

# **Thesis**

zur Erlangung des Grades

## **Master**

im Studiengang Medieninformatik Master

an der Fakultät Digitale Medien

# **Entwicklung einer Augmented Reality App zur Visualisierung historischer Gebäude der Stadt Villingen**

Erstbetreuer: Prof. Dr. Uwe Hahne

Zweitbetreuer: Prof. Dr. Thomas Schneider

Abgegeben am: 31.08.2022

Abgegeben von: Oliver Kusch

Matrikelnummer: 268513

Erbsenlachen 6, 78050 Villingen-Schwenningen

oliver.kusch@hs-furtwangen.de



## Abstract

Durch das Projekt NISABA<sup>1</sup> und der Veranstaltung Bildverarbeitung und Computergrafik im Studiengang Medieninformatik Master im Sommersemester 2021 sind 3D-Modelle der Kasernengebäude des Lyautey- und Mangin-Geländes entstanden. Um die fertigen 3D Modelle für jeden zugänglich und auf moderner Weise präsentieren zu können, wird in dieser Master-Arbeit eine Augmented Reality Anwendung für Smartphone und Tablet Geräte entwickelt.

Diese Forschungsarbeit beschäftigt sich mit der Entwicklung einer Augmented Reality AR Anwendung. Durch genaues Tracking und eine realistische Darstellung der Gebäude durch wetterspezifische Belichtung, Schattierung und Spiegelung soll eine hohe Immersion erschaffen werden. In der Anwendung werden die 3D-Objekte in der Größendarstellung 1:1 über das Videobild der Kamera gelegt und der Nutzer kann vor Ort das Gebäude in Echtzeit ein- und ausblenden lassen. Dafür wird der Begriff *world-scale Augmented Reality* herangezogen. Bekannte Beispiele sind das AR Spiel Pokemon GO<sup>2</sup> oder auch Anwendungen für Googles Holo Lens<sup>3</sup>.

Eine wesentliche Rolle bei der Immersion in AR spielt das Tracking, bei dem kontinuierlich die Positions- und Rotationsdaten des Endgeräts erfasst werden. Es gibt mehrere Verfahren, um die Position und Orientierung der Kamera relativ zur Umgebung zu bestimmen. Im begrenzten Räumen ist das Tracking durch einheitlichere Belichtung und vorhandenen Kanten und Flächen mit kamerabasierten Tracking Methoden gut umsetzbar. Im Freien kann das Tracking je nach Umgebung Probleme bereiten. Schwierigkeiten entstehen durch unterschiedliche Lichtverhältnisse und dem größeren Suchraum. Hinzu kommt, dass durch die große Distanz zwischen Kamera und virtuellem Objekt die Platzierung des 3D-Modells bereits durch kleine Bewegung der Kamera stark von der korrekten Position abweicht. Das Objekt erscheint nicht homogen in der realen Welt, wodurch die Immersion beeinträchtigt wird.

Klassischerweise wird bei AR im Freien GPS und die Neigungs- und Beschleunigungssensoren der Smartphones genutzt, um die Kameraposition und -orientierung zu ermitteln. Wie Platinsky und seine Koautoren[18] bereits erörtert ist die GPS Lokalisierung insbesondere in Städten mit Störfaktoren wie Gebäuden oder Vegetation ungenau. Deshalb wird in dieser Arbeit auf Methoden eingegangen, um die Genauigkeit der Position und der Orientierung der Kamera im Freien zu verbessern.

Um die Immersion bei der Nutzung der App zu steigern, werden die Wetterbedingungen bei der Darstellung der 3D-Modelle berücksichtigt. So sollen die Fassaden bei regnerischem Wetter dunkler und gegebenenfalls spiegelnd dargestellt werden. Die Schattierungen sollen sich anpassen, indem bei hartem Licht (z.B. durch starke Sonneneinstrahlung) auch harte Schatten und bei weichem Licht

---

<sup>1</sup><https://nisaba.dm.hs-furtwangen.de/>

<sup>2</sup><https://pokemongolive.com/de/>

<sup>3</sup><https://www.microsoft.com/de-de/hololens>

(z.B. bei einer dichten Wolkendecke) weiche Schatten dargestellt werden.



## Inhaltsverzeichnis

<b>Zusammenfassung</b>	<b>3</b>
<b>Abkürzungsverzeichnis</b>	<b>8</b>
<b>1 Einleitung</b>	<b>9</b>
1.1 Hintergrund der Arbeit . . . . .	10
1.1.1 Photogrammetrische Aufzeichnungen des SABA Geländes . . . . .	10
1.1.2 Photogrammetrische Aufzeichnungen des Lyautey Geländes . . . . .	11
1.1.3 Photogrammetrische Aufzeichnungen des Mangin Geländes . . . . .	11
1.1.4 Übersicht vorhandener 3D Modelle . . . . .	12
<b>2 Theoretische Grundlagen</b>	<b>14</b>
2.1 Einordnung VR, AR und MR . . . . .	14
2.2 Tracking . . . . .	15
2.2.1 Koordinatensysteme . . . . .	16
2.2.2 Tracking mit der Inertial Measurement Unit (IMU) . . . . .	16
2.2.3 Kamera-basiertes Tracking . . . . .	16
2.2.4 SIFT - Scale Invariant Feature Transform . . . . .	18
2.2.5 GPS Tracking . . . . .	18
<b>3 Grafik-Rendering-Pipeline</b>	<b>19</b>
3.1 Applikationsstufe . . . . .	20
3.2 Geometrieverarbeitung . . . . .	21
3.2.1 Vertex-Shading . . . . .	21
3.2.2 Projektion . . . . .	22
3.2.3 Clipping . . . . .	22
3.2.4 Window-Viewport-Transformation . . . . .	23
3.3 Rasterisierung . . . . .	23
3.4 Pixelverarbeitung (Fragment-Shader) . . . . .	24
<b>4 Forschungsstand</b>	<b>24</b>
4.1 Tracking . . . . .	24
4.1.1 Koordinatensysteme . . . . .	25
4.2 Kamera-basiertes Tracking . . . . .	25
4.2.1 GPS Tracking . . . . .	26
4.3 Software Entwicklung . . . . .	27

<b>5 Fragestellungen und Methodik</b>	<b>27</b>
<b>6 Vorbereitung der 3D Modelle in Blender</b>	<b>28</b>
6.1 Lückenhafte Wände füllen . . . . .	28
6.2 Export und Import . . . . .	31
<b>Abbildungsverzeichnis</b>	<b>33</b>
<b>Eidesstattliche Erklärung</b>	<b>34</b>

## Abkürzungsverzeichnis

**AR** Augmented Reality

**CPU** Central Processing Unit

**FPS** Frames Pro Sekunde

**GPU** Graphics Processing Unit

**GRP** Grafik-Rendering-Pipeline

**IMU** Inertial Measurement Units

**Materials** Materials affektieren das Aussehen eines 3D Objekts mit einer oder mehrerer Texturen, Farben oder den Eigenschaften von Reflexionen

**MR** Mixed Reality

**SIFT** Scale Invariant Feature Transform

**SLAM** Simultaneous Localization and Mapping

**SURF** Speeded Up Robust Features

**VE** Virtual Environments

**VR** Virtual Reality

## 1 Einleitung

Durch vorangegangene Projekte sind mit photogrammetrischen Methoden 3D Modelle von historisch bedeutenden Gebäuden des SABA, des Lyautley- und des Mangin-Geländes der Stadt Villingen entstanden. Die Gebäude sind virtuell gespeichert, jedoch werden diese nicht weiter verwendet. In dieser Arbeit wird eine Augmented Reality Anwendung für Smartphones entwickelt, die eine Visualisierung der Modelle mit der AR Technologie bietet. Somit haben Bewohner der Stadt Villingen und interessierte Personen wie Touristen, die mehr über die Geschichte der Stadt Villingen erfahren möchten, Zugriff auf die Modelle und können damit einen Teil der Stadtgeschichte erleben.

Die Aufgabe besteht darin eine Anwendung zu entwickeln, die zum einen die Gebäude mittels GPS vor Ort platziert. Die Gebäude werden dabei im Größenverhältnis 1:1 dargestellt, sodass der Nutzer einen realistischen Eindruck bekommt, wie das Gebäude in Echt ausgesehen hat. Zum anderen ist es möglich die Gebäude auf jeder beliebigen horizontalen Fläche zu platzieren. Die Anwendung baut auf aktuelle AR Software Development Kits (ARCore bzw. ARKit) auf und wird mit der Game Engine Unity und AR Foundation entwickelt.

Ziel ist es eine hohe Immersion bei der Nutzung zu schaffen. Das Gebäude soll nahtlos im Kamerabild dargestellt werden. So wird eine Wetter REST-API gebunden (OpenWeatherMap), um aktuelle Wetterdaten abzurufen. Daraufhin wird die Darstellung der Modelle angepasst. Bei Regen wird die Szene dunkler, Materialien werden nass dargestellt und spiegeln die Umgebung. Herrscht starker Sonnenschein, so wird die Intensität des Lichts verändert und harte Schatten geworfen.

Diese Arbeit untersucht die Möglichkeiten von AR und dessen Limitierungen in einer freien Umgebung. Mit der GPS Funktionalität werden Genauigkeitsprobleme der GPS-Antennen in Smartphones untersucht.

In dieser Arbeit werden zunächst einleitend theoretische Grundlagen von Augmented Reality beschrieben. Auf der technischen Seite wird das Tracking, die Algorithmen zur Erkennung von Merkmalen, GPS Systeme und die Rendering Pipeline erläutert. Anschließend wird die Umsetzung der Anwendung beschrieben, indem auf das Konzept und die Implementierung der genannten Funktionalitäten eingegangen wird. Daraufhin werden bestehende Probleme und Limitierungen, die während der Entwicklung und dem Testing vorgekommen sind, veranschaulicht. Zuletzt werden Ideen für Erweiterungen und Verbesserungen der Anwendung vorgeschlagen.

