein Bild beispielsweise 198ms nach dem zuletzt gesendeten abgefragt wurde und somit verworfen wird. Das darauf folgende Bild hat nun dadurch deutlich mehr Verzögerung.

$$1000ms/200ms = 5 \tag{1}$$

Somit ergibt sich eine maximale FPS-Zahl von etwa 5 FPS. Diese wurden, wie in Gleichung 1 zu sehen ist, berechnet. Dabei wird die Kinect auf 5FPS Bildrate gestellt, da nur die Einstellungsmöglichkeiten 5/30/60 FPS vorhanden sind. 5 FPS genügen hier um ausreichend Bilder zu liefern.
Beim Start der Applikation auf der HoloLens erscheint eine freie Sicht auf die Umgebung, da bis zum Verbinden des Hosts kein Hologramm eingeblendet wird.

Szenario 2

Szenario 2 verdeutlicht durch eine geringere Auflösung eine geringere Latenz. Da durch eine geringere Auflösung weniger Daten übertragen werden müssen, ist hier ebenfalls eine höhere Bildwiederholrate möglich. Dies basiert darauf, dass sobald weniger Punkte vorhanden sind, auch die Verarbeitungszeiten auf HoloLens und Client geringer sind. Die genauen Werte sowie ein Vergleich sind dabei Tabelle 8.1 und Tabelle 8.1 zu entnehmen.
Ebenfalls wie in Szenario 1 wurde hier die FPS-Anzahl durch die if-Bedingung geregelt. Dabei ermöglichte es allerdings die dreifach geringere Auflösung, dass nun minimal 80ms auf das nächste Bild gewartet werden müssen. Eine dreifach geringere Auflösung ist damit zu begründen, dass so ein deutlicher Unterschied in Latenz und Auflösung sichtbar war, dass Bild allerdings noch nicht zu undeutlich/lückenhaft war. Dafür wurde die Bedingung in Zeile 2 in Quelltext 17 verändert. Auch hier wird der aktuelle Zeitstempel in der Variable *now* gespeichert.

```
1 now = DateTime.Now;
2  if (((now - old).Milliseconds + (now - old).Seconds * 1000) >80){
3    old = DateTime.Now;
```

Quelltext 17: FPS-Festlegung Szenario 2

Aus diesen 80ms minimaler Wartezeit kann nun wieder die maximale FPS-Zahl von 12,5 fps (siehe Gleichung 2) errechnet werden.

$$1000ms/80ms = 12,5 \tag{2}$$

Somit kann hier mehr als eine Verdopplung der FPS zu Szenario 1 erreicht werden. Durch die geringere Punktanzahl funktioniert auch die Netzwerkübertragung erheblich schneller. Somit ergibt sich für Szenario 2 ein Bild mit geringerer Latenz und größerer FPS-Zahl als in Szenario 1.
Um bei einer geringeren Auflösung trotzdem ein gleichmäßiges Bild zu erhalten, wurde entschieden, etwa 1/3 der in Szenario 1 verwendeten Punkte zu übertragen. Dies wurde durch einen Zähler umgesetzt, welche durch modulo Berechnung mit

drei, nur jedes dritte Pixel zur Verarbeitung freigibt. Dabei ergibt sich für die Verarbeitung der in Quelltext 18 dargestellte Ablauf.

```
int j = 0;//only send unsed points
int k = 0;
for (int i = 0; i < number; i++){
 if (xyzArray[i].Z < 1500) { //check if background
  if (k % 3 == 0) {
   sendbyte[j * 9 + 4] = (byte)(xyzArray[i].Z & 255);
   sendbyte[j * 9 + 5] = (byte)(xyzArray[i].Z >> 8);

   if(xyzArray[i].X < 0) { //check if negativ
    sendbyte[j * 9] = (byte)((xyzArray[i].X * -1) & 255);
    sendbyte[j * 9 + 1] = (byte)(((xyzArray[i].X * -1)  >> 8) +
        128);
   }
   else {
    sendbyte[j * 9] = (byte)(xyzArray[i].X & 255);
    sendbyte[j * 9 + 1] = (byte)(xyzArray[i].X >> 8);
   }

   if((-xyzArray[i].Y) < 0) { //check if negative
    sendbyte[j * 9 + 2] = (byte)((-xyzArray[i].Y * -1) & 255);
    sendbyte[j * 9 + 3] = (byte)(((-xyzArray[i].Y * -1) >> 8) +
        128);
   }
   else {
    sendbyte[j * 9 + 2] = (byte)(-xyzArray[i].Y & 255);
    sendbyte[j * 9 + 3] = (byte)(-xyzArray[i].Y >> 8);
   }

   //serialize colors
   sendbyte[j * 9 + 6] = colorArray[i].B;
   sendbyte[j * 9 + 7] = colorArray[i].G;
   sendbyte[j * 9 + 8] = colorArray[i].R;
   j++;
  }
  k++;
 }
}
```

Quelltext 18: Vorbereitung der Bilder Szenario 2

Dieser Ablauf besteht aus einer for-Schleife, welche 3 Laufvariablen besitzt. Dabei beschreibt i, an welcher Stelle des originalen Datensatz aktuell gearbeitet wird. Entsprechend dient der Vergleich von i zur Länge des Datensatzes auch als Abbruchbedingung.
Vor jedem verarbeiten eines Bildes werden die beiden anderen Variablen j und k auf 0 gesetzt. j beschreibt dabei die Position im Array der verarbeiteten Daten, welche aktuell beschrieben wird. Somit wird dieser Wert nur inkrementiert, wenn auch wirklich ein neuer Datensatz abgespeichert wurde. Dies ist der Fall, falls der beschriebene Punkt nicht weiter als 1,5m in der Tiefe von der Kamera entfernt liegt und der dritte Punkt ist.
Die Variable k zählt dabei die Punkte, welche als potenziell verwertbare Punkte angesehen werden. Durch eine modulo Berechnung von k durch 3, wird hier jeder dritte verwendbare Wert erfasst. Das eigentliche Serialisieren erfolgt dann genauso wie in der der Studie zugrundeliegenden Version des Systems und wurde bereits in Unterabschnitt 6.3 ausführlich beschrieben.

Host

Für die Nutzerstudie wird die Applikation für den Host unverändert übernommen. Dabei ist zu beachten, dass die Anzahl an Punkten für die Mesh-Darstellung der vollen Auflösung des 720p Bildes entspricht, so dass selbst bei vollständiger Übertragung kein Überlauf passieren kann. Die maximale Punktanzahl beträgt dabei 368.640 Punkte. Dabei muss der Host nicht angepasst werden und nur einmalig auf die HoloLens gespielt werden. Ab diesem Zeitpunkt kann jede Testperson die Applikation auf der HoloLens starten und sieht, sofern schon ein Client aktiviert wurde, eine Ansicht wie in Abbildung 24. Hierbei handelt es sich um das Hologramm des gesendeten Streams

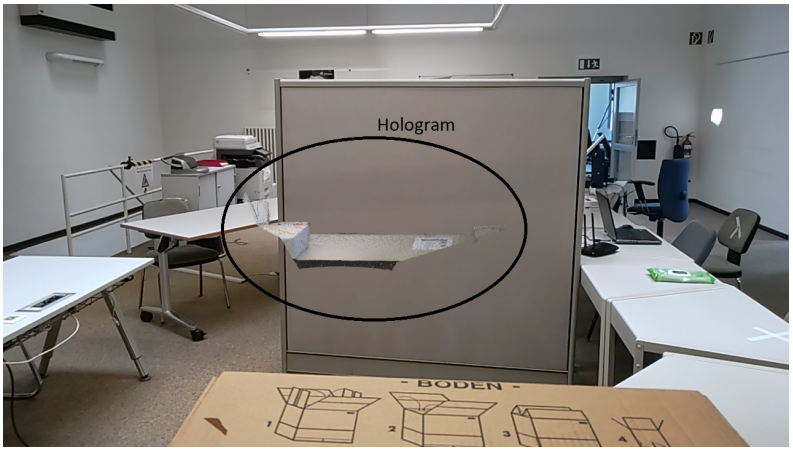

Abbildung 24: Anfangsdarstellung des Hosts

Szenario	1	2
Abfragen	72	72
Serialisieren	2	2
Netzwerkübertragung	288	159
Deserialisieren	48	33
Darstellen	45	31
Summe	455	297

Szenario	1	2
Client	74	74
Netzwerk	288	159
Host	93	64
Summe	455	297

Tabelle 1: Vergleich der Verarbeitungszeiten in ms

8. Ergebnis und Fazit

Resultierend aus den zuvor durchgeführten Methoden, können nun die Ergebnisse dieser Arbeit und eine Antwort auf die aufgeworfene Forschungsfrage erfasst werden. Dabei werden zuerst die Szenarios mithilfe erfasster Verarbeitungszeiten betrachtet. Anschließend wird eine ausführliche Auswertung der Nutzerstudie durchgeführt, welche in einen Zusammenhang mit den Messergebnissen gestellt werden kann. Schlussendlich erfolgt ein Fazit der Bachelorarbeit.

