Landesamt für Digitalisierung,
Breitband und Vermessung

Hochschule München, Fakultät für Geoinformation

Masterarbeit

Zur Erlangung des akademischen Grades Master of Engineering

Entwicklung einer Augmented Reality-App zum adaptiven Tracken physischer Kartenausschnitte

Studiengang Geomatik Sommersemester 2024

vorgelegt von

Maximilian Kinzel

Betreuer: Prof. Dr. Thomas Abmayr und Prof. Dr. Markus Oster

Kooperation: Landesamt für Digitalisierung, Breitband und Vermessung

Betreuer: Dipl. Ing. (FH) Thomas Meier (LDBV)

München, 8. Oktober 2024

Zusammenfassung

Diese Masterarbeit beschäftigt sich mit dem adaptiven Tracken physischer Kartenausschnitte und der virtuellen Erweiterung mit Geobasisdaten in einer Augmented Reality (AR) Umgebung. Dabei werden die Geobasisdaten der OpenData-Plattform der bayerischen Vermessungsverwaltung dynamisch eingebunden. Ein innovatives Konzept zur interaktiven Darstellung von Geodaten soll entstehen. Es erfolgt die Entwicklung und Implementierung eines Prototyps einer Android-Applikation. Mit dieser Applikation hat der Anwender die Möglichkeit, jeden beliebigen physischen Kartenausschnitt des Testgebiets mit seinem mobilen Endgerät aufzunehmen und diesen nach einem automatisierten Verarbeitungsprozess mit virtuellen Geobasisdaten anzureichern.

Die Umsetzung dieses Prototyps einer App erfolgt dabei in der Game Engine Unity, welche für die spezifischen Anforderungen durch das Plug-in OpenCV+ und die AR Software Development Kits Vuforia sowie ARCore erweitert wird. Die digitalen Geobasisdaten werden dabei dynamisch über die Web Map Services (WMS) der OpenData-Plattform der bayerischen Vermessungsverwaltung bezogen, ebenso wird ein neuartiges 3D-Mesh der Anwendung hinzugefügt. Dieses wurde vom Landesamt für Digitalisierung, Breitband und Vermessung (LDBV) zur Verfügung gestellt. Der Raum Würzburg wird als Testgebiet für den Prototyp gewählt und dient als Referenzkarte. Das Integrieren dieser Komponenten resultiert in einer Android-App, die das verzerrungsfreie Aufnehmen, das Verarbeiten und das digitale Erweitern beliebig vieler verschiedener Kartenausschnitte innerhalb des Testgebiets ermöglicht.

In der praktischen Umsetzung werden das Potenzial und die Effizienz einer solchen Anwendung sichtbar. Adaptives Tracking kann besonders nützlich für Anwendungen in lokalen Kontexten, wie zum Beispiel in städtischen Gebieten, Naturparks oder auf Messegeländen sein. Die Ergebnisse dieses Projekts verdeutlichen, dass WMS-Dienste hervorragend für Geovisualisierungsanwendungen geeignet sind und nur wenige Einschränkungen aufweisen.

Abstract

This master's thesis deals with the adaptive tracking of physical maps and the virtual extension with geodata in an Augmented Reality (AR) environment. The geodata of the OpenData platform of the Bavarian surveying administration is dynamically integrated. An innovative concept for the interactive display of geodata will be developed. The implementation takes place in an Android app. With this application, the user has the option of recording any physical map section of the test area with his mobile device and enhancing it with virtual geodata following an automated processing procedure.

This prototype app is being implemented in the Unity game engine, which is being extended for the specific requirements using the OpenCV+ plug-in and the AR software development kits Vuforia as well as ARCore. The digital geodata is retrieved dynamically via the web map services (WMS) of the Bavarian Surveying Administration's OpenData platform, and a new type of 3D mesh is also added to the application. It was provided by the State Office for Digitalization, Broadband and Surveying. The area of Würzburg was chosen as a test area for the prototype and serves as a reference map. The integration of these components results in an Android app that enables the distortion-free recording, processing and digital enhancement of any number of different map sections within the test area.

The practical implementation reveals the potential and efficiency of such an application. Adaptive tracking can be particularly useful for applications in local contexts, such as urban areas, nature parks or exhibition grounds. The results of this project show that web map services are ideally suited for geovisualization applications and have few limitations.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Hintergrund der Augmented Reality und deren Anwendungen . 1
- 1.2 Motivation 2
- 1.3 Stand der Forschung 4
- 1.4 Beitrag der Arbeit 6

2 Grundlagen **8**
- 2.1 Grundlegende Begriffe 8
  - 2.1.1 Mixed Reality 8
  - 2.1.2 Augmented Reality 8
  - 2.1.3 Unity 9
  - 2.1.4 Vuforia 9
  - 2.1.5 Open Source Computer Vision Library (OpenCV) 10
- 2.2 Funktionsweise AR 10
  - 2.2.1 Erfassungsmethoden 12
  - 2.2.2 Einblick ARCore 13
- 2.3 Geobasisdaten 15
- 2.4 Geodateninfrastruktur 17
  - 2.4.1 Inspire 17
  - 2.4.2 Geodateninfrastruktur (GDI) 18
- 2.5 Computer Vision 19
  - 2.5.1 Template Matching 19

3 Werkzeuge **23**
- 3.1 Opendata 23
- 3.2 Vorbereitung 3D-Mesh 24
- 3.3 Vorbereitung Referenzkarte 25

4 Vorgehensweise **28**
- 4.1 Projektvorbereitung 29

4.1.1 Untersuchungsgebiet 29
4.1.2 Software 29
4.2 Computer Vision Prozessierung 32
4.2.1 Einladen der Referenzkarte 32
4.2.2 Aufnahme eines Screenshots 32
4.2.3 Template Matching 35
4.3 3D-Verarbeitung 37
4.3.1 3D-Meshimport 37
4.3.2 Bestimmung 3D-Meshausschnitt 38
4.4 AR Workflow 41
4.5 Netzwerkprogrammierung WMS-Request 42
4.6 User Interface 43

5 Ergebnisse **48**
5.1 Versuchsfeld 48
5.2 Darstellung Funktionsweise 48
5.3 Leistungsfähigkeit auf dem Endgerät 63
5.3.1 Leistung während der AR-Darstellung 63
5.3.2 Laufzeitmessungen 63

6 Diskussion **65**
6.1 Problematiken bei der Umsetzung 65
6.2 Ergebnisse 68
6.3 Fazit 69

Anhang **71**

Literaturverzeichnis **72**

Abbildungsverzeichnis

1.1 Schweizer Landeskarte als Marker nach (Loesch et al., 2015) . . 4
1.2 AR-App im Nationalpark nach (Yonov and Petkov, 2020) 5
1.3 AR-GIS-System nach (Fenais et al., 2019) 6

2.1 Virtualitäts Kontinuum nach (Milgram and Kishino, 1994) . . . 8
2.2 Feature Tracking nach (Tsotsos and Ballan, 2020) 13
2.3 Visual-Inertial Sensor Fusion nach (Tsotsos and Ballan, 2020) . 14
2.4 Exemplarisches digitales Geländemodell nach (Bayerisches Landesamt für Digitalisierung, 2019) 15
2.5 Übersicht Inspire nach (Illert, 2009) 17
2.6 Geometrie des Template Matching nach (Burger, 2006) 19
2.7 Zielbild mit Score-Werten nach (Girod, 2013) 22

3.1 Opendata der Bayerischen Vermessungsverwaltung nach (Vermessungsverwaltung, 2024b) 23
3.2 Vorbereitung 3D-Mesh in 3ds Max 24
3.3 3D-Mesh in Infraworks 25
3.4 Referenzkarte des Interessengebiet Würzburg und Umland . . . 27

4.1 Systematische Darstellung der Arbeitsprozesse 28
4.2 Übersicht über die Unity Plug-in Packages 30
4.3 Vuforia Entwicklungsmöglichkeiten 31
4.4 Konfiguration der Referenzkarte 33
4.5 Aufbau der Szene mit ARCore 34
4.6 Vorgehen Template Matching 36
4.7 3D-Mesh im Unity Editor 37
4.8 Konfiguration des Shaders 39
4.9 Mesh mit transparenten Quadern 40
4.10 Aufbau der Szene mit Vuforia 41
4.11 User Interface nach dem Start 44
4.12 Bildschirm während des Template Matchings 45

4.13 Template Matching abgeschlossen 45
4.14 Bildschirm während der 3D-Verarbeitung 46
4.15 User Interface mit 3D-Funktionalität 46
4.16 User Interface ohne 3D-Funktionalität 47

5.1 App nach dem Start mit physischer Karte „Flusslauf“ 49
5.2 Aufnahme des Screenshots durch Einpassen der Karte „Flusslauf“ in das User Interface 49
5.3 Ergebnis Template Matching Karte „Flusslauf“ mit Angaben . . 50
5.4 Virtuelles digitales Orthophoto Karte „Flusslauf“ 50
5.5 Virtuelles digitales Geländemodell Karte „Flusslauf“ 51
5.6 Virtuelle digitale Orthokarte Karte „Flusslauf“ 51
5.7 Orthophoto Nahaufnahme physischen Karte „Flusslauf“ 52
5.8 Aufnahme der physischen Karte „Stadtzentrum“ 53
5.9 Aufnahme des Screenshots durch Einpassen der Karte „Stadtzentrum“ in das User Interface 54
5.10 Ergebnis Template Matching Karte „Stadtzentrum“ 54
5.11 Orthophoto Karte „Stadtzentrum“ 55
5.12 DGM Karte „Stadtzentrum“ 55
5.13 DOK Karte „Stadtzentrum“ 56
5.14 3D-Mesh Karte „Stadtzentrum“ 56
5.15 3D-Mesh Nahaufnahme Karte „Stadtzentrum“ 57
5.16 3D-Mesh Ansicht von Südwesten Karte „Stadtzentrum“ 57
5.17 Aufnahme der physischen Karte „Burg“ 58
5.18 Aufnahme des Screenshots durch Einpassen der Karte „Burg“ in das User Interface 59
5.19 Orthophoto Karte „Burg“ 59
5.20 DGM Karte „Burg“ 60
5.21 DOK Karte „Burg“ 60
5.22 3D-Mesh Karte „Burg“ 61
5.23 Nahaufnahme 3D-Mesh Karte „Burg“ 61
5.24 Mainufer unterhalb der Würzburg in 3D Karte „Burg“ 62
5.25 Schrebergärten unterhalb der Würzburg in 3D Karte „Burg“ . . 62

Formelverzeichnis

2.1 Summe der quadrierten Differenz 20
2.2 Normalisierte Summe der quadrierten Differenz 20
2.3 Kreuzkorrelation 20
2.4 Normalisierte Kreuzkorrelation 21
2.5 Korrelationskoeffizienten 21
2.6 Normalisierte Korrelationskoeffizienten 21

Tabellenverzeichnis

5.1 Zeitmessung in Sekunden 63
5.2 Downloadzeit WMS-Dienste in Sekunden 64

1 Einleitung

1.1 Hintergrund der Augmented Reality und deren Anwendungen

Die Technologie der Augmented Reality (AR) hat in den letzten Jahren erhebliche Fortschritte erzielt und konnte in die verschiedensten Anwendungsbereiche wie z. B. Bildung, Industrie, Unterhaltung und Medizin vordringen. Mit AR-Applikationen ist es möglich, digitale Informationen in der physischen Welt zu platzieren und damit eine interaktive, immersive Erfahrung für den Anwender zu schaffen. Während die ersten Phasen in der fortlaufenden Entwicklung von AR häufig statische Inhalte darstellten, hat sich der Fokus zunehmend auf Systeme verlagert, die in Echtzeit auf den Benutzer und seine Umgebung reagieren können. Diese Systeme bieten das Potenzial, personalisierte und kontextabhängige Erlebnisse zu schaffen, die über die Möglichkeiten statischer Darstellungen weit hinausgehen. Die Integration von Geodaten in Augmented Reality bietet eine Vielzahl an neuen Perspektiven für interaktive und kontextbezogene Anwendungen. Geodaten, insbesondere Geobasisdaten, stellen Informationen mit Raumbezug dar, sie eignen sich deshalb sehr gut dazu, digitale Inhalte präzise über die reale Welt zu legen. In AR-Anwendungen können Geodaten dazu verwendet werden, virtuelle Objekte oder Informationen basierend auf der aktuellen Position des Nutzers anzuzeigen.

Einen spezifischen Ansatz stellt außerdem die markerbasierte AR dar, bei der ein physischer Marker als Ankerpunkt dient, um virtuelle Objekte oder Informationen im Raum zu platzieren. Es gibt eine große Bandbreite an Einsatzmöglichkeiten, wie beispielsweise in der Navigation, in der AR-Nutzer durch Echtzeit-Überlagerungen der Streckenführung unterstützt werden. Auch im Tourismus können Geodaten genutzt werden, um historische Informationen und somit Geschichte für bestimmte Orte direkt erlebbar zu machen. Darüber hinaus können Geodaten in der städtischen Planung und im Bauwesen Anwendung finden, indem sie detaillierte Visualisierungen von geplanten Strukturen in ihrer realen Umgebung ermöglichen. Insgesamt kann die Nutzung von Geodaten

in AR relevante Informationen kontextualisieren und damit leichter zugänglich machen.

In dieser Masterarbeit soll das Konzept der adaptiven Augmented Reality auf amtliche Geobasisdaten angewendet werden, dabei wird ein besonderes Augenmerk auf das Tracken physischer Kartenausschnitte gelegt. Hierbei stellt sich die Frage, ob eine AR-Applikation entwickelt werden kann, die eine innovative Tracking-Methode beinhaltet, um physische Kartenausschnitte adaptiv zu erfassen, mit virtuellen Geobasisdaten zu verknüpfen und interaktiv zu präsentieren. Diese Arbeit stellt verschiedene Sensorik- und Datenverarbeitungstechnologien vor, die erforderlich sind, um die Umgebung und den Kontext des Nutzers in Echtzeit zu erfassen und entsprechende AR-Inhalte zu generieren.

1.2 Motivation

Es sollen neue Potentiale vorhandener Vermessungsdaten der OpenData-Plattform der bayerischen Vermessungsverwaltung aufgezeigt werden. Die Umsetzung einer Android-App unter der Verwendung von amtlichen Geobasisdaten in einer AR-Umgebung stellt eine innovative Art der Visualisierung dar, die beim Anwender das Verständnis und die Wahrnehmung von Inhalten verbessern kann.

Gewünscht ist die Entwicklung einer Android-Applikation, die mit einem mobilen Endgerät physische Kartenausschnitte aufnehmen und diese nach einem automatisierten Verarbeitungsprozess mit virtuellen Geodaten anreichern kann. Dabei stehen folgende Aspekte im Vordergrund zum einen die Flexibilität der Entwicklung und Nutzung von Augmented Reality-Anwendungen, zum anderen, wie bereits genannt, die Nutzung der OpenData-Plattform der bayerischen Vermessungsverwaltung (Vermessungsverwaltung, 2024b).

Die hohe Anpassungsfähigkeit eines adaptiven AR-Systems hilft Entwicklern Geovisualisierungsprojekte für ein bestimmtes Untersuchungsgebiet effizient zu erstellen. Die Einbindung von Web Map Services (WMS) vereinfacht den Entwicklungsprozess und bietet gegenüber dem herkömmlichen Einbinden von Geodaten noch weitere Vorteile wie, z. B. Aktualität und reduzierter Speicherbedarf. Die Nutzer haben die Möglichkeit mit einer einzigen Applikation eine Fülle an Karten zu tracken und virtuell zu erweitern. Diese Flexibilität ist besonders wichtig, um individuelle Benutzerbedürfnisse und unterschiedliche Anwendungsfälle effektiv abzudecken. Die Möglichkeit, AR-Inhalte dynamisch zu mo-

difizieren und zu erweitern, eröffnet ein breites Spektrum an kreativen und funktionalen Einsatzmöglichkeiten, die weit über statische Darstellungen hinausgehen. Die OpenData-Plattform der bayerischen Vermessungsverwaltung stellt einen bedeutenden Anreiz dar. Diese Plattform ermöglicht den freien Zugang zu umfangreichen Geobasisdaten, unter anderem in Form von WMS-Diensten, die als Grundlage für innovative AR-Anwendungen dienen können. Durch diese Bereitstellung ist es möglich kostenlos auf qualitativ hochwertige und aktuelle Informationen zuzugreifen. Des Weiteren wird gezeigt, dass sich neuartige Geodaten wie das 3D-Mesh in ein dynamisches Umfeld einbinden lassen und welche Herausforderungen dadurch entstehen.

Die Kombination dieser Aspekte, die Anpassungsfähigkeit der AR-Technologie und die Nutzung frei verfügbarer Geobasisdaten bilden die Grundlage für das Interesse und die Zielsetzung dieser Arbeit. Es ist das Bestreben, durch die Untersuchung und Implementierung eines adaptiven AR-Systems einen Beitrag zur Weiterentwicklung dieses spannenden und dynamischen Forschungsfeldes zu leisten.

1.3 Stand der Forschung

Der folgende Abschnitt bietet einen Überblick über den derzeitigen Forschungsstand im Bereich der Augmented Reality in Verbindung mit raumbezogenen Daten. Es werden relevante Projekte mit verschiedensten Ansätzen vorgestellt.

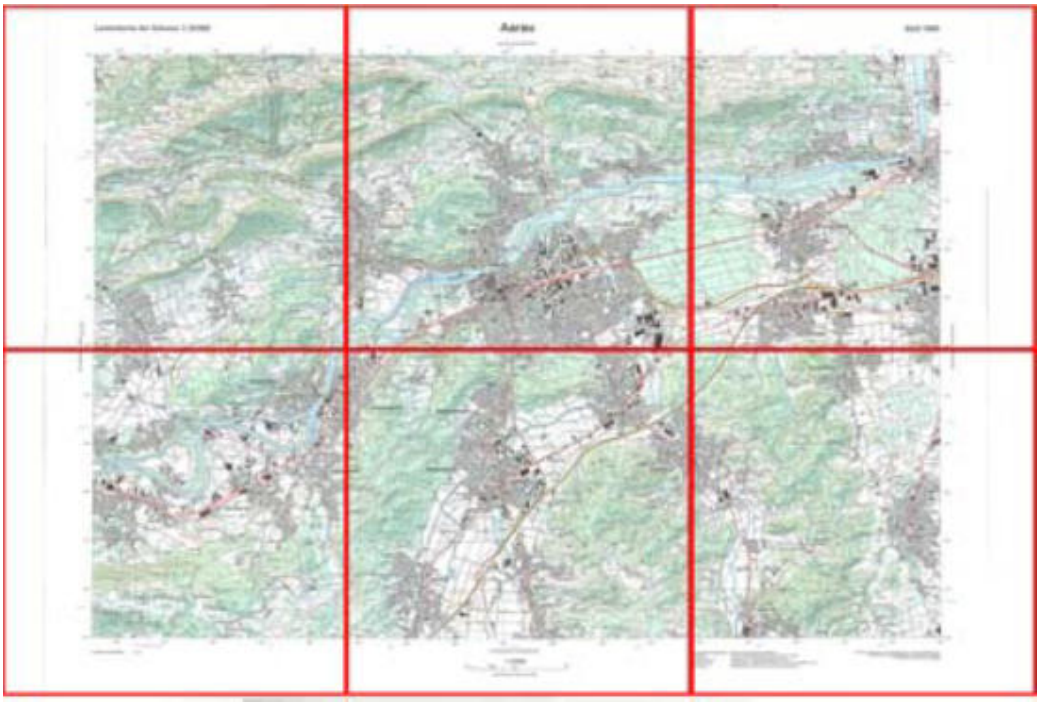

Abbildung 1.1: Schweizer Landeskarte als Marker nach (Loesch et al., 2015)

Der Artikel „Geospatial Augmented Reality – Lösungsansätze mit natürlichen Markern für die Kartographie und die Geoinformationsvisualisierung im Außenraum“ aus dem Jahr 2015 der Fachhochschule Nordwestschweiz beschäftigt sich mit dem Themenfeld Geospatial Augmented Reality. In diesem wird ein angewandtes Projekt mit dem Titel „Augmented Maps“ vorgestellt. Hier werden Möglichkeiten zur erweiterten Nutzung von gedruckten Karten untersucht. Die physische Karte soll als Marker fungieren, dabei werden lagekorrekt virtuelle Inhalte zweidimensional und dreidimensional überlagert. Zur Umsetzung wird im Beispielprojekt als Basis die schweizer Landeskarte im Maßstab 1:25000 verwendet und sechs zusammenhängende Markerbilder generiert (siehe Abbildung 1.1). Die Endanwendung wird für mobile Geräte konzipiert. Die grundlegende App wird dabei mithilfe der Game Engine Unity3D entwickelt, für das Augmented Reality Tracking wird das Plug-in Vuforia von Qualcomm verwendet. Die digitale Erweiterung der physischen Karte findet auf Basis von Geodaten

statt, die auch in klassischen Geoinformationssystemen (GIS) zu finden sind. Hierbei kommen ein Höhen- und Landschaftsmodell, aber auch Orthophototexturen und eine alternative Kartendarstellung zum Einsatz. Außerdem wurden Live-Inhalte in Form von Fahrgastinformationen der Schweizer Bundesbahn eingebunden, welche über ein Kontextmenü an Bahnhöfen in der Karte erreichbar sind. (Loesch et al., 2015)

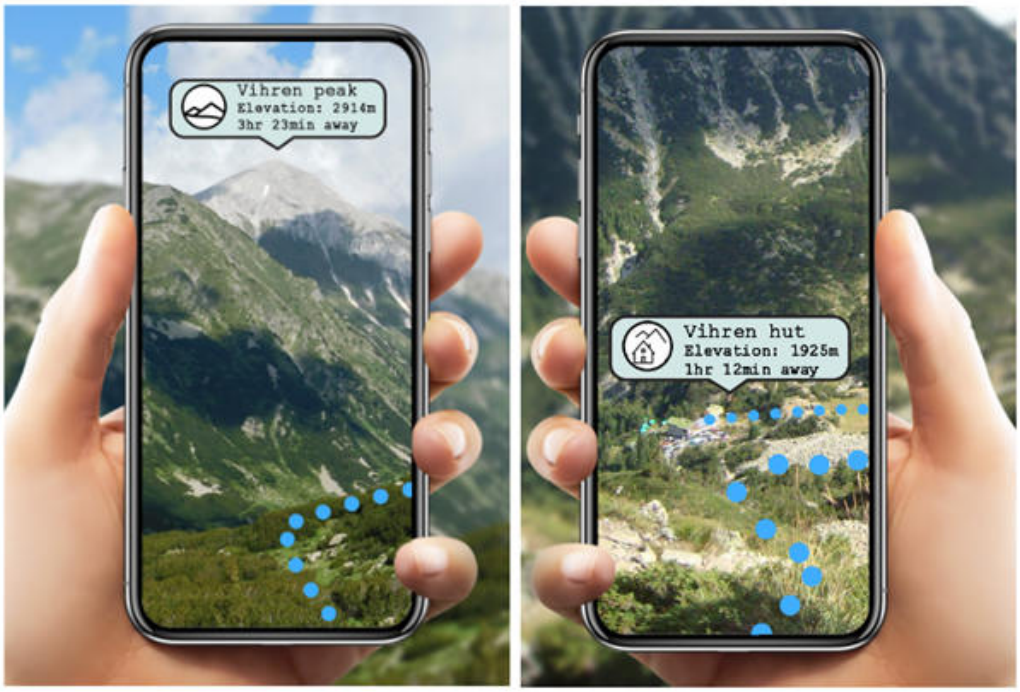

Abbildung 1.2: AR-App im Nationalpark nach (Yonov and Petkov, 2020)