## 1.1 Hintergrund der Arbeit

### 1.1.1 Photogrammetrische Aufzeichnungen des SABA Geländes

Das studentische Forschungsprojekt aus dem Wintersemester 19/20 [19] befasst sich mit der Photogrammetrischen Aufzeichnung des SABA-Geländes in Villingen-Schwenningen. SABA (Schwarzwälder Apparate-Bau-Anstalt) war ein deutsches Unternehmen, das unter anderem elektronische Geräte für den Rundfunk herstellte. Der Entwicklungs- und Produktionsstandort war das Gebäude in Villingen. Aufgrund der Größe des Unternehmens (mehr als 6000 Mitarbeiter), hatte SABA eine große Bedeutung für die Stadt. 1986 wurde das Unternehmen aufgelöst, bis das Gebäude am 11. August 2021 abgerissen wurde.<sup>4</sup>

Mithilfe von Drohnen- und Bodenaufnahmen wird ein 3D Modell des SABA Geländes mit photogrammetrischen Algorithmen erzeugt. *Regard3D*<sup>5</sup>, *RealityCapture*<sup>6</sup>, *VisualSFM*<sup>7</sup> und *Meshroom*<sup>8</sup> werden im Projektverlauf genutzt und verglichen, wobei *Meshroom* als kostenlose Open-Source Software hauptsächlich genutzt wird. Das Hauptgebäude wird nach der Erstellung des Meshes aus *Meshroom* neu modelliert. Aufgrund der hohen Dichte der Vertices ist die Aufteilung der einzelnen Gebäude-Elemente wie Fenster, Wand und Türen schwierig. Durch eine Neumodellierung wird das Modell klarer und die Texturierung wird vereinfacht. Als Grundlage dient dabei das generierte Mesh aus *Meshroom*. Als Modellierungssoftware wird *Blender* benutzt.<sup>9</sup> In Abbildung 1 ist das fertige 3D Modell zu sehen.



Abbildung 1: Das 3D Modell des SABA Hauptgebäudes in der 3D Karte aus den Drohnen-Aufnahmen.

---

<sup>4</sup><https://dewiki.de/Lexikon/SABA>, zuletzt aufgerufen am 17.03.2022

<sup>5</sup><https://www.regard3d.org/>

<sup>6</sup><https://www.capturingreality.com/>

<sup>7</sup><http://ccwu.me/vsfm/>

<sup>8</sup><https://github.com/alicevision/meshroom>

<sup>9</sup><https://www.blender.org/>

### 1.1.2 Photogrammetrische Aufzeichnungen des Lyautey Geländes

Im Projekt *NISABA* [21] aus dem Sommersemester 2020 und Wintersemester 2020/21 sind 3D Modelle von Gebäuden des ehemaligen Lyautey Kasernengeländes (das heutige "Richthofen") entstanden. Auch in diesem Projekt werden verschiedene Photogrammetrie Programme genutzt. *Meshroom*, *Pix4DMapper*<sup>10</sup>, *Agisoft Metashape*<sup>11</sup> und *WebODM*<sup>12</sup> werden für dieses Projekt verwendet. Gute Ergebnisse erzielt dabei *Pix4DMapper*, da viele Einstellungen über den Detailgrad getroffen und mehrere Projekte kombiniert werden können. Aus dem generierten Mesh wird auch hier eine Nachmodellierung in *Blender* durchgeführt.

Das Lyautey Gelände umfasst insgesamt sieben Gebäude. Für jedes Gebäude gibt es ein fertiges Modell aus *Pix4DMapper* und ein Modell, bei dem die Polygone reduziert sind. Nur das Mannschaftsgebäude gibt es nachmodelliert. Das ist das Gebäude 4 in der Abbildung 2. Alle 3D Modelle können auf der Website des *NISABA*-Projekts<sup>13</sup> begutachtet werden.

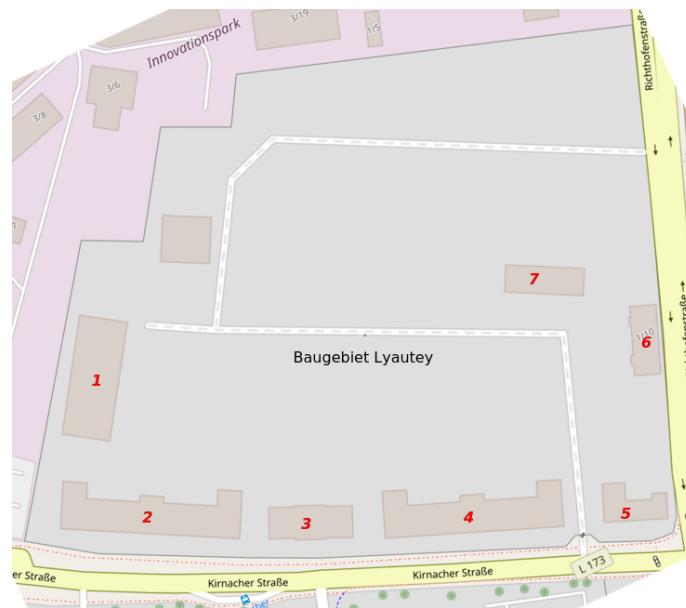


Abbildung 2: Eine Übersicht der Gebäude auf dem Lyautey-Gelände.

### 1.1.3 Photogrammetrische Aufzeichnungen des Mangin Geländes

Im Zuge der Veranstaltung Bildverarbeitung und Computergrafik im Sommersemester 2021 im Studiengang Medieninformatik Master sind weitere 3D Modelle entstanden[12]. Die Gebäude befinden sich auf dem für die Stadt historisch wichtigen Kasernengelände Mangin in Villingen-Schwenningen,

<sup>10</sup><https://www.pix4d.com/de/produkt/pix4dmapper-photogrammetrie-software>

<sup>11</sup><https://www.agisoft.com/>

<sup>12</sup><https://www.opendronemap.org/webodm/>

<sup>13</sup><https://nisaba.villingen-schwenningen.de/uebersicht/>

das sich direkt östlich vom Lyautey befindet. Dabei handelt es sich um ein verlassenes Kasernen-Gelände mit architektonisch und historisch interessanten Gebäuden. Die Aufnahmen der Gebäude erfolgte in Zusammenarbeit mit dem Vermessungsamt Villingen-Schwenningen. Es wurden Aufnahmen von den Gebäuden 2-8 und 10-12 gefertigt. In der Abbildung x ist eine Übersichtskarte des Geländes mit Nummerierungen der Gebäude zu sehen. Die Bilder aus der Luft wurden mit einer Drohne vom Vermessungsamt gemacht, während die Aufnahmen am Boden von den Studierenden aufgenommen wurden. Als Photogrammetrie-Software wird hauptsächlich *Pix4DMapper* und *Meshroom* verwendet.



Abbildung 3: Eine Übersicht der Gebäude auf dem Mangin-Gelände.

#### 1.1.4 Übersicht vorhandener 3D Modelle

Einige Modelle sind nicht vollständig, haben aufgrund der Vegetation vor Ort Löcher im Mesh oder einen niedrigen Detailgrad. Daher können in dieser Arbeit nicht alle 3D Modelle verwendet werden. Die Tabelle 1 zeigt eine Liste der vorhandenen 3D Modelle. Die Gebäude sind wie in den Übersichtskarten nummeriert. Ein geringer Detailgrad bedeutet, dass das Mesh Löcher hat oder unvollständig ist. Ein mittlerer Detailgrad zeichnet sich durch ein vollständiges Mesh mit geringen Details aus. Gebäude mit einem hohen Detailgrad können für die Augmented Reality (AR) Anwendung genutzt werden. Händisch modellierte Gebäude haben einen sehr hohen Detailgrad.

Gelände	Bezeichnung	Nr.	Detailgrad	Vertices	Texturenanzahl
SABA	Karte	-	gering	458.311	1
SABA	Hauptgebäude Neumodellierung	1	sehr hoch	44.708	34
SABA	Heizwerk	2	mittel	142	8
Lyautey	Karte	-	mittel	2.049.590	1
Lyautey	Reithalle	1	hoch	1.386.531	1
Lyautey	Mannschaftsgebäude 1	2	hoch	1.255.734	1
Lyautey	Wirtschaftsgebäude	3	hoch	1.610.983	1
Lyautey	Mannschaftsgebäude 2	4	hoch	175.352	1
Lyautey	Mannschaftsgebäude 2 Neumodellierung	4	sehr hoch	2.046	45
Lyautey	Stabshaus	5	gering	-	-
Lyautey	Familienhaus	6	mittel	119.574	1
Lyautey	Kammergebäude	7	hoch	269.460	1
Mangin	Karte	1	mittel	502.040	1
Mangin	Casino	2	mittel	681.082	1
Mangin	-	3	hoch	381.633	1
Mangin	Pferdestall	4	mittel	242.984	1
Mangin	-	6	gering	500.463	1
Mangin	-	7	gering	31.490	1
Mangin	-	8	hoch	5.093	1
Mangin	-	10	hoch	500.307	1
Mangin	-	11	mittel	427.753	1
Mangin	-	12	mittel	497.625	1
Mangin	-	17	gering	5.179	1
Mangin	-	23	gering	998	1
Mangin	Karte der Gebäude 17- 23	-	mittel	15.433	1

Tabelle 1: Eine Übersicht der vorhandenen 3D Modelle.

## 2 Theoretische Grundlagen

### 2.1 Einordnung VR, AR und MR

Bei Augmented Reality (AR) Anwendungen werden dreidimensionale virtuelle Objekte in eine echte dreidimensionale Umgebung in Echtzeit integriert. Es handelt sich um eine Variation der Virtual Environments (VE), der Virtuellen Realität (VR) [2]. Gerne wird zwischen AR, VR und Mixed Reality (MR) unterschieden. Speicher und seine Ko-Autoren [23] haben zur Eingrenzung Charakteristiken definiert, die aus Interviews mit Experten hervorgehen.

VR hat die Eigenschaft eine komplett synthetisch generierte, virtuelle Welt darzustellen. So bietet sich die Möglichkeit für den Nutzer unerreichbare Orte zu erleben. Es werden hierfür VR-Headsets wie die Oculus Rift S<sup>14</sup> oder die Playstation VR<sup>15</sup> benötigt und die Bewegung des Headsets muss getrackt<sup>16</sup> werden. Der Nutzer ist während der VR-Erfahrung von der echten Umgebung isoliert, wodurch die soziale Interaktion gering ist.

Zusätzlich zu der Definition Azumas[2] von AR wird die Eigenschaft genannt, dass der virtuelle Content dazu in der Lage ist mit der echten Welt zu interagieren. Im Gegensatz zu VR ist der Nutzer im gegenwärtigen physischen Raum gebunden. Die menschliche Wahrnehmung wird durch das augmentieren und der Kreation von Content Erfahrungen verbessert[23].

Die Eingrenzung von MR ist nicht eindeutig. So wird MR als Kombination von AR und VR gesehen. Die Anwendungen bieten die Möglichkeit sowohl AR als auch VR zu nutzen. Auch wird Mixed Reality in Verbindung mit spezifischer Hardware wie die HoloLens<sup>17</sup> gebracht. Ein weiteres Beispiel ist Pokemon GO<sup>18</sup>, bei dem zwischen der AR Funktion und einer virtuellen Umgebung umgeschaltet werden kann.

