HAW HAMBURG

Bachelorarbeit

Moritz Sundermann

Tiefenberechnung für Augmented Reality Anwendungen im Plattformvergleich mobiler Endgeräte

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Moritz Sundermann

Tiefenberechnung für Augmented Reality Anwendungen im Plattformvergleich mobiler Endgeräte

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Wirtschaftsinformatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Philipp Jenke
Zweitgutachter: Prof. Dr. Ulrike Steffens

Eingereicht am: 21. April 2021

Moritz Sundermann

Thema der Arbeit

Tiefenberechnung für Augmented Reality Anwendungen im Plattformvergleich mobiler Endgeräte

Stichworte

Mobile Augmented Reality, Tiefenberechnung, Tiefenkarte, Tracking, Objektverdeckung

Kurzzusammenfassung

In dieser Arbeit wird die Tiefenberechnung zweier mobiler Plattformen im Kontext von Augmented Reality Anwendungen betrachtet. Dazu werden mehrere Tests beschrieben, die verschiedene Funktionalitäten der Plattformen überprüfen, welche akkurate Tiefenberechnungen erfordern, wie das Erstellen einer Tiefenkarte, die Vermessung eines Objekts oder Objektverdeckung. Abschließend wird ein Vergleich der Testergebnisse vorgenommen.

Moritz Sundermann

Title of Thesis

Depth calculation for augmented reality applications in a platform comparison of mobile devices

Keywords

Mobile Augmented Reality, Depth calculation, Depth map, Tracking, Occlusion

Abstract

In this thesis the depth calculations in an augmented reality context of two mobile devices are being examined. For that purpose multiple tests will be described, that examine different functionalities, which require accurate depth calculations, like the creation of a depth map, taking an objects measurements or object occlusion. Finally the test results of the two devices will be compared to each other.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Augmented Reality bietet eine Vielzahl an Möglichkeiten, die Realität, wie sie uns bekannt ist, um virtuelle Elemente zu erweitern und auf diese Art und Weise sehr hilfreich zu sein. Hierbei kann es sich um marginale Hilfestellungen handeln, wie die virtuelle Abseitslinie im Fussball für die Zuschauer, aber auch um sehr praktische Unterstützungen im Alltag, wie die Linien von der optischen Einparkhilfe einer Rückfahrkamera. Seit mehreren Jahren halten Augmented Reality Anwendungen nun auch auf mobilen Endgeräten Einzug. Mobile Endgeräte bieten sich für Augmented Reality Anwendungen an, da sie in den meisten Fällen eine recht hochauflösende Kamera eingebaut haben deren Bilder man live mit virtuellen Elementen anreichern kann. Möchte man beispielsweise ein neues Möbelstück kaufen, könnte eine Augmented Reality App bei der Auswahl helfen, indem sie live eine virtuelle Version des Möbelstücks als Vorschau in die Kamerabilder vom entsprechenden Raum einsetzt. Diese virtuelle Version des Möbelstücks könnte man sogar aus verschiedenen Winkeln begutachten, indem man sich mit seinem mobilen Endgerät durch den Raum bewegt.

1.1 Motivation

Bei mobilen Augmented Reality Anwendungen wird innerhalb vom Bruchteil einer Sekunde das Bild der eingebauten Kamera registriert, in Zusammenhang mit den vorausgegangenen Bildern ausgewertet, für den Nutzer basierend auf der Auswertung bearbeitet und auf dem Display des Geräts dargestellt. Dieser Vorgang wird für jedes einzelne Bild wiederholt. Der wohl anspruchsvollste Teil für das Endgerät ist hierbei die Auswertung der Bilder, da mehrere Bilder miteinander verglichen werden, um so die Distanzen zwischen der Kamera und den Objekten auf den Bildern einzuschätzen. Wenn eine solche Aufgabe innerhalb von dermaßen kurzer Zeit mit der vergleichsweise eingeschränkten Rechenleistung eines mobilen Endgeräts ausgeführt wird, stellt sich automatisch die Frage nach der Qualität des Ergebnisses.

1.2 Zielsetzung

In dieser Arbeit soll die Tiefenberechnung für Augmented Reality Anwendungen zweier verschiedener mobiler Plattformen bewertet und verglichen werden. Bei den Repräsentanten handelt es sich um ein Apple iPhone 11 und ein Google Pixel 4. Sie bieten sich für diese Tests an, da es sich um weit verbreitete Modelle handelt und hier insbesondere die Leistung von recht aktuellen Geräten bewertet werden soll, die im Alltag vorkommen. Die Bewertung wird basierend auf vier Tests erfolgen, die darauf ausgelegt sind, die Kernfunktionen zu überprüfen, welche in der Praxis am ehesten Anwendung finden.
Die erste überprüfte Funktion ist die Fähigkeit vollständige Tiefendaten einer Szene zu sammeln und darzustellen. Diese Funktion ist essentiell, da Tiefendaten die Grundlage für die meisten Augmented Reality Anwendungen bilden. Daraufhin wird die Funktion getestet, präzise Distanzen einzuschätzen. Dies kann ebenfalls für viele Augmented Reality Anwendungen entscheidend sein, da virtuelle Objekte nur durch präzise Distanzeinschätzung maßstabsgetreu und an der richtigen Stelle eingesetzt werden können. Als drittes wird die Funktion der Objektverdeckung getestet, also die Fähigkeit einer Augmented Reality Anwendung virtuelle Objekte teilweise oder ganz hinter realen Objekten verschwinden zu lassen. Diese Funktion findet zwar nicht ganz so häufig Anwendung wie die ersten beiden Funktionen, aber da sie auf sehr präzise Tiefenberechnung angewiesen ist, eignet sie sich gut um die Limits der mobilen Plattformen in diesem Bereich zu testen. Zuletzt wird noch die Funktion getestet Gesichter zu erkennen und zu verfolgen. Insbesondere unter Nutzung der Frontkamera ist dies ein sehr häufig auftretender Anwendungsfall, wenn Leute ihr Gesicht mithilfe von Augmented Reality Anwendungen um virtuelle Objekte erweitern.

1.3 Gliederung

Zunächst werden die Grundlagen erläutert, die für die Umsetzung einer Augmented Reality Anwendung und insbesondere für die folgenden Tests benötigt werden. Hierzu gehören sowohl die Prinzipien, nach denen die Testgeräte arbeiten, während sie Tiefenberechnungen anstellen, als auch die Arten, in denen sie die Ergebnisse liefern und darstellen. Daraufhin werden die geplanten Tests genauer erklärt, bevor dann die Entwicklung der Test Apps sowie die Ergebnisse der Tests näher beleuchtet werden. Abschliessend werden die Ergebnisse miteinander verglichen und bewertet, bevor ein kleiner Ausblick auf die Zukunft des Themas gegeben wird.

2 Stand der Technik

2020 präsentierten Oufqir et al. bei der 2020 International Conference on Intelligent Systems and Computer Vision die Ergebnisse ihrer Untersuchung, bei der sie die verfügbaren Augmented Reality Bibliotheken von Apple (ARKit) und Google (ARCore) näher betrachteten[16]. Dabei benutzten sie als Testgeräte ein iPhone 8 und ein Samsung Galaxy S8; beides Geräte, die bereits 2017 erschienen. Dies wirft zwar die Frage auf, ob die technischen Entwicklungen seit 2017 die Ergebnisse beeinflusst hätten, aber nichtsdestotrotz ist der Vergleich der Software fair, da die Geräte etwa zum selben Zeitpunkt erschienen. Es wird insbesondere verglichen, welche Möglichkeiten Entwicklern bereitgestellt werden. Sie kommen zwar zu dem Ergebnis, dass es Unterschiede in den bereitgestellten Funktionalitäten gibt, die relevant sein können, wenn man sich als Entwickler zwischen den beiden Bibliotheken entscheiden muss, jedoch wird keine Bewertung abgegeben, was die Qualität der Funktionalitäten angeht. Also wird betrachtet, was für Möglichkeiten Entwicklern bei der Nutzung der Bibliotheken zur Verfügung stehen, aber nicht welche Bibliothek bzw. welches Endgerät jene Funktionalitäten besser umsetzt, die von beiden bereitgestellt werden. Hier befindet sich der Ansatzpunkt dieser Arbeit: die Überprüfung, welches Endgerät unter Nutzung der jeweils zugehörigen Bibliothek die Grundfunktionen von Augmented Reality Anwendungen besser umsetzt.
Die nächste Arbeit von Qiao et al. stammt aus dem Jahr 2019 und befasst sich mit den Möglichkeiten, die webbasierte Augemented Reality der mobilen Augmented Reality in Zukunft eröffnen könnte[17]. Webbasierte Augmented Reality steht bei dieser Arbeit zwar nicht im Fokus, allerdings gehen die Autoren zu Beginn der Arbeit auf den Stand der Technik der mobilen Augmented Reality ein. In diesem Rahmen werden typische aktuelle Prinzipien der mobilen Augmented Reality beschrieben, auf denen Augmented Reality Anwendungen, wie die im Rahmen dieser Arbeit entwickelten Anwendungen, aufbauen können.
In der letzten Arbeit von Goh et al. aus dem Jahr 2019 geht es um die Interaktion mit virtuellen 3D-Objekten im Kontext der mobilen Augmented Reality[7]. Hierbei werden mehrere Möglichkeiten der Interaktion beschrieben und zum Ende der Arbeit wird auf

Herausforderungen und Schwierigkeiten eingegangen. Für diese Arbeit besonders interessant ist dabei die Beschreibung der Herausforderungen für mobile Augmented Reality. Objektverdeckung wird beispielsweise als eine Herausforderungen benannt, da großer Rechenaufwand betrieben werden muss, um sie akkurat durchzuführen. Solche Herausforderungen eignen sich gut als Ansatzpunkt für diese Arbeit, da durch sie die Grenzen mobiler Augmented Reality Systeme getestet werden können.

3 Grundlagen

3.1 Augmented Reality

Um Augmented Reality konzeptuell zu beschreiben, ordnet man sie im Realitäts-Virtualitäts-Kontinuum ein[14]. Das Realitäts-Virtualitäts-Kontinuum ist eine Skala, von einer unveränderten realen Umgebung bis zu einer rein virtuellen Umgebung, auf der man verschiedene Formen der Mixed Reality einsortieren kann. Auf dieser Skala liegt AR näher an der Realität als an der Virtualität, da eine reale Umgebung um virtuelle Objekte erweitert wird. Sie steht im Gegensatz zur Augmented Virtuality, bei der eine virtuelle Umgebung mit realen Objekten erweitert wird.

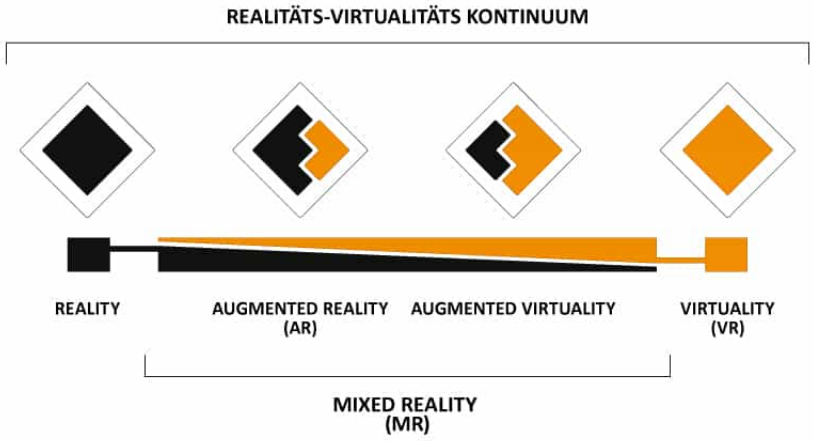

Abbildung 3.1: Das Realitäts-Virtualitäts-Kontinuum von Realität bis Virtualität [1]

Um eine Augmented Reality Anwendung erfolgreich zu gestalten, werden im Allgemeinen drei Grundfunktionen benötigt: Darstellung, Tracking und Interaktion[20]. Für Tiefenberechnungen ist allerdings insbesondere die Funktion des Trackings entscheidend, weshalb im Folgenden vor allem auf die Funktionsweisen von verschiedenen Trackingverfahren ein-

[1] Quelle: https://www.augmented-minds.com/de/erweiterte-realitaet/was-ist-augmented-reality, Zugriff: 2.12.2020

gegangen wird. Außerdem werden Methoden erläutert, wie die Ergebnisse des Trackings verwertet werden können.