8.1. Betrachtung der Verarbeitungszeiten

Nach der Fertigstellung der Szenarien wurden diese mithilfe von Zeitstempeln näher betrachtet. So konnten Erkenntnisse darüber gewonnen werden, welche Abschnitte welchen Zeitaufwand beinhalten. Dabei wurden die Anwendungen anhand der in Abbildung 25 dargestellten Abschnitte in Bereiche unterteilt.

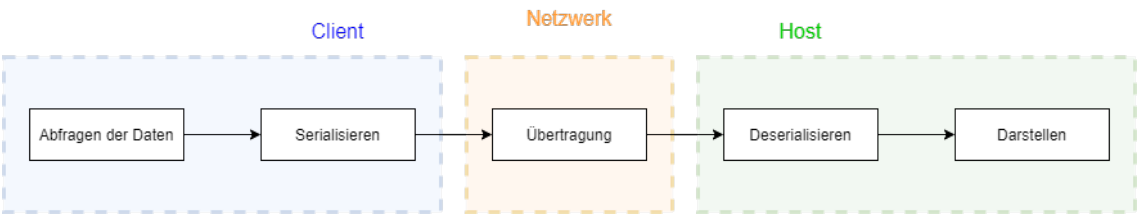

Abbildung 25: Aufteilung der Arbeitsschritte

Diese Bereiche wurden jeweils mittels Zeitstempeln erfasst. Dabei wurde die Applikation jeweils über 5 Minuten betrachtet, da dies der Dauer eines Streams in der Nutzerstudie entspricht. Die daraus resultierenden Mittelwerte sind in Tabelle 1 in ms dargestellt.

Das Abfragen der Daten dauert bei beiden Szenarien gleich lang, da diese anfangs das gesamte Bild der Kinect abrufen. Auch das serialisieren nimmt die gleiche Zeit in Anspruch. Das hierbei kein Unterschied vorliegt, ist mit der enormen Verarbeitungsgeschwindigkeit zu begründen. Da dies allgemein nur 2 ms in Anspruch nimmt, ist eine genaue Unterscheidung schwierig feststellbar, da in dieser Größenordnung auch weitere Prozesse des ausführenden Geräts einen Einfluss auf den Wert ausüben könnten. Ebenfalls wird hier vermutlich auch die Erfassung der Zeit einen gewissen Einfluss ausüben, da auch dieser Prozess Zeit benötigt.
Somit ergibt sich auf Seiten des Clients kein bedeutenden Unterschied zwischen der Zeit, welche für die Verarbeitung eines Datensatzes benötigt wird. Dies deckt sich damit, dass die Einstellung wie oft Bilder gesendet werden, vom Host abhängt.
Bei diesem sind deutliche Unterschiede zwischen den Werten sichtbar. Sowohl die Deserialisierung als auch die Darstellung bzw. Umrechnung der Daten sind bei einer geringeren Datenmenge erheblich schneller.
Dabei bildet die Menge an Daten die Ursache für diesen Unterschied. Somit ergibt sich eine Dauer für den Host von durchschnittlich 93ms pro Bild für Szenario 1 (ca. 240.000 Punkte) und 64 ms pro Bild für Szenario 2 (ca. 80.000 Punkte).
Diese Zahlen wurden für erste Tests der Absende Rate verwendet. Dabei zeigte sich allerdings deutlich, dass ein solch schnelles, allerdings schon begrenztes, absenden der Bilder weiterhin zu summierenden Verzögerungen führt.
Um einen korrekten Wert zu erhalten, muss zusätzlich der Unterschied zwischen der Verarbeitungszeit von Host und Client beachtet werden. Dieser kann bei Szenario 2 vernachlässigt werden, da der Client hier langsamer als der Host ist (74ms zu 64ms, siehe Tabelle 1 rechts). Somit wartet der Host auf neue Bilder und es kann zu keinem Stau kommen. Da sowohl die 74ms als auch die 64ms leicht schwankend sind, und einmalig vorkommende Schwankungen zu langfristigen Verzögerungen führen, wurden hier 80ms als "minimale Dauer zwischen zwei Bildern" festgelegt.
Bei Szenario 1 besteht allerdings ein beachtenswerter Unterschied (74ms zu 93ms). Würde dieser ignoriert werden, würde sich die Ausgabe des Bildes mit jedem Bild um 19ms verzögern. Ein Test mit 100ms als Ausgabe Rate zeigte allerdings, dass dieses Problem weiterhin bestehen bleibt. Vermutlich liegt hierbei zugrunde, dass zu den 93ms Darstellung noch zusätzliche Zeit für das absenden und ankommen hinzukommen, welche hier von größerer Bedeutung wäre als bei Szenario 2. Es erwies sich erst ab 200ms Wartezeit als möglich, eine dauerhaft gleichbleibende Latenz zu erreichen und keine summierende Verzögerung zu erzeugen.
Ein deutlicher Unterschied zeigt sich auch zwischen den Übertragungszeiten. Dieser ist ausschließlich auf der geringeren Datenmenge zu begründen, welche versendet werden muss. Dabei können Einflüsse durch andere Netzwerkkommunikationen aus-

geschlossen werden, da Router und Client nicht ans Internet angeschlossen waren, sondern als unabhängiges System agierten. Ein Vergleich aller weiteren technischen Daten der beiden Szenarien ist in Tabelle 2 ersichtlich.

Szenario	1	2
Punktanzahl	240.000	80.000
FPS	5	12,5
Latenz	455ms	297ms

Tabelle 2: Vergleich der Szenarien

Dabei wurden bei den Aufnahmen der Nutzerstudie die dargestellten Werte erreicht. Die Punkteanzahl von Szenario 2 entspricht genau einem drittel der vollen Punktanzahl von Szenario 1. Der Unterschied von Szenario 1 zur vollen Punktzahl (368.640) entsteht durch das weglassen der Hintergrundpunkte.
Der Unterschied der Auflösung ist anhand vor Abbildung 26 sichtbar. Dabei stellt die linke Abbildung Szenario 1 und die rechte Abbildung Szenario 2 dar.

Abbildung 26: links: Szenario 1 mit ca. 240.000 Punkten , rechts: Szenario 2 mit ca. 80.000 Punkten

Dabei sind insbesondere die scheinbar helleren Farben sichtbar. Diese beruht darauf, dass mehr Punkte gegeben sind, so dass die Farben kräftiger wirken. Die Farbwerte selbst sind identisch. Zudem ist erkennbar, dass in beiden Streams die Person als solche noch erkennbar ist.

Ebenso lassen sich die verschiedenen Bildwiederholfrequenzen grafisch darstellen. Hierfür wurden alle Einzelbilder eines Vorgangs nebeneinander als Bildstrecke gelegt (siehe Abbildung 27). Hier ist bereits an der Anzahl der Bilder sichtbar, dass Szenario 1 (oben) deutlich weniger Einzelaufnahmen liefert wie Szenario 2 (unten).

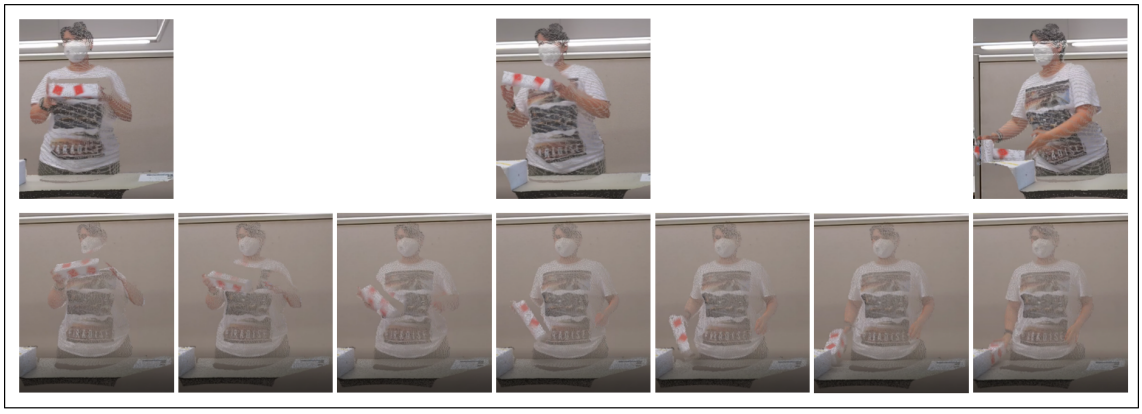

Abbildung 27: Darstellung der FPS durch eine Bildstrecke. oben: Szenario 1, unten Szenario 2

Da die eigentliche Bewegung in beiden Szenarien gleich schnell stattfand, sind die Unterschiede zwischen den einzelnen Bildern bei Szenario 2 geringer.