In Abbildung 4 haben Milgram und seine Ko-Autoren eine Grafik definiert, die das *Reality-Virtuality Continuum* darstellt. Die linke Hälfte des Kontinuums repräsentiert jede Umgebung, die rein aus realen Objekten besteht, persönlich betrachtet wird und durch jegliche Art wie den Blick durch ein Fenster oder eines Displays erweitert wird. Die rechte Hälfte besteht aus rein virtuellen Objekten, die entweder Monitor-basierend oder immersiv sind. Durch diese Darstellung wird MR zwischen diesen beiden Extremen eingeordnet und als Umgebung definiert, die die reale und die virtuelle Welt zusammen in einem Display darstellt [15].

---

<sup>14</sup><https://www.oculus.com/rift-s/>, zuletzt aufgerufen am 27.07.2022

<sup>15</sup><https://www.playstation.com/de-de/ps-vr/>, zuletzt aufgerufen am 27.07.2022

<sup>16</sup>siehe Kapitel 2.2 Tracking

<sup>17</sup><https://www.microsoft.com/de-de/hololens>, zuletzt aufgerufen am 27.07.2022

<sup>18</sup><https://pokemongolive.com/de/>, zuletzt aufgerufen am 27.07.2022

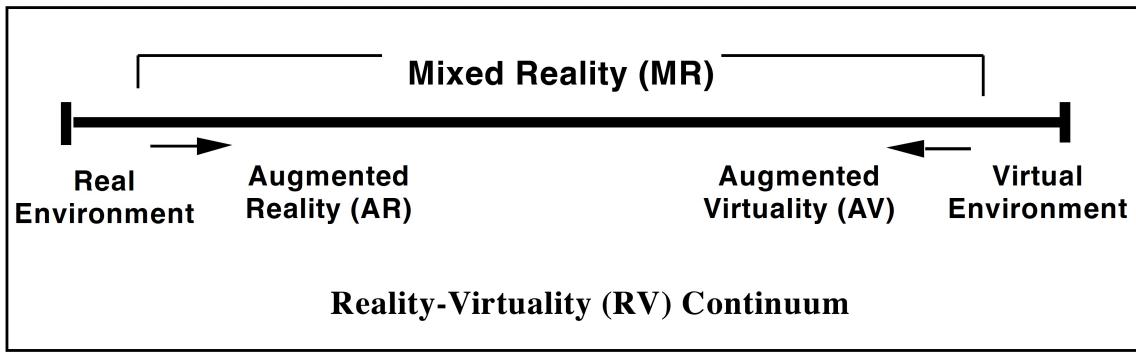


Abbildung 4: Das Reality-Virtuality Kontinuum nach Milgram[15].

Nach diesen Definitionen wird die Anwendung dieser Arbeit in die Augmented Reality eingeordnet. Die virtuellen Objekte in Form der Gebäude werden in die echte dreidimensionale Welt projiziert. Der Nutzer hat die Möglichkeit mit diesen zu interagieren. Jedoch wird es nicht möglich sein den betrachteten Raum zu verlassen und in eine reine virtuelle Welt umzusteigen.

## 2.2 Tracking

Der Begriff Tracking beschreibt die kontinuierliche Verfolgung von Positions- und Rotationsdaten, die von Eingabegeräten (z.B. VR Controller) oder Sensoren (z.B. *Inertial Measurement Units IMU*<sup>19</sup>) erfasst werden. Die Bewegung eines starren Körpers wird „durch die Angabe von sechs Werten (drei Koordinaten als Position und drei Winkel zur Beschreibung der Orientierung) für jeden Zeitschritt spezifiziert“[7, Dörner (2019) S.119f.].

Die Werte für die Position und Orientierung für ein Objekt werden als *Freiheitsgrade (engl. Degrees of Freedom – DOF)* bezeichnet. Das Ziel beim Tracking ist es, die sechs Freiheitsgrade für die Translation und Rotation der Objekte kontinuierlich zu bestimmen bzw. zu schätzen.

Es werden zwischen zwei Trackingverfahren unterschieden. Beim *Outside-In-Tracking* befindet sich die Sensorik zur Datenaufnahme in der Umgebung innerhalb eines bestimmten Raumes, sodass das Objekt von außen getrackt wird. Dies wird bei VR-Brillen eingesetzt. Der Nachteil dieser Technik ist, dass die Sensorik an einem Ort gebunden ist und bei einem Ortswechsel neu platziert werden muss. Beim *Inside-Out-Tracking* befinden sich Sensoren im Objekt, das getrackt werden soll. Die Position und Lage des Objekts wird im Verhältnis zur Umgebung gesetzt. Bei der AR Anwendung in dieser Arbeit handelt es sich somit um ein Inside-Out-Tracking.

In der Unity Manual zu AR Foundation wird der Begriff Tracking mit der Bestimmung der relativen Position und Orientierung in der physischen Welt definiert. Zusätzlich wird darauf hingewiesen, dass falls die Umgebung zu dunkel ist, das Gerät Probleme beim Tracking bekommt und

<sup>19</sup>beispielsweise Gyroskop oder Beschleunigungssensoren

die Genauigkeit der Positionsbestimmung sich verringert [25]. Damit ist davon auszugehen, dass in AR Foundation Kamera-basiertes Tracking (siehe Kapitel 2.2.3) verwendet wird.

### 2.2.1 Koordinatensysteme

Um eine Bestimmung bzw. Schätzung der Translationen und Rotationen durchzuführen, werden zwei Koordinatensysteme herangezogen. Ein Kamerakoordinatensystem und ein Objektkoordinatensystem. Weiterhin gibt es die Möglichkeit, dass für alle Objekte im Raum ein Koordinatensystem (Weltkoordinatensystem) verwendet wird. Voraussetzung für das Tracking ist, dass die Transformationen zwischen den Objekten bekannt sind. Dann kann die Transformation zwischen dem Objekt und dem Weltkoordinatensystem geschätzt werden.[7, Dörner (2019) S.124f.]. In Unity und AR Foundation hat das Gerät ein eigenes Koordinatensystem. Wird eine AR-Session gestartet, so wird ein Koordinatensystem für diese spezifische Session initialisiert. Die Koordinaten (0,0,0) referieren die Position des Gerätes, bei dem die Session gestartet ist [25].

### 2.2.2 Tracking mit der Inertial Measurement Unit (IMU)

Inertial Measurement Unit besteht typischerweise aus Beschleunigungssensoren und Drehratsensoren. Drei Beschleunigungssensoren und drei Drehratsensoren werden orthogonal zueinander eingebaut. Die Sensoren bilden ein Trägheitsnavigationssystem. Dieses misst die, Bewegungsrichtung, die Beschleunigung und die Drehung in einem kartesischen Koordinatensystem und bestimmt damit die Position und Orientierung der IMU<sup>20</sup>. Die Genauigkeit des IMU-Trackings ist insbesondere bei niedrigpreisigen Smartphones nicht akkurat. Bei der Bestimmung der Position kommt es zu durch die Ungenauigkeiten zu *Drifteffekten*. „Das heißt, wird beispielsweise ein Sensor aus dem Ruhezustand bewegt und anschließend wieder angehalten, so müssten sowohl die Summen der erfassten Beschleunigungswerte wie auch die der errechneten Geschwindigkeitswerte am Ende Null ergeben“[7, Dörner (2019) S.127f.]. Durch die Ungenauigkeit kann dies nicht erzielt werden und die errechnete Position weicht von der tatsächlichen Position ab. Zur Bestimmung der Orientierung wird die IMU oft genutzt.

### 2.2.3 Kamera-basiertes Tracking

Kamera-basiertes Tracking nutzt Informationen zu Objekten aus dem Video Datenstrom, um die relative Position und Orientierung der Objekte zur Kamera zu bestimmen. Die Position und Orientierung der Kamera in einem Weltkoordinatensystem wird von Hartley und Zisserman als extrinsische Kameraparameter bezeichnet [8, Hartley, Zisserman (2003) S.156]. Für das Kamera-basierte Tracking wird zwischen Marker-basierten und Marker-less Tracking unterschieden.

---

<sup>20</sup><https://www.5gpositioning.com/inertial-measurement-units-in-smartphones/>, zuletzt aufgerufen am 27.07.2022

Beim Marker-basierten Tracking werden schwarz-weiß Marker (sogenannte Kanji und Hiro Marker) wie in Abbildung 5 zu sehen genutzt. Die Marker werden in der Umgebung platziert und dienen als Ursprung, sodass diese als Orientierungshilfen fungieren. Das jeweilige Muster und die Größe der Marker müssen für das Tracking bekannt sein. Eine Anwendung von NGIN-Mobility arbeitet mit Markern, die auf dem Boden platziert werden können<sup>21</sup>. Da die Marker dauerhaft in der Umgebung platziert werden, hat diese Tracking-Methode Einschränkungen in der Flexibilität.

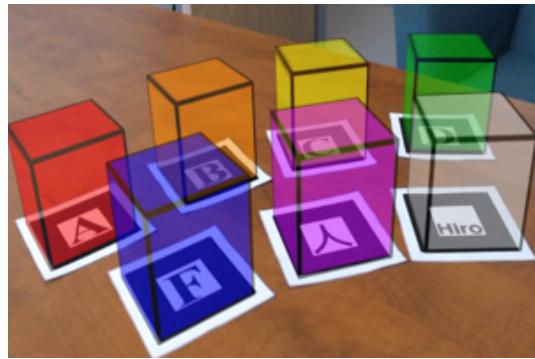


Abbildung 5: Kanji und Hiro Marker haben einfache geometrische Formen, Texte oder Buchstaben zur Erkennung in der Szene<sup>22</sup>.

Die zweite Möglichkeit erfasst über Algorithmen der Computer Vision Merkmale in der realen Umgebung. Diese Art wird als *marker-less AR* bezeichnet. Natürliche Merkmale können geometri-basiert sein, indem Kanten oder Formen wie Vierecke erkannt werden. Bekannte Detektoren wie sind z.B. *Scale Invariant Feature Transform (SIFT)*[14] oder *Speeded Up Robust Features (SURF)*[4].