3.2 Tracking

Tracking ist die Fähigkeit eines Systems, die Position und Orientierung von sich selbst und anderen Objekten in Relation zueinander zu bestimmen und über mehrere Eingaben hinweg zu verfolgen. Es gibt verschiedene Möglichkeiten Tracking umzusetzen, doch im Kontext der mobilen Augmented Reality sind insbesondere optisches Tracking und Inertial-Tracking, zu betrachten. Bei Inertial-Tracking werden Bewegungsdaten genutzt, um die eigene Position und Orientierung zu bestimmen, während beim optischen Tracking Bilder genutzt werden, um die eigene Position in der betrachteten Szene sowie Abstände von betrachteten Objekten einzuschätzen[10]. Innerhalb des optischen Tracking gibt es wiederum mehrere Verfahren, die sich grob in markerbasierte und markerlose Verfahren unterteilen lassen. Ein Marker bezeichnet hierbei ein bereits bekanntes und einfach erkennbares Objekt, an dem sich das System orientieren kann[4]. Da unsere Testgeräte ohne Marker funktionieren, stehen im Folgenden markerlose Verfahren im Fokus.

3.2.1 Merkmalsbasiertes Tracking

Beim merkmalsbasierten Tracking benötigt das System im Voraus keinerlei Informationen über seine Umgebung, sondern vergleicht Bild für Bild die Position von Merkmalen der betrachteten Szene. Dafür wird zunächst ein Bild aufgenommen. Auf dem Bild werden für das System herausstechende Merkmale gespeichert. Merkmale können hierbei Punkte, Linien, Kreise oder viele weitere Dinge sein - abhängig vom genutzten Algorithmus für die Merkmalserkennung. In der mobilen Augmented Reality sind allerdings Punkte als Merkmale beliebt, da sie einfach zu verarbeiten sind und aus verschiedenen Blickwinkeln kaum verzerrt werden[21]. Nachdem die Merkmale gespeichert wurden, wird ein weiteres Bild von der Szene (idealerweise aus einer anderen Position) gemacht und die gespeicherten Merkmale werden erneut gesucht. Basierend auf der Veränderung verschiedener Faktoren, wie beispielsweise der Größe der Merkmale oder dem Abstand zwischen mehreren Merkmalen, wird dann die Position und Orientierung, also die Pose, von den Objekten in der Szene sowie der Kamera selbst zu einander berechnet und gespeichert.

Die Ergebnisse werden dann weiter verfeinert und präzisiert, indem der Vorgang aus verschiedenen Positionen und mit weiteren Merkmalen wiederholt wird[13]. Man kann die Ergebnisse des Trackings daraufhin auf verschiedene Arten speichern. Ein paar dieser Arten werden später näher betrachtet. Eine Herausforderung für diese Art des Trackings stellen bewegliche Objekte dar, da das System die Szene als Abbildung statischer Objekte betrachtet. Daher erschweren Bewegungen der Merkmale es, präzise Einschätzungen der Distanzen und Orientierungen zu liefern[5]. Ein Ansatz um trotz solcher Erschwerungen gute Ergebnisse zu erzielen wird in 3.2.3 Hybrid-Tracking noch beschrieben.

3.2.2 Inertial-Tracking

Inertial-Tracking wird vor allem eingesetzt, um die eigene Position und Orientierung des Systems zu bestimmen. Hierfür werden mehrere sogenannte Trägheits- oder Inertialsensoren verwendet. Diese bestehen normalerweise aus Gyroskopen, die Drehungen des Systems wahrnehmen, und Beschleunigungsmessern, die Bewegungen im Raum messen[22]. Die Messungen entsprechen also der Veränderung in Position und Orientierung, beziehungsweise der Pose des Systems. Auf diese Weise kann das System seine Pose in Abhängigkeit zur eigenen Anfangspose berechnen, indem die gemessenen Veränderungen zur vorherigen Pose hinzugerechnet werden. Beim Inertial-Tracking können allerdings Ungenauigkeiten vorkommen, da plötzliche Bewegungen der Trägheitssensoren beim Tracking zu Drifteffekten führen können, welche die Genauigkeit der Berechnungen negativ beeinflussen[4].

3.2.3 Hybrid-Tracking

Wie zuvor beschrieben, kann es bei einzelnen Tracking Verfahren hin und wieder zu Situationen kommen, in denen die genutzte Methode nicht ideal funktionieren kann. Aufgrund dessen setzen Augmented Reality Systeme beim Tracking häufig auf mehrere Verfahren, die sich gegenseitig unterstützen, sollte es zu Ungenauigkeiten kommen. Verliert ein hybrides Trackingsystem beispielsweise seine Orientierungspunkte für das merkmalsbasierte Tracking aus den Augen, kann es trotzdem seine eigene Position und Orientierung bestimmen, indem es auf die Daten eines Inertial Sensors zugreift[4]. Eine solche Zusammenarbeit von Sensoren nennt man Sensor Fusion. Sensor Fusion beschreibt also die Kombination der Daten mehrerer Sensoren, um so die Schlussfolgerungen zu verbessern, die aufgrund der Daten getroffen werden. Das heißt, die Genauigkeit und Verlässligkeit

der Daten soll verbessert werden, damit die Ergebnisse der darauf basierenden Algorithmen nicht durch falsche oder ungenaue Daten beeinträchtigt werden[1].

3.2.4 Simultaneous Localization and Mapping

Simultaenous Localization and Mapping (SLAM) ist nicht direkt eine Form des Trackings, kann allerdings im Zusammenspiel mit Tracking für Augmented Reality Anwendungen eingesetzt werden. SLAM beschreibt Algorithmen, deren Ziel es ist, ähnlich wie Tracking, mithilfe eines oder mehrerer Sensoren eine Karte der Umgebung zu erstellen und sich selbst sowie betrachtete Objekte auf dieser Karte zu positionieren.[6]. Der Begriff kommt ursprünglich aus dem Bereich der Robotik, wo es eingesetzt wird, um Robotern, die sich automatisch steuern, dabei zu helfen, sich in einer unbekannten Umgebung zu orientieren. Die genaue Funktionsweise verschiedener Implementationen kann dabei je nach Anwendungsfall variieren, aber der Grundgedanke ist, mithilfe der verfügbaren Sensoren die Distanzen von Objekten um den Roboter herum zunächst einzuschätzen. Wenn er sich anschließend bewegt, wird die Einschätzung wiederholt. Aufgrund der Veränderung der Distanzen sowie der geschätzten zurückgelegten Distanz wird eine Karte erstellt und die Position des Roboters auf dieser Karte geschätzt[12]. SLAM-Algorithmen können für mobile Augmented Reality Systeme von großem Nutzen sein, da sie bis auf die Daten der Sensoren keinerlei Informationen über ihre Umgebung benötigen[4].

3.3 Nahinfrarot Kamera

Nahinfrarot (NIR) Kameras funktionieren im Grunde genommen wie normale Kameras. Der einzige Unterschied besteht in der Art des Lichts, das fotographiert wird, denn normale Kameras zeichnen Licht auf, das auch für Menschen sichtbar ist, mit Wellenlängen von 380 nm bis 750 nm, während NIR Kameras, wie der Name schon sagt, NIR-Licht aufzeichnen. Dieses hat Wellenlängen im Bereich von 750 nm bis 1400 nm[23]. Der Vorteil von NIR Kameras besteht darin, dass man eine Szene mit NIR-Licht beleuchten kann, ohne dass Nutzer es wahrnehmen. Basierend auf dem resultierenden Bild der NIR-Kamera kann man dann erkennen, wie lange das NIR-Licht an bestimmten Stellen von der Quelle bis zum Empfänger gebraucht hat. Daraus kann man wiederum direkt ableiten, wie weit die Objekte auf dem Bild entfernt sind. Also können Bilder von NIR-Kameras, sofern man die betrachtete Szene zuvor mit NIR-Licht beleuchtet hat, als Tiefenkarten fungieren[9].

Tiefenkarten werden im folgenden Abschnitt auch noch näher beschrieben.
NIR Kameras gehören, so eingesetzt, zu den aktiven Tiefensensoren. 'Aktiv', weil die Tiefendaten direkt aus der Wahrnehmung des eigens ausgesendeten NIR-Lichts hervorgehen. Alternativ gibt es auch passive Tiefensensoren, wie beispielsweise die normale Kamera. Diese nennt man passive Tiefensensoren, da ihre primäre Ausgabe normale Bilder sind, aus denen man allerdings durch weitere Schritte, wie merkmalsbasiertes Tracking, Tiefendaten gewinnen kann[11].

3.4 Tiefenkarten

Nachdem die Distanz zwischen dem System und Objekten in der betrachteten Szene erfasst wurde, gilt es die Daten auf sinnvolle Art und Weise zu speichern. Dafür werden unter anderem Tiefenkarten, auch Tiefenbilder genannt, verwendet. Angenommen, man hat Tiefeninformationen zu einer Szene gesammelt und diese in einem Raster vorliegen. Dann wird eine Tiefenkarte erstellt, indem so vielen Stellen der Szene wie möglich ihre Tiefe zugeordnet wird[11]. Aus dieser Zuordnung wird dann ein Bild erstellt, bei dem das Aussehen jedes Pixels von der Tiefe der repräsentierten Stelle abhängt. Beispielsweise können für die einzelnen Pixel verschiedene Grautöne verwendet werden, wobei hellere Töne näher und dunklere Töne weiter entfernt sind. Daraus resultierend erhält man ein Bild das vergleichbar mit einem normalen Bild der Szene ist. Aber während das normale Bild die realen Farben der Szene zeigt, repräsentiert das Tiefenbild die Entfernungen zur Kamera[18].

Abbildung 3.2: Eine Tiefenkarte (rechts) neben dem Original (links) wobei die Dunkelheit eines Pixels seine Tiefe repräsentiert[2]

3.5 3-dimensionale Koordinatensysteme

Eine weitere Art die Tiefendaten einer Szene zu speichern, ist ein 3-dimensionales Koordinatensystem für die Szene zu erstellen, bei dem sich das Gerät selbst typischerweise am Ursprung, also am Punkt 0,0,0 befindet. Die Positionen von anderen Objekten und Merkmalen werden dann innerhalb dieses Koordinatensystems gespeichert. Diese Art, eine Szene zu speichern, hat den Vorteil, dass man virtuellen Objekten jederzeit einfach einen Punkt im Koordinatensystem als Position zuweisen kann, wenn man sie an einer bestimmten Stelle einfügen will[15]. Ein weiterer Vorteil dieser Methode ist, dass sie gut für Distanzberechnung geeignet ist, da man die Distanz zwischen verschiedenen Punkten im Koordinatensystem ausrechnen kann, indem man die Länge des Vektors, der die Punkte verbindet, mit folgender Formel berechnet: $\left|\vec{V}\right| = \sqrt{x^2 + y^2 + z^2}$, wobei x, y und z die jeweiligen Koordinaten des Vektors sind[3]. 3-dimensionale Koordinatensysteme bieten natürlich noch weitere Rechenfunktionen, von denen manche auch in einem Augmented Reality Kontext nützlich sein können. An dieser Stelle soll aber nicht weiter auf sie eingegangen werden, da sie für die weitere Bearbeitung des Themas nicht relevant sind.