Eine wissenschaftliche Arbeit, die Methoden des Location-based AR und Marker-based AR vereint, ist der im Jahr 2020 erschiene Artikel mit dem Titel „Augmented Reality Navigation Map of Mountain Area". Dabei wird insbesondere auf den Anwendungsfall Navigation im Gebirge eingegangen, das Hauptziel der Arbeit liegt darin, Touristen eine bessere Orientierung im Pirin National Park zu gewährleisten. Dafür wurden im Park Schilder mit Farbmustern aufgestellt. Die Anwendung zeigt dem Nutzer beim Scannen ein 3D-Modell des Parks mit allen Wanderrouten, sowie die Position, an der er sich selbst befindet. In Abbildung 1.2 sieht man den Location-based Teil der App. Er ermöglicht es, sich Wanderrouten und Namen von Berggipfeln in der Handyaufnahme darstellen zu lassen. Die Arbeit zeigt, dass AR ein wichtiger Baustein sein kann, um abgelegene und schwer zugängliche Gebiete geografisch zu visualisieren.(Yonov and Petkov, 2020)

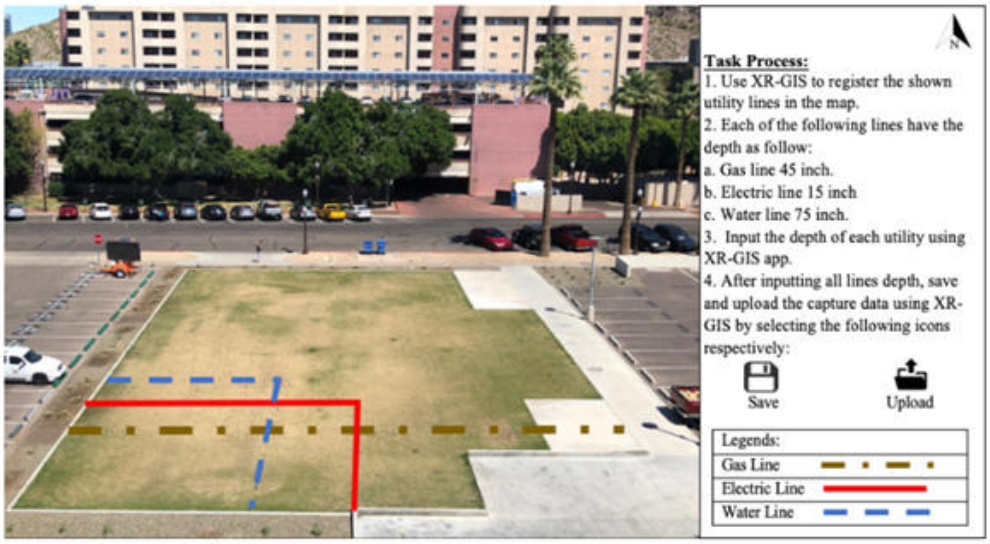

Abbildung 1.3: AR-GIS-System nach (Fenais et al., 2019)

Eine weitere Arbeit aus dem Jahr 2019 mit dem Namen „Integrating Geographic Information Systems and Augmented Reality for Mapping Underground Utilities“ beschäftigt sich mit der Verbindung von GIS-Systemen und AR-Anwendungen. Der Hauptforschungspunkt liegt dabei auf der Kartierung und Erfassung von unterirdischen Versorgungsleitungen. Hier soll eine Verbesserung der Informationsbereitstellung erfolgen, um die allgemeine Sicherheit zu erhöhen und das Risiko von Beschädigungen in Folge von Baumaßnahmen zu minimieren, da häufig fehlende Kommunikation zwischen Bauunternehmen und Versorgungsbetreibern der Hauptgrund für Beschädigungen ist. Als Lösung wird hierfür ein integriertes AR-GIS-System mit Cloud-Speicher entwickelt. Dieses System wurde von Fachleuten mithilfe von Fragebögen beurteilt, wobei 86 % der Teilnehmer die Anpassungsfähigkeit des Systems an die Gegebenheiten der unterirdischen Bauindustrie bestätigen. Insgesamt zeigt die Arbeit, dass eine Kombination aus GIS und AR in einer Anwendung die Sicherheit und Effizienz in der Bauindustrie erhöhen können.(Fenais et al., 2019)

1.4 Beitrag der Arbeit

Diese Masterarbeit soll in mehreren Bereichen einen Beitrag leisten. Es wird eine Überlegung zur Nutzung der Augmented Reality (AR) in der Geovisualisierung angestellt. Darunter im Wesentlichen das adaptive Tracking physischer Objekte

durch die Implementierung von Technologien zur Erkennung und Verfolgung von Kartenausschnitten. Zusätzlich werden WMS-Dienste integriert, um aktuelle Kartendaten in die Anwendung einzubinden. Durch die innovative Verarbeitung und Darstellung von 3D-Meshs soll eine neuartige Möglichkeit geschaffen werden, Geobasisdaten interaktiv und anschaulich zu präsentieren, welches die Benutzererfahrung und das Verständnis von Kartenmaterialien erheblich verbessert.

Im Zuge des praktischen Teils der Masterarbeit soll eine AR-App entwickelt werden, die das dynamische Tracking von physischen Kartenausschnitten des LDBV und deren Anreicherung mit zusätzlichen digitalen Informationen ermöglicht. Diese Anwendung kann perspektivisch in verschiedenen Bereichen wie Bildung, Tourismus und Stadtplanung eingesetzt werden.

Des Weiteren soll diese Arbeit zur Weiterentwicklung im Bereich AR beitragen, indem sie neue Tracking-Methoden zur Erkennung von Kartenausschnitten entwickelt und implementiert. Diese Vorgehensweise kann als Grundlage für weitere Anwendungen und Entwicklungen in der AR genutzt werden. Die Arbeit präsentiert eine systematische Herangehensweise zur Entwicklung und Evaluierung einer AR-Anwendung, die speziell auf die Nutzung von Geobasisdaten zugeschnitten ist. Dies umfasst die Konzeption, das Prototyping und die Durchführung von Performance-Tests. Die vorgestellte Methodik zur Verarbeitung von 3D-Meshs und deren Integration in AR-Anwendungen bietet ein Modell, das auf andere Projekte im Bereich der AR und Geodatenanwendungen übertragbar ist.

2 Grundlagen

2.1 Grundlegende Begriffe

2.1.1 Mixed Reality

Dieser Begriff wurde erstmals 1994 von P. Milgram und F. Kishino verwendet. Die Mixed Reality wird von ihnen als ein Kontinuum definiert, das die vollständige Bandbreite zwischen der realen und virtuellen Welt umfasst. Innerhalb dieses Mixed Reality Kontinuums ist einerseits die Augmented Reality näher an der realen Welt zu finden. Sie beschreibt Szenarien, in denen virtuelle Objekte in die reale Welt eingebettet werden. Es findet also ein Anreichern der realen Welt mit virtuellen Informationen statt. Auf der anderen Seite befindet sich die Augmented Virtuality. Sie beschreibt Anwendungsfälle, in denen reale Objekte grundsätzlich in eine virtuelle Welt eingebettet werden. Die Mixed Reality bietet ein großes Anwendungsfeld beispielsweise in den Bereichen Simulation, Training, Architektur und Design. Sie kann als Grundlage für zukunftsweisende Projekte dienen, die die Verschmelzung von Realität und Technologie vorantreiben. (Milgram and Kishino, 1994)

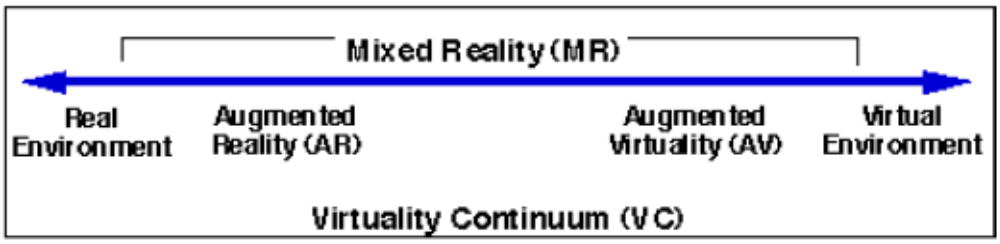

Abbildung 2.1: Virtualitäts Kontinuum nach (Milgram and Kishino, 1994)

2.1.2 Augmented Reality

Die Augmented Reality ist Teil der oben genannten Mixed Reality, sie ordnet sich dabei näher an der realen Welt ein (Milgram and Kishino, 1994).

AR ermöglicht die Überlagerung der physischen Welt mit computergenerierten virtuellen Informationen. Im Allgemeinen kann dabei zwischen drei verschiedenen Arten der Augmented Reality unterschieden werden. Der meistverwendete Fall ist Video See-Through-AR. Dabei wird die Kamera eines Handys oder Tablets verwendet, um die Umwelt aufzunehmen und auf dem Display darzustellen. Eine andere Methode zur Darstellung von AR-Inhalten ist optisches See-Through-AR, dabei wird ein semitransparentes Display z.B. in Form einer Brille verwendet, um künstliche Inhalte in die dahinterliegende Umgebung einzublenden. Bei der dritten Form handelt es sich um projektionsbasierte AR. Dabei werden digitale Inhalte durch Beamer auf reale Objekte projiziert. Bei diesem Verfahren erfährt der Nutzer die wenigsten Einschränkungen und erhält die größte Immersion. (Dörner et al., 2013, S. 241 ff.)

2.1.3 Unity

Bei Unity handelt es sich um eine Game Engine des gleichnamigen amerikanischen Unternehmens Unity Technology. Es ist die führende Plattform im Bereich der Spieleentwicklung für Smartphones. Die Engine zeichnet sich durch einen hohen Grad an Wandelbarkeit aus. Es können Spiele in 2D, 3D und für Extended Reality entwickelt werden. Neben der Spieleentwicklung lassen sich auch Anwendungen in den Bereichen Automotive, Architecture, Aerospace und Film erstellen. Durch die Möglichkeit, Plug-ins zu verwenden und eigene Skripte in der Programmiersprache C# zu schreiben, ist es in Unity möglich, individualisierte Apps für verschiedenste Anforderungen zu erstellen. (Technologies, 2024)

2.1.4 Vuforia

Vuforia ist eine Augmented Reality Plattform, die es Entwicklern ermöglicht, digitale Inhalte in die reale Welt zu integrieren. Dabei sind das Erkennen und Verfolgen einer Vielzahl von visuellen Objekten möglich. Es können Bilder, Texte und auch 3D-Objekte erfasst werden. Dabei wird das Tracking von mehreren Zielen gleichzeitig unterstützt. Vuforia unterstützt sowohl Location-based-AR

als auch Marker-based-AR. Die verschiedenen Tracking-Ziele können sowohl lokal in der App gespeichert als auch über Cloud-Services online abgerufen werden. Der Einsatz erstreckt sich dabei über ein Anwendungsspektrum von Industrie und Fertigung über Marketing und Werbung bis hin zu Spielen und Unterhaltung. (PTC, 2024)

2.1.5 Open Source Computer Vision Library (OpenCV)

Bei OpenCV handelt es sich um eine freizugängliche Softwarebibliothek für maschinelles Sehen und Bildverarbeitung. Sie ist eine der am häufigsten verwendeten Bibliotheken der Computer Vision und wird von einer aktiven Gemeinschaft gepflegt. OpenCV zeichnet sich durch umfangreiche Funktionen aus, die mit mehr als 2500 optimierten Algorithmen implementiert sind. Es werden alle gängigen Betriebssysteme unterstützt und es gibt Schnittstellen zu den Programmiersprachen C++, Python, Java und MATLAB. Die Hauptanwendungsgebiete von OpenCV sind klassische Bildverarbeitung, Gesichtserkennung, Objekterkennung, maschinelles Lernen und medizinische Bildgebung. (OpenCV, 2024)

2.2 Funktionsweise AR

Augmented Reality ist eine Technologie, die es ermöglicht, virtuelle Informationen und Objekte in die reale Welt einzubetten. Jedes System zur Nutzung von AR-Inhalten erfordert gewisse technische Voraussetzungen. So wird Hardware in Form von Kameras, Brillen oder Displays benötigt. Des Weiteren bedarf es einer Computereinheit, auf der die AR Software betrieben und die aufgenommen Daten verarbeitet und interpretiert werden können. (Dörner et al., 2013, S. 241 f.)

Dabei wird grundsätzlich zwischen zwei Arten der Augmented Reality unterschieden. Die Erste ist hierbei die sogenannte Location-based AR, bei der die Inertial Measurement Unit (IMU), der Kompass und der GPS-Empfänger des

jeweiligen Smartphones oder Tablets genutzt werden, um die Position und Orientierung des Endgerätes in der realen Welt festzustellen. Virtuelle Inhalte werden dann relativ zur eigenen Position in der Welt platziert. Diese Anwendungen stoßen allerdings bei hohen Genauigkeitsanforderungen schnell an ihre Grenzen, Messrauschen und allgemeine Ungenauigkeit der Handy- und Tabletsensorik führen zu störendem Springen und Wackeln der virtuellen Inhalte. (Loesch et al., 2015)

Bei der zweiten Art handelt es sich um die sogenannte Marker-based Augmented Reality. Dabei werden Lage und Position des Endgerätes mithilfe von Kamerabildern, die einen Marker in der realen Welt enthalten, berechnet. Die Bildmerkmale bzw. Features des Markers werden in der App gespeichert und ständig in der laufenden Kameraaufnahme gesucht. So ist es möglich, digitale Inhalte relativ zum Marker in der realen Welt zu platzieren. Der große Nachteil hierbei ist, dass das ständige Erkennen des Markers rechenintensive Algorithmen benötigt. (Loesch et al., 2015)

Eine Neuentwicklung im Bereich der Marker-based Augmented Reality ist der Einsatz von Multimarkern. Dabei handelt es sich um mehrere Einzelbilder, die dasselbe Marker-Objekt enthalten, Position und Orientierung der Bilder zueinander sind dabei bereits bekannt. Ein solches Verfahren kann helfen, das Tracken aus ungünstigen Winkeln zu verbessern. Eine weitere Einsatzmöglichkeit von Multimarkern ist das Bilden einer LoD-Pyramide (Level of Detail). Am Beispiel einer Landkarte könnte so die gesamte Ausdehnung der Karte als erstes Level definiert und immer kleiner werdende vorgegebene Ausschnitte als darunterliegende LoD-Level untergeordnet werden. (Loesch et al., 2015)

Des Weiteren ermöglicht der Einsatz von Multimarkern auch das Tracken von physischen 3D-Objekten als Marker. So können virtuelle Inhalte auf und neben dem Objekt dauerhaft dargestellt werden. Solche Anwendungen können z. B. im Maschinenbau einen großen Mehrwert in den Bereichen Fertigung und Qualitätsprüfung bieten. (PTC, 2024)

Eine andere Herangehensweise für sich ändernde Bedingungen bei Augmented Reality ist der Einsatz von cloudbasierten Markern. Dabei wird bei jedem Start der App auf dem Handy erst die Tracking-Datenbank aktualisiert, wodurch das Tracking auch bei sich ändernden äußeren Bedingungen stabil aufrechterhalten werden kann. (PTC, 2024)

2.2.1 Erfassungsmethoden

Der wohl wichtigste Teil für ein stabiles Tracking, also das Erfassen der Umwelt in der Augmented Reality, ist die Sensorik. Über diese soll die Pose, bestehend aus Orientierung und Position, des Endgerätes ermittelt werden. Im Folgenden werden einige Vorgehensweisen genannt.

Ein Ansatz ist die Bestimmung der Pose über magnetische Sensoren in einem künstlichen Magnetfeld. Es ermöglicht stabiles Tracking und 360° Bewegungsfreiheit. Die Grundvoraussetzung eines künstlichen Magnetfeldes reduziert jedoch die Einsatzmöglichkeiten. Für medizinische Anwendungen in einer konstanten Umgebung wie einem Krankenhaus bietet sich dieses System jedoch an. (Syed et al., 2022)

Eine effektive und im Bereich von Smartphones vielgenutzte Herangehensweise ist das Nutzen einer IMU-Einheit. Diese misst dauerhaft die Dreh- und Beschleunigungsraten des Endgerätes und erlaubt so eine Bestimmung der Orientierung. Über längere Zeit addieren sich jedoch die Messfehler der Sensoren und es kommt zu Fehlern in der Platzierung von Objekten. (Syed et al., 2022)

Eine Vielzahl von Ansätzen gibt es in der Auswertung von Kamerabildern. In den Bildern können markante Punkte identifiziert werden. Durch Bewegen der Kamera werden diese Punkte aus verschiedenen Perspektiven erkannt und eine Bildtriangulation durchgeführt. So kann die relative Position der Kamera zu den aufgenommenen Objekten bestimmt werden. Des Weiteren können 2D- oder 3D-Marker in einer Datenbank der Anwendung gespeichert werden. Die Perspektive im Bild kann so abgeglichen und die Aufnahmeposition der Kamera bestimmt werden. (Syed et al., 2022)

Eine Möglichkeit zur Bestimmung der Position weltweit sind Global Navigation Satellite Systems (GNSS). Diese eignen sich für Outdoor-Navigation oder das Finden von Landschaftsmerkmalen. Die Genauigkeit der Positionierung ist begrenzt, da es Abweichungen im Meterbereich gibt. (Syed et al., 2022)

Zur genaueren Bestimmung ist ebenso eine Kombination verschiedener Sensoren möglich. Ein Beispiel hierfür wäre Sensorfusion oder Simultaneous Localization and Mapping (SLAM). Bei letzterem wird versucht, ein Gerät gleichzeitig zu lokalisieren und ebenso eine Karte seiner Umgebung zu erstellen. Dabei werden Merkmale der Umgebung erkannt und gespeichert. Mit diesen Merkmalen wird

eine Karte erstellt. (Syed et al., 2022)

2.2.2 Einblick ARCore

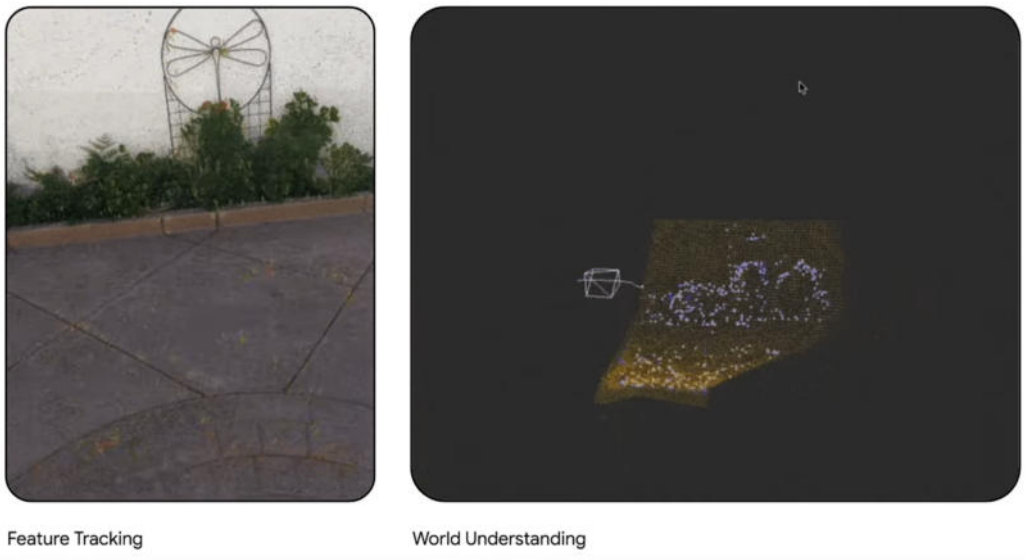

Abbildung 2.2: Feature Tracking nach (Tsotsos and Ballan, 2020)

Google liefert einen Einblick in die technische Umsetzung ihrer Augmented Reality Software ARCore. Diese stützt sich im Allgemeinen auf eine Mischung aus kamerabasierter AR mit zusätzlicher Unterstützung der IMU-Einheit. Dies wird Visual Inertial Sensor Fusion genannt. Die kamerabasierte Verarbeitung besteht zum größten Teil aus dem sogenannten Feature Tracking. Dabei werden in jedem aufgenommenen Bild markante Punkte erkannt und versucht dieselben Punkte in anderen Bildern zuzuordnen. In Abbildung 2.2 sind die markanten Punkte auf der linken Seite als kleine gelbe Punkte dargestellt. Mithilfe dieser Punkte kann eine Bildtriangulation durchgeführt und so die relative Geometrie der Kamera zu den Punkten berechnet werden. Auf der rechten Seite der Abbildung 2.2 wird die relative Position der Kamera zu den markanten Punkten in Blau angezeigt. Das Schätzen der Entfernung zwischen Kamera und den Punkten bezeichnet Google als das Tiefen-Application Programming Interface (Depth-API). (Tsotsos and Ballan, 2020)

Ein neuartiger Ansatz, den ARCore verfolgt ist, Maschine Learning based IMU Tracking. Es wird versucht, die Fehler der IMU-Einheit durch neuronale Net-

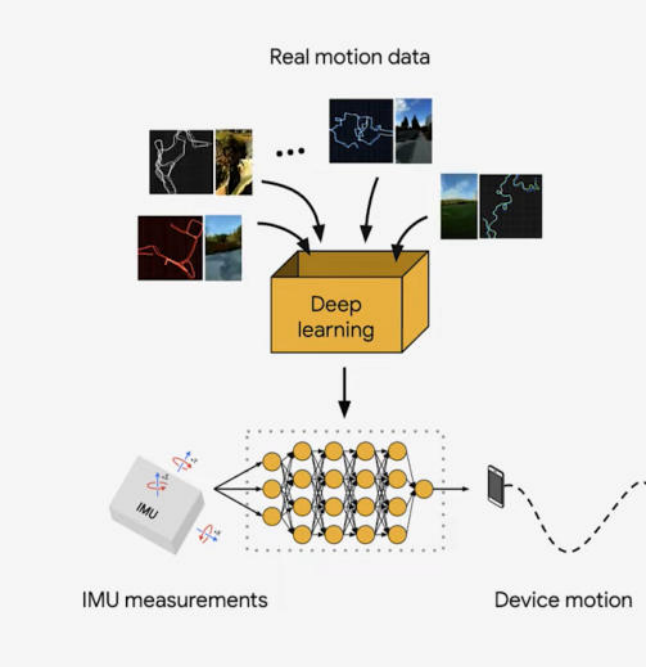

Abbildung 2.3: Visual-Inertial Sensor Fusion nach (Tsotsos and Ballan, 2020)

ze zu minimieren. Die IMU-Einheit wird hauptsächlich dann gebraucht, wenn das Feature Tracking z. B. durch verschwommene Bilder gestört ist. Die Pose-Bestimmung erfolgt dann größtenteils anhand der IMU-Messwerte. Das Problem ist hierbei, dass sich die Fehler der IMU-Messungen über die Zeit aufsummieren und so Fehler in der Darstellung von AR auftreten. Google setzt hier auf künstliche Intelligenz in Form von neuronalen Netzen. Diese wurden mit den typischen Bewegungsmustern beim Nutzen einer AR-App trainiert. Dies kann Unsicherheiten der IMU-Einheit ausgleichen und so für eine stabile Anwendung sorgen. (Tsotsos and Ballan, 2020)

2.3 Geobasisdaten

Geobasisdaten sind grundlegende geografische Daten, die als Fundament für vielfältige Anwendungen in der Praxis dienen, sei es in der Wirtschaft, Wissenschaft, Verwaltung, oder im privaten Bereich. Sie enthalten Informationen über die natürliche und gebaute Umwelt.