Diese Detektoren arbeiten nach einem ähnlichen Schema. Zunächst werden nach *Schlüsselpunkten* (engl. *Key Points*, *Feature Points*) in einem Bild gesucht, die charakteristische Eigenschaften wie Punkte, Kanten, Kontrast oder homogene Regionen aufweisen. Je nachdem wie groß das Bild ist und wie stark die Schwellwerte beim jeweiligen Detektor eingestellt sind, verändert sich die Anzahl an Schlüsselpunkten. Dies beeinflusst die Tracking Performance und die Genauigkeit. Ist eine finale Auswahl der Schlüsselpunkte erfolgt, erhält jeder Punkt einen sogenannten *Deskriptor*. Dieser enthält lokale Eigenschaften auf dem Bild in der Umgebung des Punktes. Ziel eines Deskriptors ist es Schlüsselpunkte mit so wenig Informationen wie möglich eindeutig zu beschreiben. Sind die Deskriptoren vorhanden, kann im Anschluss ein *Feature-Matching* durchgeführt werden. Dabei werden Schlüsselpunkte und Deskriptoren aus einer Datenbank<sup>23</sup> mit den determinierten Deskriptoren

<sup>21</sup><https://www.logistik-watchblog.de/startups/1479-insider-navigation-ar-loesung-lagerhalle.html>, zuletzt aufgerufen am 15.04.02022

<sup>22</sup><https://stemkoski.github.io/AR-Examples/>, zuletzt aufgerufen am 29.07.2022

<sup>23</sup>Dies kann ein vorgefertigtes Set an Schlüsselpunkten sein, die beispielsweise aus vorherigen Durchläufen entstanden sind. Meist enthält dies auch Informationen zu den Dimensionen des Objekts. Diese Datenbank wird *Feature Map* genannt

verglichen und zugeordnet. Mithilfe der zugeordneten Schlüsselpunkten kann die Pose der Kamera berechnet werden [7][9]. Eine ausführliche Beschreibung der Algorithmen zu marker-less AR bietet Herling und Broll[9].

### **Herausforderungen für marker-less Tracking in AR Anwendungen**

Damit eine valide Positions und Rotation berechnet werden kann, definieren Herling und Broll[9] Voraussetzungen für Detektoren. Da AR in Echtzeit abläuft, muss die Bestimmung von Schlüsselpunkten und Deskriptoren schnell berechnet werden. Weiterhin muss der Algorithmus robust gegenüber unterschiedlichen Lichtverhältnissen sein. Dies gilt insbesondere bei AR in freier Umgebung, da sich die Lichtverhältnisse schnell verändern können. Da der Nutzer keine Einschränkungen in der Position der Kamera besitzt, kann ein schneller Perspektivwechsel erfolgen. Der Algorithmus muss daher eine Robustheit gegenüber starken Bildveränderungen haben. Letzlich muss eine Skalierungs-Invarianz bereitgestellt werden. Das bedeutet, dass das Tracking nicht auf eine bestimmte Entfernung beschränkt ist und weiter entfernte Punkte nicht verschwinden. In einem kleineren Bereich ist es einfacher Schlüsselpunkte zu detektieren. Die Skaleninvarianz ist daher insbesondere bei AR im Freien relevant.

Das SIFT Verfahren Auch die aus der Robotertechnik bekannte *Simultaneous Localization and Mapping*[6] [3] Methode wird genutzt.

#### **2.2.4 SIFT - Scale Invariant Feature Transform**

Vorteil dieser Methode ist, dass gleichzeitig eine 3D-Karte des Raumes generiert wird, die immer wieder zur Positions- und Rotationsbestimmung verwendet werden kann.

#### **SURF**

#### **SLAM**

Dabei werden entweder kamerabasiert (Visual SLAM) nach Merkmalen gesucht oder es kommen Sensoren zur Generierung von Tiefeninformationen zum Einsatz, z.B. Kinect <sup>24</sup> oder *LIDAR-Sensoren* (engl. *light detection and ranging*)[13].

#### **2.2.5 GPS Tracking**

Im Außenbereich wird bei AR auch GPS für das Tracking herangezogen. Dabei sind Positionsabweichungen von bis zu 10 Metern möglich, sodass eine genaue Bestimmung der extrinsischen Kameraparameter beeinträchtigt wird. Um die Tracking-Genauigkeit zu erhöhen, gibt es mehrere Methoden. *DGPS* (engl. *Differential GPS*) verbessert GPS-Signale, indem es ein Korrektursignal

---

<sup>24</sup><https://developer.microsoft.com/de-de/windows/kinect/>

durch eine ortsfeste Referenzstation mit bekannter Lokalisierung berechnet. Da es in Deutschland lediglich acht solcher Stationen gibt und einige Anbieter nur kommerziell die Daten bereitstellen, ist diese Methode nicht für diese Arbeit geeignet<sup>25</sup> <sup>26</sup>. Eine weitere bekannte Möglichkeit bietet *SBAS* (engl. *Satellite Based Augmentation System*), bei dem mehrere geostationäre Satelliten das GPS Signal auf bis zu einem Meter Genauigkeit zu verbessern [7].

Platinsky und seine Koautoren[18] erstellen für ein besseres Tracking bei fehlender GPS Genauigkeit ein 3D-Modell der Umgebung. Bei der anschließenden AR Nutzung in diesem Gebiet wird auf dem Smartphone SLAM betrieben. Die Daten vom Smartphone werden mit der 3D-Karte verglichen, um ein genaueres Tracking durchzuführen. Ein ähnliches System wäre für die Anwendung in dieser Arbeit denkbar, da eine große Datenmenge von Bildern des Geländes vorhanden ist. Über Structure from Motion Methoden, kann mit den Bildern eine große 3D-Karte erstellt werden.

### 3 Grafik-Rendering-Pipeline

Die Grafik-Rendering-Pipeline (GRP) dient dazu die dreidimensionalen Objekte mit einer virtuellen Kamera auf ein zweidimensionales Bild zu *rendern*<sup>27</sup>. Es handelt sich um eine Pipeline (Befehlskette), die aus mehreren Teilaufgaben bestehen. Die Geometrie, die Charakteristiken der Umgebung und die Platzierung der virtuellen Kamera in der Szene beeinflussen die Lokalisierung und die Form der 3D-Objekte. Die Materials, *Texturen*, Lichtquellen und Shader beeinflussen die Erscheinung [1, Moeller (2019)]. Materials affektieren das Aussehen eines 3D Objekts mit Texturen, Farben oder der Eigenschaften von Reflexionen. Texturen sind Bilder, die die Oberfläche des Objekts benetzen. Abbildung 6 zeigt mehrere dreidimensionale Objekte, die virtuelle Kamera und das daraus generierte zweidimensionale Bild.

Die GRP wird in vier Hauptstufen unterteilt:

- Applikation,
- Geometrieverarbeitung,
- Rasterisierung,
- Pixelverarbeitung.

Die Abarbeitung der Aufgaben in den verschiedenen Stufen wird parallel ausgeführt. Durch die Parallelisierung ist eine verbesserte Performance möglich, da nicht jeder Prozess Schritt für Schritt

<sup>25</sup><https://www.heise.de/newsticker/meldung/Differential-GPS-und-WLAN-RTT-Praezise-Ortung-mit-Android-P-4046935.html>

<sup>26</sup> Liste von DGPS-Sendern: <https://www.ndblist.info/datamodes/worldDGPSdatabase.pdf>

<sup>27</sup> „ein Bild (eine Grafik) aus Rohdaten (..) aus einer 3D-Szene(..)) durch einen Webbrowser/ein Programm erzeugen.“ <https://de.wiktionary.org/wiki/rendern>

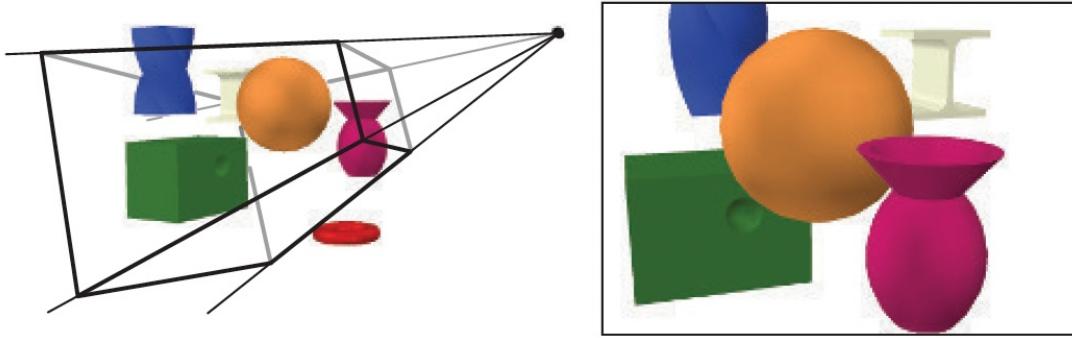


Abbildung 6: Die dreidimensionale Szene (links) und die zweidimensionale Sicht der virtuellen Kamera. Wird das Objekt perspektivisch gerendert, gibt es ein Raumvolumen (frustum). Es werden nur die Objekte gerendert, die sich innerhalb dieses Volumens befinden. So wird der rote Donut nicht und das blaue Objekt abgeschnitten dargestellt[1, Moeller (2019)].

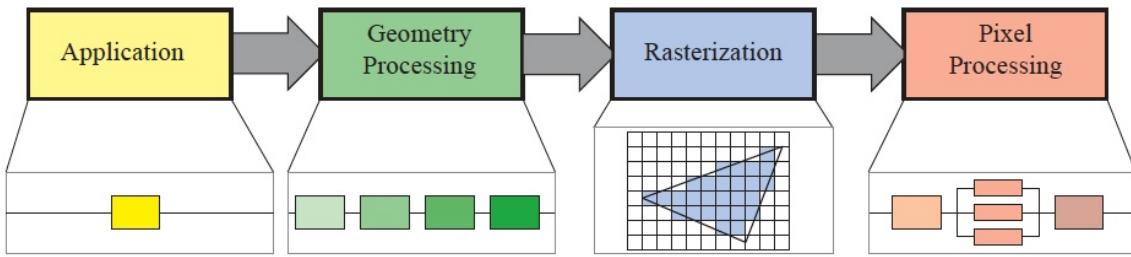


Abbildung 7: Die Basis der GRP, die in weitere Sub-Pipelines eingeteilt und parallel ausgeführt werden können [1].

durchgeführt werden muss. Jede Stufe kann gleichzeitig eine weitere Sub-Pipeline benutzen. So hat die Stufe *Geometrieverarbeitung* in der Abbildung 7 die Aufgabe in eine Sub-Pipeline gegliedert, die wiederum parallel ausgeführt werden kann. Die render Geschwindigkeit wird in *frames pro sekunde* (FPS) angegeben und repräsentiert die Anzahl an Bildern, die pro Sekunde gerendert werden[1, Moeller (2019)].

### 3.1 Applikationsstufe

Die Hauptaufgabe der Applikationsstufe besteht darin die benötigten Daten in die Pipeline zu schicken und zu aktualisieren. Die Daten bestehen aus den Render-Primitiven, die beispielsweise Punkte, Linien, Dreiecke oder Polygone (Vielecke) darstellen. Am Ende der Applikationsstufe werden die Rendering-Primitive an die Geometrieverarbeitungsstufe weitergegeben. Weiterhin finden Interaktionen wie z.B. Kollisionsabfragen statt oder es werden Input-Daten von Maus und Tastatur

verarbeitet. Auch werden Culling-Algorithmen<sup>28</sup> ausgeführt. Die Applikationsstufe wird im Gegensatz zu den anderen Stufen meist auf der CPU ausgeführt. Entwickler haben dadurch eine hohe Kontrolle über die Prozesse in dieser Stufe. Sie können die Implementierung selbst bestimmen und im nachhinein modifizieren, um z.B. die Performance weiter zu verbessern[1, Moeller (2019)].