[2] Quelle: https://triaxes.com/articles/manual-depth-map-creation/, Zugriff: 19.12.2020

3.6 Dreiecksnetze

Dreiecksnetze werden in der Computergrafik eingesetzt, um die Form von 3D-Objekten sowie Flächen zu speichern und darzustellen, was sie für Augmented Reality sehr nützlich macht. Ein Dreiecksnetz besteht aus Vertices, Kanten und Facetten. Hierbei ist ein Vertex ein Punkt im 3-dimensionalen Raum, dessen Koordinaten gespeichert werden. Eine Kante ist eine Seite eines Dreieck und besteht aus zwei Vertices, die von der Kante verbunden werden. Eine Facette besteht aus drei Vertices, die wiederum durch drei Kanten zu dem Dreieck verbunden werden. Kanten und Vertices können dabei zu mehreren Facetten gehören, sodass aus mehreren Facetten, die Kanten und Vertices teilen, ein Dreiecksnetz wird[15]. Je mehr Facetten man für die Darstellung eines Objekts benutzt, desto detaillierter wird die Darstellung. Allerdings müssen dann natürlich auch mehr Vertices, Kanten und Facetten gespeichert und verarbeitet werden. Dies kann bei begrenzter Rechenleistung und einer hohen Anzahl an Facetten zu Problemen führen[19].

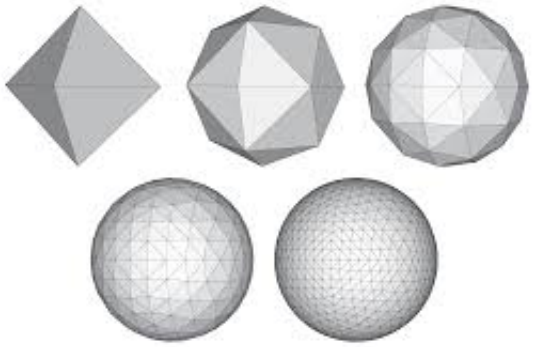

Abbildung 3.3: Mehrere Dreiecksnetze mit zunehmender Anzahl an Facetten[3]

[3]Quelle: https://kronuz.io/Xapiand/docs/reference-guide/schemas/field-types/geospatial-type/htm/ Zugriff: 21.12.2020

4 Konzept

Im Folgenden wird beschrieben, welche Tests mithilfe der Testgeräte durchgeführt werden sollen, was die Anforderungen an die Apps sind, die die Tests durchführen und die Herangehensweise an die Entwicklung dieser Apps.
Die geplanten Tests sind Folgende:

4.1 Tests

Mithilfe von jeweils einer App pro Testgerät soll eine Testreihe durchgeführt werden, bei der sowohl die Rück- als auch die Frontkamera der beiden Geräte getestet werden. Der primäre Fokus der Testreihe liegt darauf, die Präzision der Tiefeneinschätzung der Geräte unter Nutzung der jeweiligen Kamera zu testen. Darüber hinaus sollten die Tiefendaten idealerweise auf eine Art und Weise zugänglich sein, die es Entwicklern erlaubt direkt mit den Daten weiterzuarbeiten.

1. Tiefenbilder: Dieser Test soll damit beginnen, dass mithilfe der vier Kameras jeweils, soweit möglich, eine Tiefenbild der selben Szene erstellt und als Bild gespeichert wird. Die verschiedenen Tiefenbilder sollen daraufhin miteinander und mit einem normalen Bild der Szene verglichen werden, um so die Qualität der Tiefenbilder zu bewerten.
2. Distanzmessung: In diesem Test soll mithilfe der Rückkameras die Länge eines Objekts aus verschiedenen Entfernungen eingeschätzt werden. Aus sehr kurzer Distanz, aus kurzer Distanz, aus mittlererer Distanz und letztlich aus großer Distanz. Die Bewertung der Testergebnisse basiert zum einen darauf, ob die Messung vom System mit den realen Maßen übereinstimmt, und zum anderen, ob sich die Messergebnisse bei verschiedenen Entferungen signifikant verändern.

3. Objektverdeckung: Als nächster Test soll mithilfe der Rückkameras ein virtuelles Objekt in die Szene eingesetzt werden, bevor ein reales Objekt hinzugefügt wird, und zwar in einer Position, in der es das virtuelle Objekt teilweise verdecken sollte. Auch dieser Test wird wieder mit verschiedenen Distanzen zum Objekt durchgeführt. Die Bewertung der Ergebnisse folgt daraus, wie akkurat das virtuelle Objekt verdeckt wird.

4. Gesichtserkennung: Abschließend erfolgt noch ein Test für die Frontkameras. Da die Funktionalitäten dieser Kameras vor allem auf Gesichtserkennung ausgerichtet sind, sollen für diesen Test, sobald ein Gesicht erkannt wird, alle Punkte, die das System im Gesicht trackt, live visualisiert werden, sodass man die Präzision der Gesichtserkennung insbesondere bei Bewegung erkennen kann.

4.2 Anforderungen

Primäre Bedeutung hat bei den zu entwickelnden Apps, dass diese präzise Testergebnisse ausgeben. Sekundärer Fokus liegt darauf, dass die Apps intuitiv zu benutzen sind, für den zugegebenermaßen unwahrscheinlichen Fall, dass die Apps außerhalb dieser Arbeit zum Einsatz kommen. Es sollte eine Möglichkeit geben, mittels einzelner Knopfdrücke zwischen den verschiedenen Testmodi zu wechseln und eine kleine Erläuterung zum jeweiligen Modus zu erhalten. Zudem wird eine Möglichkeit benötigt, ebenfalls mithilfe eines einzelnen Knopfdrucks, zwischen den beiden Kameras eines Geräts zu wechseln. In jedem der Testmodi wird zunächst das Bild der Kamera live angezeigt, damit diese richtig auf die zu betrachtende Szene ausgerichtet werden kann. Im ersten Testmodus wird darüber hinaus, unabhängig von der derzeit genutzten Kamera, lediglich ein Auslöser benötigt, um ein Tiefenbild der Szene zu speichern. Im zweiten Testmodus für die Rückkamera benötigt man eine Möglichkeit, zwei Punkte der Szene zu markieren und sich ihre Distanz von einander ausgeben zu lassen, um so die Länge von Objekten zu messen. Im dritten Testmodus der Rückkamera benötigt man die Möglichkeit, ein virtuelles Objekt in die Szene einzusetzen, dessen Distanz zur Kamera ausgegeben wird und idealerweise über einen Schieberegler oder etwas ähnliches variiert werden kann. Abschließend benötigt man noch einen Modus für die Frontkamera, in dem automatisch die getrackten Punkte von erkannten Gesichtern visualisiert werden.

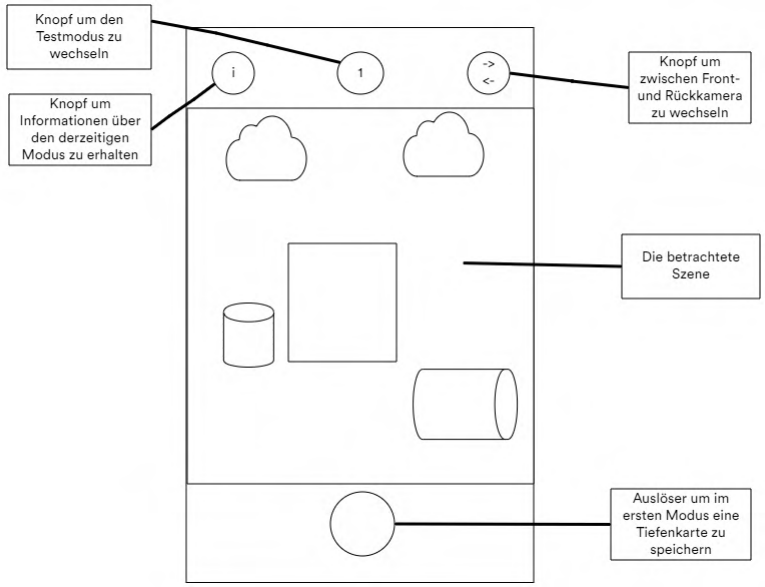

Abbildung 4.1: Mockup - Entwurf der App

4.3 Herangehensweise

Für mein Vorgehen an dieses Softwareprojekt habe ich eine Methode gewählt, die auf Eckhart Hansers [8] Beschreibung des Scrum-Prozesses basiert. Hierfür beginnt man mit einer Beschreibung der Vision des Kunden für das fertige Produkt. Da es bei dieser Arbeit keinen Kunden gibt, entspricht die vorab in Abschnitt 4.2 erfolgte Beschreibung der Anforderungen an die fertige Software diesem Schritt. Die Entwicklung wird daraufhin in mehrere sogenannte Sprints eingeteilt, die alle gleichlang sind und jeweils zu einem Meilenstein in der Entwicklung führen. Zunächst einmal muss im Setting für diese Arbeit die Unterteilung stattfinden, dass zwei Apps mit den selben Anforderungen in zwei verschiedenen Programmiersprachen entwickelt werden. Für die Entwicklung jeder dieser Apps sind jeweils vier Wochen eingeplant. Diese vier Wochen werden in vier einwöchige Sprints eingeteilt. Während des ersten Sprints sollen das User Interface erschaffen werden, das zu den verschiedenen Tests führt, und der erste Testmodus implementiert werden. Die folgenden drei Sprints werden jeweils für die Implementation einer der übrigen drei Testfunktionen genutzt, sodass man am Ende der vier Sprints alle vier Tests mithilfe der App durchführen kann. Normalerweise gibt es während der Sprints sogenannte Daily Scrums: tägliche Meetings bei denen der bisherige Fortschritt, mögliche Probleme usw. besprochen werden. Da ich alleine arbeite, entfallen diese Meetings. Aber indem ich am

Ende eines Arbeitstages meine Gedanken zum bisherigen Fortschritt und zu Problemen notiere, wird es beim nächsten Mal durch das Lesen der Notizen vom Vortag einfacher, an der selben Stelle weiterzuarbeiten.

5 Implementierung

5.1 Technische Grundlagen

In diesem Abschnitt wird die Technik beschrieben, die von den Testgeräten für Augmented Reality Anwendungen bereitgestellt wird. Das bedeutet, hardwaretechnisch, welche Sensoren insbesondere für Augmented Reality Anwendungen verwendet werden. Was die Software angeht, werden sowohl angewendete Methoden des Trackings beschrieben, als auch die Art und Weise, wie die daraus resultierenden Ergebnisse gespeichert und dem Nutzer präsentiert werden.

5.1.1 Hardware

Google Pixel 4

Das Google Pixel 4 hat zwei eingebaute Rückkameras. Eine 16 Megapixel Kamera und eine 12,2 Megapixel Kamera. Die Frontkamera ist eine 8 Megapixel Kamera, die zudem von einem NIR-Emitter, einem NIR-Punktprojektor und zwei NIR-Kameras unterstützt wird. Darüber hinaus gibt es einen Beschleunigungsmesser, auch Gyrometer oder Inertialsensor gennant.[1] Es gibt noch weitere Sensoren und technische Eckdaten, doch diese sind für Augmented Reality Anwendungen nicht relevant.