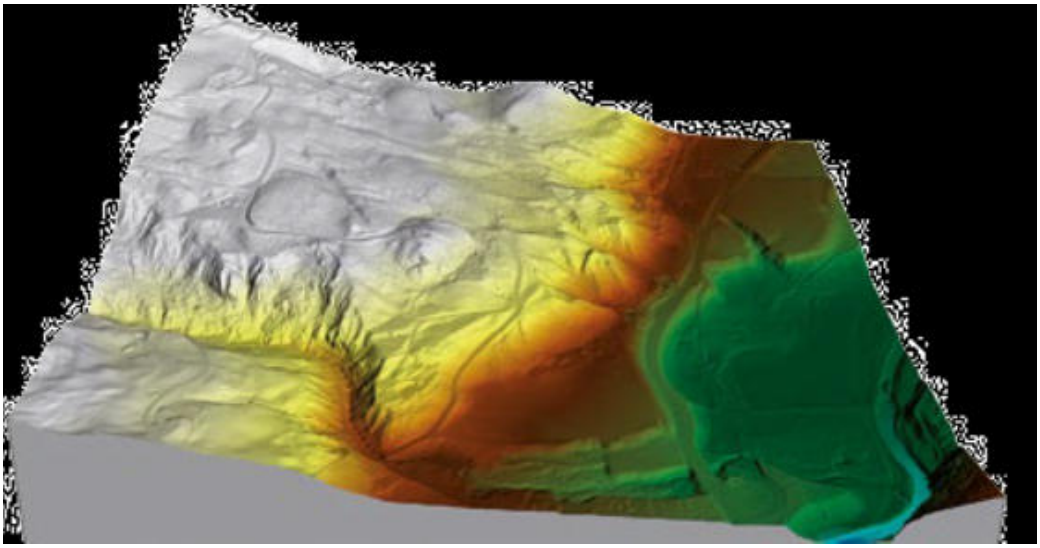

Abbildung 2.4: Exemplarisches digitales Geländemodell nach (Bayerisches Landesamt für Digitalisierung, 2019)

In Bayern werden die Geobasisdaten durch die Bayerische Vermessungsverwaltung bereitgestellt. Geodaten vereinen eine Sachauskunft mit einer Position oder einem geografischen Bereich der Erde. Ihnen wird eine bestimmte örtliche Lage auf der Oberfläche der Erde mittels Koordinaten verliehen. Die Besonderheit der amtlichen Geobasisdaten ist ihre hohe Aktualität, ein einheitliches Qualitätsniveau sowie ihre flächendeckende Bereitstellung. Im weiteren Verlauf werden einige für diese Arbeit wichtige Geobasisdaten des Freistaats Bayern angeführt und erklärt. (Bayerisches Landesamt für Digitalisierung, 2019)

Digitale Orthophotos (DOP): Bei den Digitalen Orthophotos handelt es sich um verzerrungsfreie Luftbilder, die die Erdoberfläche maßstabsgetreu abbilden und lagerichtig wiedergeben. Verfügbar sind sie in Echtfarben und im Infrarotbereich. Der Infrarotbereich eignet sich für Analysen der Vegetation. Orthophotos werden in verschiedenen Auflösungen erzeugt. Sie sind vollständig für ganz Bayern erhältlich. (Bayerisches Landesamt für Digitalisierung, 2019)

Digitales Oberflächenmodell (DOM): Bei dem bildbasierten Digitalen Oberflächenmodell handelt es sich um eine Darstellung der Oberfläche der Erde mit den darauf befindlichen Objekten, wie z.B. Bebauung oder Pflanzenbewuchs, in Gitterform. Das digitale Oberflächenmodell wird aus Luftbildern abgeleitet und zusammen mit den Orthophotos berechnet. (Bayerisches Landesamt für Digitalisierung, 2019)

Digitales Geländemodell (DGM): Das Digitale Geländemodell präsentiert die Oberfläche der Erde ohne Vegetation und Bauwerke als 3D-Modell. Das Modell ist als Gitternetz verfügbar. Jedem Gitterpunkt wird hierbei eine Geländehöhe zugeordnet. Die erhältliche Auflösung wird von 1 Meter bis 200 Meter Gitterweite angeboten. Die Daten für das DGM werden aus Befliegungen mit dem Laserscanner abgeleitet. (Bayerisches Landesamt für Digitalisierung, 2019)

3D-Gebäudemodelle: Die 3D-Gebäudemodelle werden in Bayern flächendeckend in verschiedenartigen Varianten bzw. Genauigkeitsabstufungen (LoD = Level of Detail) angeboten. Das einfachere LoD1, enthält lediglich die Gebäudeumrisse in 3D-Darstellung während das LoD2 die Dachform tatsächlicher Dächer in standarisierter Weise zusätzlich modelliert. Die Modelle werden durch Laserscanning-Befliegungen und aus Daten im Liegenschaftskataster abgeleitet und fortlaufend durch Einmessungen der Gebäude vor Ort ergänzt. (Bayerisches Landesamt für Digitalisierung, 2019)

Digitale Topografische Karte (DTK): Die Topographische Karte bereitet den sichtbaren Teil der Oberfläche der Erde kartographisch auf. Sie zeichnen sich durch einen beträchtlichen Umfang an Informationen aus und sind sowohl digital als auch gedruckt erhältlich. Kartenauszüge sind in verschiedenen Maßstäben zu erhalten und werden ständig aktualisiert. (Bayerisches Landesamt für Digitalisierung, 2019)

Webkarte Bayern: Die Webkarte lässt sich gut für Applikationen im Web einsetzen und wurde eigens zu diesem Zweck überarbeitet. Ihre Darstellung umfasst Inhalte aus dem Amtlichen Topographischen Kartographischen Informationssystem (ATKIS), der digitalen topographischen Karte im Maßstab 1:25000, den Hausumringen und der Gemeindenamendatenbank. (Vermessungsverwaltung, 2024a)

2.4 Geodateninfrastruktur

Die Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland beschreibt eine Geodateninfrastruktur (GDI) als die Bereitstellung vorhandener Geodatenbestände von allgemeinem Interesse über Geo Web Services (GWS). Diese Dateninfrastruktur soll öffentlich zugänglich sein, sodass sie in allen raumbezogenen Vorgängen des öffentlichen und wirtschaftlichen Handelns genutzt werden kann. (Vogel, 2002)

2.4.1 Inspire

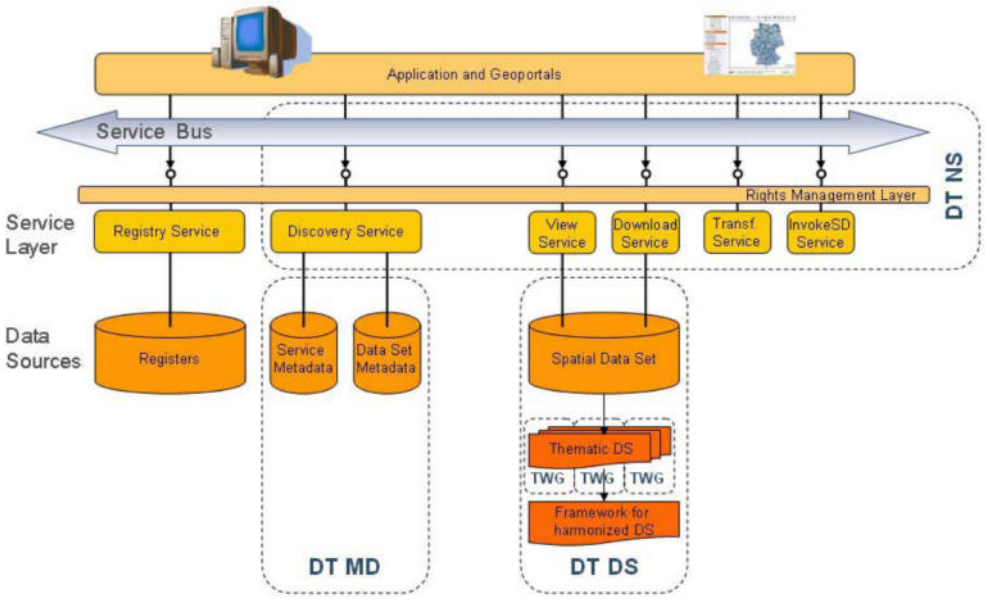

Abbildung 2.5: Übersicht Inspire nach (Illert, 2009)

2007 verabschiedete die Europäische Union die Direktive 2007/2/EC. Diese sieht vor, eine EU-weite Infrastruktur für raumbezogene Daten zu schaffen. Sie soll die Zugänglichkeit von Raumdaten erleichtern. Um dies zu gewährleisten, wurden die Erfassung, Verwaltung und Verbreitung von raumbezogenen Daten standardisiert. Die Kompatibilität zwischen den Diensten und Datensätzen ist ein weiteres zentrales Thema. Dazu wurden verschiedene Leitlinien entwickelt. Zum einen zur Erstellung von Metadaten, die die Auffindbarkeit und Nutzung erleichtern sollen. Die Datenhaltung selbst wird auch auf Austauschbarkeit ausgerichtet. Zum anderen die Architektur der Geodatendienste und ihre Administration.

Die Richtlinien sind für alle Mitgliedsstaaten verpflichtend und müssen in Landesrecht umgesetzt werden. Die Direktive ist also entscheidend für die Schaffung einer konsistenten, zugänglichen und interoperablen europäischen Geodateninfrastruktur. (Vogel, 2002; Union, 2007)

2.4.2 Geodateninfrastruktur (GDI)

In Deutschland erfolgt die Umsetzung der Inspire Initiative mit der Geodateninfrastruktur Deutschland, kurz GDI-DE. Dabei werden die Verwaltungsebenen in der Bearbeitung ihrer Aufgaben unterstützt und Verfahren mit Raumbezug automatisiert. Des Weiteren sollen die raumbezogenen Daten bei der Entwicklung der Wirtschaft und Forschung helfen. Außerdem stellt die GDI sich die Aufgabe, die Transparenz der Verwaltung durch die öffentliche Bereitstellung von Geodaten zu fördern. Diese sollen leicht zugänglich gemacht werden und die Kompatibilität untereinander gewährleistet sein. Flexibilität wird durch dezentral aufgebaute IT-Komponenten gegeben. (GDI-DE, 2021)

Eine wichtige Komponente der GDI-DE ist die Geodateninfrastruktur Bayern (GDI-BY). In ihr erfolgt die Umsetzung auf Landesebene. Das Geoportal Bayern stellt einen wesentlichen Bestandteil der GDI-BY dar. Auf dieser Plattform werden raumbezogene Daten von der staatlichen Verwaltung, Kommunen, Firmen und Privatperson zur Verfügung gestellt. Eine Suchfunktion erleichtert das Auffinden relevanter Datensätze. Die Bereitstellung der Daten erfolgt in Form von Datensätzen oder Darstellungsdiensten wie z. B. Web Map Services. (Staatsregierung, 2024)

2.5 Computer Vision

Computer Vision ist ein sehr weit gefasstes Forschungsgebiet. Im Allgemeinen befasst es sich mit der Analyse und Auswertung von Daten, die durch verschiedenste Sensor- und Aufnahmesysteme erfasst werden. Das Ziel ist dabei die vorhandene Information so zu verarbeiten, dass sie der menschlichen Auffassungsgabe entsprechen. Computer Vision befasst sich mit mathematischen Methoden, die dazu dienen, die dreidimensionale Form und das Erscheinungsbild von Objekten anhand von Bilddaten zu rekonstruieren. Allerdings ist es eine Herausforderung, die visuelle Welt in ihrer ganzen Komplexität präzise abzubilden. (Szeliski, 2022)

Im Folgenden wird das für diese Arbeit relevante Template Matching, ein Teilbereich der Computer Vision, näher beleuchtet.

2.5.1 Template Matching

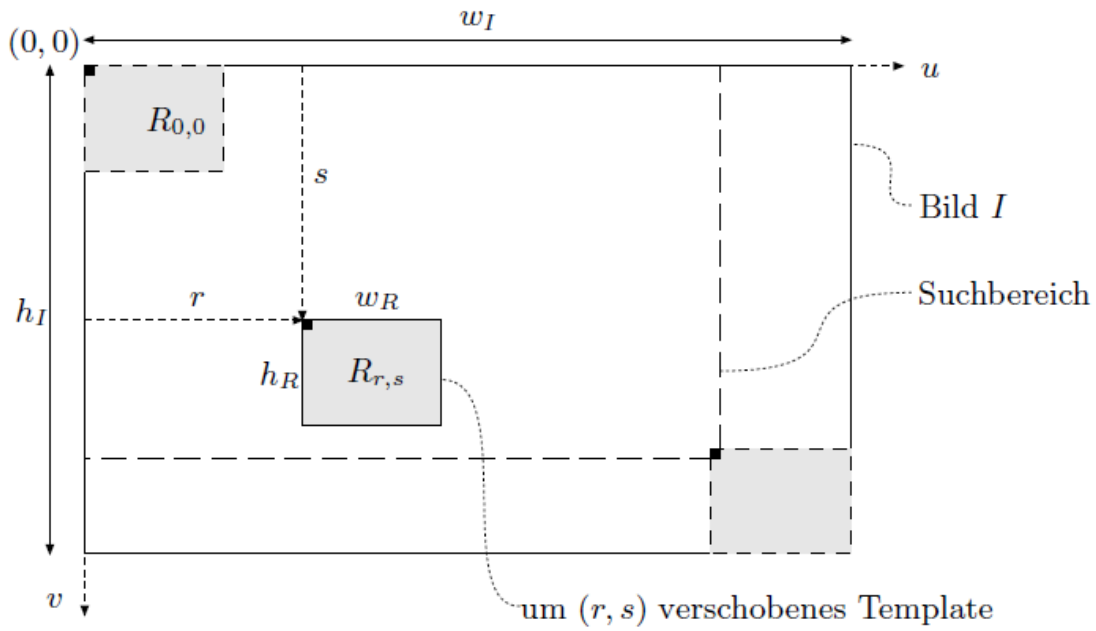

Abbildung 2.6: Geometrie des Template Matching nach (Burger, 2006)

Hierbei handelt es sich um eine Technik, die in der Bildverarbeitung weit verbreitet ist. Sie wird verwendet, um kleine Ausschnitte (templates) einem größeren

Bild zuzuordnen (matching) (Burger, 2006, S. 412 f.).

Beim Template Matching wird von folgenden Voraussetzungen ausgegangen. Das in Abbildung 2.6 dargestellte Referenzbild R wird über das Zielbild I verschoben. Dabei dient der Koordinatenursprung in der oberen linken Ecke als Referenzpunkt. Ziel ist es die Verschiebung $R_{r,s}$ zu finden in dem die Ähnlichkeit zwischen Referenz- und Zielbild maximal ist. (Burger, 2006, S. 412 ff.)

Um Aufschluss über die Übereinstimmung des Referenzbildes an jeder Stelle des Zielbildes zu bekommen, muss eine Ähnlichkeitsmetrik eingeführt werden. Dazu gibt es verschiedene Ansätze und Methoden. In OpenCV sind unter anderem folgende Metriken implementiert. (OpenCV, 2024)

$$\text{SQDIFF: } R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x', y+y'))^2 \tag{2.1}$$

Bei SQDIFF werden für jede Iteration die Summe der quadrierten Differenz zwischen dem Referenzbild und dem Zielbild berechnet.

$$\text{SQDIFF\_NO: } R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x', y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x', y+y')^2}} \tag{2.2}$$

In Gleichung 2.2 wird die normalisierte Variante der Summe der quadrierten Differenz dargestellt. Eine Normalisierung dient dazu die Beleuchtungseinflüsse zwischen den beiden Bildern zu verringern.

$$\text{CCORR: } R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x', y+y')) \tag{2.3}$$

In Formel 2.3 wird die Kreuzkorrelation zwischen dem Ziel- und Referenzbild berechnet.

$$\text{CCORR\_NO: } R(x,y) = \frac{\sum_{x',y'}(T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \quad (2.4)$$

Zur Verbesserung von verschiedenen Beleuchtungseinflüssen wird in Gleichung 2.4 eine normalisierte Variante der Kreuzkorrelation dargestellt.

$$\text{CCOEFF: } R(x,y) = \sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y')) \quad (2.5)$$

Der mathematische Ausdruck 2.5 berechnet die Korrelationskoeffizienten zwischen dem Ziel- und Referenzbild. Es wird die lineare Abhängigkeit zwischen den beiden Bildern gemessen.

$$\text{CCOEFF\_NO: } R(x,y) = \frac{\sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \quad (2.6)$$

Bei der Berechnung der Korrelationskoeffizienten gibt es ebenso eine normalisierte Variante 2.6.

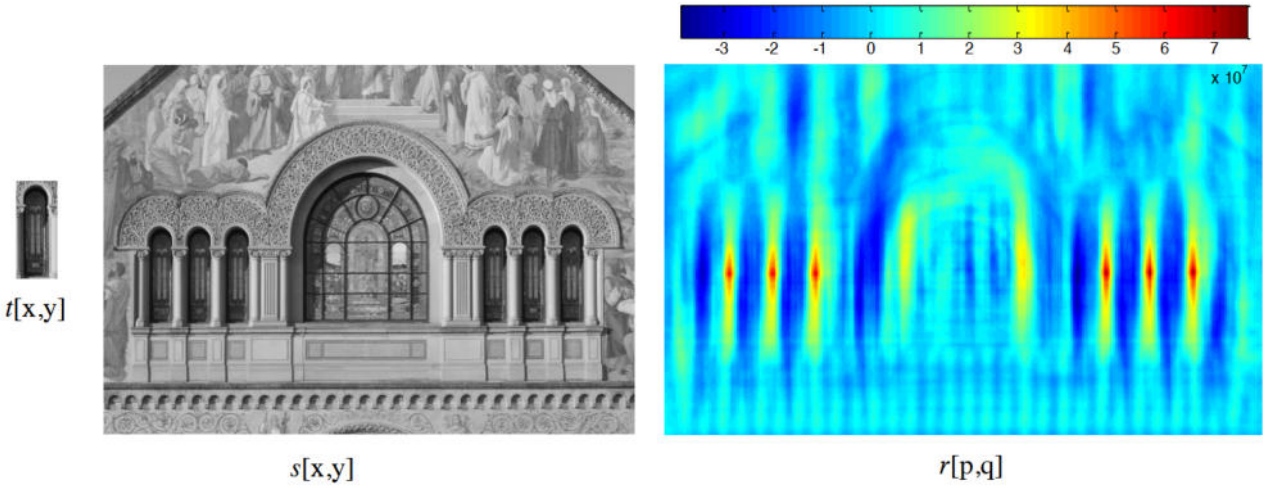

Abbildung 2.7: Zielbild mit Score-Werten nach (Girod, 2013)

Mit dieser Ähnlichkeitsmetrik wird für jede Position des Template-Bildes t[x,y] ein Wert (Score) berechnet. Je höher dieser ist, desto wahrscheinlicher ist die korrekte Zuordnung. In Abbildung 2.7 wird das Zielbild s[x,y] in Abhängigkeit jedes einzeln berechneten Scores auf der rechten Seite dargestellt. In diesem Beispiel wird ersichtlich, dass das Template-Bild mehrfach im Zielbild vorhanden ist. Je näher das Template-Bild an der richtigen Position ist, desto höher wird auch der Score-Wert, bis an der richtigen Stelle das Maximum erreicht wird. (Girod, 2013)

3 Werkzeuge

3.1 Opendata

In diesem Projekt sollen primär die Geobasisdaten des Freistaates Bayern präsentiert werden. Im Zuge der Umstellung der Bayerischen Vermessungsverwaltung auf OpenData, einem kostenfreien Geodatenportal, sind diese frei zugänglich (Vermessungsverwaltung, 2024b). Da die Geodaten später dynamisch verarbeitet werden, bietet es sich an, alle Rastergeodaten über WMS-Services zu beziehen. Diese Services sind über eine URL zu erreichen und benötigen als Eingabeparameter lediglich die Koordinaten, das Bezugssystem und andere optionale Parameter. Durch WMS-Services kann garantiert werden, dass die Geobasisdaten immer dem aktuellen Stand der Bayerischen Vermessungsverwaltung entsprechen. Der einzige Nachteil für den Nutzer besteht darin, dass eine Internetverbindung erforderlich ist.

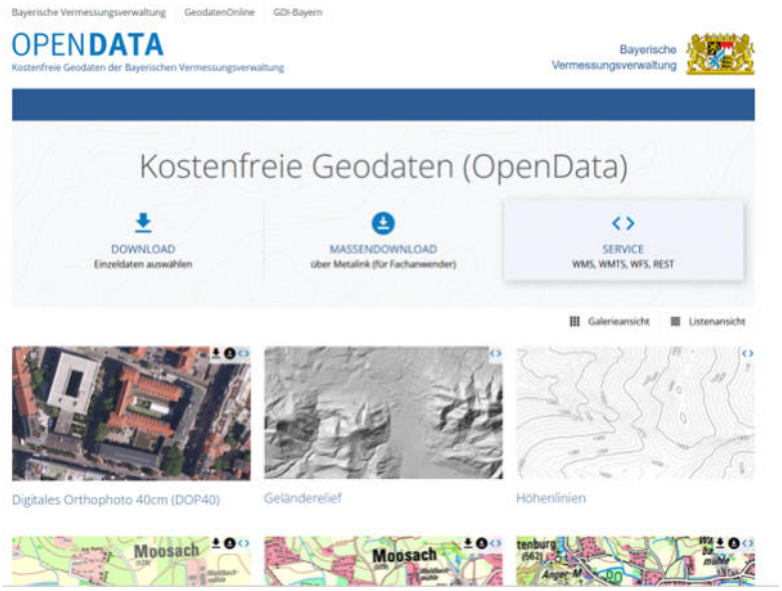

Abbildung 3.1: Opendata der Bayerischen Vermessungsverwaltung nach (Vermessungsverwaltung, 2024b)

3.2 Vorbereitung 3D-Mesh

Eine weitere wichtige Komponente in der späteren Anwendung ist die Darstellung eines 3D-Meshs. Es handelt sich dabei um ein detailliertes 3D-Modell. Es wird durch die Triangulation von Bildern einer Schrägbefliegung der Erdoberfläche berechnet. Das 3D-Mesh befindet sich in der Entwicklungsphase und ist deswegen noch nicht für die breite Öffentlichkeit zugänglich. Für dieses Projekt wurde ein Testgebiet des 3D-Meshs für den Raum Würzburg vom Landesamt für Digitalisierung, Breitband und Vermessung zur Verfügung gestellt.

Das 3D-Mesh wird in Form von Kacheln im Format obj. bereitgestellt. Diese Aufteilung ist für Unity jedoch nicht optimal. Daher wird zuerst ein Vorverarbeitungsprozess durchgeführt.

Im ersten Schritt werden die einzelnen Kacheln in die Software 3ds Max geladen. Sie erhalten dort eine fortlaufende Nummerierung. Das Gesamtgebiet von Würzburg wird in vier große Kacheln als fbx.-Dateien abgespeichert.

