Hochschule München, Fakultät für Geoinformation

Bachelorarbeit

Verwendungsmöglichkeit amtlicher Geobasisdaten in einer Augmented Reality Umgebung

Studiengang Angewandte Geodäsie und Geoinformatik
Sommersemester 2022

vorgelegt von
Maximilian Kinzel

Betreuer: Prof. Dr. Thomas Abmayr

Kooperation: Landesamt für Digitalisierung, Breitband und Vermessung

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

München, 14. September 2022

Zusammenfassung

Die gegenständige Arbeit befasst sich mit den Verwendungsmöglichkeiten amtlicher Geobasisdaten in einer Augmented Reality (AR) Umgebung. Dabei soll ein neuartiges Konzept zur interaktiven Präsentation von Geodaten entstehen. Dieses umfasst das Projizieren von digitalen 3D-Geodaten auf eine physische Karte. Die amtlichen Daten werden so aufbereitet, dass sie zusätzlich in einer Routingfunktion Anwendung finden. Da heutzutage bereits die neueren Smartphones und Tablets mit der benötigten Rechenleistung ausgestattet sind, fiel die Entscheidung auf Augmented Reality. Somit ist eine große Basis zur Nutzung von AR-Projekten vorhanden.

Nun soll ein Versuchsträger mit amtlichen Geodaten entstehen. Hierzu wurde eine Vorgehensweise in der Game-Engine Unity entwickelt, die die Kombination und Integration verschiedenster Datentypen in einer Anwendung beinhaltet und die auf modernen Zielgeräten lauffähig ist. Dabei werden die AR-Funktionalitäten unter Zuhilfenahme des Plug-ins AR Foundation erstellt. Die Geodaten werden mit dem Standort des Benutzers und mit Informationen zu Sehenswürdigkeiten in Form von Infotafeln innerhalb des Interessensgebietes verknüpft. Als Testgebiet für den Augmented Reality Prototyp wird die Münchner Innenstadt verwendet. Für die Wegfindung wurde das neuartige Unity Data-Oriented Technology Stack als Grundgerüst genutzt. Dies erlaubt das schnellere Verarbeiten größerer Datenmengen und ist darauf ausgelegt, eine große Anzahl an mathematischen Operationen effizient durchzuführen.

Die Arbeit zeigt, dass es möglich ist amtliche Geobasisdaten in Form einer Augmented Reality Anwendung darzustellen und funktional in einen Wegfindungsprozess aufzunehmen. Es geht nicht nur um die reine Darstellung, sondern auch um eine Interaktion mit den Daten. Zusätzlich wird gezeigt, welches Potenzial sich entwickeln kann durch die Darstellung der Geodaten in Verbindung mit weiteren Sachinformationen.

Abstract

This bachelor thesis deals with the possible uses of official geodata in an augmented reality (AR) environment. Thereby, a new concept for the interactive presentation of geodata shall be developed. This involves the projection of 3D digital geospatial data onto a physical map. In addition, the official data will be prepared in such a way that they can be used in a routing function. AR was chosen because nowadays all newer smartphones and tablets are already equipped with the necessary computing power. Thus, a large basis for the use of AR projects is available.

An experimental App with official geodata is to be created. For this purpose, an approach was developed in the game engine Unity, which involves the combination and integration of a wide variety of data types into one application which is executable on all modern smartphones and tablets. The AR functionalities are created with the help of the AR Foundation plug-in. The geodata will be linked with the user's location and with further information of buildings worth seeing within the area of interest. Munich's city center is used as a test area for the AR prototype. For wayfinding, the new Unity Data-Oriented Technology Stack was used as the basic framework. This allows for faster processing of larger amounts of data and is designed to efficiently perform many mathematical operations.

The work shows that it is possible to display official geospatial data in an augmented reality application and functionally incorporate it into a routing process. It wasn't just about the pure presentation, but also about an interaction with the data. In addition, it is shown what potential the representation of the geodata can develop in combination with further detail information.