Apple iPhone 11

Die eingebaute Rückkamera des iPhones ist eine 12 Megapixel Zweifach-Kamera mit Ultraweitwinkel- und Weitwinkelobjektiv. Die Frontkamera ist eine 12 Megapixel TrueDepth Kamera.[2] Auch wenn in den technischen Daten zu dem Gerät eine Infrarot-Kamera

[1] Quelle: https://store.google.com/de/product/pixel_4_specs Zugriff: 10.12.2020
[2] Quelle: https://www.apple.com/de/iphone-11/specs/ Zugriff: 10.12.2020

nicht explizit genannt wird, kann man schlussfolgern, dass eine solche in der TrueDepth Kamera enthalten ist, da in der Erklärung der Funktionsweise vom Face ID Feature beschrieben wird, dass die TrueDepth Kamera ein Infrarotbild aufzeichnet. Das iPhone hat ebenfalls einen Beschleunigungssensor und darüber hinaus einen weiteren 3-Achsen Gyrosensor, der bei der Bestimmung der Lage des Geräts hilft.[3] Auch hier gibt es weitere Sensoren und technische Daten, die aber für Augmented Reality Anwendungen nicht relevant sind.

5.1.2 Software

ARCore

Auf dem Google Pixel 4 wird in dieser Testreihe die Version 1.21.0 des Frameworks ARCore verwendet. Dieses Augmented Reality Framework wird von Google selbst für Android Systeme angeboten und ist besonders interessant, da es die Depth API für Tiefenberechnung beinhaltet. Die Depth API benutzt Motion Tracking, eine Form von Hybrid-Tracking, welches merkmalsbasiertes sowie Inertial-Tracking mit einem SLAM Algorithmus kombiniert. Dies eignet sich am besten zum Testen von Augemented Reality Anwendungen, welche die Rückkamera verwenden.[4] Die Ergebnisse des Trackings werden dann wahlweise als Tiefenkarte zurückgegeben, auf der Pixel, je nach Entfernung, rot (nah), orange, gelb, grün und blau (fern) dargestellt werden. Gleichzeitig wird ein 3-dimensionales Koordinatensystem mit der Einheit Meter für die Szene erstellt, um die Position vom Gerät selbst sowie von erkannten Flächen und Objekten zu verwalten.[5] Zum Testen von Anwendungen, die die Frontkamera nutzen, eignet sich die Klasse AugmentedFace innerhalb von ARCore am besten als Basis, da diese mithilfe der Frontkamera Tiefendaten sammelt und neben dem 3-dimensionalen Koordinatensystem auch noch Dreiecksnetze erstellt, welche gefundene Gesichter repräsentieren.[6]

[3] Quelle: https://support.apple.com/de-de/HT208108 Zugriff: 10.12.2020
[4] Quelle: https://developers.google.com/ar/discover/concepts Zugriff: 9.12.2020
[5] Quelle: https://developers.google.com/ar/develop/java/depth/overview, Zugriff: 9.12.2020
[6] Quelle: https://developers.google.com/ar/reference/java/com/google/ar/core/AugmentedFace Zugriff: 10.12.2020

ARKit

Auf dem iPhone 11 wird für die Testreihe das Augmented Reality Framework ARKit 4 verwendet, das von Apple für iOS Systeme angeboten wird. ARKit bietet unterschiedliche Tracking Methoden für verschiedene Kontexte an.[7] Die variabelste Methode ist das World Tracking, welches "visual-inertial odometry" benutzt. Das bedeutet, dass auch hier eine hybride Methode bestehend aus merkmalsbasiertem und Inertial-Tracking eingesetzt wird. Aus den Ergebnissen des Trackings wird dann ein 3-dimensionales Koordinatensystem mit der Einheit Meter erstellt, in dem die Position vom Gerät selbst sowie erkannten Flächen und Objekten eingeordnet werden. World Tracking ist allerdings speziell auf die Rückkamera zugeschnitten.[8]
Zum Testen der Frontkamera bietet sich Face Tracking an, da dieses mithilfe der Infrarot-Technik der TrueDepth Kamera Tiefendaten sammelt und nach Gesichtern sucht, von denen Modelle in Form von Dreiecksnetzen erstellt werden. Auch hier wird neben den Dreiecksnetzen wieder ein 3-dimensionales Koordinatensystem erstellt, um wichtige Punkte zu speichern.[9]

5.2 Durchführung

Aufgrund der unterschiedlichen Betriebssysteme mussten die beiden Apps in zwei verschiedenen Programmiersprachen geschrieben werden. Die Android-App wurde in Java geschrieben und die iOS-App in Swift. Trotzdem sollten die Funktionalitäten und Strukturen möglichst gleich sein. Im Folgenden werden die verschiedenen Funktionalitäten durchgegangen und für die jeweilige Programmiersprache bzw. das jeweilige Framework erläutert, wie die Funktionalität umgesetzt wurde.

5.2.1 Renderer

Die Grundlage für jeden Testmodus besteht aus einem Renderer, der das Kamerabild live verarbeitet und bereit ist, virtuelle Elemente einzubauen oder Tiefendaten zu visualisieren.

[7] Quelle: https://developer.apple.com/documentation/arkit Zugriff: 10.12.2020
[8] Quelle: https://developer.apple.com/documentation/arkit/world_tracking/understanding_world_tracking
[9] Quelle: https://developer.apple.com/documentation/arkit/tracking_and_visualizing_faces Zugriff:10.12.2020

Innerhalb von ARKit gibt es dafür die Klasse ARSCNView. Wenn man einen ARSCNView in sein User Interface einbaut, enthält dieser automatisch eine Augmented Reality Session, welche man mit bestimmten Einstellungen starten kann, je nach dem, welchen Kamera Feed man beispielsweise verarbeiten möchte. Diese Session wird dann durch einen ebenfalls enthaltenen Renderer automatisch auf dem ARSCNView angezeigt, sodass Entwicklern ein schneller und einfacher Einstieg in die eigene Augmented Reality App ermöglicht wird.
In ARCore stellt sich der Einstieg etwas komplizierter dar, da es kein Interface Element gibt, das genau auf Augmented Reality Sessions zugeschnitten ist. Stattdessen muss man mit einem GLSurfaceView aus der OpenGL ES API arbeiten, da die Augmented Reality Session von ARCore den Zugriff auf derzeitige Frames nur innerhalb eines GL Kontexts erlaubt. Um zu vermeiden, einen von Grund auf neuen Renderer zu schreiben, der mittels der OpenGL ES API die Frames der Augmented Reality Session auf einen GLSurfaceView zeichnet, wurde für die Basis der Android App ein Beispiel Projekt von Google gewählt, in dem bereits ein solcher Renderer enthalten ist.[10]

5.2.2 Test 1: Tiefenbilder

Für diesen Testmodus sollten die Tiefendaten live anstelle des normalen Kamerabildes angezeigt werden und bei Bedarf das derzeige Tiefenbild gespeichert werden. Dafür kann man in ARCore eine Methode namens acquireDepthImage benutzen, welche jederzeit ein Tiefenbild des derzeit betrachteten Frames liefern kann. Wechselt man allerdings von der Rück- zur Frontkamera ist diese Methode nicht mehr verfügbar, da ARCore nur unter Nutzung der Rückkamera detaillierte Tiefendaten sammelt, sodass die Möglichkeit fehlt, mithilfe der Frontkamera ein Tiefenbild zu erstellen. Genauer gesagt sind unter Nutzung der Frontkamera bei ARCore fast ausschließlich Funktionalitäten verfügbar, die dabei helfen Gesichter zu finden und ihnen Animationen hinzuzufügen.
Innerhalb von ARKit stellt es sich gewissermaßen andersherum dar: Unter Nutzung der Frontkamera werden detaillierte Tiefendaten mithilfe der TrueDepth-Kamera gesammelt und in Form eines Tiefenbildes der Szene jederzeit zur Verfügung gestellt. Benutzt man mit ARKit allerdings die Rückkamera, steht die Option sich ein Tiefenbild liefern zu lassen, nicht mehr zur Verfügung. Dies ist verwunderlich, da anders als bei der Frontkamera des Google Pixels, mithilfe der Rückkamera des iPhones sogar außerhalb des Augmented

[10] Grundlage: https://github.com/google-ar/arcore-android-sdk/tree/master/samples/hello_ar_java Zugriff: 6.1.2021

Reality Kontexts Tiefendaten von derzeitigen Kamerabildern gesammelt und als Tiefenbild geliefert werden können. Dies führt zu der Frage, ob ARKit diese Tiefendaten dem Nutzer einfach nicht zur Verfügung stellt oder ob es sie selbst nicht nutzt.[11] Trotzdem werden die Tiefendaten für diesen Testmodus bei der Rückkamera des iPhones außerhalb des Augmented Reality Kontexts angezeigt, um die Qualität der Daten vergleichen zu können.

5.2.3 Test 2: Distanzmessung

Die Implementation von diesem Testmodus verlief in beiden Frameworks ziemlich parallel. Die Grundlage für diese Funktion ist die Fähigkeit der Systeme, in einer betrachteten Szene automatisch Flächen erkennen und dann gegebenenfalls bestimmte Punkte der Fläche markieren zu können. Hierzu kann man in diesem Modus einen beliebigen Punkt der Szene antippen und beide Systeme sind in der Lage zu überprüfen, ob man eine erkannte Fläche berührt hat. Ist dies der Fall, kann man ein virtuelles Objekt an dem Punkt einsetzen, an dem die Fläche berührt wurde, um die Stelle zu markieren. Hat man auf diese Art und Weise zwei Objekte platziert, gilt es nur noch den Abstand zwischen ihren Positionen auszurechnen. Ein hilfreicher Umstand dabei ist, dass beide Frameworks ein Koordinatensystem mit der Einheit Meter für die Szene erstellen, in dem die Positionen von virtuellen Objekten gespeichert werden. Also kann man eine Einschätzung des realen Abstands zwischen zwei virtuellen Objekten erhalten, indem man den Abstand ihrer Positionen im Koordinatensystem, wie in Abschnitt 3.5 erläutert, ausrechnet.

5.2.4 Test 3: Objektverdeckung

Für diesen Testmodus galt es, zunächst ein virtuelles Objekt mit einem bestimmten Abstand zur Kamera in der Szene platzieren zu können. Hierfür kann man sich bei beiden Frameworks sowohl Position als auch Orientierung der Kamera ausgeben lassen. Hat man diese beiden Informationen, kann man ein Objekt von der Position der Kamera aus eine bestimmte Entfernung weit in Richtung der Orientierung verschieben. In diesem Fall wird die Entfernung über einen Schieberegler festgelegt. Der wichtigere Teil dieses

[11] Auf diese Frage habe ich auch nach intensiver Recherche keine Antwort gefunden, aber es gibt reichlich Nutzer die sich darüber wundern, dass ARKit keine Tiefendaten zur Verfügung stellt solange man die Rückkamera nutzt, wie beispielsweise: https://developer.apple.com/forums/thread/127279 Zugriff: 12.4.2021

Tests ist allerdings, wie sich das System verhält, wenn man nach Platzierung des virtuellen Objekts ein reales Objekt zwischen Kamera und virtuellem Objekt platziert. In der ARCore App gibt es dafür das Boolean Feld useDepthForOcclusion. Stellt man dieses auf True, versteckt der bereitgestellte Renderer automatisch virtuelle Objekte hinter realen Objekten, sollte dies basierend auf den Tiefendaten nötig sein. Innerhalb von ARKit gibt es nur eine ähnliche Konfiguration namens personSegmentationWithDepth. Wie der Name vermuten lässt, benutzt auch hier das System Tiefendaten, um automatisch virtuelle Objekte zu verstecken. Allerdings mit dem Unterschied, dass diese Option speziell für das Verdecken durch Personen konzipiert ist. Während es zwar beeindruckend ist, dass ARKit zwischen Personen und unbelebten Objekten unterscheiden kann, fehlt trotzalledem die Möglichkeit, virtuelle Objekte auch hinter realen Objekten, die keine Personen sind, verschwinden zu lassen. Würde das Framework vollständige Tiefendaten für eine betrachtete Szene zur Verfügung stellen, könnte man möglicherweise selbst eine allgemeine Objektverdeckung implementieren. Wie in Abschnitt 5.2.2 beschrieben, stellt ARKit jedoch unter Nutzung der Rückkamera keine vollständigen Tiefendaten der Szene zur Verfügung. Nichtsdestotrotz kann man die Präzision der Objektverdeckung mithilfe einer Person als verdeckendes Objekt überprüfen, was für diesen Test ausreichend ist.