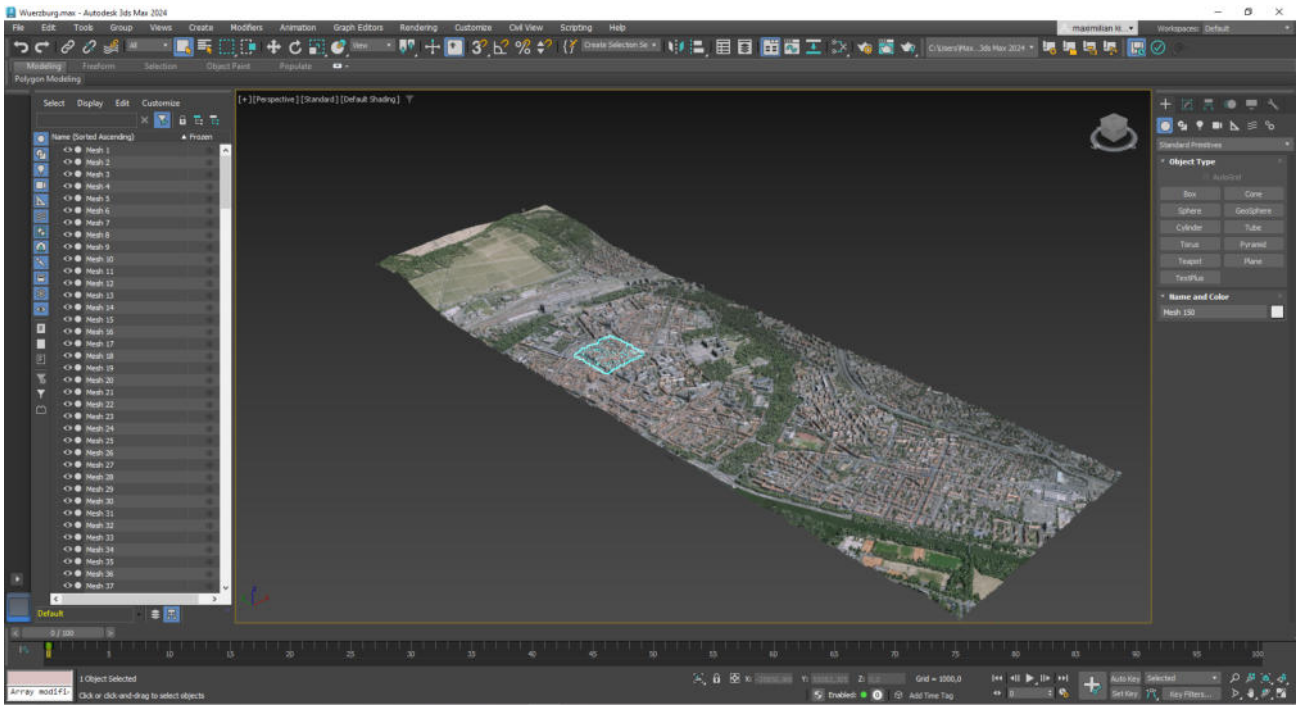

Abbildung 3.2: Vorbereitung 3D-Mesh in 3ds Max

Bei fbx.-Dateien werden die Texturen stets in einem extra Ordner namens fbm abgespeichert. Diese Texturen lassen sich auch in Photoshop öffnen. Hier wird

die Radiometrie verbessert und das spätere Modell bekommt ein sehr realistisches Aussehen. Um diese Änderungen zu übernehmen, muss das 3D-Mesh nochmals in 3ds Max mit den neuen Texturen abgespeichert werden. Nun kann das Mesh mit der Funktion „Batch Optimizer“ komprimiert werden. In diesem Fall ließ sich die Anzahl der Polygone, aus denen die 3D-Körper bestehen, um ca. 50 % reduzieren. Ein weiterer wichtiger Schritt ist das Setzen des Drehhauptpunktes (Pivot-Punkt) jeder einzelnen Kachel. Zur besseren späteren Prozessierung in Unity wird der Pivot-Punkt auf das Center jeder Kachel gesetzt.

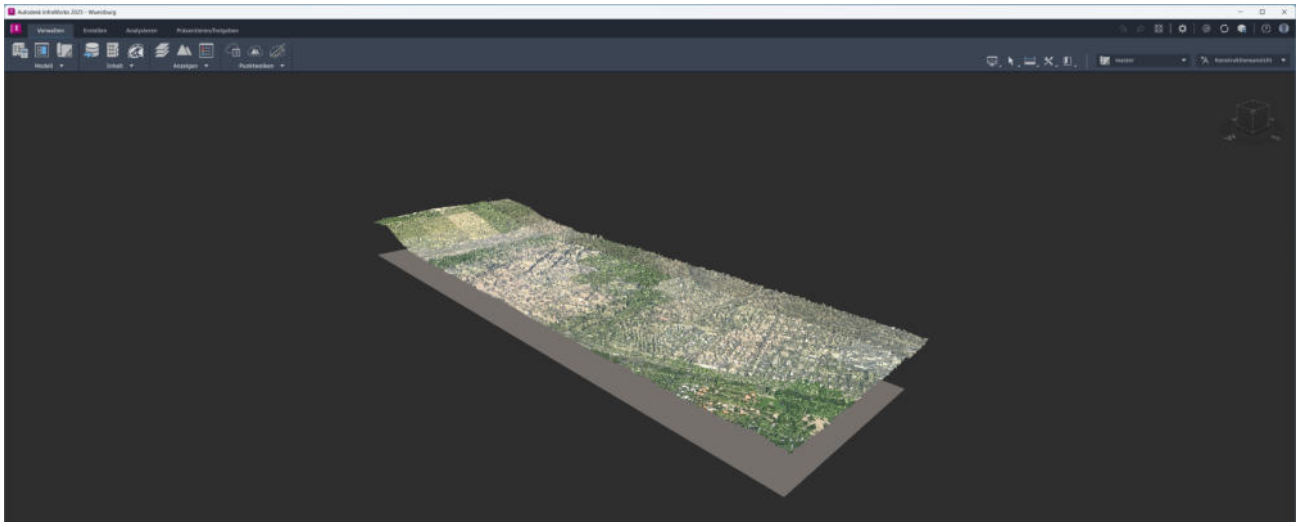

Abbildung 3.3: 3D-Mesh in Infraworks

Abschließend werden die vier 3D-Mesh Kacheln in Infraworks geladen. Hier werden die metrischen Einheiten und das Geobezugssystem gesetzt, die Kacheln als fbx.-Dateien exportiert und in 3ds Max in die von Unity unterstützte Version umgewandelt.

3.3 Vorbereitung Referenzkarte

Eine zentrale Komponente in der späteren Anwendung ist die Referenzkarte 3.4 für das Template Matching. Sie ist hauptausschlaggebend dafür, welche Kartenausschnitte später in der praktischen Umsetzung der Anwendung erkannt werden können. In diesem Fall wird die Webkarte als Referenzkarte herangezogen. Dabei muss darauf geachtet werden, dass die Symbolisierung im richtigen Maßstab ist. Bei modernen Webkarten passt sich die Beschriftung automatisch an, um eine gute Lesbarkeit zu garantieren. Aufgrund dessen muss die Ausgabe

der Webkarte so angepasst werden, dass zwar das gesamte Interessengebiet vorhanden ist, aber die Beschriftung so dargestellt wird, dass sie auf den späteren kleineren Kartenausschnitten gut lesbar ist. Ein weiterer wichtiger Punkt ist die Auflösung der Referenzkarte. Sie muss verhältnismäßig hoch sein, damit die Inhalte detailliert wiedergegeben werden können, um ein stabiles Template Matching zu ermöglichen. Für die praktische Umsetzung wird initial eine Auflösung von 26240 x 21504 Pixeln mit einer horizontalen und vertikalen Auflösung von 635 dpi ausgewählt.

Die Erstellung der Referenzkarte 3.4 erfolgt mit dem internen Geodatenbestell-Tool der bayerischen Vermessungsverwaltung. Die resultierende tif. Bild-Datei ist 1,7 GB groß. Die Dateigröße verhindert eine effiziente Verarbeitung in Unity auf Computern und mobilen Endgeräten. Daher wird eine Komprimierung in Photoshop durchgeführt. Mithilfe eines Komprimierungsverfahren kann die Größe der Referenzkarte auf 45 MB verkleinert werden. Eine visuelle Überprüfung zeigt keine gravierenden Unterschiede und ein stabiles Template Matching kann weiterhin gewährleistet werden.

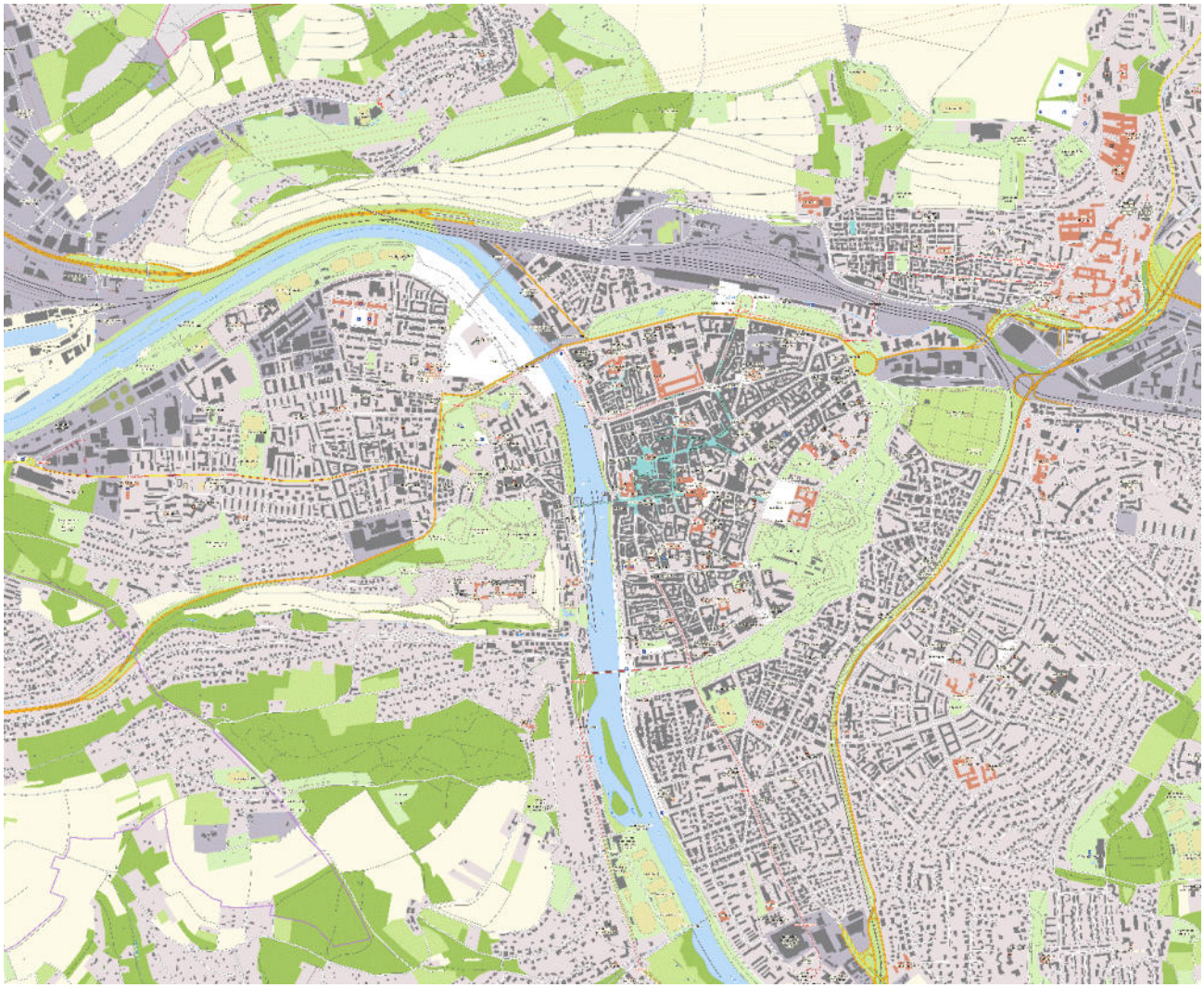

Abbildung 3.4: Referenzkarte des Interessengebiet Würzburg und Umland

4 Vorgehensweise

In diesem Abschnitt wird die praktische Umsetzung des Projektes erläutert. Abbildung 4.1 bietet folgend einen Überblick über die systematische Vorgehensweise. Zur schnelleren Verarbeitung werden die verschiedenen Unity C# Skripte auf verschiedene Unity-Szenen aufgeteilt. Die Szenen-Mechanik in Unity kommt aus dem Game Engine-Hintergrund, und ist dazu gedacht, verschiedene Levels zu managen. Im Kontext dieses vorliegenden Projektes werden Unity-Szenen zur Datenaufnahme, Verarbeitung und AR-Darstellung genutzt.

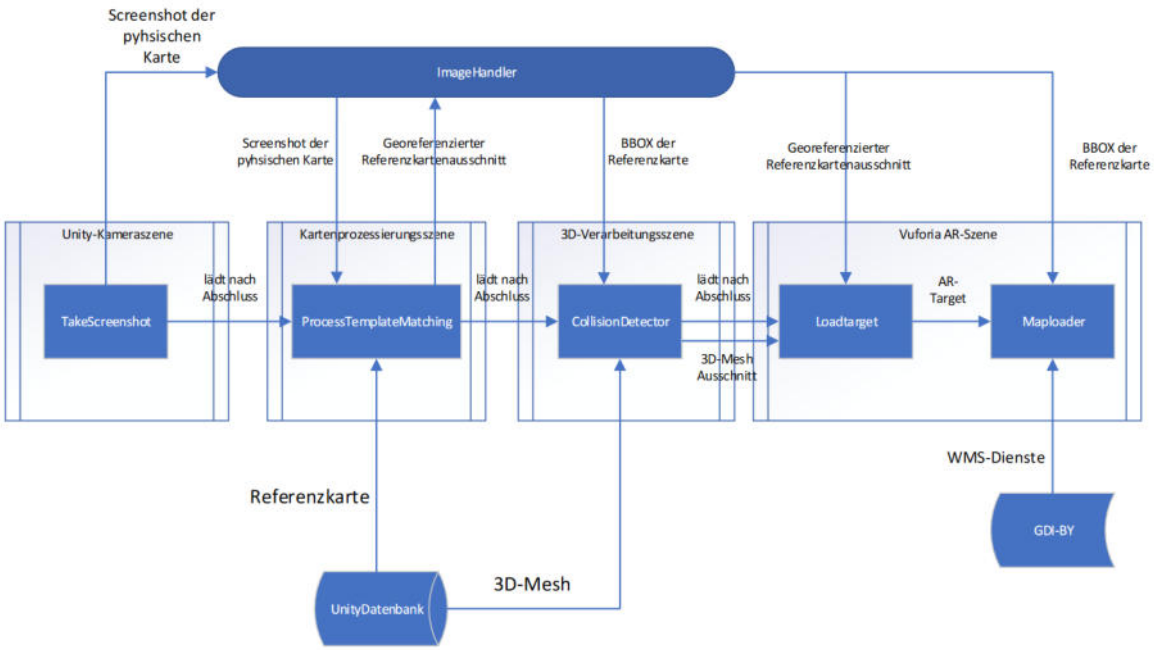

Abbildung 4.1: Systematische Darstellung der Arbeitsprozesse

4.1 Projektvorbereitung

4.1.1 Untersuchungsgebiet

Als Untersuchungsgebiet wird die Stadt Würzburg mit Umland gewählt, wie in Abbildung 3.4 dargestellt. Würzburg ist dank seiner geografischen Lage ideal für Geovisualisierungen geeignet. Es gibt rein urbane Gebiete und außerhalb der Stadtgrenzen verschiedene Vegetationsarten. Ein besonderes Highlight ist außerdem die Würzburg, die durch ihre erhöhte Lage sofort ins Auge fällt. Der Hauptgrund liegt jedoch darin, dass für Würzburg das aktuellste 3D-Mesh innerhalb der bayerischen Vermessungsverwaltung verfügbar ist. Bei einer Oblique-Befliegung beziehungsweise Schrägluftbild-Aufnahme wurde Würzburg mit Kameras aus mehreren Perspektiven fotografiert, dadurch ergibt sich der Vorteil, dass bei einer späteren Berechnung eines 3D-Meshs auch Texturen für vertikale Wände interpolationsfrei berechnet werden können. Diese Befliegung ist nicht Teil der Bayernbefliegung, sondern eine Sonderbefliegung mit 5500 Luftbildern und einer Bodenpixelgröße von etwa 7 cm.

4.1.2 Software

Die praktische Umsetzung der AR-Anwendung erfolgt in der Game Engine Unity. Dafür wird die Version 2021.3.10f1 ausgewählt. Diese Version von Unity bietet eine stabile und dauerhaft gepflegte Entwicklungsumgebung mit einer umfangreichen Unterstützung für verschiedene Plattformen und eine Vielzahl von Tools, die den Entwicklungsprozess erleichtern. Um die App auf Android-Geräten lauffähig zu machen, wird die Android Build-Unterstützung in Unity installiert. Dies ermöglicht es, die App direkt für Android zu entwickeln, zu testen und zu debuggen. Die Installation der Android Build-Unterstützung erfolgt über die Software Unity Hub, die Konfigurationen der Android Software Development Kit (SDK) und Java Development Kit (JDK) verbleiben bei den Unity Default Versionen, um eine optimale Leistung und Kompatibilität sicherzustellen. Eine Anpassung der SDK und JDK ist jederzeit über das Entwicklungstool Android Studio möglich. Dies war jedoch im Verlauf des Projekts nicht notwendig. Android Studio wird nur zum Auslesen von Log-Dateien und zur allgemeinen Konfiguration der Testgeräte verwendet.

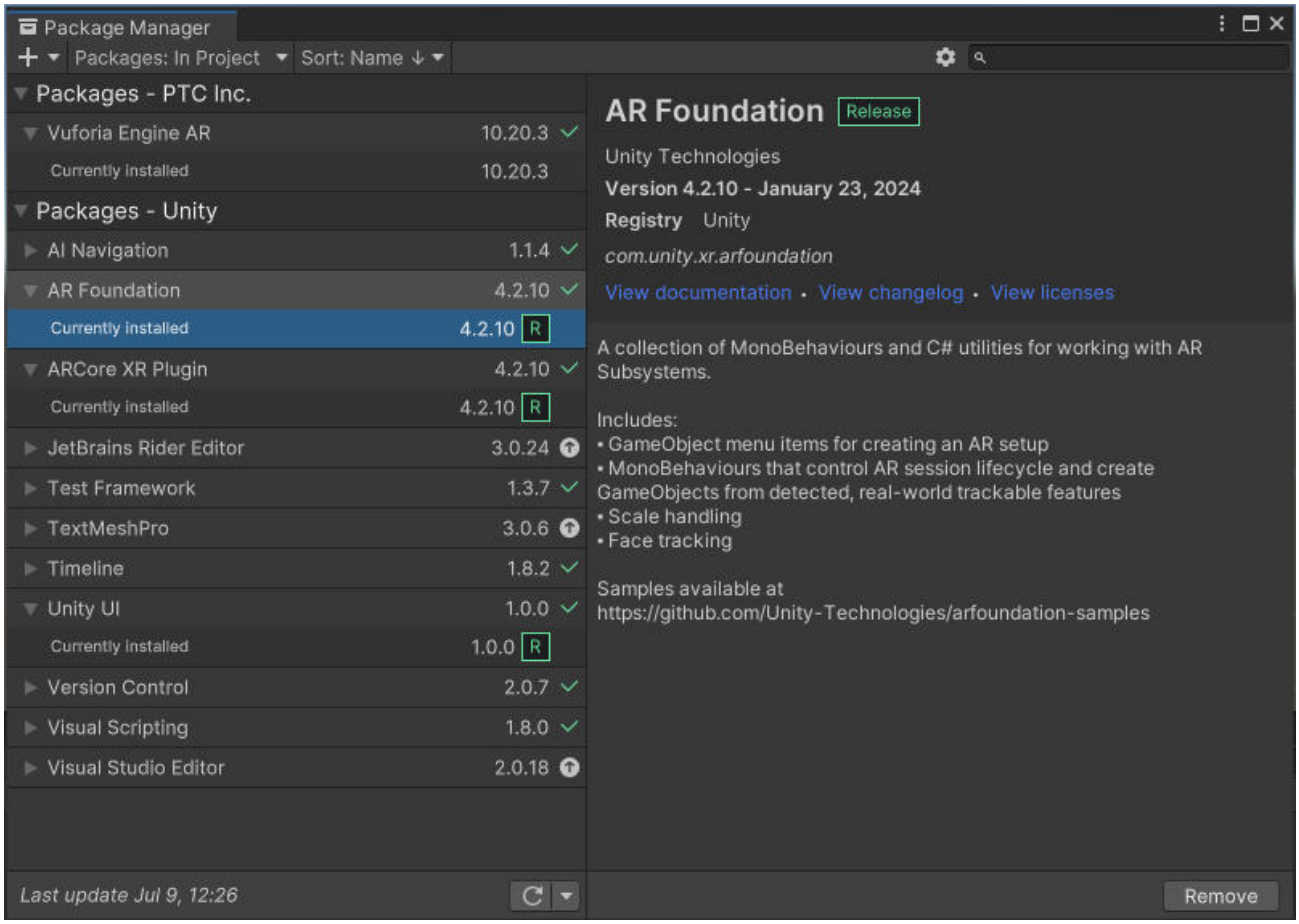

Abbildung 4.2: Übersicht über die Unity Plug-in Packages

Um alle angedachten Funktionalitäten der App umsetzen zu können, müssen mehrere Plug-ins in das Projekt integriert werden.

Das erste Plug-in, AR Foundation, wird verwendet, um grundlegende Augmented Reality-Funktionen zu implementieren. Es handelt sich um das kostenlose Standard AR Entwicklungstool von Unity. AR Foundation bietet ein einheitliches Application Programming Interface (API) für verschiedene AR-Plattformen wie ARKit und ARCore, wodurch die Entwicklung plattformübergreifender AR-Anwendungen erleichtert wird. Die Integration von AR Foundation ermöglicht es, AR-Erlebnisse zu schaffen, die sowohl auf iOS- als auch auf Android-Geräten funktionieren. Das effiziente Entwickeln für eine große Bandbreite an potenziellen Endgeräten ist somit gewährleistet. In dieser Arbeit dient es dazu die primäre AR-SDK Vuforia zu unterstützen, aber auch den Kamerazugriff auf dem späteren Mobilgerät unkompliziert zugänglich zu machen.