8.2. Auswertung Nutzerstudie

Um beurteilen zu können, wie die Wahrnehmung eines solchen Streams bei Testpersonen erfolgt, wurde eine Nutzerstudie mit dem Fragebogen aus Unterabschnitt 7.4 durchgeführt. Aus den Ergebnissen wird zuerst der SUS-Wert berechnet und anschließend eine Auswertung der einzelnen Antworten durchgeführt. An der Studie haben insgesamt 15 Personen teilgenommen, 4 Frauen und 11 Männer, im Alter zwischen 20 und 36 Jahren, mit einem Durchschnittsalter von 26,4 Jahren.

SUS-Ergebnis

Für die Berechnung des Wertes wurden die Fragen 4 bis 13 verwendet. Für diese wurden jeweils für Szenario 1 und Szenario 2 die Mittelwerte bestimmt. Diese sind in Tabelle 3 in der zwei und vierten Spalte dargestellt. Um diese als SUS-Score verwenden zu können, müssen sie so skaliert werden, dass die Bewertungsskala der einzelnen Fragen von 4 (positivste Wertung für die Nutzerstudie) bis 0 (negativste Wertung für die Studie) verläuft.[51] Dabei wird anhand der Frage entschieden, ob bei dieser die Antwort 1 oder 5 das gewünschte Ergebnis wäre. Wurde die Frage negativ/negiert gestellt, so ist 1 das gewünschte Ergebnis. Dies ist bei *I needed to learn more before I could get going with the system.* der Fall. Hier ist gewünscht, dass der Nutzer 1 ankreuzt, dieser Aussage also widerspricht. Um dies auf der Bewertungsskala abzubilden, wird "5 - Likertwert" gerechnet. So würde einer negierten Frage mit der Antwort eins auf der Bewertungsskala ein Wert von 4 zugeordnet werden.

[51] vgl. Tho.

Ist die Frage positiv gestellt, wie etwa in *I found the stream very consistent.*, so ist eine 5 auf der Likert-Skala die beste Wertung. Um wieder auf die 0 - 4 Skala zu übertragen, wird hier "Likertwert - 1" berechnet. Somit wäre eine Wertung 5 bei der genanten Frage auf der Auswertungsskala eine 4.[52]
Nach diesem Prinzip wurden alle betrachteten Fragen verarbeitete und die resultierenden Werte sind in Spalte drei und fünf vermerkt.
Diese Werte werden addiert und anschließend mit 2.5 multipliziert. Die so errechnete Zahl entspricht dem SUS-Score. Dieser Wert kann nur begrenzt mit der originalen Skala verglichen werden, da diese andere Fragen beinhaltete. Dabei ist grundsätzlich festzustellen, dass das Ergebnis besser ist, je höher der SUS-Wert ausfällt (siehe Abbildung 28).

Frage	Szenario 1		Szenario 2	
	Durchschnitt	skaliert	Durchschnitt	skaliert
4	1,60	3,40	1,47	3,53
5	3,40	2,40	4,33	3,33
6	2,87	2,13	1,87	3,13
7	3,60	2,60	3,20	2,20
8	4,13	3,13	4,53	3,53
9	1,33	3,67	1,20	3,80
10	3,13	2,13	3,47	2,47
11	3,93	2,93	4,00	3,00
12	1,67	3,33	1,47	3,53
13	2,20	2,80	1,73	3,27
$\sum$		28,53		31,80
SUS-Score		71		80

Tabelle 3: Auswertung des SUS-Fragebogens

[52] vgl. Tho.

Grade	SUS	Percentile range
A+	84.1 - 100	96 - 100
A	80.8 - 84.0	90 - 95
A-	78.9 - 80.7	85 - 89
B+	77.2 - 78.8	80 - 84
B	74.1 - 77.1	70 - 79
B-	72.6 - 74.0	65 - 69
C+	71.1 - 72.5	60 - 64
C	65.0 - 71.0	41 - 59
C-	62.7 - 64.9	35 - 40
D	51.7 - 62.6	15 - 34
F	0 - 51.6	0 - 14

Abbildung 28: Auswertung SUS-Wert[53]

Dabei zeigt sich, dass das zweite Szenario mit einer geringeren Auflösung und einer geringeren Latenz sowie höherer FPS-Zahl besser bewertet wurde. Allerdings sind die Werte zwischen 71 und 80 beide über dem Durchschnitt. Nach Lewis und Sauro, welche den SUS-Wert sowie dessen Auswertung 2013 beschrieben[54], kann der SUS-Wert wie in Abbildung 28 sichtbar ausgewertet werden.
Somit kann Szenario 1 eine eher leicht überdurchschnittliche Bewertung zugeschrieben werden, was bedeutetet, dass die Nutzerfreundlichkeit besser ist als bei vergleichbaren Applikationen. Szenario 2 weißt schon über 85% Zustimmungsrate auf und liegt somit deutlich über dem Durchschnitt.
Auffällig hierbei ist, dass sofern die SUS-Scores für jeden Nutzer einzeln berechnet werden, sich die folgenden Verteilungen ergeben (siehe Abbildung 29).
Dabei wird deutlich, dass die Verteilung der Werte bei Szenario 1 deutlich höher ist, als bei Szenario 2. Somit scheint sich hier die Wahrnehmung einzelner Personen deutlicher zu unterscheiden. Dies wurde auch anhand der Freitextfragen der Fragebögen deutlich, da die Meinungen dazu, wie viele Bildpunkte nötig sind, deutlich unterschieden.

[53] LS18, S.4.
[54] vgl. LS18.

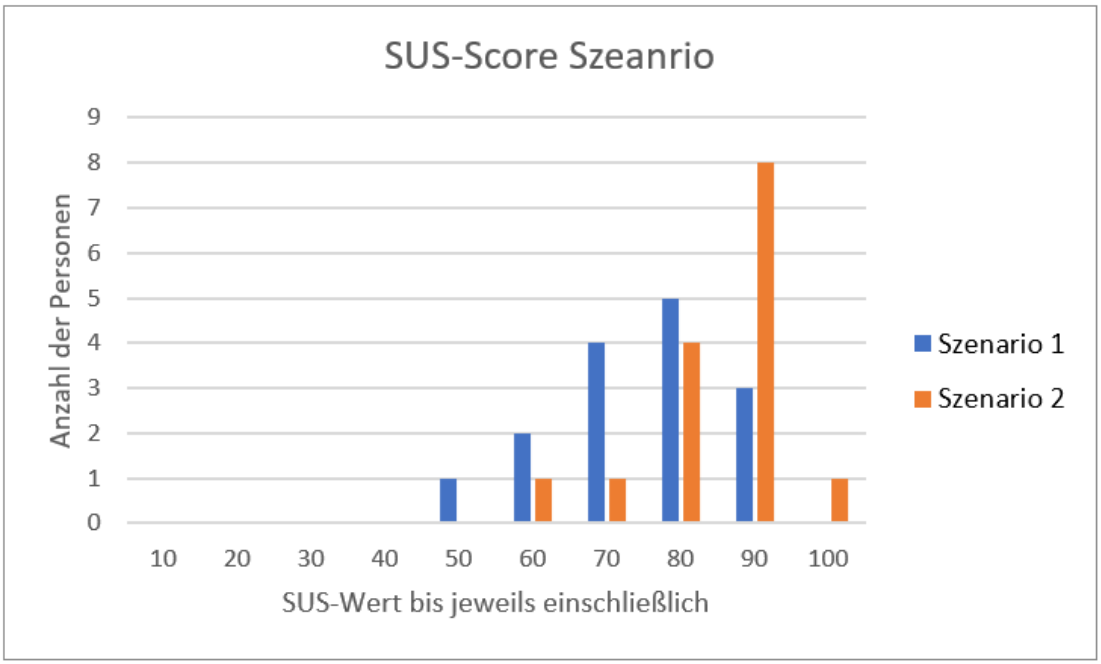

Abbildung 29: Verteilung der SUS-Werte

Auswertung der einzelnen Fragen

Im folgenden werden die Antworten auf die einzelnen Fragen betrachtet und Interpretationsansätze geliefert. Dabei sind Szenario 1 und Szenario 2 zu Vergleichszwecken immer in einem Diagramm dargestellt, wobei die Werte von Szenario 1 immer blau und von Szenario 2 immer orange sind.

1. Do you have experience with AR ? (much experience / no experience)
Anhand der in Abbildung 30 links dargestellten Verteilung wird sichtbar, dass in der Teilnehmergruppe Personen mit verschiedenen Erfahrungen mit AR enthalten waren.

2. Do you often work with AR ? (daily / never)
Bei den Antworten auf Frage 2 wird die Umgebung der Nutzerstudie sichtbar. Da diese in einer Abteilung durchgeführt wurde, in welcher einige Personen regelmäßig mit AR beschäftigt sind, ist hier die Anzahl derer, die täglich damit arbeiten deutlich erhöht. Aber auch einige Personen, welche eher selten bis gar nicht mit der Technologie arbeiten, wurden befragt. (siehe Abbildung 30 mitte).

3. Do you often use videostreams ? (daily / never)
Hierbei zeigt sich, dass gerade zum Zeitpunkt der Durchführung viele Personen täglich mit Videostreams arbeiten bzw. diese privat verwenden. Somit ist allen Personen die Problematik der Auslösung zu Verzögerung/FPS bekannt. Auffällig ist hierbei eine Person, welche angibt, niemals Videostreams zu verwenden (siehe Abbildung 30 rechts).