5.2.5 Test 4: Gesichtserkennung

In diesem Testmodus soll geprüft werden, wie akkurat die Systeme Gesichter erkennen und verfolgen. Um es Nutzern zu ermöglichen, mit gefundenen Gesichtern zu interagieren, können beide Frameworks die Koordinaten der Vertices eines Dreiecksnetztes liefern, welches das Gesicht repräsentiert. Indem man kleine virtuelle Kugeln an jeder erhaltenen Position in die Szene einsetzt, kann man live visualisieren, wie die Systeme das Gesicht wahrnehmen. Dies funktioniert bei beiden Systemen gleichermaßen, allerdings ist anzumerken, dass ARCore die Koordinaten in Form eines unsegmentierten FloatBuffers liefert. Darin befinden sich alle Positionen fortlaufend mit ihren einzelnen X- Y- und Z-Koordinaten. Auf der anderen Seite liefert ARKit die Positionen in Form eines Arrays, das vector_float3 Objekte enthält, welche, wie der Name schon andeutet, 3-dimensionale Vektoren repräsentieren. Beide Arten, die Daten zu liefern, erfüllen ihren Zweck, jedoch kann man mit ARKits Methode intuitiver arbeiten, was insbesondere bei der Implementation von komplizierteren Funktionen als dieser wichtig ist.

6 Testergebnisse und Evaluation

6.1 Test 1: Tiefenbild

Die folgenden Bilder zeigen jeweils vier Darstellungen der selben Szene. Jeweils ein normales Bild und drei Tiefenbilder: eins mithilfe der Rückkamera des Google Pixels und jeweils eins mithilfe der Rück- und Frontkamera des iPhones erstellt. Mithilfe der Frontkamera des Google Pixels war dies, wie oben erläutert, nicht möglich. Hierbei ist immer das normale Bild links oben, das Tiefenbild vom Google Pixel rechts oben, das Tiefenbild von der Rückkamera des iPhones links unten und das Tiefenbild von der Frontkamera des iPhones rechts unten.

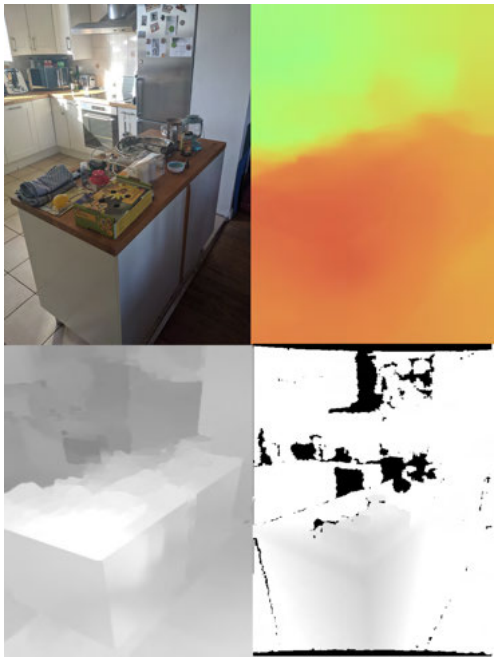

Abbildung 6.1: Tiefenbilder Szene 1

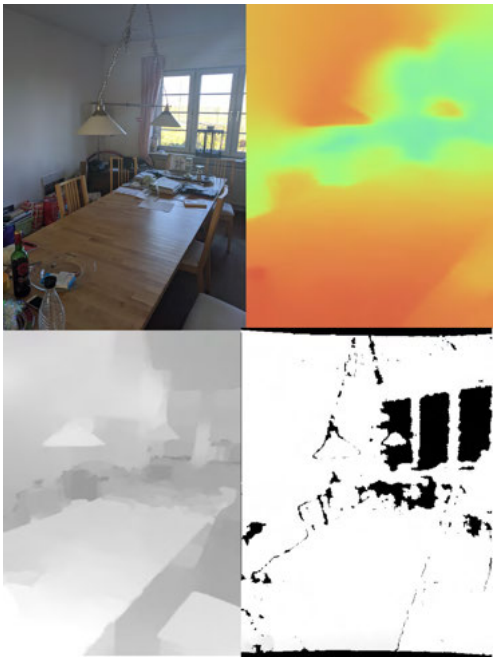

Abbildung 6.2: Tiefenbilder Szene 2

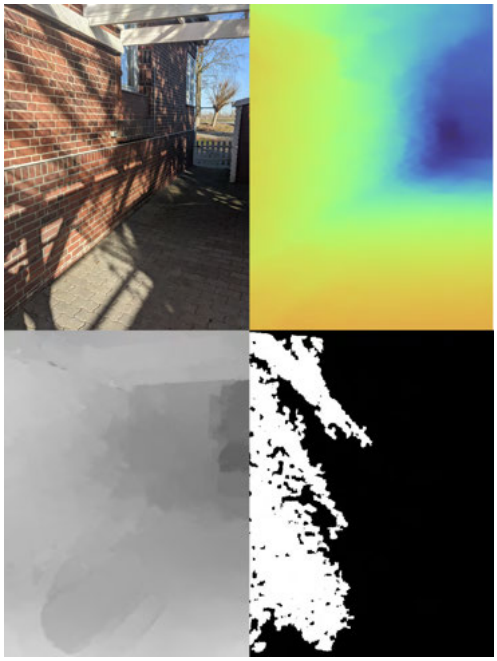

Abbildung 6.3: Tiefenbilder Szene 3

Abbildung 6.4: Tiefenbilder Szene 4

Zunächst fallen bei den Bildern die Unterschiede darin auf, wie die Tiefendaten dargestellt werden: Beim Google Pixel mit einem Farbspektrum von Rot bis Blau und bei dem iPhone lediglich mit Weiß bis Schwarz bzw. verschiedenen Grautönen. Intuitiv würde man möglicherweise vermuten, dass ein Spektrum mit mehr Farben auch eine detailliertere Darstellung ermöglicht, allerdings sind diese sogenannten Regenbogenkarten für die Darstellung von Daten nicht ideal, da sie keiner Wahrnehmungsordnung entsprechen. Bei der Darstellung des iPhones dagegen kann man dunkele und helle Grautöne als Gegensätze identifizieren und so klarer die Unterschiede in den Daten erkennen[2]. Betrachtet man die ersten beiden Szenen, sind sowohl das Tiefenbild der Frontkamera als auch das der Rückkamera des iPhones dementsprechend etwas detaillierter als das Tiefenbild des Google Pixels, was besonders gut an Kanten und Übergängen zu erkennen ist. Guckt man sich daraufhin aber die dritte Szene an, bei der draußen eine Hauswand mit vielen Schatten und ungleichmäßigen Lichtverhältnissen betrachtet wird, scheint das Google Pixel, trotz des schlechter gewählten Farbspektrums, die beste Darstellung der Entfernungen in der Szene zu liefern, da der fließende Übergang der Farben, welcher zuvor an Kanten zu einer ungenauen Darstellung führte, die immer weiter entfernte Fläche der Wand sehr gut darstellt. Die Rückkamera des iPhones schafft es ebenfalls, noch eine gute Darstellung zu liefern, während die Frontkamera ein sehr abstraktes Ergebnis erbrachte, das mehr an die

Schatten auf der Wand erinnert als an die Wand selbst. Bei der letzten Szene kann die Frontkamera des iPhones dafür ihre Präszision demonstrieren, als aus kürzerer Entfernung eine Person betrachtet wird. Hier liefert sie das beste Ergebnis, da die Ränder der Person am deutlichsten dargestellt sind und sogar teilweise Details wie die Ränder der Brille erkennbar sind. Dass die Frontkamera bei dieser letzen Szene besser abschneidet als die beiden Rückkameras ergibt Sinn, da dies der häufigste Anwendungsfall für Frontkameras ist. Alles in allem scheint es so, als ob die Frontkamera des iPhones am besten für Szenen geeignet ist, die nicht weit entfernt sind und ausgeglichene Lichtverhältnisse haben. Die Rückkamera des iPhones dagegen kommt auch mit höheren Entfernungen und schwierigeren Lichtverhältnissen zurecht, wobei die besten Ergebnisse ebenfalls bei gleichmäßiger Belichtung erzielt werden. Die Darstellungen des Google Pixels sind unter idealen Umständen zwar ein wenig ungenauer, da Kanten und Übergänge weniger klar dargestellt werden, dafür leidet die Qualität so gut wie gar nicht unter schwierigeren Verhältnissen, wie hoher Entfernung oder ungleichmäßiger Belichtung. Also kann man bei diesem Test das Fazit ziehen, dass die beiden Kameras des iPhones unter den richtigen Verhältnissen, auch durch die bessere Wahl des Farbspektrums, etwas genauere Tiefenbilder liefern, aber die Rückkamera des Google Pixels dafür flexibler einsetzbar ist. Außerdem ist abschließend nicht zu vergessen, dass das Google Pixel unter Nutzung der Frontkamera anscheinend nicht genug Tiefendaten sammelt, um eine Tiefenkarte zu erstellen.

6.2 Test 2: Distanzmessung

Für diesen Test wurde die Breite eines 30cm breiten Blocks aus verschiedenen Entfernungen mithilfe der Testgeräte vermessen. Die Entfernungen zum Block betrugen erst einen halben Meter, dann einen Meter, daraufhin drei Meter und zuletzt fünf Meter. Im Folgenden ist die Messung aus einem halben Meter Entfernung als Beispiel zu sehen, wobei die Zahl am unteren Bildrand das Messergebnis ist. Die unterschiedliche Höhe der virtuellen Marker ist irrelevant. Bilder von den übrigen Messungen sind im Anhang zu finden.

Abbildung 6.5: Beispiel Messung aus 50cm, iPhone

Abbildung 6.6: Beispiel Messung aus 50cm, Google Pixel

Entfernung	iPhone	Google Pixel
50cm	30,7cm	31,8cm
1m	30,6cm	32,2cm
3m	30,2cm	30,8cm
5m	N/A	33,9cm

Tabelle 6.1: Messergebnisse

Die Testergebnisse führen zu einer Reihe von Erkenntnissen. Zunächst sind die Messergebnisse des iPhones näher an den realen Maßen (30cm). Diese Genauigkeit ist allerdings begrenzt: Es war dem iPhone aus 5 Metern Entfernung gar nicht mehr möglich, den Stein zu vermessen. Wie zuvor erläutert, funktioniert dieser Testmodus, indem man eine beliebige Stelle der Szene antippt, und wenn das System an dieser Stelle eine Fläche findet, wird die Stelle auf der Fläche markiert. Wenn man beim iPhone aus 5 Metern Entfernung auf oder vor den Stein tippte, geschah nichts. Also ist es dem iPhone aus dieser Entfernung anscheinend nicht möglich, den Stein selbst als Fläche wahrzunehmen, noch den Boden unmittelbar davor. Beim Google Pixel war es zwar noch möglich, aber die Qualität der Distanzeinschätzung zwischen den Punkten litt auch hier. Die Entfernung von 3 Metern schien dagegen die beste Entfernung zu sein, um Distanzen einzuschätzen, da hier die Ergbenisse bei beiden Systemen näher an der Realität sind als aus den kürzeren Distanzen. Also kann man zusammenfassen, dass beide Systeme an Präzision verlieren, wenn man dem Objekt zu nah kommt oder zu weit vom Objekt entfernt ist. Außerdem liefert das iPhone im Allgemeinen zwar präzisere Ergebnisse, ist aber aus zu hoher Entfernung gar nicht mehr in der Lage zu messen, was das Google Pixel wiederum flexibler erscheinen lässt.