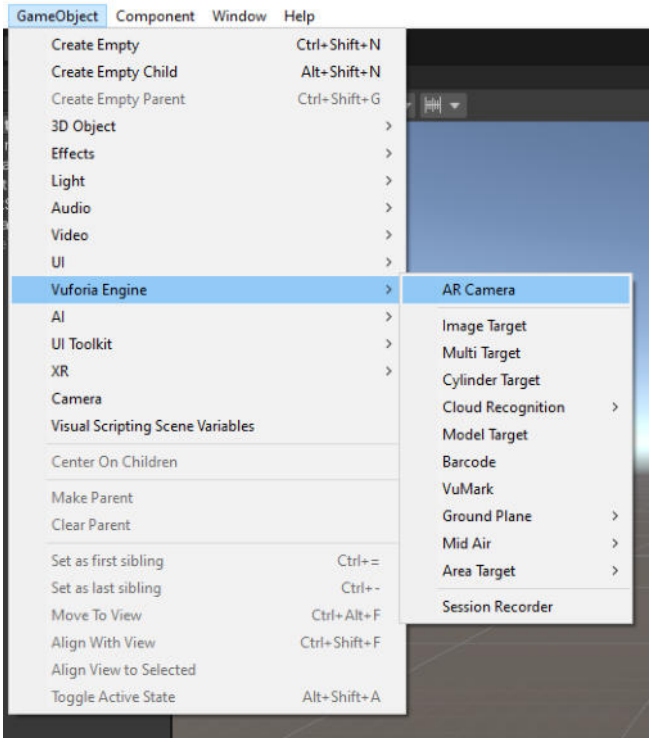

Abbildung 4.3: Vuforia Entwicklungsmöglichkeiten

Das Plug-in, das hauptsächlich für den AR Workflow verantwortlich ist, ist Vuforia. Es bietet fortschrittliche Bilderkennung und Tracking-Funktionalitäten, die auch bei schwierigen Umgebungsbedingungen stabile Ergebnisse liefern. Es kann genutzt werden, um markerbasierte AR-Erlebnisse zu erstellen. Zusätzlich bietet Vuforia leistungsstarke Features wie die Model Target-Erkennung, die das Tracking komplexer 3D-Modelle ermöglicht, sowie eine benutzerfreundliche Oberfläche für die einfache Verwaltung und Implementierung von AR-Inhalten. Die Integration von Vuforia in das Projekt umfasst die Registrierung und Einrichtung eines Vuforia-Entwicklerkontos, das Herunterladen und die Konfiguration des Vuforia SDK innerhalb von Unity. Für die besten Entwicklungsmöglichkeiten kommt die aktuelle Version 10.20.3 von Vuforia zum Einsatz.

Zusätzlich wird das Plug-in OpenCV+ integriert. OpenCV ist eine Bibliothek, die für Echtzeit-Computer-Vision-Anwendungen verwendet wird. Das OpenCV+ Plug-in bietet erweiterte Bildverarbeitungsfunktionen, die weit über die Standardmöglichkeiten von Unity hinausgehen. Es handelt sich um das einzige kostenlose OpenCV Plug-in für Unity. Die Integration von OpenCV+ umfasst die Installation des Plug-ins, die Einrichtung der entsprechenden Bibliotheken und das Umbenennen verschiedener Klassen, die mit den AR Plug-ins in Kon-

flikt standen. Außerdem wird eine ältere Dynamic Link Library (DLL) Datei aus einem anderen Package manuell installiert und konfiguriert, um App Builds auf Android ARM64 zu erlauben.

4.2 Computer Vision Prozessierung

4.2.1 Einladen der Referenzkarte

Zunächst muss die Referenzkarte in Unity geladen werden, was per Drag-and-Drop geschehen kann. Bilder werden hierbei automatisch in eine 2D-Textur umgewandelt. Um später von den C# Skripten verarbeitet werden zu können, muss die Konfiguration der importierten Daten angepasst werden. Das Feld „None Power of 2“ wird auf „None“ gesetzt, da sonst Verzerrungen entstehen. Das Feld „Read/Write“ wird aktiviert, um eine Veränderung während der Laufzeit zuzulassen. Unity unterstützt nur eine maximale Texturauflösung von 16384 x 16384 Pixeln, die eingeladene Referenzkarte wird automatisch heruntergerechnet. Des Weiteren lässt sich insbesondere für Android Builds eine Kompression einstellen. In diesem Fall wurde das Preset „Fast“ verwendet. Bei der späteren Optimierung des Template Matchings kann so die Auflösung der Referenzkarte schrittweise heruntergesetzt werden. Es stellte sich heraus, dass eine Auflösung von 4096 x 4096 Pixeln ausreicht, um ein exaktes Ergebnis beim Template Matching zu erreichen. Die Laufzeit der Computer Vision Prozessierung wird so auf ein Drittel reduziert, ebenso kann der Speicherbedarf der App auf Androidgeräten verringert werden.

In der Übersicht 4.1 ist die Referenzkarte in der „Unity-Datenbank“ zu verordnen.

4.2.2 Aufnahme eines Screenshots

Damit die Position der physischen Karte in der Referenzkarte ermittelt werden kann, muss ein Screenshot des Bildschirms des jeweiligen Endgerätes aufgenommen werden. Dafür wird die erste Szene der Unity-App aufgesetzt. In 4.1 entspricht dieser Vorgang der Szene „Unity-Kameraszene“. Damit auf dem Bildschirm das aktuelle Livebild der Kamera angezeigt wird, muss eine AR-SDK hinzugefügt werden. Das schnellste Ergebnis liefert hier das Plug-in ARCore.

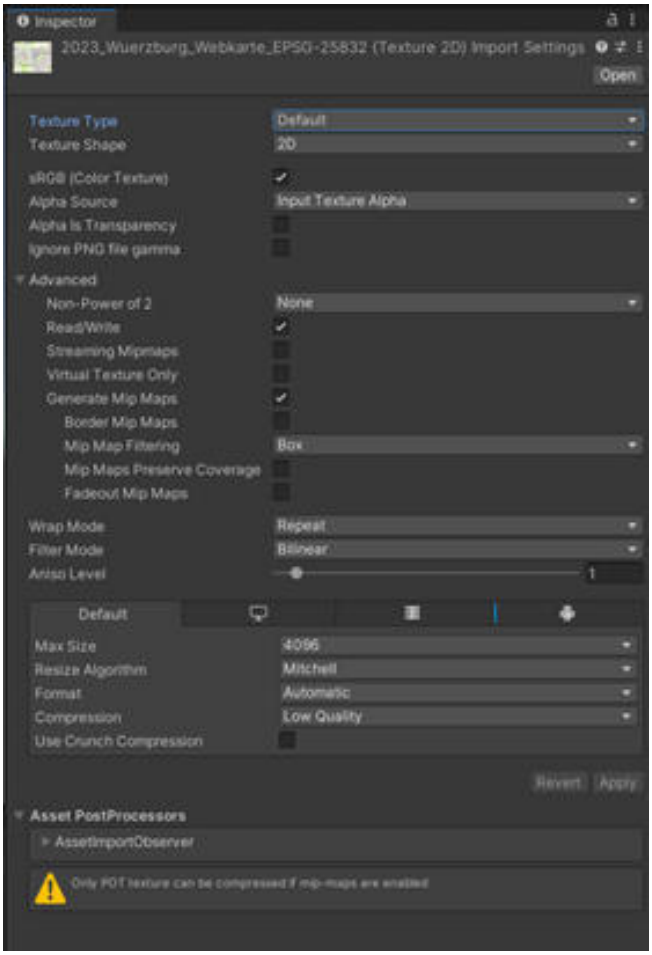

Abbildung 4.4: Konfiguration der Referenzkarte

In Abbildung 4.5 werden die Komponenten der Szene dargestellt. Die normale Unity-Kamera wird entfernt und stattdessen die „AR Session“, das „AR Session Origin“ mit der „AR Camera“ und die „AR Default Plane“ hinzugefügt. Das Endgerät stellt die Kameraaufnahme somit umgehend dar.

An den Rändern des User Interfaces werden semitransparente Balken, die einen Rahmen ergeben hinzugefügt. Dieser zeigt dem User, in welchem Teil des Bildschirms sich der physische Kartenausschnitt zur Bildaufnahme befinden soll. Der Benutzer hat so die Möglichkeit, den Zeitpunkt der Screenshot-Aufnahme selbst zu bestimmen, wofür dem User Interface zwei Buttons hinzugefügt werden.

Für die Erstellung und Verarbeitung der Aufnahme wird ein C#-Skript erstellt. Zu Beginn werden die Maße eines Rechtecks definiert, welches dem Ausschnitt entspricht, in dem die physische Karte im User Interface dargestellt wird. Es dient später beim Ausschneiden als Maske. In Unity werden Screenshots immer

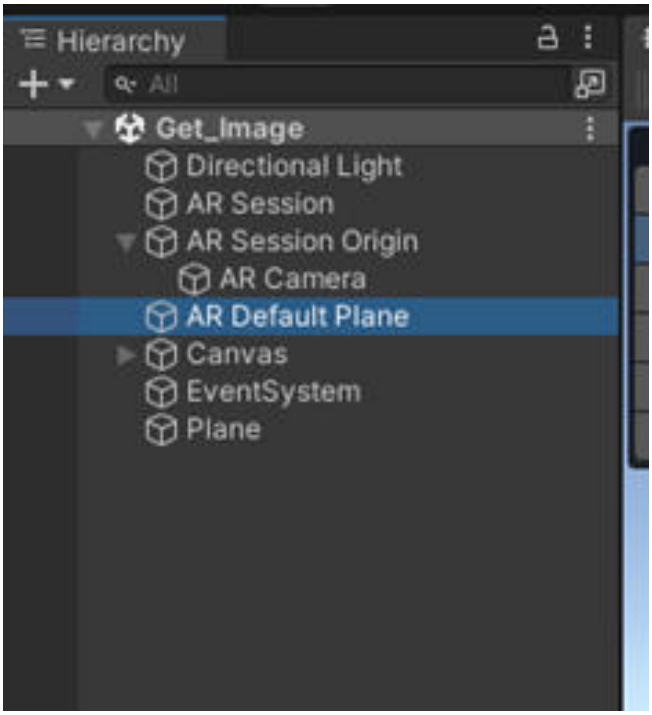

Abbildung 4.5: Aufbau der Szene mit ARCore

in den sogenannten „persistentDatapath“ gespeichert, welcher stets konstant ist und sich bei einem Wechsel der Szene nicht ändert. Wird die App geschlossen und wieder geöffnet, bleiben die Dateien in diesem Pfad trotzdem erhalten. Die Pfade können je nach Build-Plattform variieren. Für eine bessere Performance und Speichergröße der App werden nach dem Starten des Skriptes alle Dateien innerhalb des Ordners gelöscht. In der realen Umsetzung dieses Projekts handelt es sich nur um den alten Screenshot der physischen Karte der vorherigen Aufnahme.

Nachdem dieser Vorgang abgeschlossen ist, wird eine Unity Coroutine Methode gestartet. Im Normalfall wird in Unity die Spielszene so lange pausiert, bis alle Methoden abgeschlossen sind. Dauert die Ausführung länger, wie z.B. in diesem Fall durch das Abspeichern eines Screenshots, ist die Anwendung für diesen Zeitraum unterbrochen. Die Coroutine hat die Eigenschaft, dass sie bei längeren Laufzeiten die Anwendung nicht anhält, sondern von der Klasse nebenher ausgeführt wird. In der vorliegenden Anwendung wird zunächst ein String für den Filenamen des Screenshots erstellt. Dieser besteht aus dem Datum und der Uhrzeit der Aufnahme. Der eigentliche Screenshot wird mit der Funktion „ScreenCapture.CaptureScreenshot“ ausgeführt. Als Parameter wird der vorher erstellte Name übergeben. Nachdem das Speichern abgeschlossen ist, muss ein

weiterer Berechnungszyklus abgewartet werden, damit andere Prozesse ebenso auf den Screenshot zugreifen können.

Eine zweite Coroutine wartet zur gleichen Zeit auf die Screenshot-Datei. Sobald diese verfügbar ist, wird sie als Byte stream eingelesen. Im Anschluss wird dieser in eine Textur umgewandelt. Die Textur wird der ImageHandler-Klasse zur weiteren Bearbeitung übergeben und zuletzt das Laden der „Kartenprozessierungsszene“ ausgelöst.

4.2.3 Template Matching

Das Template Matching wird in der „Kartenprozessierungsszene“ ausgeführt, siehe Abbildung 4.1. Der Benutzer hat in dieser Szene keinerlei Einflussmöglichkeiten. Mit Beginn der Szene wird das Skript „ProcessTemplateMatching“ gestartet, im ersten Schritt wird die Bounding Box der Referenzkarte in UTM-Koordinaten festgelegt. Diese Koordinaten müssen manuell im Editor oder in ein Skript eingetragen werden, da das Einlesen eines World Files hier zeitlich nicht umgesetzt werden konnte. Alternativ können die Koordinaten der Bounding Box ermittelt werden, indem die Referenzkarte in ein GIS-Programm geladen wird. Die Referenzkarte muss auch im Editor dem Skript hinzugefügt werden. Anschließend wird der bereits erstellte Screenshot dem ImageHandler entnommen. Damit eine Verarbeitung mit dem OpenCV+ Plug-in stattfinden kann, muss eine Umwandlung der Unity Texturen in OpenCV Materialien stattfinden, welche sowohl für die Referenzkarte als auch für den Screenshot durchgeführt wird. Der Screenshot wird nun für die weitere Verarbeitung zugeschnitten. Das Rechteck wurde bereits in der „Unity-Kameraszene“ definiert.

Ein wesentlicher Nachteil des Template Matchings ist die Empfindlichkeit gegenüber Maßstabsveränderungen zwischen Templatebild und Referenzbild. Diese Problematik wird jedoch umgangen, indem über eine Bandbreite an Maßstäben der Templatekarte iteriert wird.

In Abbildung 4.1 wird der systematische Vorgang visualisiert. Für das Template Matching findet die OpenCV Methode „Cv2.MatchTemplate“ Anwendung, als Ähnlichkeitsmetrik werden hierbei die normalisierten Korrelationskoeffizienten 2.6 verwendet.

Nun wird eine Schleife für das Template Matching geschrieben. Für jeden Durchgang dieser wird eine beste Platzierung errechnet, die einen Score enthält. Je

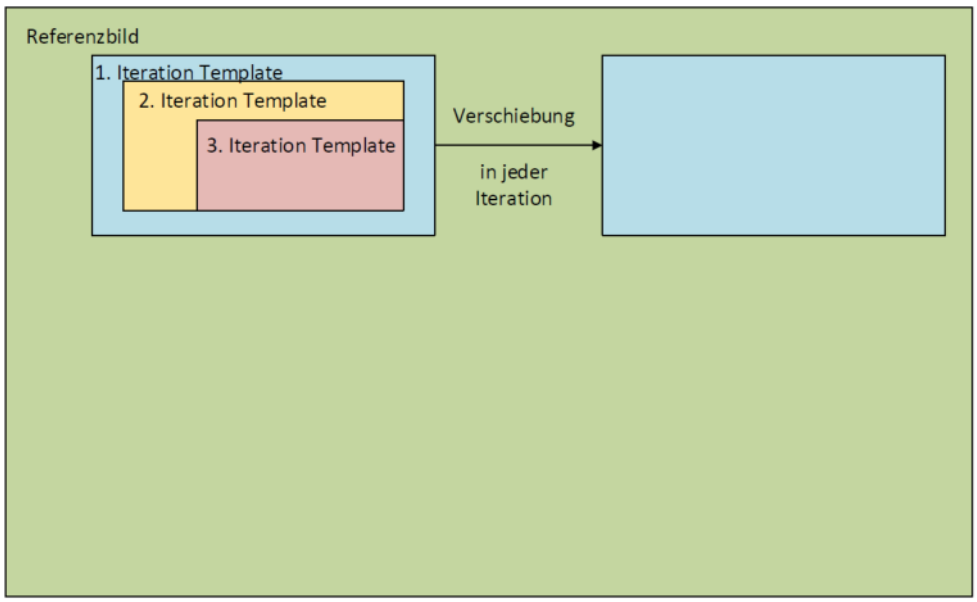

Abbildung 4.6: Vorgehen Template Matching

höher der Score ist, desto wahrscheinlicher handelt es sich um die richtige Platzierung des Templates. Die Scores eines jeden Durchgangs werden miteinander verglichen, der höchste Score ist sehr wahrscheinlich die richtige Position des Templates in der Referenzkarte. Um die Performance zu verbessern, wird ein Schwellenwert eingebaut, welcher annähernd dem Scorewert der richtigen Platzierung entspricht und durch Erfahrungswerte beim Testen bestimmt wurde. Sobald der Score höher als der Schwellenwert ist, bricht der Algorithmus den Suchvorgang ab und liefert die Position der Templatekarte.

Durch die Bestimmung der Position der Templatekarte kann diese nun aus der Referenzkarte ausgeschnitten werden. Dieser Ausschnitt wird dem Nutzer zur visuellen Überprüfung kurz angezeigt. Damit der fertige Ausschnitt später verarbeitet werden kann, wird er im nächsten Schritt georeferenziert. Dadurch sind nun die Ausmaße der Referenzkarte in Pixeln, die Position der Templatekarte in Pixeln und die Ausmaße der Referenzkarte in UTM-Koordinaten bekannt. Somit kann die Transformation über ein Verhältnis errechnet werden.

Der fertige Ausschnitt der Referenzkarte wird anschließend dem ImageHandler übergeben, ebenso die Bounding Box des Ausschnittes der Referenzkarte. Sobald alle Vorgänge abgeschlossen sind, wird die „3D-Verarbeitungsszene“ geladen.

4.3 3D-Verarbeitung

4.3.1 3D-Meshimport

Die Vorbereitung des 3D-Meshs wurde in Kapitel 3.2 bereits erläutert. Fbx.-Dateien können in Unity über Drag-and-Drop importiert werden. Nach diesem Schritt ist jedoch nur die Geometrie des Modells ohne Texturen sichtbar. Die Texturpfade müssen in den Einstellungen des Meshs noch konfiguriert und die Texturen anschließend importiert werden. Unity erstellt automatisch ein neues Unterverzeichnis, in dem alle Texturen abgespeichert werden. Um die Ladezeiten der späteren App zu verringern, wird das Mesh in Unity bezüglich der Laufzeit optimiert, wobei die Mesh-Kompression auf „Medium“ gestellt wird. Bei einer vergleichenden Betrachtung mit dem bloßen Auge lassen sich keine Unterschiede zwischen einem normalen und einem komprimierten Mesh in der Android-App feststellen. Die Komprimierung sorgt für eine performante Darstellung beim Ausführen der App. Um auch die Ladezeit zu verringern, wird die maximale Größe jeder Textur auf 1024 x 1024 Pixel begrenzt. Des Weiteren wird die Kompressionsqualität auf „schnell“ gesetzt. Das koordinierte Zusammenspiel aller Maßnahmen gewährleistet eine schnelle und reibungslose Performance des 3D-Meshs in der Endanwendung. In Abbildung 4.1 ist das importierte Mesh in der Unity-Datenbank abgebildet.

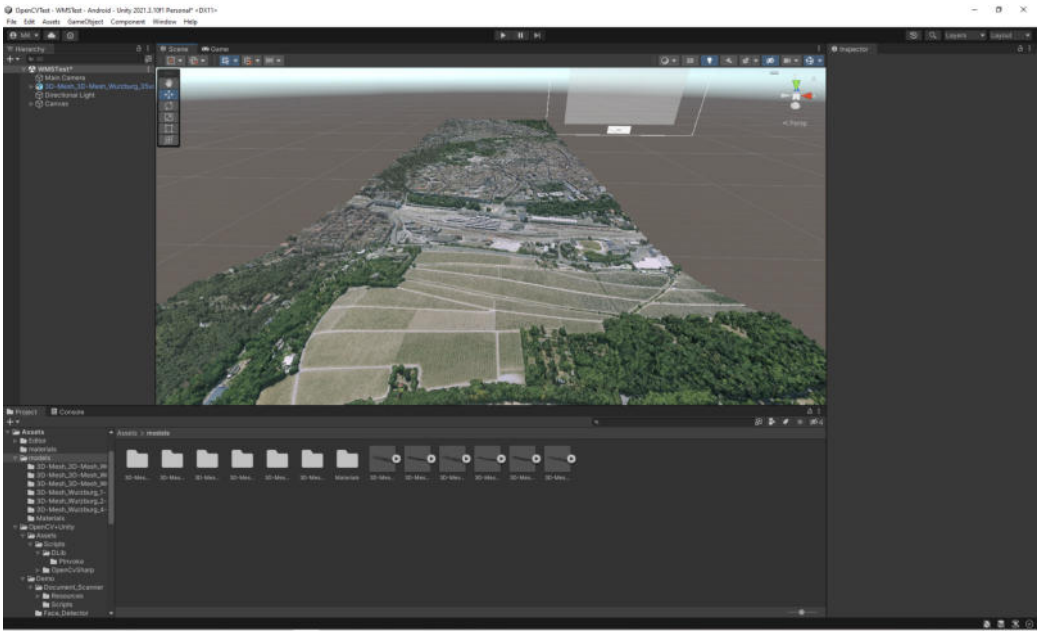

Abbildung 4.7: 3D-Mesh im Unity Editor

4.3.2 Bestimmung 3D-Meshausschnitt

Die Prozessierung findet, siehe Abbildung 4.1 in der „3D-Verarbeitungsszene“ statt. Der Verarbeitungsprozess wird so konzipiert, dass der Mesh-Ausschnitt exakt der räumlichen Ausdehnung der Referenzkarte in 3D entspricht. Zuerst wird das 3D-Mesh in die Szene eingeladen. Ein wichtiger Punkt ist hierbei, dass das Mesh deaktiviert wird, das heißt, es wird auf keine Art und Weise dargestellt. Das bringt den Vorteil mit sich, dass sich die Ladezeit der Szene deutlich verkürzt und Prozesse währenddessen schneller ablaufen können. Trotz all dieser Maßnahmen war es nicht möglich, das komplette Interessensgebiet als 3D-Mesh zu integrieren, da Unity nur eine maximale Anwendungsgröße von 4GB erlaubt. Um diesen Wert zu unterschreiten, wurden die äußeren Kacheln des Meshs entfernt. Es ist in der Anwendung somit nur das Stadtzentrum verfügbar.

Zunächst muss die Georeferenzierung des Meshs hinzugefügt werden, wofür die Bounding Box des unverarbeiteten Meshs verwendet werden kann. Dies muss jedoch außerhalb von Unity in einer Viewing Software wie Cloud Compare erfolgen. Die Minimal- und Maximalwerte der Bounding Box werden ihren korrespondierenden Variablen in Unity zugewiesen und die UTM-Koordinaten des aktuellen Kartenausschnittes werden aus dem ImageHandler geladen, siehe Abbildung 4.1. Es folgt eine Prüfung, ob sich der Kartenausschnitt auch innerhalb des vorhandenen Meshbereichs befindet. Sollte dies nicht der Fall sein, wird der Verfügbarkeitsstatus des Meshs auf „negativ“ gesetzt und anschließend die nächste Szene geladen. Wenn das Mesh für diesen Ausschnitt verfügbar ist, wird die Prozessierung angestoßen. Dazu muss zuerst das Ausmaß des 3D-Meshs in Unity-Koordinaten bestimmt werden, wofür die Bounds-Eigenschaft des Unity-Renderers verwendet wird. In einer Hilfsfunktion werden die Bounds aller einzelnen Kacheln zu einem großen Objekt zusammengefasst, mit welchem sich die Größe des gesamten 3D-Meshs in Unity Koordinaten ausgeben lässt. Mit diesen Maßen kann der Ausschnitt in die relativen Koordinaten des 3D-Meshs umgerechnet werden. Dies wird in einer Hilfsfunktion für den Minimal- und Maximalpunkt der Bounding Box des Ausschnittes ausgeführt. Anschließend werden die Werte so angepasst, dass der Nullpunkt sich in der unteren linken Ecke des Ausschnittes befindet.