## 3.2 Geometrieverarbeitung

Auf dieser Stufe finden Operationen an den Polygonen und Vertices (Eckpunkte) statt. Sie wird auf der GPU ausgeführt und lässt sich in eine Sub-Pipeline in Vertex-Shading, Projektion, Clipping und Window-Viewport-Transformation unterteilen.

### 3.2.1 Vertex-Shading

Ein Vertex-Shader ist ein programmierbarer Shader in der Rendering-Pipeline, der die Eigenschaften jedes einzelnen Vertex bestimmt. Ein Vertex beinhaltet die Koordinaten von Punkten im dreidimensionalen Raum. Mithilfe von Informationen, welche Vertices miteinander verbunden sind, werden Linien und Polygone definiert [17, Vgl. Nischwitz (2012) S.48.]. Die Aufgabe des Vertex-Shaders ist es, die Modellkoordinaten der Vertices durch Translation, Rotation oder Skalierung zu transformieren, um die Modelle zunächst im Weltkoordinatensystem, dann im Kamerakoordinatensystem und zuletzt im Clipping-Koordinatensystem zu lokalisieren. Die Transformationen werden mit 4x4 Transformationsmatrizen durchgeführt. Der Vertex-Shader berechnet die Position eines Vertex von seinem Modellkoordinatensystem zum Clipping-Koordinatensystem:

$$p' = P * K * W * p \quad (1)$$

wobei:

$p'$  = Vertex in Clip-Koordinaten

$P$  = Projektionsmatrix

$K$  = Kamera-Transformation

$W$  = Welt-Transformation

$p$  = Vertex in Modellkoordinaten

Die Reihenfolge der Transformationen spielt eine wichtige Rolle, da die Matrizenmultiplikation nicht kommutativ ist. Die Transformationen in der Gleichung 1 müssen daher von rechts nach links durchgeführt werden.

Die Vertices der Modelle befinden sich ursprünglich in ihrem eigenen Modellkoordinatensystem. Die Szene hat ein Weltkoordinatensystem und ein Kamerakoordinatensystem. Die Vertices

---

<sup>28</sup>ein Algorithmus, bei dem bestimmte (z.B. verdeckte) Polygone nicht gerendert werden, um eine bessere Performance zu erzielen [5].

werden zunächst in das Weltkoordinatensystem transformiert. Die Kamera in der Szene hat Koordinaten im Weltkoordinatensystem und eine Richtung, in die die Kamera zeigt. Mit der Kamera-Transformation werden die Vertices und die Kamera so platziert, dass die Kamera in der Position des Ursprungs des Weltkoordinatensystems liegt und (meist) in negativer z-Richtung zeigt, während die y-Achse nach oben und die x-Achse nach rechts zeigt. Die genaue Definition der Richtungen der Koordinatenachsen hängt von der jeweiligen Anwendung ab. Das daraus folgende Koordinatensystem wird Kamerakoordinatensystem genannt. Somit befinden sich die Vertices nach den Kamera-Transformationen im Kamerakoordinatensystem. Schließlich wird eine Projektionsmatrix angewandt, um die Kamerakoordinaten der Vertices in Clipping-Koordinaten zu transformieren.

Zusätzlich zu den Transformationen bestimmt ein Vertex-Shader Eigenschaften wie die Farbe, Texturkoordinaten oder Normalenvektoren für jeden Vertex. Diese beschreiben das Material des Objekts und Effekte von Lichtquellen, die auf die Oberfläche scheinen. Die Wirkung von Licht auf ein Material wird Shading genannt. Es werden "[..]Normalenvektoren und Materialeigenschaften der Objekte auf der einen Seite und den Eigenschaften der Lichtquellen auf der anderen Seite für jeden Vertex die Beleuchtungsrechnung durchgeführt und somit eine Farbe ermittelt" [Nischwitz (2012) S.48, 17]. Die Ergebnisse der Berechnungen werden zur Rasterisierung und zur Pixelverarbeitung weitergeschickt [1, Moeller (2019)].

Mit Vertex-Shadern ist es möglich Oberflächen realistisch darzustellen, ohne die Polygonanzahl zu erhöhen. So nutzen Mitchell [16] und Isidoro [11] Vertex-Shader, um mit Heightmaps und Normalmaps einen realistischen Ozean zu rendern.

### 3.2.2 Projektion

Bei der Projektion werden zwei Aufgaben durchgeführt. Zum einen wird ein Clipspace (Kanonisches-Sichtvolumen) definiert, das meist in Form eines Einheitswürfels<sup>29</sup> gegeben ist. Der Würfel ist für das Clipping von Bedeutung.

Zum anderen wird eine Projektionsmethodik je nach Anwendungsfall bestimmt. Beispielsweise gibt es eine orthographische Projektion, bei der parallele Linien auch parallel erscheinen, während bei der perspektivischen Projektion parallele Linien Fluchtpunkten folgen. Die Projektionsmethodik bestimmt die Projektionsmatrix für die Transformation in die Clipping-Koordinaten.

### 3.2.3 Clipping

Nur mit den im Vertex-Shader transformierten Clipping-Koordinaten der Vertices kann Clipping durchgeführt werden. Es werden nur die Primitiven zur Window-Viewport-Transformation weitergeschickt, die sich innerhalb des kanonischen-Sichtvolumens befinden. Clipping wird dann benötigt,

---

<sup>29</sup>ein Würfel mit der Skalierung (1,1,1)

wenn sich ein Teil eines Primitiven innerhalb und ein anderer Teil außerhalb des Sichtvolumens befinden. Die Primitiven, die sich mit dem Einheitswürfel schneiden, erhalten an der Schnittkante neue Vertices[Moeller (2019) 1].

### 3.2.4 Window-Viewport-Transformation

Im nächsten Schritt wird mit den Primitiven eine Window-Viewport-Transformation durchgeführt. Dadurch wird es möglich die x- und y-Koordinaten der Primitiven auf ein Bildschirmausschnitt (Viewport) darzustellen. Die Primitiven erhalten Bildschirm- bzw. Window-Koordinaten<sup>30</sup>. Es handelt sich dabei um eine Translation gefolgt von einer Skalierung[Moeller (2019) S.20f., 1] [Nischwitz (2012) S.150, 17].

## 3.3 Rasterisierung

Bei der Rasterisierung werden mit einer perspektivischen Projektion Pixel gesucht, die sich auf dem Viewport innerhalb der gerenderten Primitiven befinden. Es findet eine Übersetzung der 3D-Koordinaten in 2D-Pixelkoordinaten statt. Klassische Rasterisierungs-Algorithmen definieren einen zugehörigen Pixel genau dann, wenn ein Pixel sich innerhalb eines Primitiven befindet. Es werden sogenannte *Fragmente* für den Teil des Pixels erzeugt, der sich mit dem Dreieck überschneidet[22, zuletzt aufgerufen am 11.07.2022] cite\*[Moeller (2019) S.21f.,] moeller2019.

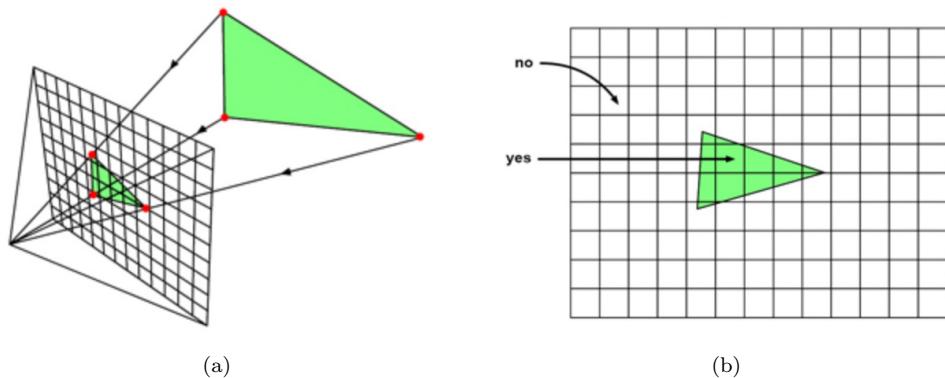


Abbildung 8: Die Primitiven werden in eine 2D-Ebene projiziert(a). Befindet sich ein Pixel innerhalb des Primitiven, werden diskrete Fragmente gebildet, die im in der Pixelverarbeitung weitergeschickt werden(b) [22, zuletzt aufgerufen am 11.07.2022].

<sup>30</sup>Bildschirm-Koordinaten sind zweidimensionale (x,y)-Koordinaten, während Window-Koordinaten auch die z-Koordinate beinhaltet (x,y,z).

### 3.4 Pixelverarbeitung (Fragment-Shader)

Bei der Pixelverarbeitung werden die diskreten Fragmente aus der Rasterisierung mit einem Fragment-Shader verarbeitet. So werden im einfachsten Fall die endgültigen Farben der Fragmente festgelegt oder auch Texturen an das Modell angeheftet.

Nach dem Fragment-Shader wird überprüft, ob sich im dreidimensionalen Raum zwei oder mehrere Fragmente überlappen. Für diesen Fall gibt es einen z-Buffer, der für jedes Fragment einen z-Wert gespeichert hat. Dieser z-Wert repräsentiert wie weit das Fragment von der Kamera entfernt ist. Ein z-Buffer Algorithmus entscheidet dann, welches Fragment näher zur Kamera liegt und welche Farbe das Fragment letztendlich haben soll. Dieser Algorithmus ist simpel und hat den Vorteil, dass die render Reihenfolge der Primitiven trivial ist. Probleme gibt es jedoch bei transparenten Objekten. Hier ist die Reihenfolge für eine korrekte Darstellung wichtig[Moeller (2019) S.21f., 1]. Für eine ausführliche Erklärung für den Umgang mit transparenten Objekten wird auf Möller, Kapitel 5.5 verwiesen[Moeller (2019) S.148ff., 1].