6.3 Test 3: Objektverdeckung

Bei diesem Test wurde zunächst eine virtuelle Box bei einer bestimmten Entfernung platziert und dann ein reales Objekt in den Weg gestellt, um zu sehen, wie präzise das virtuelle Objekt verdeckt wird. Da das verdeckende Objekt, wie zuvor erläutert, beim iPhone als Teil einer Person erkannt werden muss, damit die Objektverdeckung bei ARKit funktioniert, handelt es sich bei dem realen Objekt um eine Person.
Das sah dann folgendermaßen aus:

Abbildung 6.7: Verdeckung aus 0.5m, iPhone

Abbildung 6.8: Verdeckung aus 0.5m, Google Pixel

Abbildung 6.9: Verdeckung aus 1m, iPhone

Abbildung 6.10: Verdeckung aus 1m, Google Pixel

Abbildung 6.11: Verdeckung aus 3m, iPhone

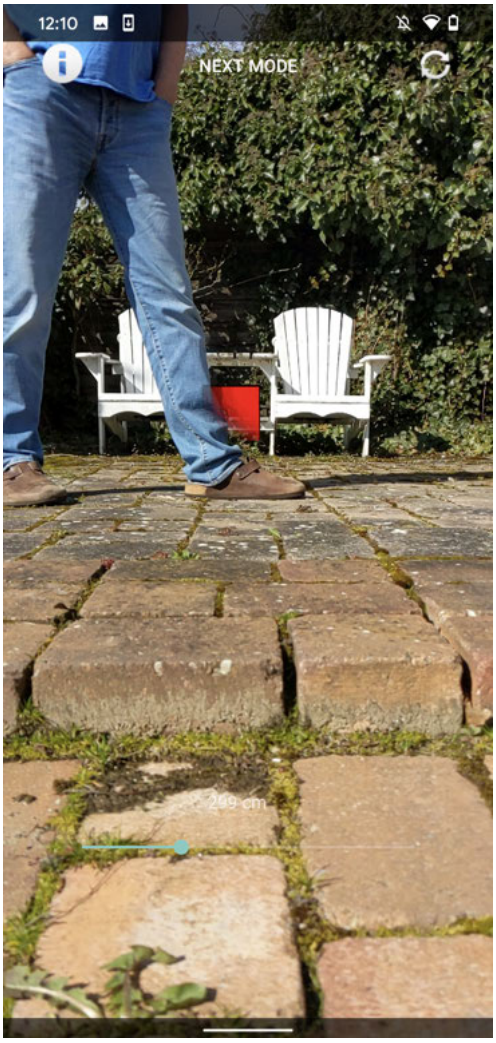

Abbildung 6.12: Verdeckung aus 3m, Google Pixel

Aus allen drei Distanzen ist deutlich zu erkennen, dass die Objektverdeckung beim iPhone besser funktioniert. Bei der Entfernung von einem halben Meter wird es am deutlichsten, denn während beim iPhone die virtuelle Box sehr akkurat von der Hand verdeckt wird, scheint das Google Pixel gar nicht zu registrieren, dass die Hand näher am Gerät ist als die Box. Bei einem Meter verbessert sich das Google Pixel zwar insofern, dass es den Fuß als verdeckendes Objekt erkennt, aber was die Genauigkeit der Verdeckung angeht, gibt es nach wie vor einen klaren Qualitätsunterschied zum iPhone. Das ändert sich auch bei einer Entfernung von drei Metern nicht. Dementsprechend scheint das iPhone dem

Google Pixel, was präzise Objektverdeckung angeht, einen Schritt voraus zu sein. An dieser Stelle möchte ich allerdings erneut darauf hinweisen, dass ARKit nur in diesem ziemlich spezifischen Anwendungsfall in der Lage ist Objektverdeckung durchzuführen. Versucht man es beispielsweise mit dem Stein aus dem vorherigen Test sieht das so aus:

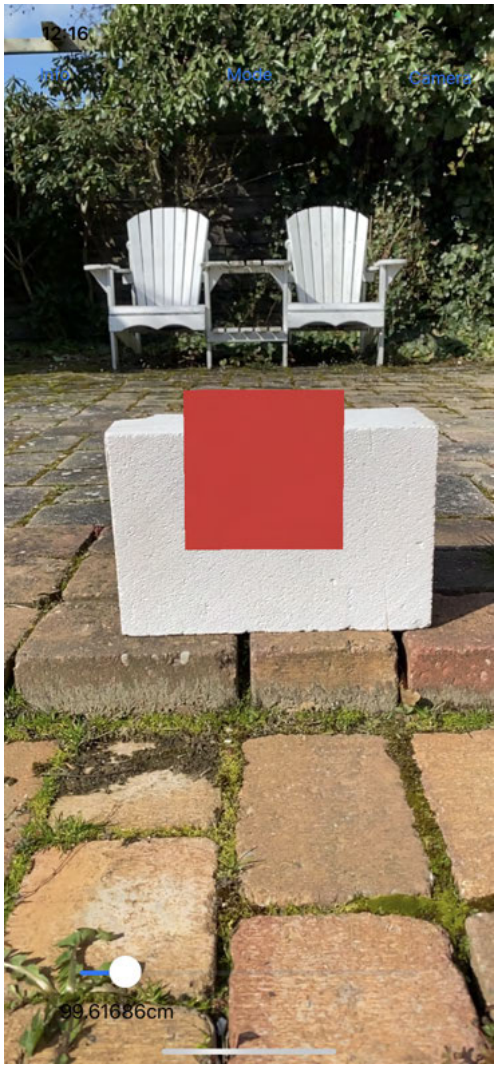

Abbildung 6.13: Verdeckung aus 1m mit Stein, iPhone

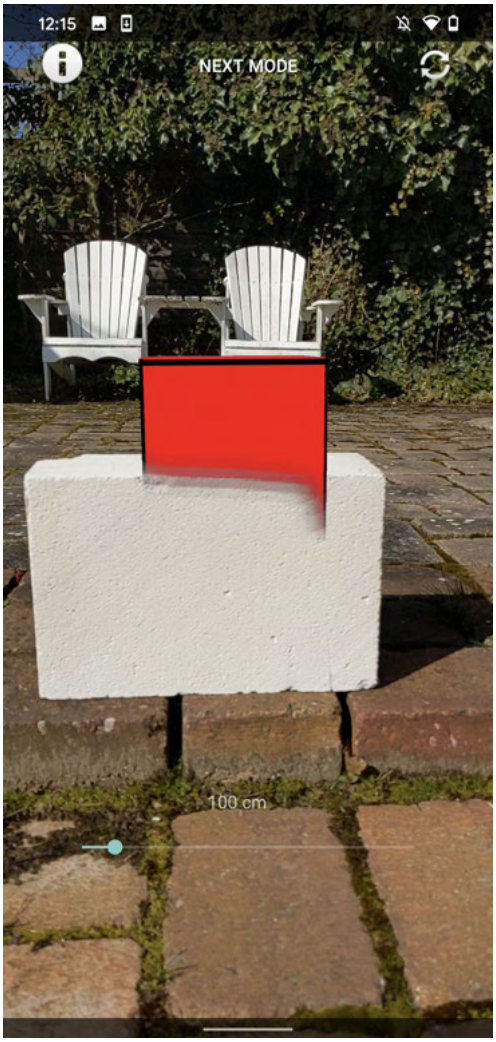

Abbildung 6.14: Verdeckung aus 1m mit Stein, Google Pixel

Dies führt zu der Erkenntnis, dass ARKit zunächst das Kamerabild der Szene nach Personen durchsucht, um nur spezifisch für gefundene Personen Objektverdeckung durchzuführen. Über den Grund hierfür könnte ich nur Mutmaßungen anstellen und werde

deshalb nicht weiter darauf eingehen. Trotzdem lässt sich insgesamt schlussfolgern, dass das iPhone unter der Bedingung des richtigen Anwendungsfalls die Objektverdeckung präziser durchführt. Das Google Pixel auf der anderen Seite kann in mehr Fällen für Objektverdeckung eingesetzt werden, wenn auch weniger präzise.

6.4 Test 4: Gesichtserkennung

Bei diesem letzten Test sollte abschließend die primäre Funktionalität der Frontkameras getestet werden: Gesichter erkennen und tracken. Dafür sollte, sobald ein Gesicht erkannt und ein Dreiecksnetz für das Gesicht erstellt wurde, eine kleine virtuelle Kugel für jeden Vertex des Dreiecksnetzes in die Szene eingesetzt werden. Das Ergebnis sah dann so aus:

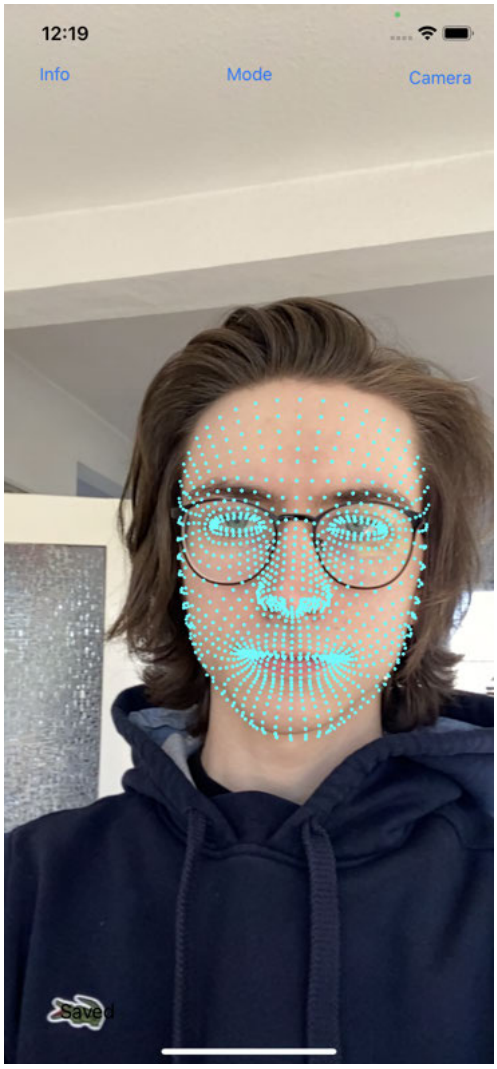

Abbildung 6.15: Gesichtserkennung, iPhone

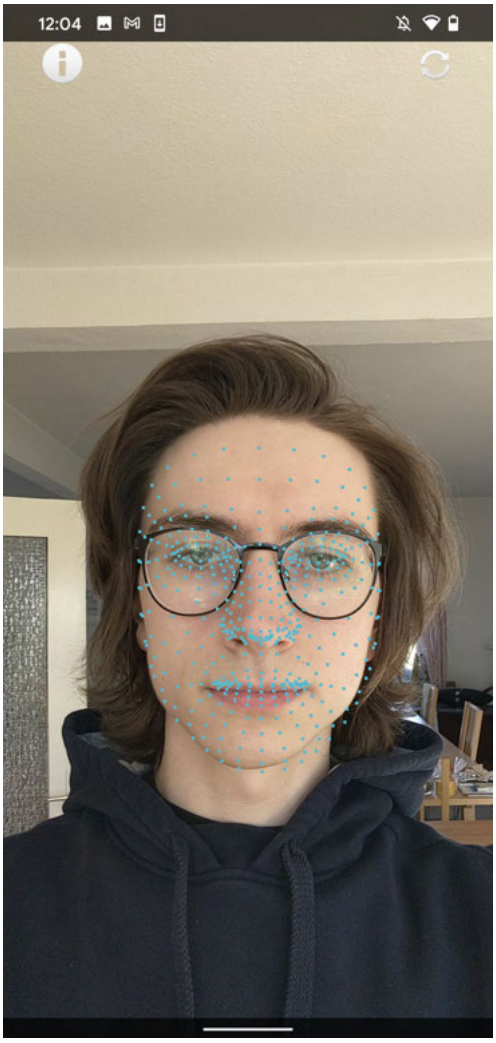

Abbildung 6.16: Gesichtserkennung, Google Pixel

Wie man sieht, enthält das Dreiecksnetz des iPhones deutlich mehr Vertices, was zu erkennen gibt, dass die virtuellen Modelle von erkannten Gesichtern beim iPhone deutlich detaillgetreuer sind als beim Google Pixel. Dies ist vermutlich auf die Unterschiede der Frameworks zurückzuführen, denn während ARCore von Google für viele unterschiedliche Android Geräte entwickelt wurde und relativ niedrige Ansprüche an die Hardware stellt, ist ARKit von Apple genau auf die Hardware der neuesten iPhone Modelle zugeschnitten. So benutzt das iPhone mittels ARKit die eingebaute TrueDepth Kamera, um ein sehr detailliertes Modell des Gesichts zu erstellen, während ARCore die Nutzung der

im Google Pixel 4 enthaltenen NIR-Kameras gar nicht ermöglicht, sondern mithilfe der Bilder der normalen Frontkamera das Gesicht modelliert.
Ein weiterer Qualitätsunterschied zwischen den beiden Systemen tritt zutage, wenn sich das betrachtete Gesicht dreht:

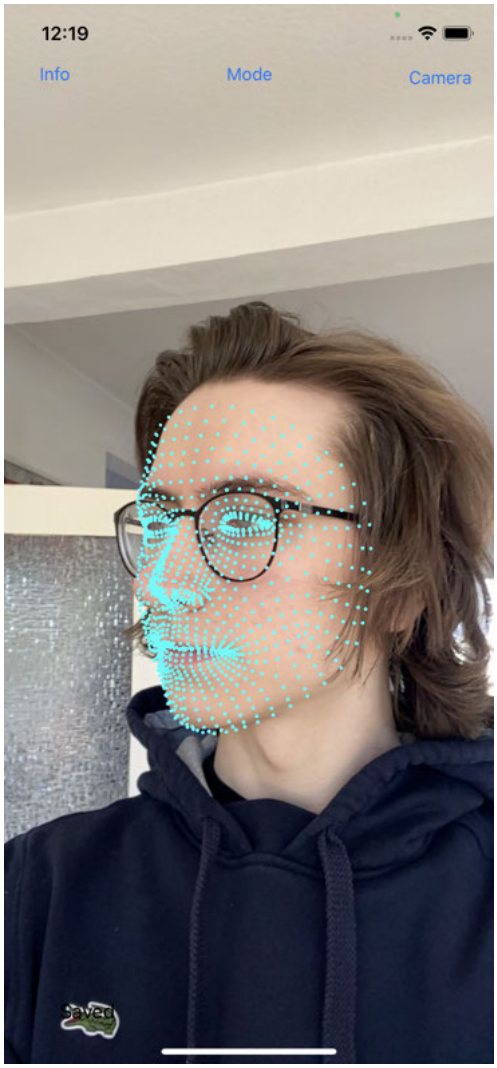

Abbildung 6.17: Gesichtserkennung nach Links gedreht, iPhone

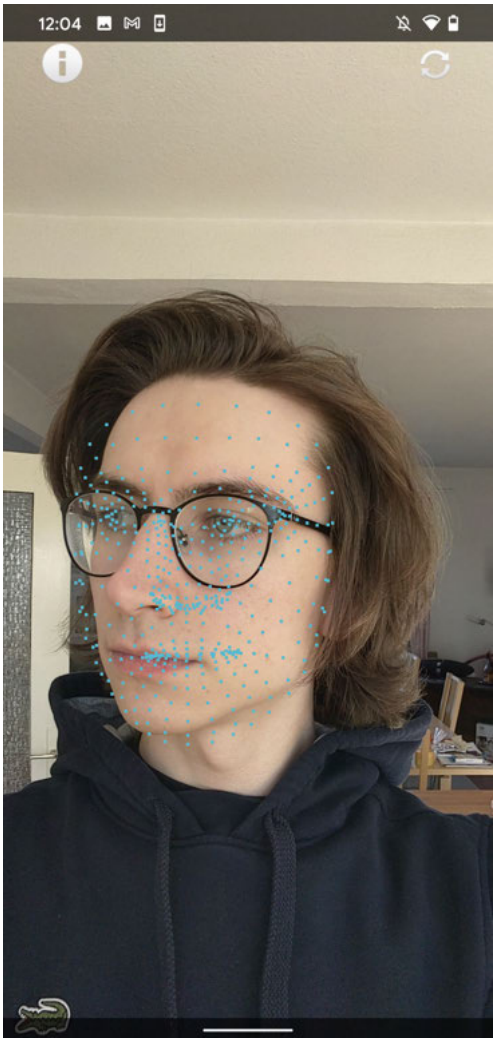

Abbildung 6.18: Gesichtserkennung nach Links gedreht, Google Pixel

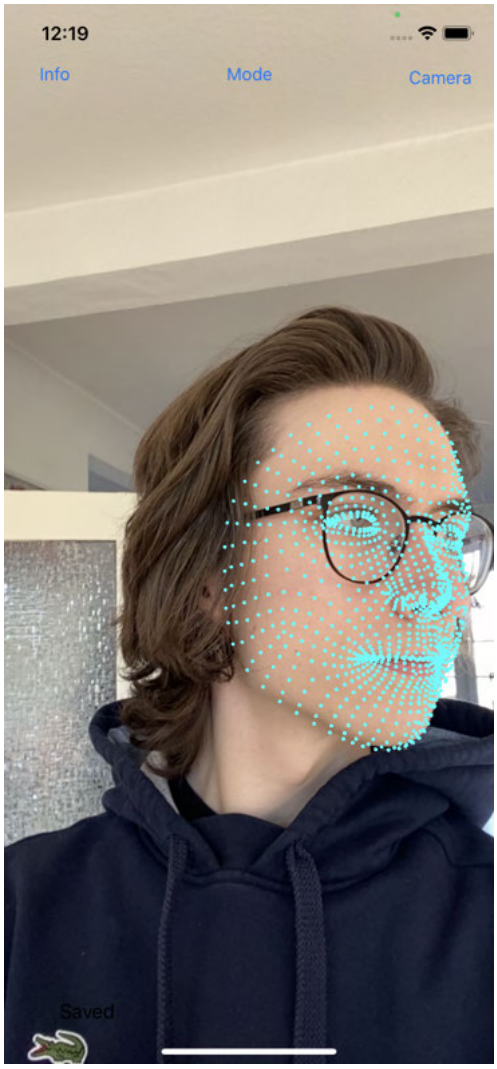

Abbildung 6.19: Gesichtserkennung nach Rechts gedreht, iPhone

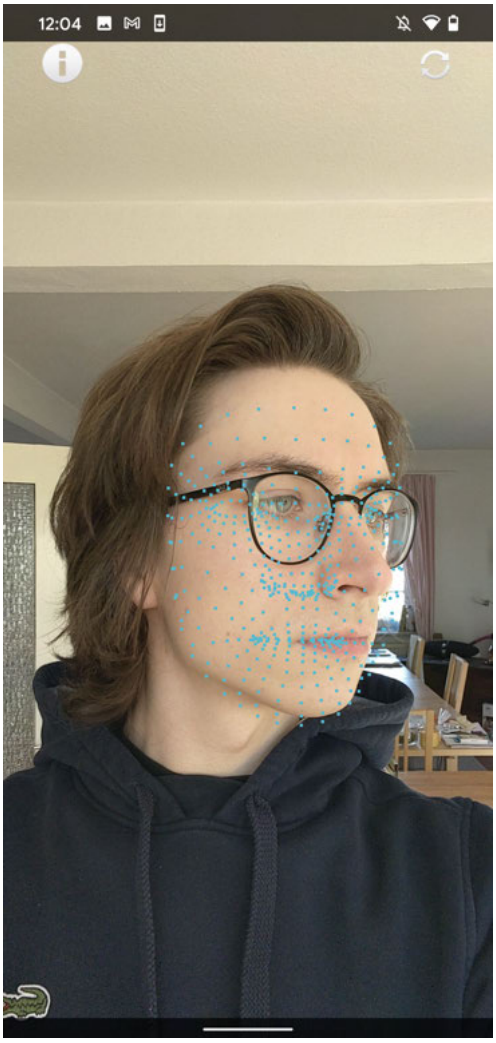

Abbildung 6.20: Gesichtserkennung nach Rechts gedreht, Google Pixel

Während das Dreiecksnetz des iPhones sich mit dem Gesicht dreht und so eindrucksvoll die Qualität des Trackings demonstriert, bewegt sich das Dreiecksnetz des Google Pixels lediglich mit, ohne sich zu drehen, sodass die Punkte des virtuellen Modells teilweise deutlich von den zugehörigen Punkten im realen Gesicht abweichen, wie man beispielsweise an der Nase gut erkennen kann. Alles in allem erstellt das iPhone also nicht nur ein detaillierteres Modell des erkannten Gesichts, sondern trackt es auch noch deutlich genauer. Dies ist, wie zuvor erläutert, nicht zwangsläufig nur auf bessere Tiefenberechnun-

gen zurückzuführen, sondern auf die effektivere Nutzung der zur Verfügung stehenden Hardware durch das zugehörige Framework. Die Erkenntnis, dass das Framework von Google selbst, die Möglichkeiten eines Google Pixels nicht voll ausschöpft, während das Framework von Apple genau auf neue Hardware zugeschnitten ist, ist aber auch durchaus relevant.

6.5 Fazit

Alles in allem haben definitiv beide Systeme ihre Vor- und Nachteile, die sich Entwickler bewusst machen sollten, wenn sie eine Plattform für ihre Augmented Reality Anwendung auswählen. Der Einstieg in eine eigene Augmented Reality Anwendung ist bei ARKit einfacher, da vorgefertigte Renderer und Views speziell für Augmented Reality bereitgestellt werden. Wenn man den Einstieg geschafft hat, ermöglicht ARCore allerdings einfacheren Zugriff auf die Tiefendaten der Szenen. Genauer gesagt ermöglicht ARKit gar keinen vollständigen Zugriff auf die Tiefendaten der Rückkamera des iPhones, obwohl das iPhone diese Daten in einem anderen Kontext liefern kann. Im Vergleich der Tiefendaten der Testgeräte in Bildform, erkennt man sogar, dass die Tiefendaten des iPhones unter den richtigen Umständen präziser sind als die des Google Pixels. Legt man wiederum mehr Wert auf Flexibilität, erscheint einem das Google Pixel besser, da die Qualität der Tiefenbilder auch unter schwierigen Lichtverhältnissen nicht schlechter wird. Ein ähnliches Fazit ergibt sich, wenn die Systeme die Distanz zwischen zwei Punkten berechnen sollen. Die Ergebnisse des iPhones sind im Allgemeinen näher an der Realität als die des Google Pixels, aber auch hier ist das Google Pixel wieder flexibler. Will man nämlich die Distanz von zwei weit von dem Gerät entfernten Punkten wissen, liefert das iPhone gar kein Ergebnis mehr, während das Google Pixel zumindest ein weniger präzise werdendes Ergebnis liefert. Dass ARCore flexibler in seinen Anwendungsfällen ist als ARKit, wurde bei dem letzten Test der Rückkamera noch einmal besonders deutlich, da das iPhone virtuelle Objekte ausschließlich von Personen verdecken lässt und nicht von anderen realen Objekten. Verdeckt eine Person ein virtuelles Objekt beim iPhone, scheint die Verdeckung sogar deutlich genauer zu funktionieren als beim Google Pixel, allerdings ist dies ein sehr spezifischer Anwendungsfall, auf den das Google Pixel, im Gegensatz zum iPhone, nicht angewiesen ist. Was die Rückkameras der Testgeräte angeht, kann man also das Fazit ziehen, dass das iPhone unter den richtigen Umständen präzisere Tiefenberechnungen anstellt, das Google Pixel dafür aber auch unter schwierigeren Umständen relativ konstante Ergebnisse liefert.