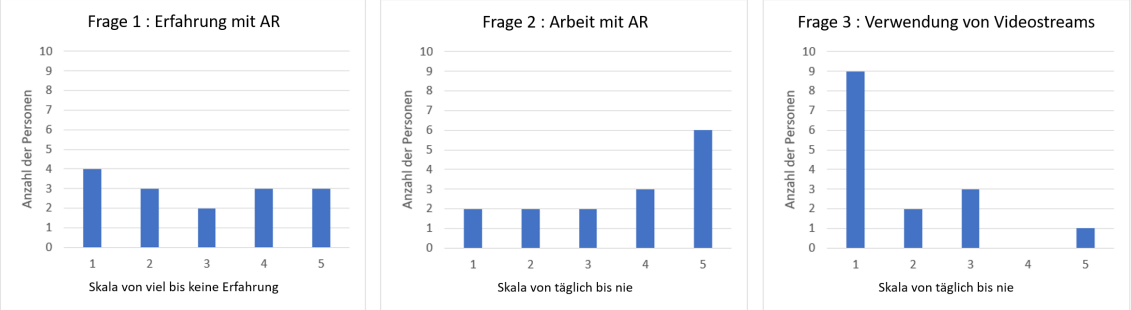

Abbildung 30: Auswertung Vorerfahrung

Die folgenden Fragen bilden den eigentlichen Fragebogen, welcher für jeden Stream beantwortet werden muss.

4. I needed to learn more before I could get going with the system. (strongly disagree / strongly agree)
Die Ergebnisse dieser Frage (siehe Abbildung 31 links) zeigen, dass die Nutzer mit der kurzen Einweisung in Form einer Erklärung, dass sie gleich eine Punktwolke sehen, welche ein Objekt darstellt, zufrieden waren. Dabei unterscheiden sich die Meinungen bei den Szenarien kaum. Es lässt sich schlussfolgern, dass das aktuelle System einfach zu verwenden ist.

5. I found the stream very consistent. (strongly disagree / strongly agree)
Bei den Ergebnissen dieser Frage (siehe Abbildung 31 mitte) zeigt sich erstmals ein deutlicher Unterschied zwischen den beiden Szenarien. Die Nutzer empfanden Szenario 2 hierbei besser als Szenario 1. Dies erscheint aber auch im Zusammenhang mit dem Unterschied der FPS-Zahlen (5 FPS in Szenario 1 zu 12,5 in FPS Szenario 2) als sinnvoll. Interessant ist hierbei, dass mehr als eine Verdopplung der FPS-Zahl einen Unterschied von etwa einem Zähler hervorbringt. Deutlich weniger als die 5FPS bei Szenario 1 sollten allerdings nicht verwendet werden, da diese schon ein grenzwertiges Ergebnis liefern.

6. I thought there was too much delay between voice and picture. (strongly disagree / strongly agree)
Auch die Antwort auf diese Frage entspricht dem erwarteten Ergebnis. Da Szenario 2 eine geringere Latenz aufwies, sind hier die Ergebnisse besser als bei Szenario 1 (siehe Abbildung 31 rechts). Auffällig bleibt, dass trotz einer Verzögerung von etwa 455ms die Testpersonen diese nicht als zu viel einschätzten.

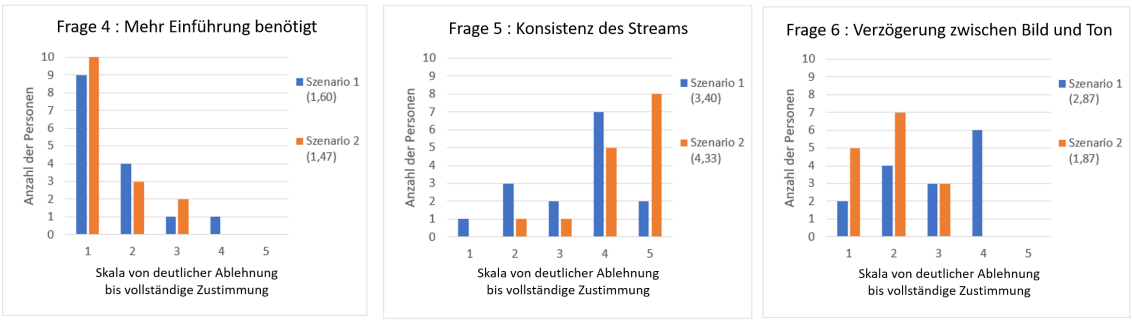

Abbildung 31: Auswertung Frage 4 bis 6

7. I could see all details clearly. (strongly disagree / strongly agree)
Die Ergebnisse von Frage 7 (Abbildung 32 links) zeigen bei beiden Szenarien ein interessantes Muster. Es ist jeweils eine kurze Erhebung, welche von einem Tal gefolgt ist und danach wieder eine hohe Erhebung zeigt. Dabei liegt die Vermutung nahe, dass einige Personen allgemein mit der Auflösung/Detailgenauigkeit des gesamten Systems nicht zufrieden waren. Dies ist zusätzlich mit den Daten der Nutzerstudie zu belegen. Durchschnittlich hat sich die Wahrnehmung der Detailgenauigkeit um 1,4 Stellen auf der Likert-Skala verschoben. Dabei sind zwei Stimmen aufgetreten, welche den Unterschied zwischen Szenario 1 und Szenario 2 an den Enden der Skala sahen. Zudem gab es zwei Personen, für welche kein Unterschied ersichtlich wahr.

8. I could follow all explanations very well. (strongly disagree / strongly agree)
Die Antworten auf diese Frage (Abbildung 32 mitte) bietet einen ersten direkten Anhaltspunkt für die Beantwortung der Forschungsfrage dieser Arbeit. Die Wahrnehmung des mittels des Streams übertragenen Inhalts war eingeschränkt, wenn diese Frage mit Ablehnung beantwortet wurde. Somit ist es äußerst Positiv, dass beide Streams ein gutes Verständnis der Erklärung ermöglichten. Auffällig ist hierbei wieder die eine Person, welche bei Szenario 1 der Erklärung überhaupt nicht folgen konnte. Eine mögliche Begründung wäre, dass durch die starke Verzögerung der Proband dem aktuellen Geschehen nicht mehr folgen konnte.

9. I think I could follow better without video. (strongly disagree / strongly agree)
Anhand dieser Frage wird sichtbar, dass die Personen den Stream als Hilfe wahrnahmen. Hierbei ist Szenario 2 minimal besser bewertet, dieser minimale Unterschied ist allerdings aufgrund der geringen Teilnehmerzahl nicht besonders beachtenswert.

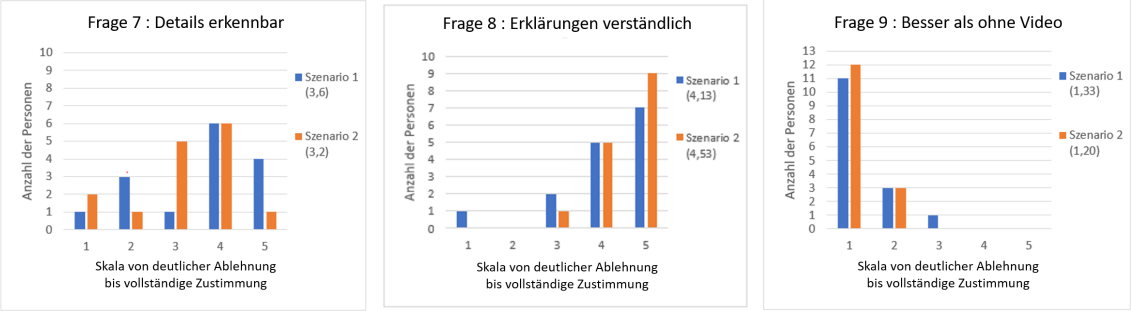

Abbildung 32: Auswertung Frage 7 bis 9

10. I think that I would like to use the system frequently. (strongly disagree / strongly agree)
Die Ergebnisse auf diese Frage (Abbildung 33 links) zeigen, dass noch nicht alle Personen von der Verwendung dieses Systems überzeugt werden konnten. Dabei ist die Zustimmungsquote bei Szenario 2 etwas höher als bei Szenario 1. Um diesen Wert weiter zu heben werden in Abschnitt 9 Verbesserungsmöglichkeiten aufgezeigt.

11. I felt very confident using the system. (strongly disagree / strongly agree)
Hier schneidet Szenario 2 minimal besser ab (Abbildung 33 rechts), als Szenario 1. Allerdings handelt es sich dabei wieder um einen minimalen Unterschied. Der Wert, welchen beide erreichen, kann als gut, aber noch nicht perfekt, angesehen werden. Auffällig ist dabei, dass die Werte bei Szenario 2 eher gleichmäßig auf die Wertungen 3 bis 5 verteilt sind, während Szenario 1 einen Stimmenzuwachs von 1 bis 5 beinhaltet. Somit ist auch hier wieder sichtbar, dass sich die Meinungen zu Szenario 1 sehr unterscheiden.