## 4 Forschungsstand

Augmented Reality ist ein nachgefragtes Thema und wird in unterschiedlichen Bereichen wie Industrie, Medizin oder in der Computerspiel-Branche untersucht[24][20][10]. Es gilt Probleme zu lösen, die sowohl Hardware als auch Software betreffen. Die Hardwareentwicklung betrifft dabei hauptsächlich die Mobilität bei der Nutzung. Beispielsweise funktioniert die HTC Vive nur mit einem Computer, der mit einem Kabel am Headset verbunden ist. Die Microsoft Holo Lens hingegen benötigt eine gute Internetverbindung über Wi-Fi. In der Software ist es eine Herausforderung aus den vielfältigen Software-Paketen eigene AR Anwendungen zu entwickeln. Nicht jede Software Bibliothek ist miteinander kompatibel, sodass die Entwicklung eine lange Einarbeitungszeit benötigt[Vgl. 20].

Ein weiteres Problem in der Nutzung von AR ist die Beleuchtung der Umgebung. In Fabriken oder Lagern ist es dunkel und Lichter kommen aus verschiedenen Richtungen, während es im Freien die Wetterbedingungen wie starke Sonne oder Wolken, die störende Schatten verursachen, zu beachten gilt. Die unterschiedlichen Lichtverhältnisse rufen Ungenauigkeiten z.B. beim Tracking hervor.

### 4.1 Tracking

Der Begriff Tracking beschreibt die kontinuierliche Verfolgung von Positions- und Rotationsdaten, die von Eingabegeräten (z.B. VR Controller) oder Sensoren (z.B. *Inertial Measurement Units* (Gyroskop und Beschleunigungssensoren)) erfasst werden. Die Bewegung eines starren Körpers kann

„durch die Angabe von sechs Werten (drei Koordinaten als Position und drei Winkel zur Beschreibung der Orientierung) für jeden Zeitschritt spezifiziert werden“[7, Dörner (2019) S.119f.].

Die Werte für die Position und Orientierung für ein Objekt werden als *Freiheitsgrade (engl. Degrees of Freedom – DOF)* bezeichnet. Beim Tracking ist es das Ziel, die sechs Freiheitsgrade für die Translation und Rotation der Kamera zu bestimmen bzw. zu schätzen[7]. Es gibt zwei Tracking Systeme. Beim *Inside-Out-Tracking* befinden sich Sensoren im Objekt, das getrackt werden soll, während beim *Outside-In-Tracking* sich die Sensorik in der Umgebung befinden und das Objekt von außen getrackt wird. In dieser Arbeit wird ein Inside-Out-Tracking verwendet, da sich die Sensorik im Smartphone bzw. Tablet befinden.

#### 4.1.1 Koordinatensysteme

Um eine Bestimmung bzw. Schätzung der Translationen und Rotationen durchzuführen, können zwei Koordinatensysteme herangezogen werden. Ein Kamerakoordinatensystem und ein Objektkoordinatensystem, womit die relativen Transformationen zwischen den Koordinatensystemen bestimmt werden können. Weiterhin gibt es die Möglichkeit, dass für alle Objekte im Raum ein Koordinatensystem (Weltkoordinatensystem) verwendet wird. Voraussetzung für das Tracking ist, dass die Transformationen zwischen den Objekten bekannt sind. Dann kann die Transformation zwischen der Kamera und das Weltkoordinatensystem geschätzt werden. Sind einzelne Transformationen von Objekten im Weltkoordinatensystem nicht bekannt, kann es auch Mischformen geben Zitat [7, Dörner (2019) S.124f.].

## 4.2 Kamera-basiertes Tracking

Kamera-basiertes Tracking nutzt Informationen zu Objekten aus dem Video Datenstrom, um die relative Position und Orientierung der Objekte zur Kamera zu bestimmen. Hartley und Zisserman bezeichnen diese als (extrinsische Kameraparameter) [8, Hartley, Zisserman (2003) S.156]. Für das Kamera-basierte Tracking wird zwischen Marker-basierten und Marker-less Tracking unterschieden.

Beim Marker-basierten Tracking werden schwarz-weiß Marker (sogenannte Kanji und Hiro Marker) (Verweis) mit einfachen geometrischen Formen platziert und erkannt, sodass diese als Orientierungshilfen fungieren. Eine Anwendung von NGIN-Mobility arbeitet mit Markern, die auf dem Boden platziert werden können <sup>31</sup>. Da die Marker in dauerhaft in der Umgebung platziert werden müssen, ist die Trackingmöglichkeit für die Nutzung im Freien ungeeignet.

Die zweite Möglichkeit erfasst über Algorithmen der Computer Vision Merkmale in der realen Umgebung. Diese Art wird als *marker-less AR* bezeichnet. Merkmale können geometrisch basiert sein (z.B. Kanten, Formen wie Vierecke) oder es werden Detektoren wie z.B. *SIFT* (engl. *Scale Invariant*

---

<sup>31</sup><https://www.logistik-watchblog.de/startups/1479-insider-navigation-ar-loesung-lagerhalle.html>

*Feature Transform)[14]* oder *SURF (engl. Scale Invariant Feature Transform)[4]* verwendet. Auch die aus der Robotertechnik bekannte *SLAM (engl. Simultaneous Localization and Mapping)[6] [3]* Methode wird genutzt. Dabei werden entweder kamerabasiert (Visual SLAM) nach Merkmalen gesucht oder es kommen Sensoren zur Generierung von Tiefeninformationen zum Einsatz, z.B. Kinect <sup>32</sup> oder *LIDAR-Sensoren (engl. light detection and ranging)[13]*. Vorteil dieser Methode ist, dass gleichzeitig eine 3D-Karte des Raumes generiert wird, die immer wieder zur Positions- und Rotationsbestimmung verwendet werden kann.

#### 4.2.1 GPS Tracking

Im Außenbereich wird bei AR auch GPS für das Tracking herangezogen. Dabei sind Positionsabweichungen von bis zu 10 Metern möglich, sodass eine genaue Bestimmung der extrinsischen Kameraparameter beeinträchtigt wird. Um die Tracking-Genauigkeit zu erhöhen, gibt es mehrere Methoden. *DGPS (engl. Differential GPS)* verbessert GPS-Signale, indem es ein Korrektursignal durch eine ortsfeste Referenzstation mit bekannter Lokalisierung berechnet. Da es in Deutschland lediglich acht solcher Stationen gibt und einige Anbieter nur kommerziell die Daten bereitstellen, ist diese Methode nicht für diese Arbeit geeignet<sup>33</sup> <sup>34</sup>. Eine weitere bekannte Möglichkeit bietet *SBAS (engl. Satellite Based Augmentation System)*, bei dem mehrere geostationäre Satelliten das GPS Signal auf bis zu einem Meter Genauigkeit zu verbessern [7].

Platinsky und seine Koautoren[18] erstellen für ein besseres Tracking bei fehlender GPS Genauigkeit ein 3D-Modell der Umgebung. Bei der anschließenden AR Nutzung in diesem Gebiet wird auf dem Smartphone SLAM betrieben. Die Daten vom Smartphone werden mit der 3D-Karte verglichen, um ein genauereres Tracking durchzuführen. Ein ähnliches System wäre für die Anwendung in dieser Arbeit denkbar, da eine große Datenmenge von Bildern des Geländes vorhanden ist. Über Structure from Motion Methoden, kann mit den Bildern eine große 3D-Karte erstellt werden.

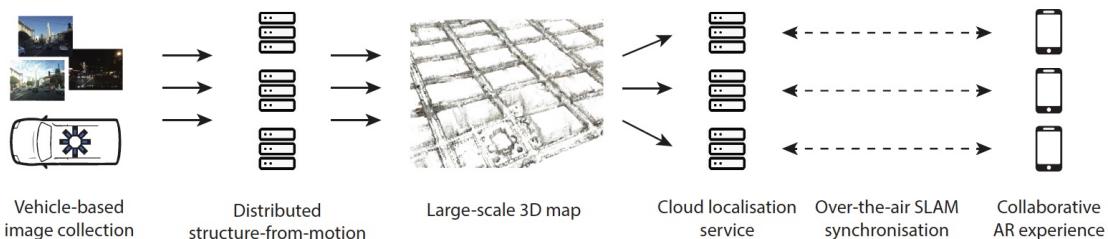


Abbildung 9: Das Grundprinzip der Methode von Platinsky.

<sup>32</sup><https://developer.microsoft.com/de-de/windows/kinect/>

<sup>33</sup><https://www.heise.de/newsticker/meldung/Differential-GPS-und-WLAN-RTT-Praezise-Ortung-mit-Android-P-4046935.html>

<sup>34</sup>Liste von DGPS-Sendern: <https://www.ndblist.info/datamodes/worldDGPSdatabase.pdf>

### 4.3 Software Entwicklung

Für Augmented Reality Anwendungen gibt es im Smartphone und Tablet Bereich mehrere Softwarepaket (SDK (*engl. Software Development Kit*)). Die bekanntesten sind ARKit von Apple, das nur auf iOS Endgeräten läuft und ARCore von Google, das für Android Endgeräte entwickelt wird.<sup>35</sup> <sup>36</sup> Andere SDK's wie Vuforia<sup>37</sup>, Wikitude<sup>38</sup>, ARToolkit<sup>39</sup> oder Lightship<sup>40</sup> sind unabhängig vom Endgerät nutzbar.

## 5 Fragestellungen und Methodik

Für die Entwicklung der AR Anwendung können die vorgestellten SDK's genutzt werden. Diese haben Vor- und Nachteile, die in dieser Arbeit erörtert werden, um daraus die passende SDK für dieses Projekt auszuwählen. Die Methode von Platinsky und seinen Koautoren[18], in der das Tracking durch eine vorgefertigte 3D-Karte der Stadt verbessert wird, wird in dieser Arbeit weiter untersucht. Im Beispiel von Platinsky ist die GPS Genauigkeit durch die hohen Gebäude in der Großstadt stark beeinträchtigt. Das Gelände in Villingen befindet sich am Stadtrand und die Gebäude könnten keinen großen Einfluss auf die GPS Genauigkeit haben. Daher wird untersucht, ob der Mehraufwand, der durch die Erstellung der 3D Map und des SLAM Systems über eine Cloud entsteht, für eine mittelgroße Stadt sinnvoll ist. Hierfür wird die GPS Genauigkeit mit und ohne dieser Methode gemessen und miteinander verglichen.

Hinzu kommt die Frage, wie detailreich die 3D-Karte und die SLAM Daten des Smartphones für ein genaues Tracking sein müssen. Ein Problem der Methode ist, dass eine konstante Verbindung zum Internet bestehen muss, um die SLAM Daten mit der 3D-Karte zu vergleichen. Die 4G Verbindung war nicht ausreichend schnell [18, sinngemäß aus]. Um an diesem Problem anzuknüpfen, wird untersucht, wie stark die Qualität der 3D-Karte reduziert werden kann, ohne die Vorteile beim Tracking zu verlieren. Eine Optimierung der Dateigröße der 3D-Karte wird im Paper zwar erwähnt, jedoch nicht umgesetzt. So wird in dieser Arbeit ein Experiment durchgeführt, bei der drei Qualitätsstufen (Hohe Details, mittlere Details, grobe Details) auf die GPS Genauigkeit untersucht werden.