Eindeutiger fällt das Fazit aus, wenn man die Frontkameras vergleicht. Hier weist das iPhone in jeder Hinsicht Vorteile auf. Es liefert vollständige Tiefendaten der Szene, wozu das Google Pixel unter Nutzung der Frontkamera gar nicht in der Lage ist. Nun könnte man aus Sicht des Google Pixels argumentieren, dass Frontkameras zumeist nicht zum Sammeln von Tiefendaten eingesetzt werden, sondern vor allem, um Gesichter zu erkennen und zu tracken. Doch auch bei dieser Funktionalität sind klare Qualitätsunterschiede zu erkennen, da das iPhone sehr viel mehr Punkte von Gesichtern trackt und so präzisere Modelle erstellt. Selbst den Vorteil der Flexibilität, den man dem Google Pixel bei der Rückkamera noch zuschreiben konnte, hat beim Vergleich der Frontkameras das iPhone, da sich die virtuellen Modelle von erkannten Gesichtern geradezu perfekt mit dem realen Gesicht mitbewegen, während die Modelle des Google Pixels bereits bei einer leichten Drehung des Gesichts deutliche Abweichungen aufweisen. Ist man also ein Entwickler, der sich basierend auf der Qualität der Tiefenberechnungen entscheiden muss, auf welcher Plattform er seine Augmented Reality Anwendung entwickeln will, sollte man sich zunächst fragen, welche der beiden Kameras man primär benutzen wird. Soll vor allem die Frontkamera genutzet werden, sollte die Entscheidung in jedem Fall auf das iPhone fallen. Ist die Rückkamera wichtiger, ist die Antwort nicht ganz so einfach. Legt man mehr Wert darauf, unter den richtigen Umständen genauere Ergebnisse zu erhalten, sollte man sich auch hier für das iPhone entscheiden. Ist es aber wichtiger, dass die Anwendung unter verschiedenen Umständen konstant funktioniert, ist das Google Pixel besser geeignet.

7 Zusammenfassung

Im Rahmen dieser Arbeit wurden zwei Test Apps entwickelt, mithilfe derer die Qualität der Tiefenberechnung für Augmented Reality Anwendungen auf einem Apple iPhone 11 und einem Google Pixel 4 getestet wurden. Hierzu wurden zunächst Grundlagen von Augmented Reality Anwendungen beschrieben und insbesondere auch Funktionalitäten, welche im Zusammenhang mit Tiefenberechnungen genutzt werden. Daraufhin wurden die geplanten Tests sowie die Anforderungen an die Apps beschrieben, die die Tests durchführen sollten. Danach wurde die Herangehensweise an das Softwareprojekt näher beleuchtet, gefolgt von der Beschreibung der Durchführung des Projekts. Abschließend wurden die Ergebnisse der Tests berichtet und die Geräte basierend darauf verglichen.
Bei den Tests wurde die Unterscheidung zwischen den Front- und Rückkameras der Testgeräte vorgenommen. Bei beiden Kameras wurde die vollständige Sammlung und Darstellung von Tiefendaten getestet, während ausschließlich bei den Rückkameras die Präzision von Distanzmessung und Objektverdeckung geprüft wurde. Bei den Frontkameras dagegen wurde die Erkennung sowie das Tracking von Gesichtern zusätzlich näher betrachtet. Bei dem Vergleich der Rückkameras zeigte sich, dass beide Geräte gut dazu geeignet sind, als Plattformen für Augmented Reality zu dienen, da sie alle getesteten Funktionalitäten anbieten. Allerdings zeigten sich Unterschiede in der Präzision und Variabilität der Geräte. Während das Google Pixel in mehr verschiedenen und schwierigeren Anwendungsfällen einsetzbar war, lieferte das iPhone in den Anwendungsfällen, die beide durchführen konnten, präzisere Ergebnisse. Das beste Beispiel hierfür ist die Objektverdeckung. Während das Google Pixel diese mit jedem beliebigen Objekt durchführen konnte, musste das verdeckende Objekt beim iPhone eine Person sein, damit die Objektverdeckung funktionierte. Verglich man allerdings die Präzision der Verdeckung in diesem Anwendungsfall, waren Qualitätsvorteile beim iPhone gegenüber dem Google Pixel zu erkennen.
Weniger ausgeglichen verlief der Vergleich der Frontkameras. Das Google Pixel war im Gegensatz zum iPhone unter Nutzung der Frontkamera nicht nur nicht in der Lage vollständige Tiefendaten der Szene zu sammeln, sondern darüber hinaus schnitt es auch

deutlich schlechter als das iPhone bei der Durchführung der Gesichtserkennung und -verfolgung ab.

8 Ausblick

Augmented Reality und insbesondere mobile Augmented Reality sind weiterhin im Kommen, da sie in so vielen verschiedenen Situationen einsetzbar sind. Dass gerade mobile Augmented Reality etwas ist, worauf Apple in Zukunft noch mehr setzen will, ist spätestens seit der Ankündigung des während meiner Arbeit an diesem Projekt erschienenen iPhone 12 Pro klar. Dieses beinhaltet nämlich einen Light detection and ranging (LiDaR) Sensor. Dieser Sensor ist in der Lage, mit hoher Geschwindigkeit eine Tiefenkarte für eine betrachtete Szene zu erstellen, indem er misst, wie schnell Licht von unterschiedlichen Stellen der Szene reflektiert wird. [1] Es wäre daher sicherlich interessant, in Zukunft die hier durchgeführten oder ähnliche Tests mit einem iPhone 12 Pro durchzuführen und zu vergleichen. Aller Erwartung nach müsste das neuere Modell viel bessere Ergebnisse liefern als die vorherigen Modelle. Auch neuere Modelle von anderen Herstellern, die keine LiDaR Sensoren oder ähnliche Technologien beinhalten, sollten vom iPhone 12 Pro, was Tiefenberechnungen angeht, in den Schatten gestellt werden. Allerdings sind nur Vermutungen möglich, bis man es testet.

[1] Quelle: https://www.apple.com/de/iphone-12-pro/ Zugriff: 23.03.2021

Literaturverzeichnis

[1] Aguileta, Antonio A. ; Brena, Ramon F. ; Mayora, Oscar ; Molino-Minero-re, Erik ; Trejo, Luis A.: *Multi-sensor fusion for activity recognition—a survey.* 2019

[2] Borkin, Michelle A. ; Gajos, Krzysztof Z. ; Peters, Amanda ; Mitsouras, Dimitrios ; Melchionna, Simone ; Rybicki, Frank J. ; Feldman, Charles L. ; Pfister, Hanspeter: Evaluation of artery visualizations for heart disease diagnosis. In: *IEEE Transactions on Visualization and Computer Graphics* (2011). – ISSN 10772626

[3] Bosch, Siegfried: *Lineare Algebra.* 2021. – ISBN 978-3-662-62616-0

[4] Broll, Wolfgang ; Dröll, Ralf ; Grimm, Paul ; Jung, Bernhard: *Virtual und Augmented Reality (VR/AR).* 2019. – ISBN 978-3-662-58861-1

[5] Chatzopoulos, DImitris ; Bermejo, Carlos ; Huang, Zhanpeng ; Hui, Pan: *Mobile Augmented Reality Survey: From Where We Are to Where We Go.* 2017

[6] Concha, Alejo ; Loianno, Giuseppe ; Kumar, Vijay ; Civera, Javier: Visual-inertial direct SLAM. In: *Proceedings - IEEE International Conference on Robotics and Automation*, 2016. – ISBN 9781467380263

[7] Goh, Eg S. ; Sunar, Mohd S. ; Ismail, Ajune W.: *3D object manipulation techniques in handheld mobile augmented reality interface: A review.* 2019

[8] Hanser, Eckhart: 2010. – ISBN 978-3-642-12313-9

[9] Ionescu, Dan ; Suse, Viorel ; Gadea, Cristian ; Solomon, Bogdan ; Ionescu, Bogdan ; Islam, Shahidul ; Cordea, Marius: A 3D NIR camera for gesture control of video game consoles. In: *CIVEMSA 2014 - 2014 IEEE Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications, Proceedings*, 2014. – ISBN 9781479926138

[10] Klopschitz, Manfred ; Schall, Gerhard ; Schmalstieg, Dieter ; Reitmayr, Gerhard: Visual tracking for augmented reality. In: *2010 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2010 - Conference Proceedings*, 2010. – ISBN 9781424458646

[11] Lee, Eun K. ; Kim, Sung Y. ; Jung, Young K. ; Ho, Yo S.: High-resolution depth map generation by applying stereo matching based on initial depth informaton. In: *2008 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video, 3DTV-CON 2008 Proceedings*, 2008. – ISBN 9781424417551

[12] Leonard, J.J. ; Durrant-Whyte, H.F.: Simultaneous map building and localization for an autonomous mobile robot, 1991. – ISBN 0-7803-0067-X

[13] Lieberknecht, Sebastian ; Benhimane, Selim ; Meier, Peter ; Navab, Nassir: Benchmarking template-based tracking algorithms. In: *Virtual Reality* (2011). – ISSN 13594338

[14] Milgram, Paul ; Kishino, Fumio: Taxonomy of mixed reality visual displays. In: *IEICE Transactions on Information and Systems* (1994). – ISSN 09168532

[15] Nischwitz, Alfred ; Fischer, Max ; Haberäcker, Peter ; Socher, Gudrun: *Computergrafik*. 2019. – ISBN 978-3-658-25384-4

[16] Oufqir, Zainab ; El Abderrahmani, Abdellatif ; Satori, Khalid: ARKit and ARCore in serve to augmented reality. In: *2020 International Conference on Intelligent Systems and Computer Vision, ISCV 2020*, 2020. – ISBN 9781728180410

[17] Qiao, Xiuquan ; Ren, Pei ; Dustdar, Schahram ; Liu, Ling ; Ma, Huadong ; Chen, Junliang: Web AR: A Promising Future for Mobile Augmented Reality-State of the Art, Challenges, and Insights. In: *Proceedings of the IEEE* 107 (2019), Nr. 4. – ISSN 15582256

[18] Rosselot, Donald: *Processing real-time stereo video in disparity space for obstacle mapping*, University of Cincinnati, Diplomarbeit, 2005

[19] Salomon, David: *Data compression: The complete reference*. 2007. – ISBN 1846286026

[20] Tönnis, Marcus: *Augmented Reality einblicke in die erweiterte Realität*. 2010 ISSN 1098-6596

[21] Wang, Wei J. ; Wan, Hua G.: Real-time camera tracking using hybrid features in mobile augmented reality. In: *Science China Information Sciences* (2015). – ISSN 18691919

[22] Yang, Xin ; Guo, Jiabin ; Xue, Tangli ; Cheng, Kwang Ting (.: Robust and real-time pose tracking for augmented reality on mobile devices. In: *Multimedia Tools and Applications* (2018). – ISSN 15737721

[23] Zhang, Xiaopeng ; Sim, Terence ; Miao, Xiaoping: Enhancing photographs with near infrared images. In: *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008. – ISBN 9781424422432

A Anhang

Abbildung A.1: Messung aus 1m, iPhone

Abbildung A.2: Messung aus 1m, Google Pixel

Abbildung A.3: Messung aus 3m, iPhone

Abbildung A.4: Messung aus 3m, Google Pixel

Abbildung A.5: Messung aus 5m, Google Pixel

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	[redacted]
		Unterschrift im Original