12. How insecure, discouraged, irritated, stressed and annoyed were you? (very low / very high)
An den Antworten auf diese Frage zeigt sich, dass ein Großteil der Personen sich nicht von der Anwendung gestört fühlte. Insbesondere die beiden Personen, welche für eine 4 stimmten äußerten dazu, dass dies aufgrund der Verzögerung und damit dem Merken von zuvor gesagten zusammenhängt. Somit kann das Ergebnis im ganzen als positiv bewertet werden.

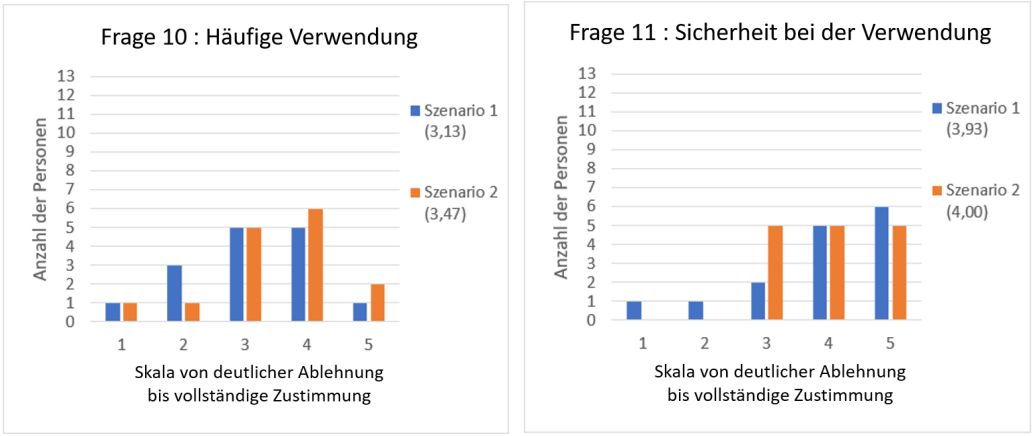

Abbildung 33: Auswertung Frage 10 und 11

13. How mentally demanding was it to follow the explanation? (very low / very high)
Anhand der Ergebnisse dieser Frage (Abbildung 34 mitte) wird sichtbar, dass die Testpersonen Szenario 1 als anstrengender erachten. Dies ist damit zu begründen, dass sich hier der gesprochenen Text kurzfristig gemerkt werden muss, bis das passende Bild erscheint. Diese Tatsache ist zwar grundsätzlich auch bei Szenario 2 nötig, allerdings scheint die hier benötigte Zeit weit weniger problematisch zu sein.

14. How hurried or rushed was the pace of the explanation? (very low / very high)
Diese Frage diente dazu abschätzen zu können, dass etwaige Verständnisprobleme nicht durch zu schnelle Erklärungen oder eiliges Erklären verursacht wurden. Da dies nicht der Fall war (siehe Abbildung 34 rechts) und zudem beide Szenarien die exakt gleichen Durchschnittswerte erreicht haben, wurden die beiden Streams gut vergleichbar vorgetragen.

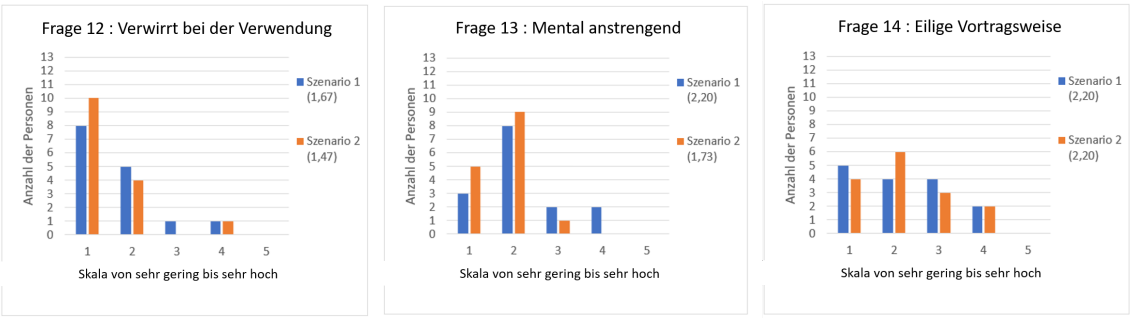

Abbildung 34: Auswertung Frage 12 bis 14

15. How much latency do you think was given ? (in milliseconds or seconds)
Mithilfe dieser Frage sollte abgeschätzt werden, wie sehr die Personen den Latenzunterschied zwischen den beiden Szenarien wahrnehmen.

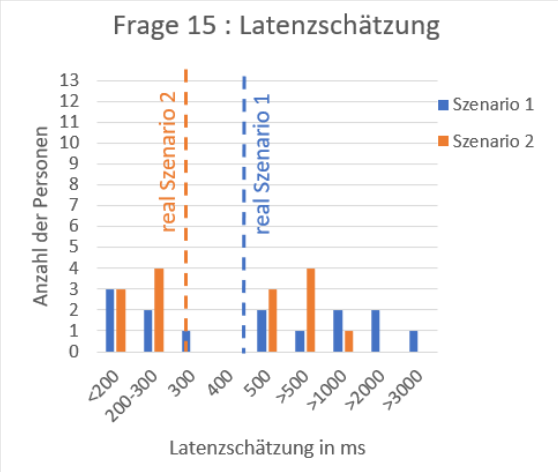

Abbildung 35: Auswertung Latenzschätzung

Die Antworten sind in Abbildung 35 zu sehen. Interessant ist dabei, dass der Unterschied zwischen den beiden Streams deutlich verschieden wahrgenommen wurde. Dabei waren Änderungen von -1500 bis 2500 von Szenario 1 zu Szenario 2 vorhanden. Besonders auffällig ist hier, dass drei Personen Szenario 1 mit geringerer Latenz als Szenario 2 wahrgenommen haben. Da dies in der Realität nicht der Fall war, kann aus dieser Frage geschlussfolgert werden, dass das Schätzvermögen der Testpersonen keine wirklich realen Werte lieferte. Mit einer durchschnittlichen geschätzten Veränderung von 800ms wurde der reale Unterschied von etwa 200ms deutlich stärker wahrgenommen.

16. Would you want to have something described that way? (yes / no / Don't now)
Frage 16 beinhaltet eine der grundsätzlichen Informationen dieser Arbeit. Dabei sollte gezeigt werden, ob Personen diese Art der Anwendung annehmen würden. Das Ergebnis zu dieser Frage (siehe Abbildung 36) lässt die Vermutung zu, dass eine Erklärung mithilfe eines RGBD Streams für einen Großteil der Personen ein positives Erlebnis wäre.

Die folgenden Fragen wurden im Rahmen des Vergleichs der Szenarien gestellt. Somit ist hier pro Person nur noch eine Antwort zu erwarten.

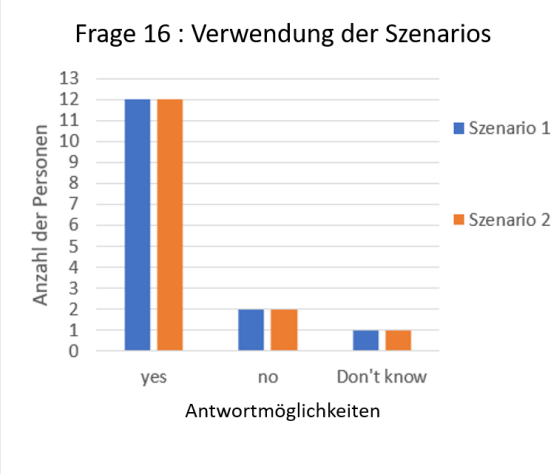

Abbildung 36: Auswertung Verwendung von Szenarios

1. Which system would you prefer to use? (first stream / second stream / other / none)
Da sich die Fragestellung dieser Arbeit damit beschäftigt, welche Eigenschaften die Wahrnehmung eines RGB-D Streams besonders beachtet werden, wird mithilfe dieser Frage eine Informationen von den Nutzern gewünscht, welches Szenario sie bevorzugen würden.
Dabei zeigte sich, dass eine Mehrheit Szenario 2 bevorzugen würde. Ebenso gab es jeweils eine Stimmt für keines der Szenarien und ein anderes. Da zu diesen Fragen um Erklärungen gebeten wurden, ist auch die Grundlage für diese Entscheidung bekannt.