Je nachdem wie die Ergebnisse der Experimente ausgehen, ist es für die Anwendung sinnvoll dem Nutzer zwei Möglichkeiten anzubieten: Eine hohe Tracking Genauigkeit bei hohem Datenverbrauch und eine möglicherweise geringe Tracking Genauigkeit, bei dem diese Methode nicht genutzt wird.

---

<sup>35</sup><https://developer.apple.com/augmented-reality/>

<sup>36</sup><https://developers.google.com/ar>

<sup>37</sup><https://www.ptc.com/en/products/vuforia>

<sup>38</sup><https://www.wikitude.com/>

<sup>39</sup><http://www.hitl.washington.edu/artoolkit/>

<sup>40</sup><https://lightship.dev/>

## 6 Vorbereitung der 3D Modelle in Blender

Einige 3D Modelle haben durch die umliegende Vegetation unvollständige oder mit der Vegetation verschmolzene Meshes. In der Projektarbeit des Wintersemesters 2020/21 wird das Problem näher beschrieben [12].

### 6.1 Lückenhafte Wände füllen

Das Gebäude 2 des Lyautey Geländes hat eine Haushälfte, die mit einem Baum verschmolzen ist und es fehlt daher ein Teil der Wand. Da das Gebäude symmetrisch und die gespiegelte Seite fehlerfrei ist, wird ein Teil dieser Wand als Lückenfüller eingesetzt. Im folgenden wird die Nachbearbeitung an diesem Modell beschrieben.



Abbildung 10: Ein Baum behinderte bei den Aufnahmen die Sicht auf die Wand, sodass diese mit dem Baum verschmolzen ist.

Zunächst wird das Modell kopiert. Anschließend werden die fehlerhaften Polygone im *Edit-Mode* ausgewählt und gelöscht. Beim kopierten Modell werden alle Polygone gelöscht, bis auf die Wand, die eingesetzt werden soll. Diese wird mit dem *Mirror-Modifier* gespiegelt. Das gespiegelte Modell wird an die Position der fehlerhaften Wand gesetzt und die beiden Modelle werden mit *STRG + J* als ein Modell zusammengefügt. Die Lücke zwischen den beiden Modellen wird anschließend händisch mit neuer Geometrie gefüllt. Dieser Prozess ist sehr zeitaufwendig, da je nach Polygonanzahl viele Kanten einzeln ausgewählt werden müssen. Daher wird zunächst ein *Decimate-Modifier* benutzt, um die Anzahl an Polygone zu reduzieren. Bei diesem Gebäude wird eine Reduzierung um 70% vorgenommen, sodass sich die gesamte Anzahl an Polygonen von über 2.5 Millionen auf ca 750.000 Polygone verringert, ohne einen starken Verlust im Detailgrad festzustellen.

Um die Lücken zu schließen werden benachbarte Kanten im *Edit-Mode* ausgewählt und mit der *F-Taste* mit neuen Flächen aufgefüllt. An einigen Stellen sind kleine Polygone vorhanden, bei dem dieser Prozess erschwert wird. Diese werden gelöscht und mit größeren Flächen ersetzt.

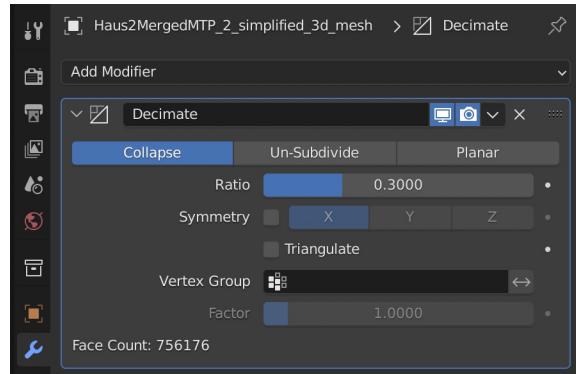


Abbildung 11: Der Decimate-Modifier reduziert die Anzahl an Polygone. Ein Wert von 0.3 bedeutet, dass die Polygonenanzahl 30% der ursprünglichen Polygonanzahl entspricht.

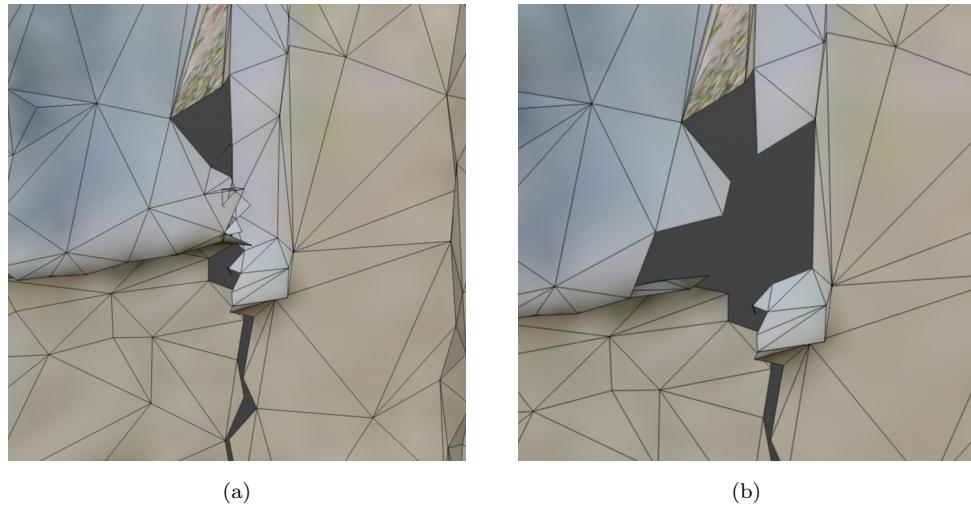


Abbildung 12: Kleine Polygone erschweren das Markieren von Kanten (a). Das entstandene Loch wird mit größeren Flächen gefüllt (b).

Ist der Füllprozess abgeschlossen, ist die Naht dennoch erkennbar (siehe Abbildung 13), da die Textur auf den neuen Polygone verzerrt wird. Aufgrunddessen werden im letzten Schritt im UV-Editing Tab die UV-Koordinaten der neuen Polygone neu vergeben, sodass die Texturen den umliegenden Flächen entsprechen. Hierfür werden die neuen Polygone im *Edit-Mode* markiert. Im *UV-Editor* werden mit dem Befehl *U + Unwrap* die markierten Polygone bestmöglich auf die 2D Textur gelegt.<sup>41</sup> Die Polygone werden auf der Textur so skaliert, bewegt und rotiert, dass sie eine ähnliche Textur wie die umliegenden Flächen besitzen. Zur Hilfe werden die umliegenden Flächen markiert und deren UV-Koordinaten auf der Textur verglichen. So wird sichergestellt, dass die Farbunterschiede an den Kanten nicht zu stark abweichen. Das fertige Modell ist im Vorher-

<sup>41</sup><https://docs.blender.org/manual/en/latest/modeling/meshes/editing/uv.html>, zuletzt augerufen am 29.05.2022



Abbildung 13: Das zusammengesetzte Gebäude.

Nachher-Vergleich in Abbildung 15 zu sehen.

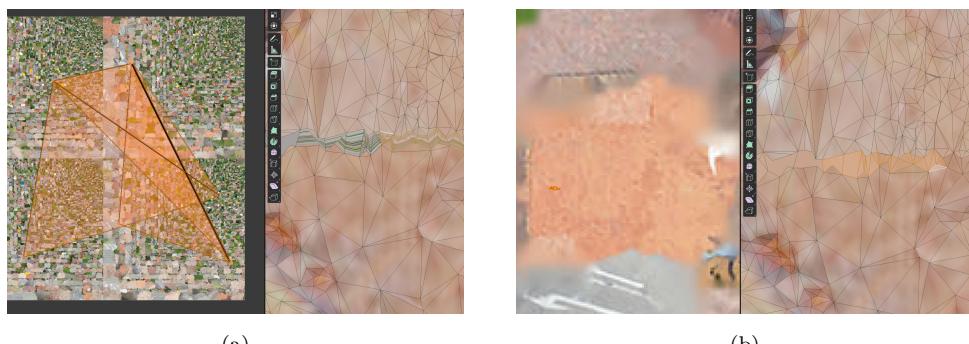


Abbildung 14: Die Polygone werden markiert(a), unwrapped und anschließend auf eine geeignete Fläche auf der Textur gelegt(b).

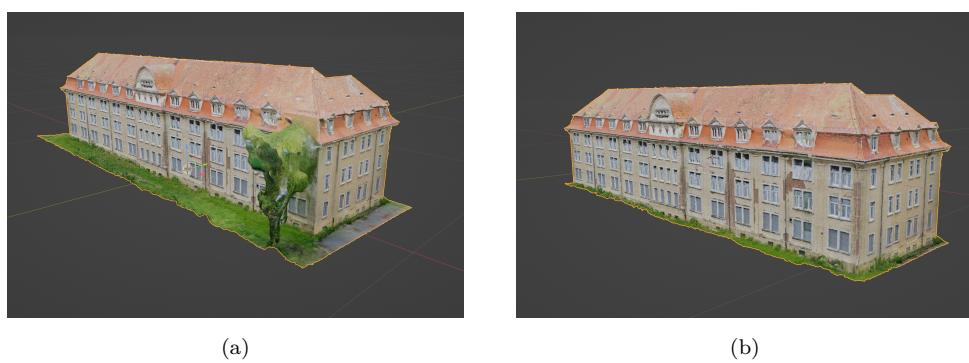


Abbildung 15: Das Mannschaftsgebäude vor(a) und nach der Bearbeitung(b).

## 6.2 Export und Import

Damit Unity die 3D Modelle korrekt darstellen kann, werden Anpassungen in Blender vorgenommen:

- Vor dem Export werden Texturen in einem Ordner gespeichert. In Blender muss dafür die Option unter *File>External Data>Automatically Pack Resources* ausgeschaltet sein. Anschließend wird mit *File>External Data>Unpack Resources* ein Ordner im Projekt mit den Namen *textures* erstellt, in dem sich die Texturen des Projekts befinden. Nun kann Unity die fehlenden Materials und Texturen im Projekt-Ordner finden,
- Unity kann die Texturen nur anwenden, wenn die Textur direkt im *Shader Node* als *Base Color* verbunden ist. Dazwischen befindliche Knoten funktionieren nur in Blender und müssen entfernt werden,
- die in Blender genutzten *Modifier*<sup>42</sup> werden angewendet,
- damit die Textur korrekt dargestellt werden, werden die Richtungen der *Normalen*<sup>43</sup> überprüft, ob diese in die richtige Richtung zeigen. Die Normalen müssen zur korrekten Darstellung nach außen zeigen. In Blender können die Normalen im *Edit Mode* im Viewport Overlay angezeigt und unter *Mesh>Normals* angepasst werden,
- Blender und Unity haben unterschiedliche Koordinatensysteme. In Blender zeigt die Z-Achse nach oben und die Y-Achse in die Tiefe, während in Unity die Y-Achse nach oben und die Z-Achse in die Tiefe zeigt. Damit die Modelle aus Blender mit der korrekten Orientierung dargestellt werden, wird das Modell in Blender um -90° gedreht. Mit *STRG + A* und *Rotation* wird die Rotation eingebettet,
- beim Export wird das Export-Format *.fbx* benötigt.