Im nächsten Schritt wird nun ein Quader-Gameobjekt im Skript referenziert. Es erhält den Mittelpunkt und die Ausmaße des in Unity Koordinaten umgerechneten Kartenausschnittes. Um alle Kacheln zu finden, die im Ausschnitt

enthalten sind, muss eine Pufferzone hinzugefügt werden, die annähernd einer Kachelbreite entspricht. Dies ist notwendig, da bei der späteren Berechnung stets nur die Mittelpunkte aller Kacheln abgefragt werden und ohne diese Pufferzone kleine Löcher an den Rändern des Ausschnittes entstehen könnten. Die eigentliche Berechnung findet in einer Foreach-Schleife statt. In dieser wird über jede Position aller Kacheln iteriert und festgestellt, ob sich die jeweilige Position innerhalb des Kartenausschnitts mit Pufferzone befindet. Sollte dies zutreffen, wird die jeweilige Kachel aktiv geschaltet und dem vorher genannten Quader untergeordnet.

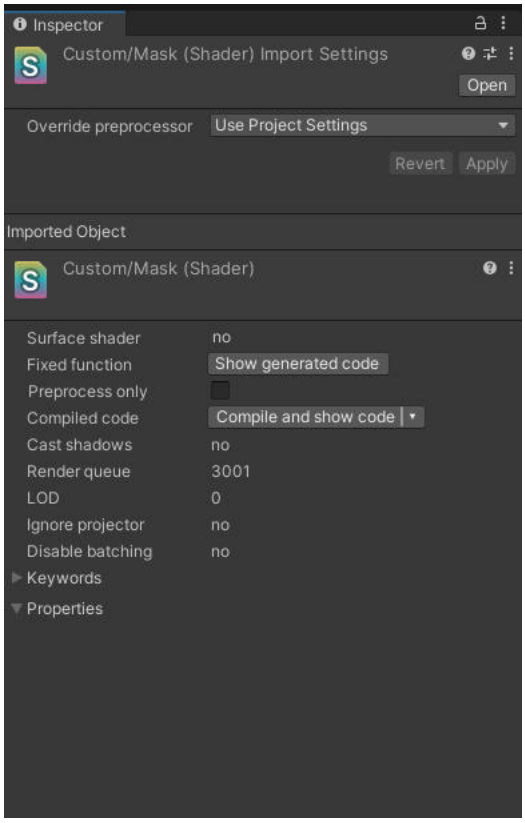

Abbildung 4.8: Konfiguration des Shaders

Zum jetzigen Verarbeitungsstand sind alle benötigten Mesh-Kacheln ausgewählt worden. Problematisch ist jedoch, dass Randkacheln über den Ausschnitt in die Pufferzone hinausstehen können. Ein Zuschneiden der 3D-Körper ist in Unity leider nicht möglich, Maskieren erweist sich hier als die beste Lösung. Dazu werden einige Vorbereitungen getroffen. Zuerst muss ein benutzerdefinierter Shader erstellt werden, welcher sich in der „Renderer Queue“ nach den normalen transparenten Materialien einreiht. Beim Abspeichern ergibt sich ein Einordnen auf Platz 3001 aller Materialien. Das Objekt, das dieses Material erhält, wird die

später zugeordneten Objekte verdecken. Das Material beeinflusst keine unbeabsichtigten Gameobjekte, die Objekte, die maskiert werden sollen, müssen durch ein Skript erfasst werden. In diesem Skript werden die Materialien der jeweiligen Objekte auf Platz 3002 in der „Renderer Queue“ gesetzt. Dies hat zur Folge, dass die unerwünschten Gameobjekte jetzt hinter dem Gameobjekt mit dem Masking-Material verschwinden.

Im Skript zur Verarbeitung des 3D-Meshs wird ein Quader mit dem genannten Masking-Material hinzugefügt. Dieser wird viermal instanziiert und um den Hauptquader mit dem 3D-Mesh platziert. Jetzt ist das 3D-Mesh nur für den gewünschten Ausschnitt sichtbar. Für eine bessere Ladegeschwindigkeit werden alle Mesh-Ausschnittelemente mit dem Tag „Respawn“ versehen und beim Hauptquader die Methode „DontDestroyOnLoad“ aktiviert. Dadurch kann die nächste Szene geladen werden, ohne dass das 3D-Mesh verschwindet. Alle vorherigen Bearbeitungen werden beibehalten.

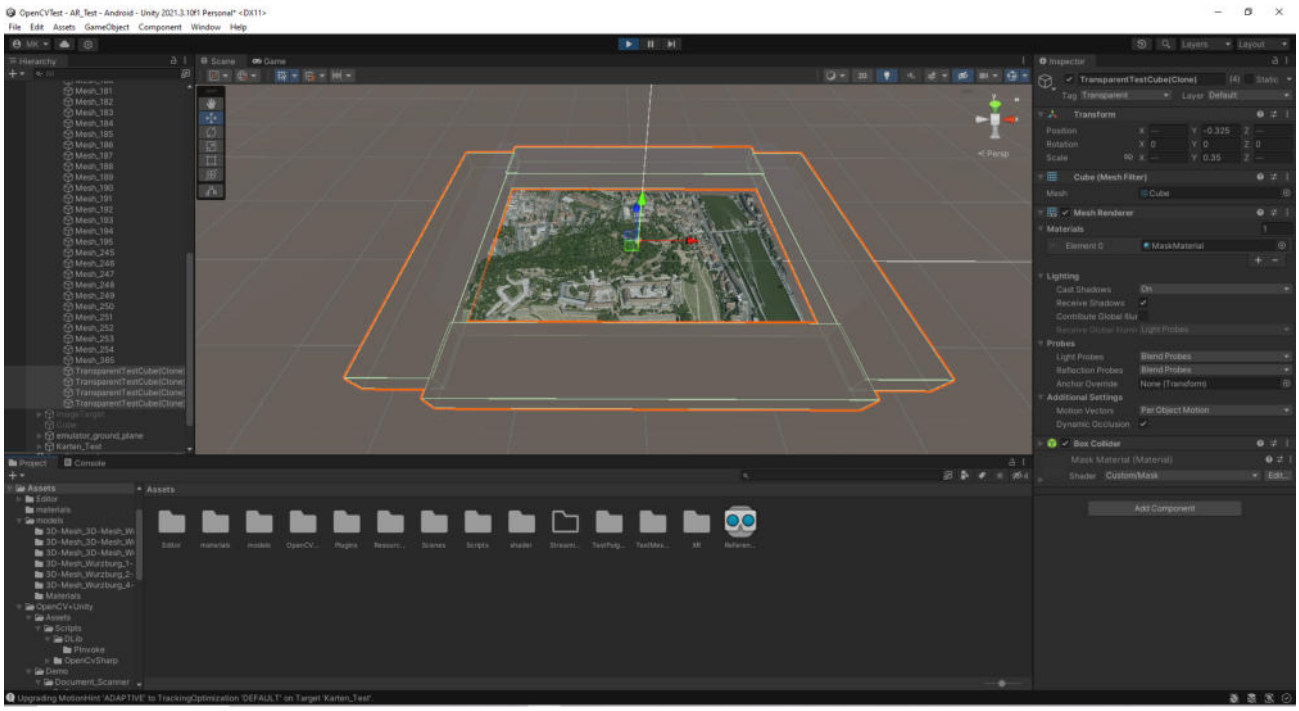

Abbildung 4.9: Mesh mit transparenten Quadern

4.4 AR Workflow

In Abbildung 4.1 wird sichtbar, dass in der letzten Unity Szene „Vuforia AR-Szene“ zwei Skripte vorhanden sind. Dies ist nötig, da beim Start der Szene die AR Runtime initialisiert wird. Für diese Szene wird das AR Software Development Kit Vuforia verwendet. In Abbildung 4.10 werden die Komponenten der Szene dargestellt. Die ARCamera ermöglicht den einfachen Zugriff auf alle Kamerasysteme des Endgerätes. Im Canvas_Parent sind die verschiedenen User Interfaces enthalten, die eine Benutzeroberfläche für die Darstellungen mit und ohne Mesh ermöglichen. Im AR_Parent sind alle Objekte untergeordnet, die später in der AR-Umgebung dargestellt werden. Die restlichen Gameobjekte dienen zum Erstellen oder Debuggen der Anwendung.

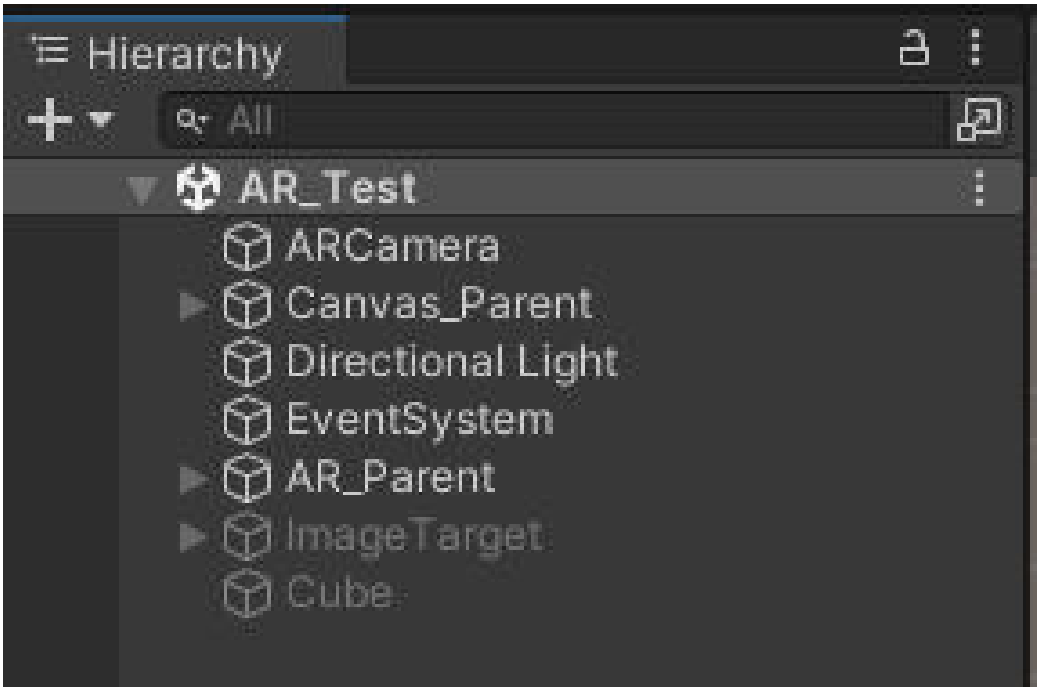

Abbildung 4.10: Aufbau der Szene mit Vuforia

Das Skript „Loadtarget“ siehe 4.1 ist für die Initialisierung aller AR-Systeme zuständig. Es wird geprüft, ob für den gegenwärtigen Ausschnitt das 3D-Mesh verfügbar ist. Je nachdem welcher Fall zutrifft, wird das passende User Interface aktiviert. Ist das Mesh verfügbar, so wird das übergebende Gameobjekt mit dem Mesh-Ausschnitt mithilfe des Tags „Respawn“ gesucht. Die Größe des Mesh-Ausschnitts wird wie zuvor beschrieben mit der „Mesh.bounds“ Methode errechnet, woraus sich ein Verhältnis zur Größe des AR_Parents ableiten

lässt, anhand dessen das Mesh entsprechend angepasst werden kann. Außerdem wird die Höhe des Meshs mit der des AR_Parent verglichen und so angepasst, dass sie genau übereinanderstehen. So wird garantiert, dass das Mesh immer bündig über dem späteren physischen Marker schwebt. Sobald dies abgeschlossen ist, kann ein AR Target erstellt werden. Dabei handelt es sich um die Karte, die später in der realen Welt erkannt werden soll. In diesem Fall wird vom ImageHandler der Ausschnitt der Referenzkarte geladen. Mithilfe der Funktion „VuforiaBehaviour.Instance.ObserverFactory.CreateImageTarget“ kann aus der Textur ein AR Target erstellt werden. Nachdem dies abgeschlossen ist, müssen dem AR Target die 3D-Inhalte zugeordnet werden. Dazu wird dem AR Target eine Instanz des AR_Parent untergeordnet. Danach ist die Initialisierung abgeschlossen.

4.5 Netzwerkprogrammierung WMS-Request

Um die Raster-Geobasisdaten aus dem Internet herunterzuladen, muss ein WMS-Request gestellt werden. Dieser erfolgt in Form einer URL. In Abbildung 4.1 ist ersichtlich, dass dies vom „Maploader“ durchgeführt wird. Zunächst muss definiert werden, welche Rasterdaten in der späteren App angezeigt werden sollen. Für die praktische Umsetzung wurde sich für das digitale Orthophoto, das digitale Geländemodell und die digitale Orthokarte der bayerischen Vermessungsverwaltung entschieden. Für jeden Typ wird die Base URL definiert, welche aus dem Domain- und Servernamen sowie dem angehängten Dateipfad besteht. Diese Base URLs unterscheiden sich durch verschiedene Dateipfade. Nachdem diese als Variablen definiert worden sind, können die Parameter für den Query-String erstellt werden. Dieser wird durch ein Fragezeichen „?“ in der URL eingeleitet und ermöglicht es, dass Parameter an den Server übergeben werden können. In der praktischen Umsetzung müssen mehrere Parameter weitergeleitet werden. Für jeden Raster-Geobasisdatentyp ist es nötig, einen individuellen Layer auszuwählen. Für das Orthophoto wird das Farbbild gewählt, für die beiden anderen WMS-Dienste die jeweilige Standardansicht. Die restlichen Parameter sind für alle Abfragen identisch. Als Geobezugssystem wird UTM32 verwendet. Die Bildauflösung entspricht der Auflösung des aufgenommenen Kartenausschnitts (siehe Kapitel 4.2), als Bildformat wird jpeg. verwendet. Der Bereich, der später dargestellt werden soll, wird durch eine Bounding Box definiert. Dazu wird die bereits bekannte Bounding Box aus dem ImageHandler aufgerufen und die Ko-

ordinaten werden so modifiziert, dass sie dem Syntax einer URL entsprechen. Nachdem all diese Werte vorhanden sind, kann die eigentliche Abfrage stattfinden. Dafür wird erneut eine Coroutine verwendet, um die Hauptanwendung währenddessen nicht zu unterbrechen. Zugleich müssen Fehler, die durch eine mangelhafte Internetverbindung auftreten, abgefangen werden. Die Abfrage erfolgt mit der Methode „UnityWebRequestTexture.GetTexture“. Sobald diese erfolgreich ausgeführt wird, kann der Byte stream in eine Textur umgewandelt werden. Die Textur wird einem Gameobjekt innerhalb des AR_Parents übergeben und auf diesem dargestellt. Um die Ladezeiten zu verkürzen, werden alle Texturen gespeichert und bei erneuter Abfrage aus dem internen Speicher geladen. So kann die Prozessierungszeit verringert und das Datenvolumen des Nutzers geschont werden.

4.6 User Interface

Um die App später steuern zu können, muss ein User Interface erstellt werden. Wie in Abbildung 4.1 ersichtlich, gibt es vier Unity-Szenen, wovon zwei steuerbar sein müssen, während die anderen beiden ausschließlich den Fortschritt der Verarbeitung darstellen.

In der ersten Szene „Unity-Kameraszene“ siehe Abbildung 4.1 muss der Nutzer lediglich eine Eingabe machen. Das User Interface wird durch semitransparente Balken, die einen Rahmen ergeben, an den Rändern begrenzt. Hier soll der physische Kartenausschnitt eingefügt werden. Sobald dies geschehen ist, kann der Screenshot aufgenommen werden. Zu Beginn war der Button zum Aufnehmen des Screenshots unten mittig platziert worden, in Praxistests stellte sich dies jedoch als unpraktikabel heraus. Beim Halten des Endgerätes mit beiden Händen hat der Daumen die meiste Bewegungsfreiheit. Damit eine Aufnahme durch beide Hände erfolgen kann, wurden seitlich Buttons zur Aufnahme platziert. Die Funktionsweise beider Buttons ist identisch.

In der zweiten Szene „Kartenprozessierungsszene“ werden keine Eingaben vom Nutzer verlangt. Während des Template Matchings erscheint nur die Meldung „Bild wird verarbeitet“.

Sobald das Template Matching abgeschlossen ist, wird der gefundene Ausschnitt der Karte gezeigt. Hier kann der Nutzer visuell überprüfen, ob der gefundene Ausschnitt wirklich der physischen Karte entspricht. Die dargestellten Zahlen

Abbildung 4.11: User Interface nach dem Start

geben den Scorewert, den dabei verwendeten Maßstab und die Zeit der Verarbeitung in Sekunden wieder.

In der dritten Szene „3D-Verarbeitungsszene“ wird dem Nutzer die Datenverarbeitung angezeigt. Sollte das Mesh für den jeweiligen Bereich nicht verfügbar sein, erscheint eine Meldung, dass die 3D-Daten nicht verfügbar sind.

In der letzten Szene mit dem Titel „Vuforia AR-Szene“ gibt es zwei User Interfaces. Diese unterscheiden sich durch die Anzahl der Buttons. Für den Fall, dass 3D-Daten verfügbar sind, sowie für den Fall, dass sie nicht verfügbar sind, werden die entsprechenden Buttons geladen. Bei Betätigen des Buttons werden die gewünschten Geodaten abgerufen, die Navigation erfolgt durch Bewegen des Endgeräts.