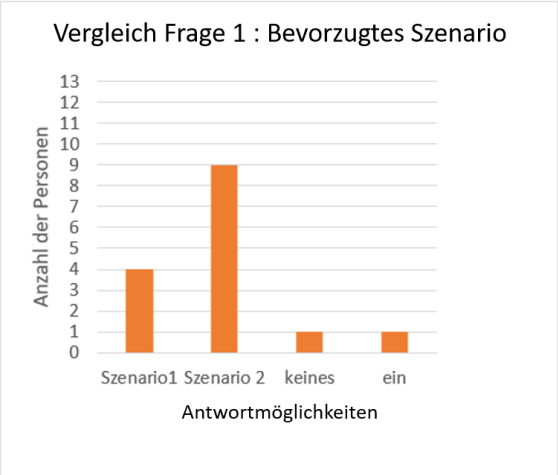

Abbildung 37: Auswertung bevorzugtes Szenario

Für die Entscheidung, keines der Systeme zu verwenden, wurde die Begründung gebracht, dass Szenario 2 eine zu geringe Auflösung und Szenario 1 eine zu geringe FPS-Zahl hatte. Die Begründung, einen völlig anderen Stream zu wünschen lautete "Am Besten hohe Auflösung UND niedrige Latenz"[55].
Als Begründung für Szenario 2 wurde hauptsächlich genannt, dass die Auflösung nicht so schlecht gewesen wäre und das das Gehirn in der Lage ist, fehlende Auflösung auszugleichen, aber keine zusätzlichen Bilder zu erzeugen. Zudem wurde angemerkt, dass sich so kein Text gemerkt werden muss.
Die Begründungen für die Wahl von Szenario 1 waren ähnlich. Hier wurde gesagt, dass eine Kompensation der Latenz möglich ist, allerdings keine Details geistig dazugewonnen werden können. Zudem wurde die Genauigkeit über die FPS-Zahl gestellt, da die zwischen liegenden Bilder ergänzbar sind, die fehlenden Details allerdings nicht.

2. What is more important for you? (high resolution / no latency)
Aus der vorhergehenden Frage konnten schon erste Vermutungen über die Präferenzen der Testpersonen gewonnen werden. Dabei wäre zu erwarten, dass sich diese auch in Frage 2 zeigen. Die Ergebnisse (siehe Abbildung 38 links) liefern allerdings ein interessantes Bild. Es gibt nur eine kleine Mehrheit für welche die Latenz wichtiger als die Auflösung ist. Dabei haben alle Personen, mit Ausnahme von einer, welche zuvor für Szenario 2 gestimmt haben, auch für Latenz als wichtigeren Punkt gestimmt. Ebenfalls auffällig ist, dass keinerlei Wert in der Mitte, also eine kompromissfähig zu finden ist. Die Personen entscheiden immer, dass eines der beiden Kriterien wichtiger ist, als das andere.
Dabei wurde bei dieser Frage bewusst auf die persönliche Meinung abgezielt. Diese konnte als Begründung zusätzlich abgeben werden. Dabei wurden die gleichen Erklärungen wie in Frage 1 abgegeben. Eine höherer Auflösung wurde mit mehr Details begründet. Keine Latenz wurde hauptsächlich damit begründet, dass diese störend wirkt und so eine Erklärung unnötig kompliziert gestaltet.

3. What do you think is more important for the system in general? (high resolution / no latency)
Im Gegensatz zu Frage 2 des Vergleichs wird hier danach gefragt, was allgemein als wichtiger angesehen wird. Dabei verlagern sich die Meinungen deutlich näher an eine Kompromisslösung als zuvor (siehe Abbildung 38 rechts). Diese ist nun die Variante mit den meisten Stimmen. Dabei haben alle Personen entweder näher an der Mitte oder gleichbleibend gestimmt.

[55]Teilnehmer der Nutzerstudie

Diese Entscheidungen werden damit begründet, dass je nach Anwendungsfall verschiedenen Faktoren wichtig sind und somit eine Kompromisslösung möglichst viele Fälle abdecken könnte. Dabei wird immer wieder "Frame rate is as impotant as resolution and latency"[56] in verschiedenen Variationen als Begründung gegeben.

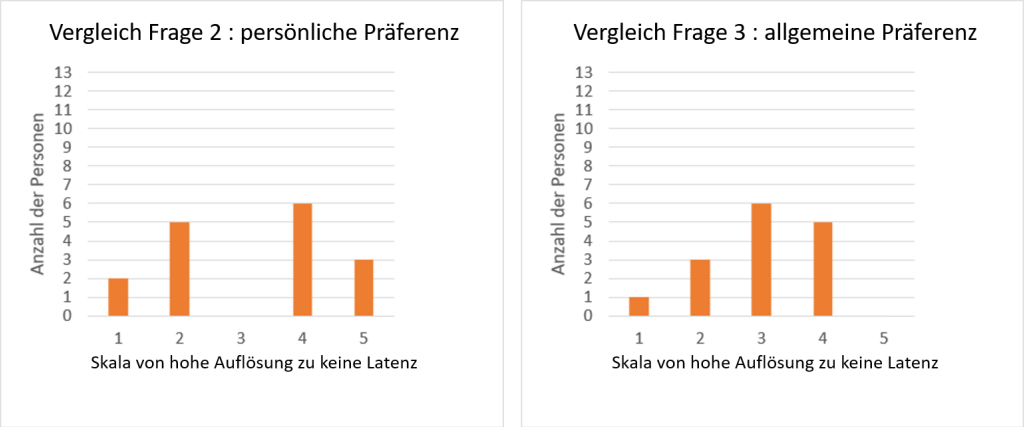

Abbildung 38: Auswertung Vergleich persönliche vs. allgemeine Präferenz

4. Further comments:
Zusätzlich konnten die Teilnehmer weitere Informationen hinter dem Fragebogen notieren. Diese Möglichkeit wurde hauptsächlich verwendet um auf zwei Probleme aufmerksam zu machen. Einerseits wurden die auftretenden Schatten beschrieben, welche die Erkennbarkeit der Boxen behinderten. Diese Schatten entstehen, da die Abtastung wie eine Decke auf den Objekten liegt. Da die Kamera sich nicht in der exakt gleichen Höhe wie der Betrachter befinden, erscheint für diesen kein durchgängiges Bild, sondern Punkte, welche an einigen Stellen hintereinander liegen, an anderen Stellen aber keinerlei Punkte zu sehen sind. Die Entstehung dieser Schatten ist in Abbildung 39 dargestellt. Dabei ist die blaue Aufnahme das Bild der Kamera und orange die Aufnahme des Nutzers. Werden diese beiden Bilder übereinander gelegt, entsteht die dritte Zeichnung. Die grünen Flächen sollten sichtbar sein, und werden auch vom Kamerabild erfasst. Die schwarze Fläche entspricht einem Schatten. Hier sollte eine Darstellung sein, diese ist jedoch aufgrund der Position der Kamera nicht aufgenommen wurden. Die rosa Bereiche beinhalten ein weiteres Phänomen. Hier sind Strukturen erkennbar, welche für den Nutzer eigentlich verdeckt sind. Da allerdings andere Punkte vor diesen Objekten sind, werden diese kaum wahrgenommen und nicht als Problem erkannt.

[56] Umfrageteilnehmer

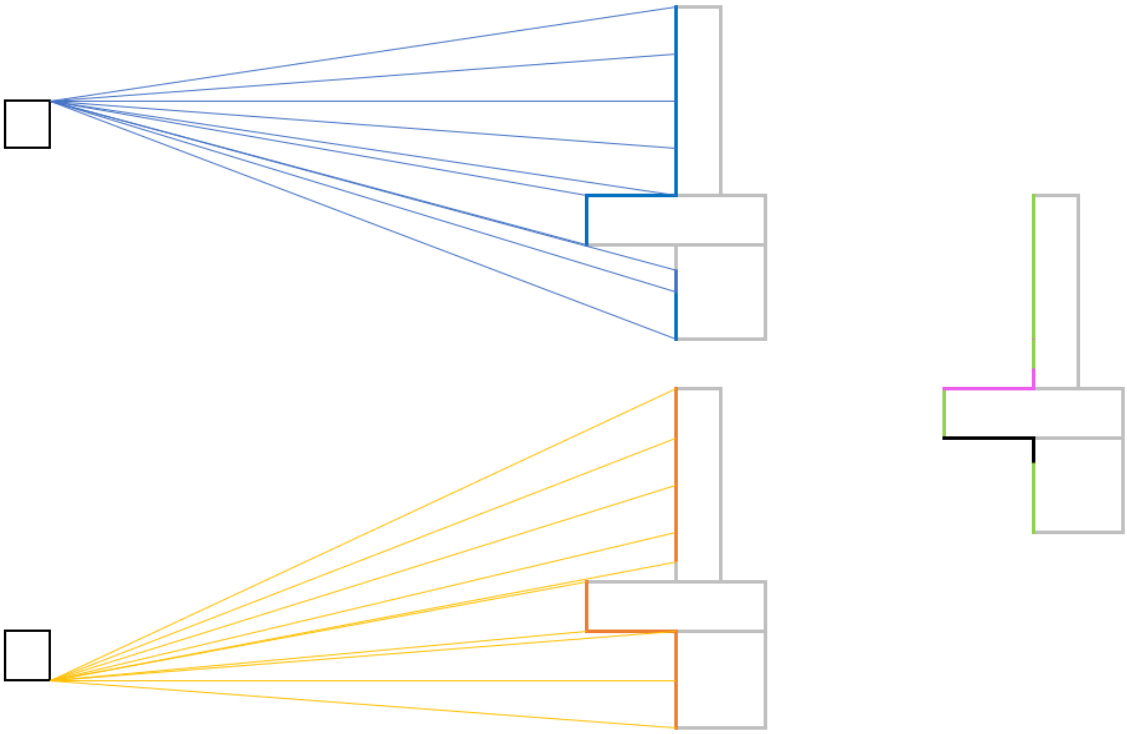

Abbildung 39: Entstehung von Schatten im Hologramm

Eine reale Darstellung dieses Problems ist in Abbildung 40 zu sehen, wobei das linke Bild die Ansicht der Nutzers und das rechte Bild eine seitliche Ansicht auf das Hologramm ist.