Inhaltsverzeichnis

1 Einleitung **8**

2 Stand der Forschung **11**
2.1 Augmented Reality (AR) 11
2.2 Geovisualisierung 11
2.3 Augmented Reality Geovisualisierung 12

3 Grundlagen **14**
3.1 Grundlegende Begriffe 14
3.2 Funktionsweise AR 16
3.3 Geobasisdaten 19
3.4 Routing 21
3.5 Lokalisierung 23

4 Werkzeuge **27**
4.1 Erzeugung 27
4.2 Zusammenführung 28

5 Vorgehensweise **29**
5.1 AR-Workflow 29
5.2 Vektordatenprozessierung 31
5.3 Routing 32
5.4 User-Interface 36

6 Ergebnisse **39**
6.1 Versuchsfeld 39
6.2 Darstellung Funktionsweise 40
6.3 Performance 44

7 Diskussion **45**

8 Fazit **48**

Danksagung **49**

Literaturverzeichnis **50**

Abbildungsverzeichnis

3.1 Virtualitäts-Kontinuum nach [10] 14
3.2 Feature Tracking nach [8] 17
3.3 Visual -Inertial Sensor Fusion nach [8] 17
3.4 Maschine Learning based IMU Tracking nach [8] 18
3.5 A*-Algorithmus 22
3.6 GNSS Schaubild nach [4] 25

4.1 Workflow in Infraworks 28

5.1 AR Session Grundlagen 29
5.2 Reference Image Library 30

6.1 Versuchsfeld 39
6.2 Start der Anwendung 40
6.3 Bewegung der Kamera 40
6.4 Vergrößern des 3D-Modells 40
6.5 Verkleinern des 3D-Modells 40
6.6 Benutzer-Menü 41
6.7 Orthophoto 41
6.8 Gebäudemodell mit Webkarte 41
6.9 Digitales Geländemodell 41
6.10 Webkarte 42
6.11 Uraufnahme 42
6.12 3D-Modell mit Infotext 42
6.13 Gebäudemodell mit Infotext 42
6.14 Starten der Routingfunktion 43
6.15 Eingabe des Endpunktes 43
6.16 Berechnete Route im 3D-Modell 43
6.17 Route mit einer Sehenswürdigkeit 43

Formelverzeichnis

3.1 Kostenfunktion A*-Algorithmus 3.1 21
3.2 Räumlicher Pythagoras 3.2 24
3.3 Pseudodistanzen 3.3 24

1 Einleitung

In den vergangenen Jahrzehnten hat die fortschreitende Technologie den Alltag der Menschen stark verändert. Die Digitalisierung hält sowohl in der Arbeitswelt als auch im privaten Umfeld Einzug. Jeden Tag werden unglaubliche Datenmengen verschickt, verarbeitet und abgespeichert. Viele dieser Daten sind abstrakt und bleiben im Verborgenen, andere wie Fotos oder Videos werden millionenfach geteilt und auf sozialen Medien dargestellt. Die immer leistungsfähigeren Computer-Chips erlauben das Entwickeln von immer komplexeren Anwendungen. Es werden fortwährend neue Möglichkeiten geschaffen, Daten zu verarbeiten und abzubilden.

Dies hielt auch schon früh Einzug in das Vermessungswesen und hält bis heute an. Das manuelle Vermessen mit Maßband und Winkelmesser wurde früh durch elektronische Messgeräte ersetzt. Diese sind effizienter, genauer und arbeiten digital. Der letzte Punkt bietet einen entscheidenden Vorteil. Die frühere Arbeitsweise, Schreiben und Zeichnen auf Papier nahm viel Zeit in Anspruch und war nicht ressourcenschonend. Die Datenhaltung auf digitalen Speichermedien erweist sich als einfacher und nachhaltiger. Die fortschreitende Technik ermöglicht, mehr und genauere Daten aufzunehmen. Heutzutage gibt es eine große Fülle an Geodaten, die mit den unterschiedlichsten Verfahren aufgenommen werden. In Deutschland sind die wesentlichsten Datenbestände bei den Vermessungsverwaltungen der Länder beheimatet. Das Landesamt für Digitalisierung, Breitband und Vermessung (LDBV) ist in Bayern für die sogenannten Geobasisdaten zuständig.