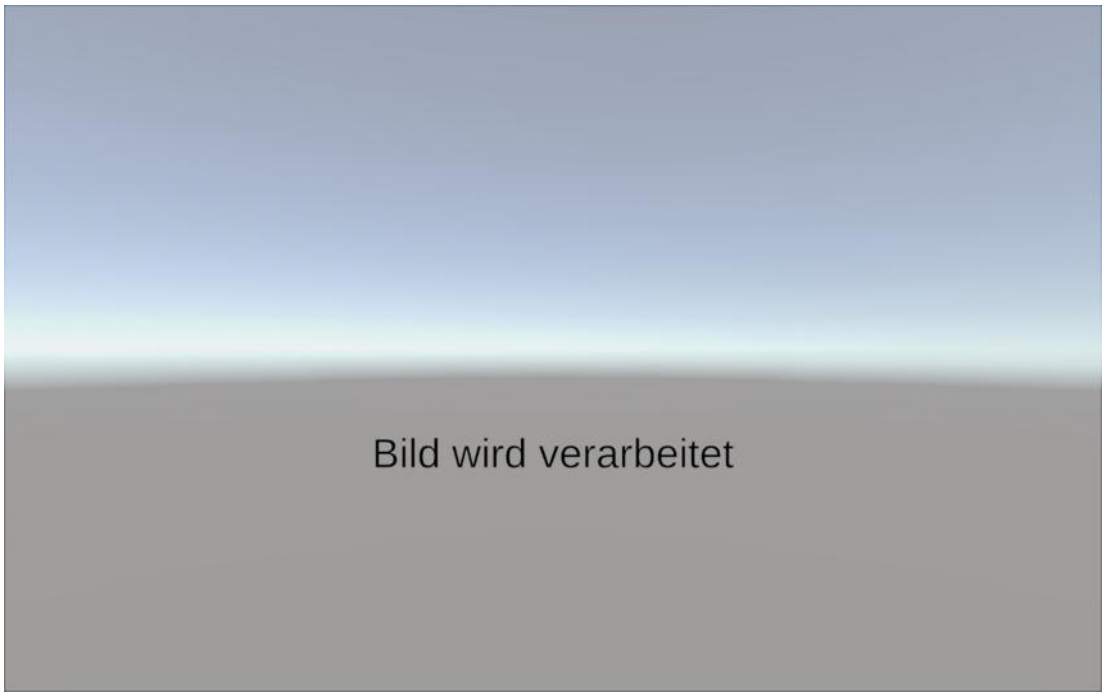

Abbildung 4.12: Bildschirm während des Template Matchings

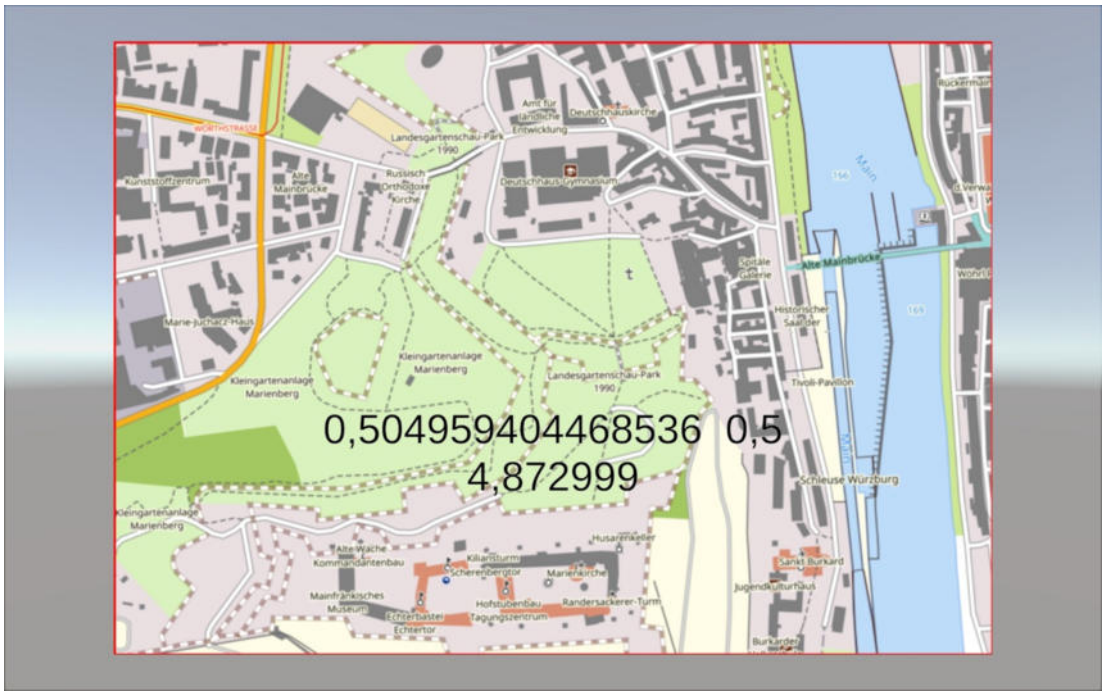

Abbildung 4.13: Template Matching abgeschlossen

Abbildung 4.14: Bildschirm während der 3D-Verarbeitung

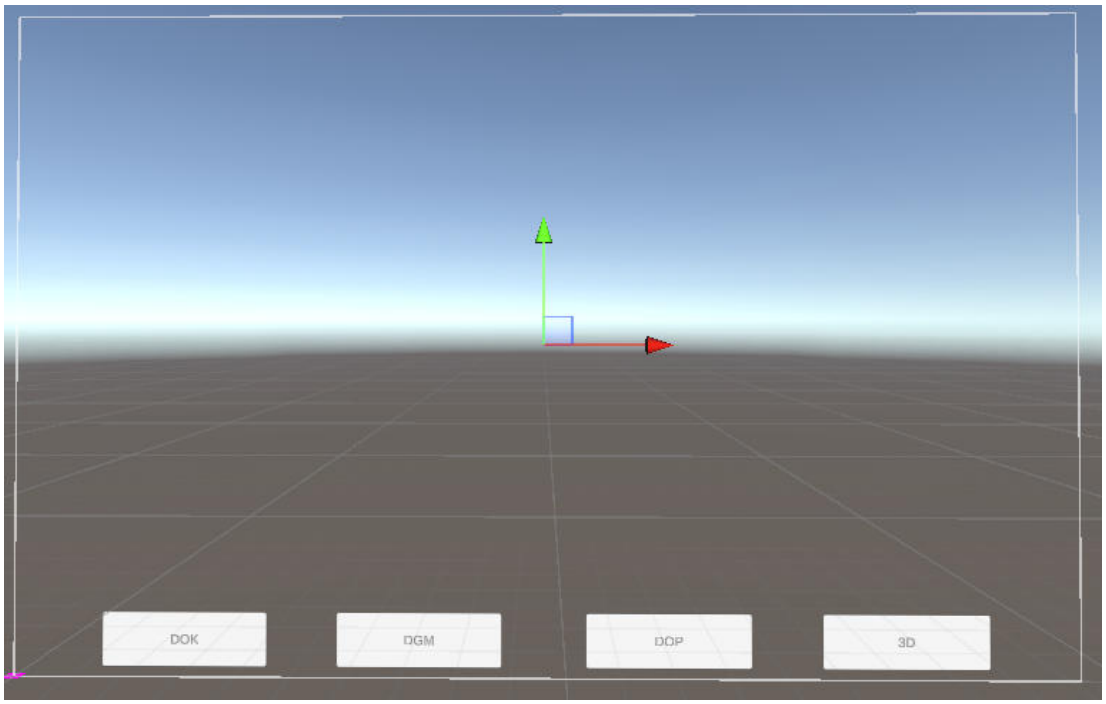

Abbildung 4.15: User Interface mit 3D-Funktionalität

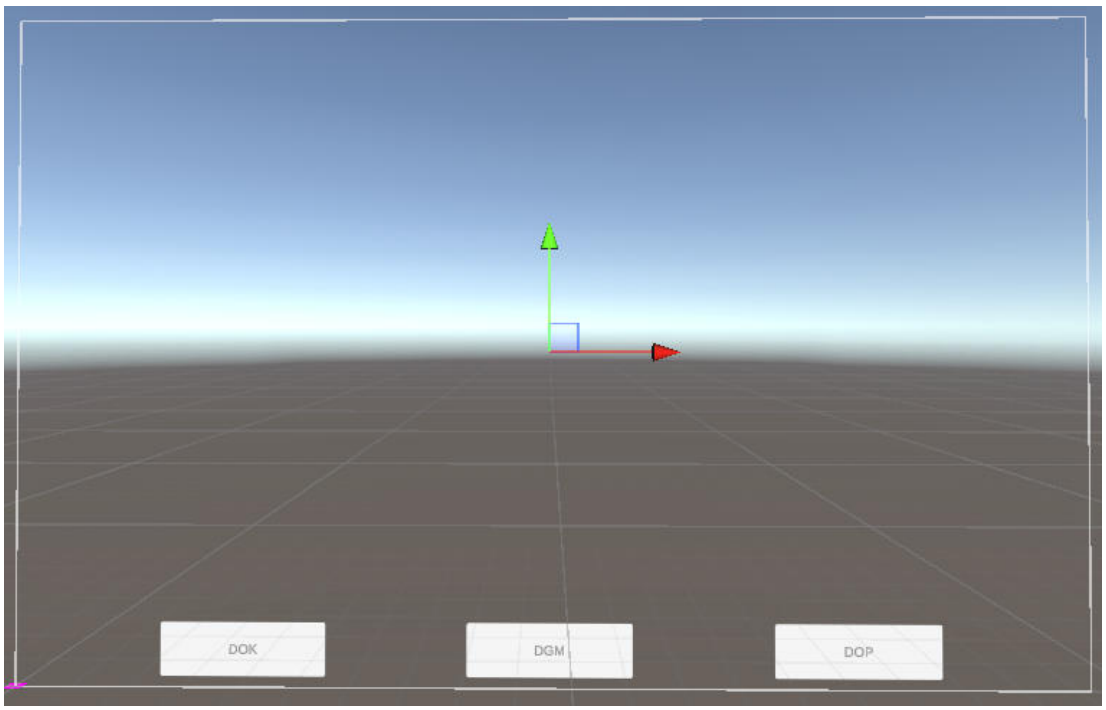

Abbildung 4.16: User Interface ohne 3D-Funktionalität

5 Ergebnisse

5.1 Versuchsfeld

Wie bereits in Kapitel 4.1.1 ausführlich erwähnt, wurde als Interessensgebiet Würzburg mit Umland als Referenzkarte gewählt. Für diesen Bereich liegen dem LDBV mit die aktuellsten Daten aus einer Oblique-Befliegung vor, anhand derer sich eine hervorragende 3D-Darstellung in Form eines 3D-Meshs ableiten lässt. Dies führt dazu, dass sich selbst Details auf den adaptiv getrackten Kartenausschnitten der App eindrucksvoll in 3D darstellen lassen. Das 3D-Mesh wurde vom LDBV zur Verfügung gestellt.

Das Mesh in der App ist nicht für das gesamte Versuchsfeld verfügbar, da die Speicherkapazität des Unity Android Build Systems nicht für diese Größe ausgelegt ist. Dies wird in Kapitel 4.3.2 dargelegt.

5.2 Darstellung Funktionsweise

Exemplarisch werden im Folgenden drei verschiedene physische Kartenausschnitte aufgenommen und getrackt. Dabei handelt es sich im ersten Beispiel um ein Gebiet, an dem das 3D-Mesh aus Speicherkapazitätsgründen nicht verfügbar ist.

Abbildung 5.1: App nach dem Start mit physischer Karte „Flusslauf“

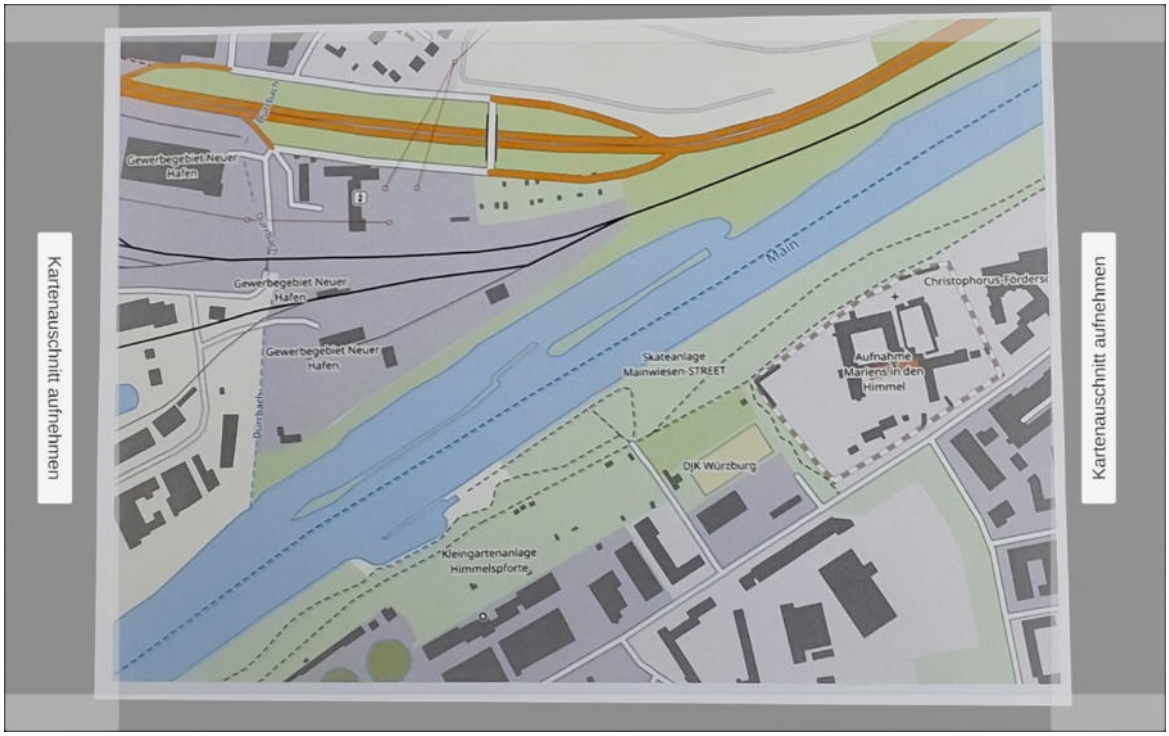

Abbildung 5.2: Aufnahme des Screenshots durch Einpassen der Karte „Flusslauf“ in das User Interface

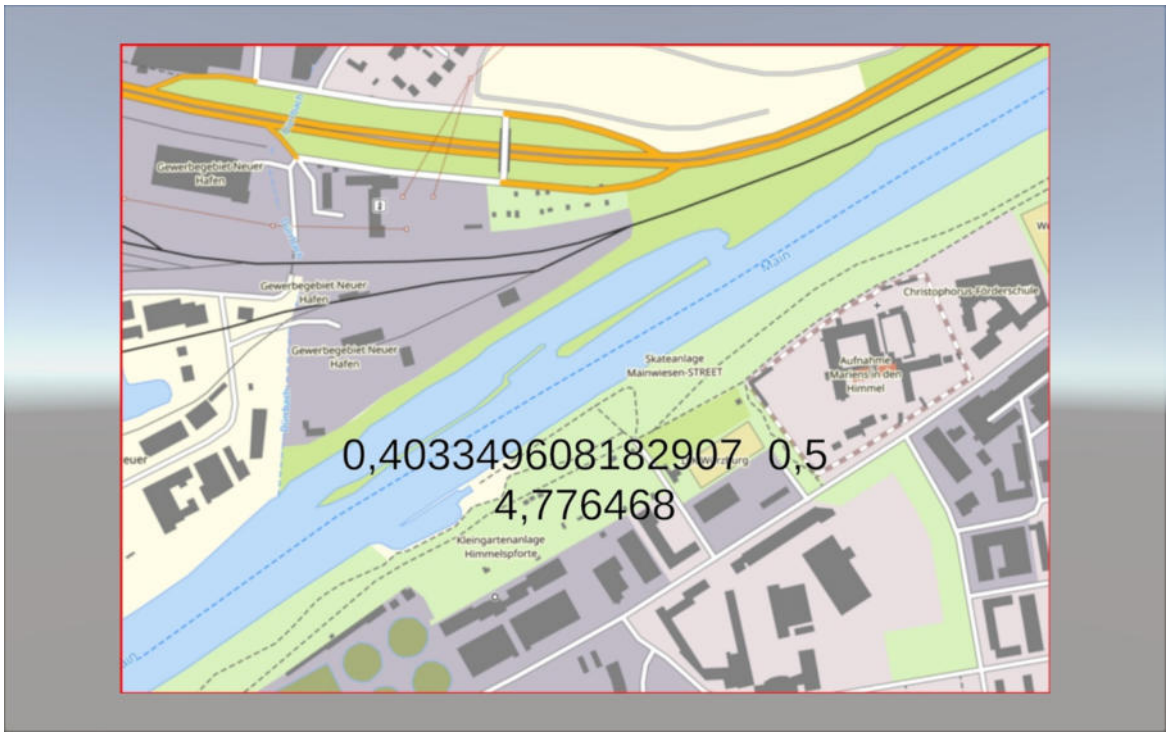

Abbildung 5.3: Ergebnis Template Matching Karte „Flusslauf“ mit Angaben

Abbildung 5.4: Virtuelles digitales Orthophoto Karte „Flusslauf“

Abbildung 5.5: Virtuelles digitales Geländemodell Karte „Flusslauf“

Abbildung 5.6: Virtuelle digitale Orthokarte Karte „Flusslauf“

Abbildung 5.7: Orthophoto Nahaufnahme physischen Karte „Flusslauf“

Die zweite Aufnahme stellt das Zentrum Würzburgs auf der physischen Karte dar, für die 3D-Daten implementiert wurden. Hier kommt das angepasste User Interface zum Einsatz, das vier Buttons aufweist.

Abbildung 5.8: Aufnahme der physischen Karte „Stadtzentrum“

Abbildung 5.9: Aufnahme des Screenshots durch Einpassen der Karte „Stadtzentrum“ in das User Interface

Abbildung 5.10: Ergebnis Template Matching Karte „Stadtzentrum“

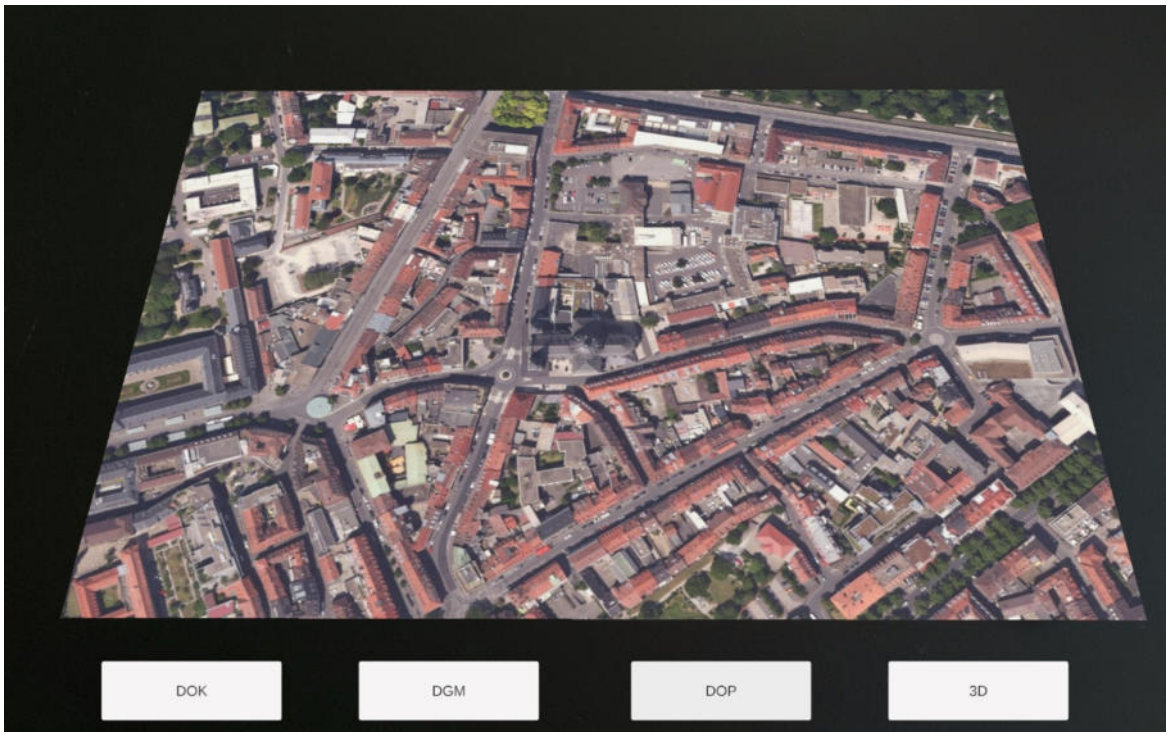

Abbildung 5.11: Orthophoto Karte „Stadtzentrum“

Abbildung 5.12: DGM Karte „Stadtzentrum“

Abbildung 5.13: DOK Karte „Stadtzentrum“

Abbildung 5.14: 3D-Mesh Karte „Stadtzentrum“

Abbildung 5.15: 3D-Mesh Nahaufnahme Karte „Stadtzentrum“

Abbildung 5.16: 3D-Mesh Ansicht von Südwesten Karte „Stadtzentrum“

Die dritte physische Karte zeigt die Würzburg auf der Anhöhe über der Stadt mit 3D-Mesh.

Abbildung 5.17: Aufnahme der physischen Karte „Burg“

Abbildung 5.18: Aufnahme des Screenshots durch Einpassen der Karte „Burg“ in das User Interface

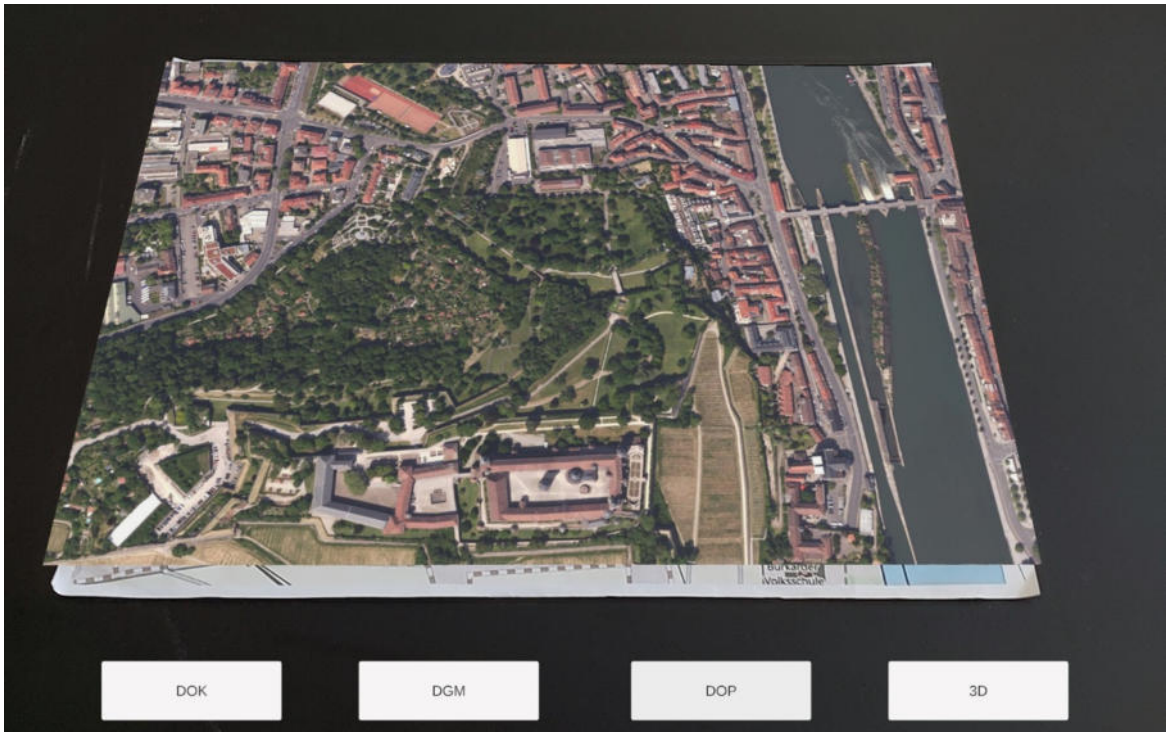

Abbildung 5.19: Orthophoto Karte „Burg“

Abbildung 5.20: DGM Karte „Burg“

Abbildung 5.21: DOK Karte „Burg“

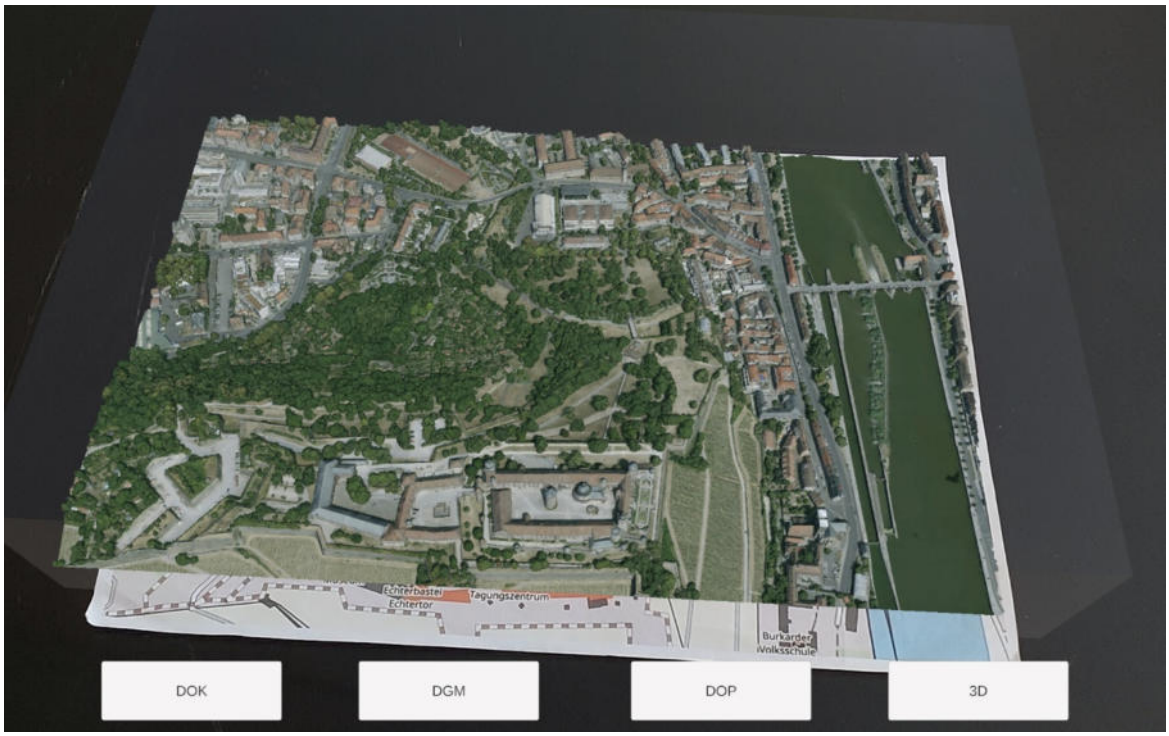

Abbildung 5.22: 3D-Mesh Karte „Burg“

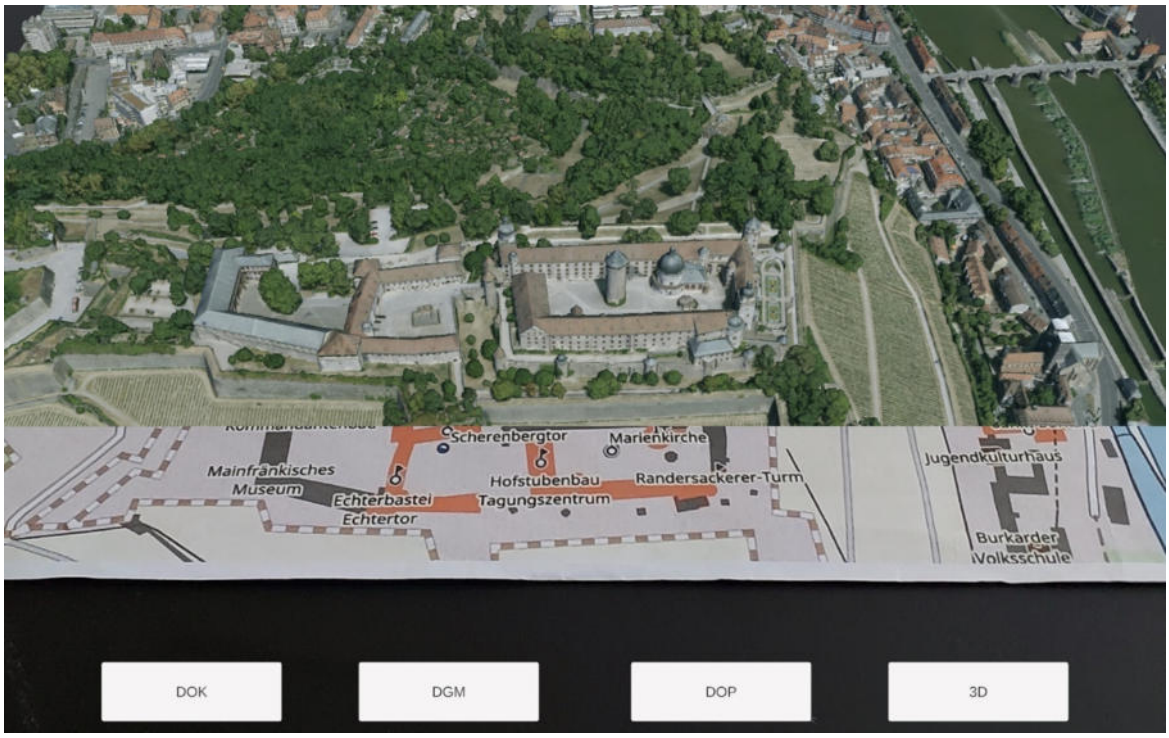

Abbildung 5.23: Nahaufnahme 3D-Mesh Karte „Burg“

Abbildung 5.24: Mainufer unterhalb der Würzburg in 3D Karte „Burg“

Abbildung 5.25: Schrebergärten unterhalb der Würzburg in 3D Karte „Burg“

5.3 Leistungsfähigkeit auf dem Endgerät

Die Leistungsfähigkeit wurde auf zwei verschiedenen Geräten getestet. Auf einem Windows Laptop, mit einem Intel i7-Prozessor der achten Generation mit 16GB Arbeitsspeicher und einer dezidierten Grafikkarte. Das andere Testgerät ist ein Samsung Android Tablet mit einem Qualcomm Snapdragon 778G Prozessor und 6 GB Arbeitsspeicher.