Abbildung 40: Darstellung von Schatten im Hologramm

Das zweite beschriebene Problem besteht darin, dass sofern einmal eine große Latenz erreicht wurde, diese nicht wieder abgebaut wird. Dies ist damit zu begründen, dass

die Bilder hintereinander angezeigt werden, egal wie alt sie sind. Somit würden auch bei einer größeren Verzögerung danach alle Bilder gleichbleibend angezeigt werden. Da die Kalkulation, wie viele Bilder übertragen werden, sehr eng erfolgt ist, besteht hierbei auch kaum die Möglichkeit, eine solche Verzögerung wieder aufzuholen.

8.3. Fazit

Zusammenfassen kann gesagt werden, dass die Wahrnehmung von RGB-D-Videostreams zwischen verschiedenen Personen sehr variiert. Die meisten befragten Personen würden für sich selbst eine auf die persönlichen Bedürfnisse zugeschnittene Variante bevorzugen. Dabei unterscheidet sich die Wahrnehmung von Auflösung und insbesondere Latenz von Person zu Person stark. Besonders Veränderungen der Latenz werden deutlich höher geschätzt, als sie wirklich sind. Entsprechend kann schon eine geringe Abänderung an dieser Eigenschaft eine große Reaktion bei den Nutzern auslösen. Unterschiede der Auflösung werden nicht so kritisch betrachtete, hier sind kleine Variationen vermutlich ohne größerer Meinungsänderungen möglich.
Da kaum Interesse an einer Kompromisslösung besteht und oftmals darauf hingewiesen wird, dass bei verschiedenen Anwendungen vermutlich andere Gewichtungen für Auflösung/Latenz/FPS gefordert werden, scheint eine allgemeine Lösung nicht angestrebt zu sein. Entsprechend wäre es ein Ansatz, die Einstellungen für einen Stream je nach Anwendung einzeln wählen zu können.

9. Ausblick

Um die Anwendbarkeit dieses Systems weiter zu verbessern, ergeben sich mehrere Weiterführungsmöglichkeiten. So war bei der aktuellen Version der Software es noch nicht möglich, dass gleichzeitig zwei Kameras angeschlossen werden können, bzw. aktuell würden ausschließlich die Daten des zuerst verbundenen Clients angezeigt werden. Dies könnte so erweitert werden, dass mehrere Kamerabilder nebeneinander oder sogar in Verbindung zueinander dargestellt werden könnten. Dabei wäre insbesondere bei zweiter Version darauf zu achten, dass gleiche Punkte erkannt werden müssten um unnötiges Darstellen zu hindern.
Eine andere Erweiterung wäre es, dass das Bild auf mehrere HoloLens übertragen werden könnte. Dies wäre beispielsweise so umsetzbar, dass es zwei verschiedene Typen von Clients gibt. Einerseits die bereits existierenden, welche die Daten einer Kamera senden. Andererseits könnte eine neue Art von Clients erstellt werden, welche vom Host die erhaltenen Daten weitergeleitet bekommen und so dass gleiche Hologramm darstellen wie der Host. Dabei wäre darauf zu achten, dass zwischen den einzelnen HoloLens ebenfalls eine Verzögerung besteht.
Um die Verzögerung gänzlich auszuschließen wäre es eine Option neben dem Bild der Azure Kinect ebenfalls die Daten des Mikrofone-Arrays zu versenden. So könnten Ton und Bild synchronisiert beim Anwender dargestellt werden und dieser würde eine vorhandene Latenz nicht bemerken.
Um eine universell einsetzbare Applikation zu erhalten, wäre eine Nutzeroberfläche vor der eigentlichen Applikation denkbar, in welcher ausgewählt werden kann, ob für das folgende Szenario eher Auflösung oder Verzögerung/FPS entscheidend sind. Somit könnte der Stream sogar individuell bestimmt werden.
Um für alle Versionen eine möglichst geringe Verzögerung zu erreichen, könnten die Daten weiter komprimiert werden. Hierfür wäre eine klassische Videokomprimierung möglich. Dabei besteht allerdings das Problem, dass nicht jedes mal die gleichen Punkte übertragen werden und somit eine Information mit gesendet werden müsste, welche Punkte aktualisiert werden würde. Diese zusätzlichen Daten müssten unterhalb der eingesparten Datenmenge liegen, um einen positiven Effekt zu erzielen.
Eine eigens entwickelte Idee der Komprimierung wäre es, die Farb- und Entfernungswerte zu kombinieren. Aktuell sind 255^3, also etwa 16,5 Mio. Farbwerte möglich. Eine solch hohe Anzahl an Farbwerten ist höchstwahrscheinlich nicht notwendig. Somit wäre es möglich, diese auf jeweils 5 statt bisher 8 Bit zu skalieren. Somit wären immer noch 32^3 Farbwerte möglich, was 32768 Farben entspricht. Eine solche Zahl an Farben sollte für die Betrachtung eines Hologramms genügen.

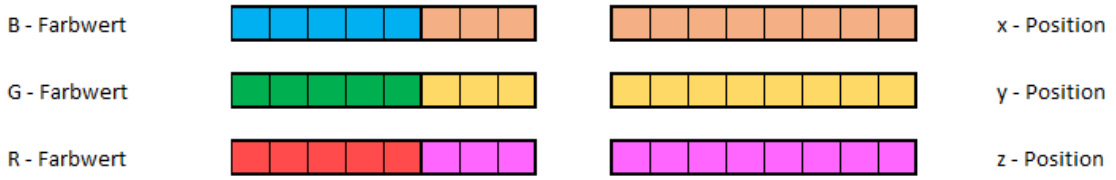

Abbildung 41: Mögliche Komprimierung der Daten

Durch die Skalierung der Farben wäre es möglich, jeweils ein Positionswert und einen Farbwert in zwei Byte zu speichern. Die Positionswerte benötigen nicht mehr als 10 Byte für x und y, da hier 1000 der größte Wert ist. Zusätzlich muss noch ein Bit für Plus und Minus eingeplant werden. Der z-Wert kann zwei bis 1500 betragen, benötigt aber kein Vorzeichen. Somit wären jeweils maximal 13 Byte belegt. Darauf würde sich der in Abbildung 41 dargestellte Aufbau eines Datenelements ergeben. Durch die Einsparung von 1/3 der Datenmenge könnte entweder eine noch geringere Latenz oder eine höherer Auflösung bei gleicher Latenz erreicht werden. Dies würde vermutlich zu einer noch besseren Nutzerfreundlichkeit führen.

Literaturverzeichnis

[A J17] M. Rao A. Jana M. Sharma. *HoloLens Blueprints: Build immersive AR and Mixed Reality Applications*. 1. Aufl. Packt Publishing, Juni 2017. ISBN: 978-1787281943.

[Abh17] Mallikarjuna Rao Abhijit Jana Manish Sharma. *HoloLens Blueprints: Build immersive AR and Mixed Reality Applications*. Packt Publishing, 2017.

[Alb+20] Justin Amadeus Albert u. a. „Evaluation of the Pose Tracking Performance of the Azure Kinect and Kinect v2 for Gait Analysis in Comparison with a Gold Standard: A Pilot Study“. In: *Sensors* 20.18 (2020). ISSN: 1424-8220. DOI: 10.3390/s20185104. URL: https://www.mdpi.com/1424-8220/20/18/5104.

[App] Apple. *Informationen zur fortschrittlichen Technologie von Face ID*. URL: https://support.apple.com/de-de/HT208108.

[Bas21] Matthias Bastian. *AR-Brille Nreal Light: Hersteller bietet PC-Verbindung an*. 26. Juli 2021. URL: https://mixed.de/ar-brille-nreal-light-hersteller-bietet-pc-verbindung-an/.

[Brü11] Christian Brümmer. *Avatar: Deppth-based Multi-Camera Motion Capturing*. Bachelorarbeit. 2011.

[DLR21] DLR. *Institut für Softwaretechnologie*. 31. Aug. 2021. URL: https://www.dlr.de/sc/desktopdefault.aspx/tabid-1185/1634_read-3062/.

[Doc] Microsoft Docs. *Hardwarespezifikationen für Azure Kinect DK*. URL: https://docs.microsoft.com/de-de/azure/kinect-dk/hardware-specification.