Diese breit gefächerten Geodaten lassen sich heute mit der Augmented Reality (AR) darstellen und so neue Möglichkeiten der Präsentation entstehen. Bei der AR-Anwendung werden Bilder der Realität aufgenommen und auf dem Ausgabemedium wie z. B. Handy oder Tablet durch virtuelle Inhalte ergänzt. Die virtuellen Objekte werden auf dem Display des Endgerätes so dargestellt, als würden sie sich in der realen Welt befinden. Für den Anwender bedeutet dies

eine innovative Aufbereitung von Inhalten. Heutzutage sind bereits die neueren Smartphones und Tablets mit der benötigten Rechenleistung ausgestattet, die eine AR-Anwendung benötigt. Somit ist eine große Basis zur Nutzung von AR-Projekten vorhanden. Die sehr beliebte App Pokémon GO hat bereits gezeigt, dass raumbezogene Daten und Augmented Reality weitreichendes Potenzial besitzen.

Thema dieser Bachelorarbeit ist die Entwicklung eines Prototyps der Geodaten verschiedenster LDBV-Produkte in eine mobile Augmented Reality Anwendung integriert. Diese Arbeit konzentriert sich darauf, die Fülle amtlicher Vermessungsdaten für den Endnutzer auf neue Art aufzubereiten und vorzustellen. Hierbei steht die Entwicklung im Fokus einer performanten Augmented Reality Applikation, die die Beziehung des Nutzers zu den Geodaten zugänglicher macht und die Variabilität dieser aufzeigt. Dabei muss die Anwendung mit dem heutigen Stand der Technik sowohl in den Bereichen Hardware mobiler Endgeräte als auch mit der aktuellen Software lauffähig sein. Die Frage, welche Entwicklungsumgebung und welches AR-Framework geeignet sind, stellte sich. Die Entscheidung fiel auf Unity und AR Foundation, weil diese sehr große Entwicklungsfreiräume bieten und gut erweiterbar sind.

In der zu entwickelnden App soll eine Karte der Münchner Innenstadt entstehen, die durch Augmented Reality Inhalte erweitert wird. So werden Informationen zu ausgewählten Sehenswürdigkeiten durch Antippen der Benutzeroberfläche angezeigt, Routenplanung innerhalb des Interessengebietes zur Verfügung gestellt und die Art der Darstellung der Karte (3D-Mesh, Orthophoto, Gebäudemodell, Geländemodell, Webkarte und Uraufnahme) auf dem Display ausgewählt. Die Schwierigkeit der Verwirklichung einer solchen App besteht darin, durch Programmierung die verschiedenen Dateiformate in den Unity-Editor zu importieren und diese AR-kompatibel zu gestalten. Ebenso müssen die Touch-Funktionen auf der Benutzeroberfläche des Displays für Augmented Reality angepasst werden.

Im Folgenden wird in Kapitel 2 ein Abriss über den Stand der Forschung in den Bereichen Augmented Reality und Geovisualisierung aufgezeigt. Daran schließt sich Kapitel 3 mit der Darlegung von Grundbegriffen und Grundlagen aus den

Gebieten Technik, Software und Algorithmus an. Ebenso werden wichtig Geobasisdaten vorgestellt. Kapitel 4 Werkzeuge behandelt die Vorbereitung der Geodaten für den weiteren Import in die Entwicklungsumgebung. Die Vorgehensweise der Programmierung wird in Kapitel 5 ausführlich erläutert und anschließend die Ergebnisse in Kapitel 6 dargestellt. Hier wird auch auf die Leistungsfähigkeit der fertigen Anwendung eingegangen. Abschließend werden die Resultate mit Anregungen und Ausblicken in Kapitel 7 diskutiert. Ein Fazit schließt mit möglichen Anpassungen und Erweiterungen unter Kapitel 8 die Arbeit ab.