Jetzt wird das 3D Modell als *.fbx* Datei in Unity importiert. Im *Inspector*-Fenster der *.fbx* Datei werden unter Materials mit *Extract Materials* Materials in Unity generiert, die die Texturen beinhalten. Der Texturen-Ordner muss im gleichen Ordner liegen. Werden die Texturen nicht gefunden, so können diese händisch in das Projekt eingefügt werden. Um die Texturen den von Unity generierten Materials zu geben, wird im *Inspector* die Textur in das Feld neben *Albedo* eingefügt werden.

---

<sup>42</sup><https://docs.blender.org/manual/en/latest/modeling/modifiers/index.html>

<sup>43</sup><https://docs.blender.org/manual/en/latest/modeling/meshes/editing/mesh/normals.html?highlight=normals>



## Abbildungsverzeichnis

1	Das 3D Modell des SABA Hauptgebäudes in der 3D Karte aus den Drohnen-Aufnahmen.	10
2	Eine Übersicht der Gebäude auf dem Lyautey-Gelände.	11
3	Eine Übersicht der Gebäude auf dem Mangin-Gelände.	12
4	Das Reality-Virtuality Kontinuum nach Milgram[15].	15
5	Kanji und Hiro Marker haben einfache geometrische Formen, Texte oder Buchstaben zur Erkennung in der Szene.	17
6	Die dreidimensionale Szene (links) und die zweidimensionale Sicht der virtuellen Kamera. Wird das Objekt perspektivisch gerendert, gibt es ein Raumvolumen (frustum). Es werden nur die Objekte gerendert, die sich innerhalb dieses Volumens befinden. So wird der rote Donut nicht und das blaue Objekt abgeschnitten dargestellt[1, Moeller (2019)].	20
7	Die Basis der GRP, die in weitere Sub-Pipelines eingeteilt und parallel ausgeführt werden können [1].	20
8	Die Primitiven werden in eine 2D-Ebene projiziert(a). Befindet sich ein Pixel innerhalb des Primitiven, werden diskrete Fragmente gebildet, die im in der Pixelverarbeitung weitergeschickt werden(b) [22, zuletzt aufgerufen am 11.07.2022].	23
9	Das Grundprinzip der Methode von Platinsky.	26
10	Ein Baum behinderte bei den Aufnahmen die Sicht auf die Wand, sodass diese mit dem Baum verschmolzen ist.	28
11	Der Decimate-Modifier reduziert die Anzahl an Polygonen. Ein Wert von 0.3 bedeutet, dass die Polygonenanzahl 30% der ursprünglichen Polygonanzahl entspricht.	29
12	Kleine Polygone erschweren das Markieren von Kanten (a). Das entstandene Loch wird mit größeren Flächen gefüllt (b).	29
13	Das zusammengesetzte Gebäude.	30
14	Die Polygone werden markiert(a), unwrapped und anschließend auf eine geeignete Fläche auf der Textur gelegt(b).	30
15	Das Mannschaftsgebäude vor(a) und nach der Bearbeitung(b).	30

## **Eidesstattliche Erklärung**

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Thesis selbständig und ohne unzulässige fremde Hilfe angefertigt habe. Alle verwendeten Quellen und Hilfsmittel sind angegeben.

Villingen-Schwenningen, 31.08.2022

---

Oliver Kusch

## Literatur

- [1] Tomas Akenine-Moeller, Eric Haines und Naty Hoffman. *Real-Time Rendering*. A K Peters/CRC Press, Jan. 2019. ISBN: 9781315365459. DOI: [10.1201/9781315365459](https://doi.org/10.1201/9781315365459).
- [2] Ronald T. Azuma. „A Survey of Augmented Reality“. In: *Presence: Teleoperators and Virtual Environments* 6 (4 Aug. 1997), S. 355–385. ISSN: 1054-7460. DOI: [10.1162/pres.1997.6.4.355](https://doi.org/10.1162/pres.1997.6.4.355).
- [3] T. Bailey und H. Durrant-Whyte. „Simultaneous localization and mapping (SLAM): part II“. In: *IEEE Robotics and Automation Magazine* 13 (3 Sep. 2006), S. 108–117. ISSN: 1070-9932. DOI: [10.1109/MRA.2006.1678144](https://doi.org/10.1109/MRA.2006.1678144).
- [4] Herbert Bay u. a. „Speeded-Up Robust Features (SURF)“. In: *Computer Vision and Image Understanding* 110 (3 Juni 2008), S. 346–359. ISSN: 10773142. DOI: <https://doi.org/10.1016/j.cviu.2007.09.014>.
- [5] Satyan Coorg und Seth Teller. „Real-time occlusion culling for models with large occluders“. In: ACM Press, 1997, 83–ff. ISBN: 0897918843. DOI: [10.1145/253284.253312](https://doi.org/10.1145/253284.253312).
- [6] H. Durrant-Whyte und T. Bailey. „Simultaneous localization and mapping: part I“. In: *IEEE Robotics and Automation Magazine* 13 (2 Juni 2006), S. 99–110. ISSN: 1070-9932. DOI: [10.1109/MRA.2006.1638022](https://doi.org/10.1109/MRA.2006.1638022). URL: <https://ieeexplore.ieee.org/document/1638022/>.
- [7] Ralf 'Dörner u. a. *Virtual und Augmented Reality (VR/AR)*. Bd. 2. Springer Berlin Heidelberg, 2019. ISBN: 978-3-662-58860-4. DOI: [10.1007/978-3-662-58861-1](https://doi.org/10.1007/978-3-662-58861-1).
- [8] Richard Hartley und Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Bd. 2. Cambridge University Press, März 2004. ISBN: 9780521540513. DOI: [10.1017/CBO9780511811685](https://doi.org/10.1017/CBO9780511811685).
- [9] Jan Herling und Wolfgang Broll. *Markerless Tracking for Augmented Reality*. Springer New York, 2011, S. 255–272. DOI: [10.1007/978-1-4614-0064-6\\_11](https://doi.org/10.1007/978-1-4614-0064-6_11).
- [10] Bo-Chen Huang u. a. „ARBIN: Augmented Reality Based Indoor Navigation System“. In: *Sensors* 20 (20 Okt. 2020), S. 5890. ISSN: 1424-8220. DOI: [10.3390/s20205890](https://doi.org/10.3390/s20205890).
- [11] John Isidoro, Alex Vlachos und Chirs Brennan. „Rendering ocean water“. In: *Direct3D ShaderX: Vertex and Pixel Shader Tips and Tricks*, Wordware (2002).
- [12] Oliver Kusch u. a. *Kreation von 3D Modellen der alten Kaserne in Villingen-Schwenningen - Eine Analyse der Arbeitspipeline zur Erstellung von 3D Modellen mittels Pix4d*. Hochschule Furtwangen, 2021.
- [13] Weiquan Liu u. a. „Learning to Match 2D Images and 3D LiDAR Point Clouds for Outdoor Augmented Reality“. In: IEEE, März 2020, S. 654–655. ISBN: 978-1-7281-6532-5. DOI: [10.1109/VRW50115.2020.00178](https://doi.org/10.1109/VRW50115.2020.00178).

- [14] David G. Lowe. „Object recognition from local scale-invariant features“. In: IEEE, 1999, 1150–1157 vol.2. ISBN: 0-7695-0164-8. DOI: [10.1109/ICCV.1999.790410](https://doi.org/10.1109/ICCV.1999.790410).
- [15] Paul Milgram u. a. „Augmented reality: a class of displays on the reality-virtuality continuum“. In: Hrsg. von Hari Das. Dez. 1995, S. 282–292. DOI: [10.1117/12.197321](https://doi.org/10.1117/12.197321).
- [16] Jason L. Mitchell. „Real-Time Synthesis and Rendering of Ocean Water“. In: *ATI Research Technical Report* (2005), S. 121–126.
- [17] Alfred Nischwitz u. a. *Computergrafik und Bildverarbeitung*. Vieweg+Teubner Verlag, 2012. ISBN: 978-3-8348-1304-6. DOI: [10.1007/978-3-8348-8323-0](https://doi.org/10.1007/978-3-8348-8323-0).
- [18] Lukas Platinsky u. a. „Collaborative Augmented Reality on Smartphones via Life-long City-scale Maps“. In: (Nov. 2020). URL: [http://arxiv.org/abs/2011.05370](https://arxiv.org/abs/2011.05370).
- [19] Andreas Reich u. a. *Photogrammetric recordings of the SABA area*. Hochschule Furtwangen University, 2020.
- [20] Gian Maria Santi u. a. „Augmented Reality in Industry 4.0 and Future Innovation Programs“. In: *Technologies* 9 (2 Apr. 2021), S. 33. ISSN: 2227-7080. DOI: [10.3390/technologies9020033](https://doi.org/10.3390/technologies9020033).
- [21] Andre Schaefer u. a. *NISABA: Photogrammetrische Erfassung historischer Gebäudekomplexe in Villingen-Schwenningen*. Hochschule Furtwangen University, 2021.
- [22] Scratchapixel. *Rasterization: a Practical Implementation*. 2022. URL: <https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/overview-rasterization-algorithm>.
- [23] Maximilian Speicher, Brian D. Hall und Michael Nebeling. „What is Mixed Reality?“ In: ACM, Mai 2019, S. 1–15. ISBN: 9781450359702. DOI: [10.1145/3290605.3300767](https://doi.org/10.1145/3290605.3300767).
- [24] Sud Sudirman und Abdennour El-Rhalibi. „Improving Camera Pose Estimation for Indoor Marker-less Augmented Reality“. In: IEEE, Okt. 2015, S. 994–999. ISBN: 978-1-5090-0154-5. DOI: [10.1109/CIT/IUCC/DASC/PICOM.2015.150](https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.150).
- [25] *Unity Dokumentation, About AR Foundation*. 2021. URL: <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual/index.html> (besucht am 27.07.2022).