[Doc20a] Microsoft Docs. *Einführung in die Mixed Reality-Entwicklung*. 16. Dez. 2020. URL: https://docs.microsoft.com/de-de/windows/mixed-reality/develop/development?tabs=unity.

[Doc20b] Microsoft Docs. *HoloLens 2 Overview*. 2. Dez. 2020. URL: https://docs.microsoft.com/en-gb/hololens/hololens2-options?tabs=device.

[Doc20c] Microsoft Docs. *HoloLens 2-Hardware*. 2. Dez. 2020. URL: https://docs.microsoft.com/de-de/hololens/hololens2-hardware.

[Doc21] Microsoft Docs. *HoloLens 1 Overview*. 12. Jan. 2021. URL: https://docs.microsoft.com/de-de/hololens/hololens1-hardware.

[Edi] Natalie Ediger. *Star Wars und der Fluch des CGI*. URL: https://cleverclipstudios.com/de-ch/blog/star-wars-und-der-fluch-des-cgi/.

[eV21] Initistive D21 e.V. *D21-Digital-Index 2020/2021 – Jährliches Lagebild zur Digitalen Gesellschaft*. Studie. 2021. URL: https://initiatived21.de/app/uploads/2021/02/d21-digital-index-2020_2021.pdf.

[J B20] B. Rush J. Brown Kramer L. Sabalka. „Automated depth video monitoring for fall reduction: A case study“. In: *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2020).

[Jun] Hanseul Jun. *KinectToHoloLens*. URL: https://github.com/hanseuljun/kinect-to-hololens/tree/master/unity/KinectToHololens.

[Kra] Marco Kratzenberg. *Apples neue Animojis im iPhone X - so funktionieren sie*. URL: https://www.giga.de/smartphones/iphone-x/gallery/apples-neue-animojis-im-iphone-x-so-funktionieren-sie/.

[LS18] James R Lewis und Jeff Sauro. „Item benchmarks for the system usability scale.“ In: *Journal of Usability Studies* 13.3 (2018).

[Mica] Microsoft. *Azure-Kinect-Sensor-SDK*. URL: https://github.com/microsoft/Azure-Kinect-Sensor-SDK/blob/develop/docs/usage.md.

[Micb] Microsoft. *Microsoft.Azure.Kinect.Sensor.Short3 Struct Reference*. URL: https://microsoft.github.io/Azure-Kinect-Sensor-SDK/master/struct_microsoft_1_1_azure_1_1_kinect_1_1_sensor_1_1_short3.html.

[Mic20] Microsoft. *Azure Kinect DK*. 23. Juli 2020. URL: https://azure.microsoft.com/de-de/services/kinect-dk/.

[Muh+21] Azeem Syed Muhammad u. a. „A Suggestion-Based Interaction System for Spacecraft Design in Augmented Reality“. In: *2021 IEEE Aerospace Conference (50100)*. 2021, S. 1–10. DOI: 10.1109/AERO50100.2021.9438526.

[PLH18] Thammathip Piumsomboon, Gun A. Lee und Jonathon D. Hart. „MiniMe: An Adaptive Avatare for Mixed Reality Remote Collaboration“. In: *CHI* (2018).

[Tho] Natahn Thomas. *How To Use The System Usability Scale (SUS) To Evaluate The Usability Of Your Website*. URL: https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/.

[Ton14] HaiHong Gao Tong Jia ZhongXuan Zhou. „Depth Measurement Based on Infrared Coded Structured Light". In: *Journal of Sensors* 2014 (2014). ISSN: 1424-8220. DOI: 10.1155/2014/852621. URL: https://doi.org/10.1155/2014/852621.

[Tra] Advanced Realtime Tracking. *MOTION CAPTURE*. URL: https://ar-tracking.com/en/product-program/motion-capture.

[Unia] Unity. *Mesh.indexFormat*. URL: https://docs.unity3d.com/ScriptReference/Mesh-indexFormat.html.

[Unib] Unity. *TextureFormat.RGBA32*. URL: https://docs.unity3d.com/ScriptReference/TextureFormat.RGBA32.html.

[Uni21] Unity. *Startseite*. 4. Jan. 2021. URL: https://unity.com/de.

[Utz+19] Sebastian Utzig u. a. „Augmented Reality for Remote Collaboration in Aircraft Maintenance Tasks". In: *2019 IEEE Aerospace Conference*. 2019, S. 1–10. DOI: 10.1109/AERO.2019.8742228.

[Xia08] Marc Aurel Schnabel Xiangyu Wang. *Mixed Reality In Architecture, Design, And Construction*. Springer Science und Business Media, 2008.

[Yos] Takashi Yoshinaga. *Azure-Kinect-Unity-Sample*. URL: https://github.com/TakashiYoshinaga/Azure-Kinect-Sample-for-Unity.

[Zer] ZeroMQ. *ZeroMQ An open-source universal messaging library*. URL: https://zeromq.org/.

Anhang

A. Dokumente für Nutzerstudie

Evaluation of RGB-D-Videostreams in Augmented Reality (AR)

User Number : Age : Gender : Part: Scenario 1

		1	2	3	4	5	
1. Do you have experience with AR ?	much experience						no experience
2. Do you often work with AR ?	daily						never
3. Do you often use videostreams ?	daily						never
4. I needed to learn more before I could get going with the system.	Strongly disagree						Strongly agree
5. I found the stream very consistent.	Strongly disagree						Strongly agree
6. I thought there was too much delay between voice and picture.	Strongly disagree						Strongly agree
7. I could see all details clearly.	Strongly disagree						Strongly agree
8. I could follow all explanations very well.	Strongly disagree						Strongly agree
9. I think I could follow better without video.	Strongly disagree						Strongly agree
10. I think that I would like to use the system frequently.	Strongly disagree						Strongly agree
11. I felt very confident using the system.	Strongly disagree						Strongly agree
12. How insecure, discouraged, irritated, stressed and annoyed were you?	Very Low						Very High
13. How mentally demanding was it to follow the explanation?	Very Low						Very High
14. How hurried or rushed was the pace of the explanation?	Very Low						Very High

15. How much latency do you think was given ? (in milliseconds or seconds) _____

16. Would you want to have something described that way? ☐ yes ☐ no ☐ Don't know

Evaluation of RGB-D-Videostreams in Augmented Reality (AR)

User Number :

Part: Comparison

1. Which system would you prefer to use?

☐ first stream ☐ other

☐ second stream ☐ none

Please explain why:

2. What is more important for you?

high resolution ☐ ☐ ☐ ☐ ☐ no latency

Please explain why:

3. What do you think is more important for the system in general?

high resolution ☐ ☐ ☐ ☐ ☐ no latency

Please explain why:

4. Further comments: _____

_____

_____

_____

Evaluation of RGB-D-Videostreams in Augmented Reality (AR)

User Number : Age : Gender : Part: Scenario 2

		1	2	3	4	5	
1. Do you have experience with AR ?	much experience						no experience
2. Do you often work with AR ?	daily						never
3. Do you often use videostreams ?	daily						never
4. I needed to learn more before I could get going with the system.	Strongly disagree						Strongly agree
5. I found the stream very consistent.	Strongly disagree						Strongly agree
6. I thought there was too much delay between voice and picture.	Strongly disagree						Strongly agree
7. I could see all details clearly.	Strongly disagree						Strongly agree
8. I could follow all explanations very well.	Strongly disagree						Strongly agree
9. I think I could follow better without video.	Strongly disagree						Strongly agree
10. I think that I would like to use the system frequently.	Strongly disagree						Strongly agree
11. I felt very confident using the system.	Strongly disagree						Strongly agree
12. How insecure, discouraged, irritated, stressed and annoyed were you?	Very Low						Very High
13. How mentally demanding was it to follow the explanation?	Very Low						Very High
14. How hurried or rushed was the pace of the explanation?	Very Low						Very High

15. How much latency do you think was given ? (in milliseconds or seconds) _____

16. Would you want to have something described that way? ☐ yes ☐ no ☐ Don't know

User study consent form

Please read and sign this form.

In this user study:

- You will be asked to watch videostreams on HoloLens 2
- You will be asked to fill out some Questionnaires regarding the stream you watched.

Participation in this user study is voluntary. All information will remain strictly confidential. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time without giving reasons. The results from this user study will help 3D interaction design for streaming RGB-D-Videostreams in Augmented Reality.

If you have any questions later, please contact Anna-Lena Ehmer at Anna-Lena.Ehmer@dlr.de

I have read and understood the information on this form and had all of my questions answered.

_____ _____

Subject's Signature Date

B. Likert Skala Fragebogen

Standortfragebogen um den SUS-Wert zu bestimmen[57]: I think that I would like to use this system frequently.
I found the system unnecessarily complex.
I thought the system was easy to use.
I think that I would need the support of a technical person to be able to use this system.
I found the various functions in this system were well integrated.
I thought there was too much inconsistency in this system.
I would imagine that most people would learn to use this system very quickly.
I found the system very cumbersome to use.
I felt very confident using the system.
I needed to learn a lot of things before I could get going with this system.

[57] Tho.