2 Stand der Forschung

2.1 Augmented Reality (AR)

Es gibt einige Felder, in denen Augmented Reality bereits eingesetzt werden konnte. Darunter zum Beispiel das Science Center to go. Ziel dieses Science Center to go ist, eine naturwissenschaftliche Ausstellung interaktiv und transportabel zu machen. Dazu wurde ein Ausstellungskonzept entwickelt, welches in einen Koffer passt. Es enthält die notwendige Hardware, um die späteren Augmented Reality Programme zu betreiben, aber auch naturwissenschaftliche Modelle, die um eine AR-Komponente erweitert werden können. Hauptziel des Science Center to go ist es, Wissen durch selbstständiges Ausprobieren und visuelles Auffassen zu erlernen. Dabei wird darauf geachtet, das System transportabel zu gestalten, sodass es an vielen Orten gezeigt werden kann. Als Ziel für die Zukunft wird hier angegeben, die AR-Anwendungen auch für handelsübliche Tablets zu konzipieren, um so Kosten zu sparen und die Zugänglichkeit noch weiter zu erhöhen. [6]

In einem weiteren Projekt wird Augmented Reality für Soll/Ist Vergleiche in der Fertigungsplanung genutzt. Diese Anwendung wurde in der Automobilindustrie entwickelt und soll dazu dienen, die steigende Komplexität und Geschwindigkeit der Produktion kontrollierbar zu machen. AR wird in diesem Fall eingesetzt, um digitale Modelle mit der realen Fertigung zu vergleichen. Der Einsatz dieser Technologie belegt ein merkliches Optimierungspotenzial in Prozess und Anwendung. Räumliche Abweichungen zwischen dem digitalen Modell und realem Objekt wurden besonders gut erkannt. Die Bedienung von AR-Anwendungen erwies sich außerdem als sehr intuitiv. [2]

2.2 Geovisualisierung

Geovisualisierung dient dazu, Informationen mit ihrem Raumbezug darzulegen, dazu gehört auch komplexe Daten möglichst einfach abzubilden. Ein Beispiel

hierfür ist ein Projekt aus Italien, welches den Energieverbrauch einer Region in einer dreidimensionalen Karte präsentiert. Hier werden unübersichtliche Datenmengen von Zahlenwerten zu einer Karte verarbeitet, in der die Höhe der Häuser dem Energieverbrauch entsprechen. Die unterschiedliche Färbung der Gebäude nach Industrie-, Privat– und staatlichen Gebäuden macht es möglich, schnell Aussagen aus den Daten zu treffen. Dieser Workflow könnte die Prozessplanung in Energiefragen verbessern und außerdem Energieeinsparpotenziale verständlich für die Öffentlichkeit darstellen. [12]

Ein anderes Projekt zeigt das Potenzial von interaktiver Geovisualisierung. Die Absicht ist auch hier, komplexe Daten möglichst verständlich der Öffentlichkeit zu zeigen. Dabei wurde eine Webanwendung entwickelt, die den Verschmutzungsgrad einer Gemeinde mit gesundheitsschädlichen Stoffen zeigt. Plan war es, die gemessenen Daten der Allgemeinheit kostengünstig zu präsentieren. Hierzu wurden Web Gis-Anwendungen eingesetzt, die es ermöglichen, Informationen über verschiedene Zeiträume zu visualisieren und abzufragen. [3]

2.3 Augmented Reality Geovisualisierung

Es gab bereits frühe erste Ansätze, Augmented Reality zur Navigation bzw. zur Erweiterung von Karten zu nutzen. In einer Feldstudie wurde der Einsatz einer frühen mobilen Version von Google Maps mit einer markerbasierten Routing AR-Anwendung verglichen. Es sollte herausgefunden werden, welche der beiden Anwendungen das Navigieren, allein und in der Gruppe, einfacher machen. Die damalige Technik erlaubte nur sehr begrenzte Applikationen, die das genaue Halten über einem Marker nötig machten. Unter diesen Voraussetzungen erwiesen sich beide Systeme als effektiv. Die AR-Anwendung zeigte sich jedoch als besser geeignet beim gemeinsamen Navigieren und Erkunden der Umgebung. [11]