Bei der Leistungsfähigkeit wird zwischen zwei Arten unterschieden. Auf der einen Seite gibt es die Reaktionsgeschwindigkeit der AR-Anwendung während der Darstellung von virtuellen Inhalten. Auf der anderen Seite gibt es die Verarbeitungszeit zum Finden des Kartenausschnittes und Vorbereiten des 3D-Mesh.

5.3.1 Leistung während der AR-Darstellung

Die Darstellung von AR-Inhalten wird auf beiden Testgeräten mit 50 Bildern pro Sekunde wiedergegeben. Es kommt zu keinerlei spürbaren Unterbrechungen oder anderen Aussetzern. Auch bei schnellen Bewegungen ist ein stabiles Tracking gewährleistet.

5.3.2 Laufzeitmessungen

Im Folgenden werden die Ergebnisse der Messungen der Verarbeitungszeit zum Finden des Kartenausschnittes in Tabellenform dargestellt.

Für die Prozessierung des Template Matchings wurden folgende Werte gemessen.

Tabelle 5.1: Zeitmessung in Sekunden

	Android Tablet	Windows Laptop
Karte Flusslauf	4,7765	2,9869
Karte des Stadtzentrum	36,2262	18,3985
Karte der Würzburg	4,8729	3,1665

Die Downloadzeiten für die WMS-Dienste konnten nur auf dem Windows-Laptop gemessen werden.

Tabelle 5.2: Downloadzeit WMS-Dienste in Sekunden

	Windows Laptop
DOP	0.71991
DGM	0.71949
DOK	0.99587

6 Diskussion

In dieser Arbeit wurde eine Augmented Reality-App zum adaptiven Tracken von physischen Kartenausschnitten entwickelt, um Potenziale und Herausforderungen zu analysieren. AR bietet spannende Möglichkeiten, die physische Welt mit digitalen Informationen zu erweitern, während OpenCV als leistungsstarke Bibliothek zur Bildverarbeitung und Computer Vision dient. Die Ergebnisse dieser Arbeit in Form einer prototyphaften Android-Anwendung demonstrieren die vielfältigen Anwendungsmöglichkeiten anhand des Untersuchungsgebietes Großraum Würzburg. Sie können zum Beispiel in den Bereichen Bildung, Tourismus und Stadtplanung zum Einsatz kommen. Gleichzeitig werden verschiedene technische und konzeptionelle Herausforderungen identifiziert, die bei der Implementierung solcher Systeme auftreten können. Im Folgenden werden die wichtigsten Erkenntnisse und ihre Implikationen detailliert diskutiert.

6.1 Problematiken bei der Umsetzung

Die initiale Umsetzung des Projekts erwies sich als herausfordernd. Es war erforderlich, das Plug-in OpenCV+ vor der Integration von Vuforia hinzuzufügen und sämtliche Klassen mit identischen Namen umzubenennen. Kurzzeitig wurde die Möglichkeit in Erwägung gezogen, OpenCV als natives Plug-in in C++ unter Verwendung von Android Studio zu erstellen, um potenzielle Fehler zu vermeiden und die vollständige Funktionalität zu gewährleisten. Dieser Ansatz wurde jedoch verworfen, da neue Fehler auftraten und die Anpassung der OpenCV+ Bibliothek als effizientere Lösung erschien. Diese Schritte legten den Grundstein für die erfolgreiche Entwicklung der App, indem sie eine unempfindliche und flexible Entwicklungsumgebung schufen, die den spezifischen Anforderungen des Projekts gerecht wurden.

Im ursprünglichen Konzept der AR-App war vorgesehen, dass der physische Kartenausschnitt ohne ein Einpassen in das User Interface aufgenommen werden sollte. In dem Bild des Screenshots sollte die Karte mithilfe eines Skriptes

ausgeschnitten werden. Erste Ansätze wurden mit OpenCV umgesetzt, unter anderem kamen Algorithmen wie GrabCut, Watershed oder Canny zum Einsatz. Damit ließen sich auch unter optimalen Bedingungen die gewünschten Ergebnisse erzielen. Das Problem mit diesen Algorithmen bestand darin, dass wechselnde Umgebungsfaktoren keine konsistenten Ergebnisse erlaubten. Inhomogene Hintergründe störten den Findungsprozess erheblich. Selbst bei neutralen Hintergründen kamen bei unterschiedlicher Beleuchtung keine gleichbleibenden Ergebnisse bei den getesteten physischen Karten zustande.

Ein weiterer Ansatz war das Hinzufügen einer Histogrammequalisierung und das Ausschneiden über Farbwerte, jedoch konnten keine Schwellenwerte für alle Umweltbedingungen gefunden werden. Eine Dokumentscanner-Klasse innerhalb des Plug-ins OpenCV+ brachte ebenfalls keine zufriedenstellenden Ergebnisse für die ausgewählten Testdaten. Auch die Implementierung des Android Documentscanners von Google mithilfe eines nativen Plug-ins in Unity schlug fehl. Es war somit im durch die Masterarbeit gegebenen Rahmen zeitlich nicht möglich, ein sinnvolles User Interface mit automatischer Scannerfunktion zu entwickeln.

Eine wichtige Überlegung in Hinblick auf die Computer Vision Prozessierung ist die allgemeine Entscheidung für Template Matching. Dies ist darin begründet, dass sich Template Matching für den Vergleich von Karten sehr gut eignet. Es gibt hier keine Rotationsänderungen, da Karten allgemein nach Norden ausgerichtet sind. Durch das Design des User Interfaces wird die Karte ohne perspektivische Verzerrungen durch schräge Aufnahmewinkel aufgenommen. Die Maßstab-Problematik wurde durch die zweite Iteration gelöst. Eine mögliche Alternative wäre das Feature Matching gewesen, bei dem das Berechnen von markanten Punkten in der Referenzkarte und der Templatekarte zu einer Platzierung dieser führt. Die markanten Punkte der Referenzkarte könnten bereits bei Erstellung der App vorprozessiert werden, damit wären die Berechnungen des Feature Matchings während der Laufzeit schon weitestgehend abgeschlossen. Es wäre innerhalb der App-Ausführung nur nötig die markanten Punkte der Template Karte zu berechnen und die Punkte beider Karten zu vergleichen. Sicherlich wäre es interessant gewesen, durch die Entwicklung eines weiteren Prototyps mit Feature Matching einen umfassenden Vergleich anzustellen. Es hätten sich zusätzliche Optimierungsmöglichkeiten aufzeigen lassen. Leider war dies im zeitlichen Rahmen der Masterarbeit nicht möglich.

Ein weiterer Punkt ist die Berechnungsdauer. Die Skalierbarkeit ist nicht ge-

geben. Grund dafür sind die Iterationen über die Referenzkarte und in dieser Anwendung zusätzlich die Iteration über alle Maßstäbe. Dieser Prozess ist sehr rechenintensiv. Die Grenzen dieses Algorithmus würden bei größeren Karten vermutlich schnell erreicht werden, da die Laufzeit enorm ansteigt. Die Ansätze zur weiteren Optimierung der Berechnungszeit wurden bereits weitgehend ausgeschöpft. Die Berechnung der Ähnlichkeitsmetrik mit den normalisierten Korrelationskoeffizienten erwies sich als effizienteste Berechnungsmethode. Eine weitere Reduktion der Auflösung des Referenzbildes ist nicht möglich, ohne ein starkes Ansteigen von falschen Zuordnungen des Template Bildes zu erhalten.

Ein anderer wesentlicher Aspekt ist die Zuverlässigkeit der Ergebnisse des Template Matchings. Bei guten Lichtverhältnissen werden die getesteten physischen Kartenausschnitte konsequent präzise erkannt. Bei schlechten Lichtverhältnissen, insbesondere in dunkleren Räumen oder mit einer zu dominanten Lichtquelle von einer Seite, werden manche Kartenausschnitte schlechter getrackt. Kartenausschnitte mit einem hohen Kontrast der beispielsweise durch graue Wohngebiete, grünem Wald und blauen Flüssen entsteht, werden auch bei schlechteren Lichtverhältnissen zuverlässig erkannt. Ein interessantes Verhalten zeigte sich beim Aufnehmen von zufälligen Bildern. Der Algorithmus ordnet jedem aufgenommenen Bild einen Bereich der Referenzkarte zu, wobei die Ergebnisse stark den fehlerhaften Ergebnissen bei dunkleren Lichtverhältnissen ähneln. Die Problematik scheint hier im Allgemeinen im optischen Erkennen der Karte zu liegen und nicht in der richtigen Zuordnung innerhalb der Referenzkarte. All diese Faktoren sprechen dafür dass, sobald eine Karte in hinreichender Qualität aufgenommen wurde, der Algorithmus dauerhaft korrekte Ergebnisse liefert.

Neben dem bereits erwähnten Computer Vision Teil spielt auch die 3D-Mesh Verarbeitung eine bedeutende Rolle. Die Vorprozessierung erwies sich als langwierig. Auf den zur Verfügung stehenden Arbeitsrechnern reichte der Arbeitsspeicher selbst für ein Viertel der Daten des Interessensgebiet kaum aus. Nachdem alle diese Arbeitsschritte abgeschlossen waren, erfolgte der Import in Unity. Das Einladen und Verarbeiten des gesamten Interessengebietes dauerte ca. 60 bis 70 Minuten. Erst nachdem das Mesh in Unity komprimiert und die Texturen verkleinert wurden, war es möglich das Gesamtgebiet im Editor zu betrachten. Nachdem ein Android Build mit dem ganzen Mesh erstellt wurde, kam es zum Absturz der App beim Laden der Szene mit dem 3D-Mesh. Erst das Verkleinern des Mesh-Ausschnitts durch Weglassen der Randkacheln erlaubte einen dauerhaft stabilen Betrieb.

Auch das Zuschneiden des Meshs lief nicht wie ursprünglich geplant. Anfangs war angedacht, die betroffenen Meshkacheln über die Collider-Komponente zu bestimmen. Diese Funktion ist Teil der Unity Physikengine. Dafür wurde für jede Kachel eine rechteckige Collider-Bounding Box erstellt. Dadurch war es möglich, alle Kacheln zu identifizieren, die die platzierte Box der Referenzkarte berühren. In den Testläufen zeigte sich jedoch, dass dieses Verfahren bereits auf dem Computer sehr lange dauerte. Als effizientere Lösung erwies sich die manuelle Berechnung der Ausmaße des 3D-Mesh über den Mittelpunkt jeder Kachel. Die Prozessierung dauert nur wenige Sekunden, sowohl auf dem Computer als auch auf Android-Geräten.

6.2 Ergebnisse

In der entwickelten AR-App konnten erfolgreich alle vorgesehenen Hauptfunktionen, darunter das Tracken einer physischen Karte, das Darstellen von WMS-Diensten und die Verarbeitung eines 3D-Mesh implementiert werden. Diese beinhalten die präzise Erkennung und Verfolgung von Markern in Echtzeit. Die App zeigt eine gute Zuverlässigkeit bei der Verfolgung von Markern unter günstigen Lichtverhältnissen und Winkeln, selbst bei ungünstigen Verhältnissen werden viele Marker noch erkannt. Dies bestätigt die Unempfindlichkeit des implementierten adaptiven Tracking-Algorithmus. Die Genauigkeit der Markererkennung und -verfolgung ist ein zentraler Aspekt der App. Nutzer können Marker problemlos platzieren und deren korrekte Verfolgung beobachten. Die Dauer des Template Matchings liegt bei den meisten Kartenausschnitten bei unter 5 Sekunden. Bei ungünstigen Maßstäben oder Bildern mit schlechten Screenshotaufnahmen kann die Dauer ca. 30 bis 40 Sekunden betragen, in seltenen Einzelfällen sogar 180 Sekunden. Die Zuordnung ist aber bei langen Berechnungszeiten dennoch effektiv. Die Reaktionsgeschwindigkeit der App bei der Darstellung von 3D-Daten ist sehr gut. Es kommt zu keinerlei Stocken und alle Handybewegungen werden auf dem Endgerät sehr fließend wiedergegeben. Auch das Heranzoomen in 2D- und 3D-Daten funktioniert ohne Probleme. Im 3D-Mesh lassen sich so sehr viele Details erkennen. Testnutzer fanden die App intuitiv und berichteten von einer einwandfreien Augmented Reality-Erfahrung. Es waren keine langen Einführungen zur Bedienung nötig. Die Performance der App war insgesamt sehr gut. Während der Nutzung mit dem aktuellen Prototyp gab es keine signifikanten Abstürze oder Fehlfunktionen. Die Marker wurden

in Echtzeit verfolgt. In einigen Szenarien, insbesondere bei schwacher Beleuchtung, kam es, wie bereits erwähnt, zu Fehlzuordnungen. Diese Beobachtungen unterstreichen die Notwendigkeit einer weiteren Optimierung des Aufnahmealgorithmus für alle Bedingungen.

6.3 Fazit

Diese Masterarbeit demonstriert, wie anpassungsfähig die Geovisualisierung mit Augmented Reality sein kann. Vorangegangene Untersuchungen, unter anderem im Abschnitt Stand der Forschung 1.3 dargestellt, zeichnen sich durch ihre starre, nicht adaptive Ausrichtung hinsichtlich des Untersuchungsgebietes aus. Dadurch sind sie gegenüber Änderungen des Interessengebietes unflexibel. Mit dieser App wurde ein Ansatz entwickelt, der AR in der Geovisualisierung dynamischer gestaltet. Dies kann als Basis dienen, um zukünftige AR-Projekte effektiv zu konzipieren.

Es konnte gezeigt werden, dass es erreichbar ist, einen lauffähigen Prototyp einer Android-Applikation in einer Augmented Reality-Umgebung zu entwickeln, der eine innovative Tracking-Methode beinhaltet, um physische Kartenausschnitte digital zu erkennen und mit virtuellen Geobasisdaten verknüpft interaktiv zu veranschaulichen. Die Implementierung von WMS-Diensten bildet dafür eine wichtige Grundlage. Es wurde verdeutlicht, dass diese bei der Entwicklung einfach zu integrieren sind und auch für spätere Weiterentwicklungen unbegrenzt skaliert werden können. Das Interessensgebiet in dieser App wurde im Vergleich zu herkömmlichen Anwendungen bereits erheblich erweitert und kann Gebiete von der Größe ganzer Städte wie hier Würzburg und Umland bis ins Detail darstellen.

Obwohl die Ergebnisse insgesamt positiv sind, wurden einige Bereiche identifiziert, die verbessert werden könnten. Die Einführung zusätzlicher Funktionen, wie z. B. die Unterstützung benutzerdefinierter Marker oder erweiterte Analysemöglichkeiten, könnten die Attraktivität und Funktionalität der App weiter steigern. Es bestünde die Möglichkeit eine Liste mit WMS-Diensten in die App hinzuzufügen und so die App noch anwenderfreundlicher und individueller zu gestalten. Diese Aspekte und das Testen von Feature Matching als Bildzuordnung bieten interessante Ansätze für zukünftige Forschungsarbeiten.

Danksagung

An dieser Stelle möchte ich mich bei allen beteiligten Personen bedanken, die diese Masterarbeit ermöglicht haben.

Mein erster Dank gilt dabei meinen Betreuern der Hochschule München Herrn Prof. Dr. Thomas Abmayr und Herrn Prof. Dr. Markus Oster für die hilfreichen Anregungen während des gesamten Betreuungszeitraums. Die regelmäßigen Meetings halfen sehr bei dem Entwicklungsprozess dieser Arbeit.

Ich bedanke mich außerdem ausdrücklich bei Herrn Thomas Meier Mitarbeiter des LDBV für die Unterstützung. Die Fachexpertise war stets sehr hilfreich und das zur Verfügung gestellte 3D-Mesh stellt einen bedeutenden Teil dieses Projekts dar.

Anhang

Der Anhang befindet sich in einem Verzeichnis auf der LRZ Sync + Share Plattform.

Dieses Verzeichnis enthält folgende Inhalte:

- LaTEX Projekt der Masterarbeit
- Vollständiges Unity-Projekt
- Einzelskripte aus dem Unity-Projekt
- Verwendete Geodaten (3D-Mesh und Referenzkarte des LDBV)
- Trailer der fertigen Android-App

Literaturverzeichnis

Breitband und Vermessung (LDBV) Bayerisches Landesamt für Digitalisierung. Produktenübersicht: Open data des ldbv, May 2019. URL https://www.ldbv.bayern.de/file/pdf/4628/Produkt%C3%BCbersicht_DIN%20A%204.pdf.

Wilhelm Burger. *Digitale Bildverarbeitung Eine algorithmische Einführung mit Java*. Springer Berlin Heidelberg, 2006. ISBN 9783540309406. doi: 10.1007/3-540-30941-1.

Ralf Dörner, Wolfgang Broll, Paul Grimm, and Bernhard Jung. *Virtual und Augmented Reality (VR / AR)*. Springer Vieweg, Berlin, Heidelberg, 2013. ISBN 978-3-642-28903-3. doi: 10.1007/978-3-642-28903-3.

Amr Fenais, Samuel T. Ariaratnam, Steven K. Ayer, and Nikolas Smilovsky. Integrating geographic information systems and augmented reality for mapping underground utilities. *Infrastructures*, 4(4):60, 2019. doi: 10.3390/infrastructures4040060. URL https://www.mdpi.com/2412-3811/4/4/60/pdf.

Koordinierungsstelle GDI-DE. Architektur der geodateninfrastruktur deutschland (gdi-de), 2021. URL https://www.gdi-de.org/download/Flyer_Architektur_GDI-DE_de.pdf. Flyer.

Bernd Girod. Digital image processing. Online, 2013. URL https://web.stanford.edu/class/ee368/Handouts/Lectures/2019_Winter/9-TemplateMatching.pdf. Accessed: 2024-06-18.

Andreas Illert. Infrastructure for spatial information in europe (inspire) status report on the development of implementing rules for geographical names data, 2009. URL https://unstats.un.org/unsd/geoinfo/ungegn/docs/25th-gegn-docs/wp%20papers/wp34-inspire-germany.pdf. Accessed: 2024-06-26.

B. Loesch, S. Nebiker, M. Christen, and R. Wüst. Geospatial augmented reality lösungsansaetze mit natürlichen markern für die kartographie und die geoinformationsvisualisierung im auÃŸenraum. In *DGPF Tagungsband 24 / 2015*, pages 89–97. Deutsche Gesellschaft fÃ¼r Photogrammetrie, Fernerkundung und Geoinformation, 2015. URL https://www.dgpf.de/src/tagung/jt2015/proceedings/papers/11_DGPF2015_Loesch_et_al.pdf.

P. Milgram and F. Kishino. A taxonomy of mixed reality visual displays. *IEICE Trans. Information Systems*, E77-D(12):1321–1329, Dec 1994.

Team OpenCV. About opencv. Online, 2024. URL https://opencv.org/about/. Accessed: 2024-06-18.

Inc. PTC. Vuforia engine. https://www.ptc.com/de/products/vuforia, 2024. Accessed: 2024-06-18.

Bayerische Staatsregierung. Geodateninfrastruktur bayern (gdi bayern), 2024. URL https://geoportal.bayern.de/geoportalbayern/seiten/info. Accessed: 2024-07-07.

TA Syed, MS Siddiqui, HB Abdullah, S Jan, and A Namoun. In-depth review of augmented reality: Tracking technologies, development tools, ar displays, collaborative ar, and security concerns. *Sensors*, 23(1):146, 2022. URL https://www.mdpi.com/1424-8220/23/1/146.

Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer International Publishing, 2022. ISBN 9783030343729. doi: 10.1007/978-3-030-34372-9.

Unity Technologies. Our company. https://unity.com/de/our-company, 2024. Accessed: 2024-06-18.

Konstantine Tsotsos and Luca Ballan. Ar foundation with arkit and arcore - unite now. Online, 2020. URL https://www.youtube.com/watch?v=MM786WD-Gv8&t=362s. Accessed: 2024-06-24.

European Union. Directive 2007/2/ec of the european parliament and of the council of 14 march 2007 establishing an infrastructure for spatial information in the european community (inspire), 2007. URL https://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:52022SC0196. Accessed: 2024-06-26.

Bayerische Vermessungsverwaltung. Open data - webkarte, May 2024a. URL https://geodaten.bayern.de/opengeodata/OpenDataDetail.html?pn=webkarte. Accessed: 2024-07-07.

Bayerische Vermessungsverwaltung. Bayern opengeodata, 2024b. URL https://geodaten.bayern.de/opengeodata/. Accessed: 2024-07-07.

Jürgen Vogel. Adv-positionspapier zur geodateninfrastruktur. *zfv - Zeitschrift fÃ¼r Geodäsie, Geoinformation und Landmanagement*, 127(2):91–97, 2002. URL https://geodaesie.info/images/zfv/127-jahrgang-2002/downloads/zfv_2002_2_Vogel_AdV.pdf.

N. Yonov and D. Petkov. Augmented reality navigation map of mountain area. In *Proceedings of the International Conference on Cartography and GIS*, pages 89–97. ResearchGate, 2020. URL https://www.researchgate.net/profile/Dobrin-Petkov/publication/344337226_AUGMENTED_REALITY_NAVIGATION_MAP_OF_MOUNTAIN_AREA/links/5f69b245a6fdcc008634379f/AUGMENTED-REALITY-NAVIGATION-MAP-OF-MOUNTAIN-AREA.pdf.

Hochschule München University of Applied Sciences

Erklärung gemäß § 26 Abs. 7 ASPO

Name	Kinzel
Vorname	Maximilian
Studiengang	Master Geomatik
Matrikel-Nr.	06736418

Betreuer:in Prof. Dr. Thomas Abmayr, Prof. Dr. Markus Oster

Betreuer:in Extern: Dipl. Ing. (FH) Thomas Meier (LDBV)

Hiermit erkläre ich, dass ich die Abschlussarbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

München, 18.08.2024

Ort, Datum

Unterschrift