Für die Stadt Basel wurde bereits eine Augmented Reality App entwickelt, die den Standplan virtuell erweitert. Unter Verwendung der Unity Entwicklungsumgebung mit der AR SDK Vuforia wurde eine App realisiert, die der Bevölkerung von Basel neue Einblicke geben soll. Sämtliche verfügbaren Geodaten des Geoportals des Kantons Basel Stadt fanden hier Verwendung. Dies erlaubt die Überlagerung der realen 2D-Karte mit 3D-Stadtmodellen. Außerdem

kamen Features wie Fahrradrouten, Müllabfuhrzonen, aber auch die Simulationen von Straßenbahnen und Schiffen zum Tragen. In einem weiteren Schritt realisierten die Entwickler auch Live-Wetterdaten über den digitalen Modellen. Zuletzt wurde sogar die Interaktion mit einem realen 3D-Stadtmodell im Maßstabe 1:1000 hinzugefügt. Das Projekt zeigt sehr anschaulich, welche Wirkung vorhandene Geodaten in Verbindung mit Augmented Reality haben können. [7]

3 Grundlagen

3.1 Grundlegende Begriffe

Mixed Reality

Dieser Begriff wurde erstmalig 1994 von Paul Milgram und Fumio Kishino definiert [10]. Die fortgeschrittene Technik erlaubte bereits damals die Präsentation immer komplexerer digitaler Inhalte. In diesem Zuge versuchte Milgram eine Definition und Einordnung des sogenannten „Virtualitäts-Kontinuum“. Er unterschied auf der einen Seite zwischen der Realität als ein Ende des Kontinuums und einer komplett virtuellen Umgebung auf der anderen Seite. Den Bereich dazwischen definierte er als Mixed Reality siehe Abbildung (3.1). Die Grenzen in diesem Bereich sind sehr fließend und es lässt sich allgemein zwischen der Augmented Reality (erweiterten Realität) und Augmented Virtuality (erweiterte Virtualität) unterscheiden. [10]

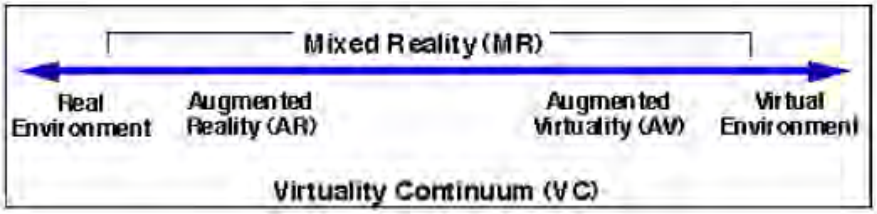

Abbildung 3.1: Virtualitäts-Kontinuum nach [10]

Augmented Reality

Dieser Teil der Mixed Reality ist näher an der Realität anzusiedeln. Im Allgemeinen versteht man unter Augemented Reality das Anreichern der Realität durch virtuelle Inhalte. Dabei ist entscheidend, dass die Erweiterung nicht statisch bleibt, sondern sich aktiv an die Interaktion mit dem Benutzer anpasst. Es lassen sich drei Arten von AR unterscheiden. Bei Video See-Trough-AR wird die Realität durch eine Kamera aufgenommen und durch virtuelle Inhalte ergänzt

auf einem Display dargestellt. Eine andere Möglichkeit ist optisches See-Trough-AR, das ein semitransparentes Display benötigt, mit dem die dahinterliegende Realität durch Einblendung auf das spezielle Ausgabegerät erweitert werden kann. Zuletzt gibt es noch projektionsbasiertes AR, bei dem virtuelle Inhalte auf reale Objekte projiziert werden. Bei dieser Art ist der Nutzer am wenigsten durch Displays oder sonstige Darstellungstechniken eingeschränkt, dafür lassen sich hier nur sehr begrenzt virtuelle räumliche Strukturen schaffen. Dieses Verfahren ermöglicht eine Manipulation der realen Oberflächeneigenschaften. [5]

Unity

Unity ist eine Game-Engine des gleichnamigen amerikanischen Unternehmens Unity Technology. Sie ist eine der am weltweit meistverbreiteten Game-Engines und ermöglicht das Entwickeln von Spielen auf allen möglichen Plattformen wie z.B. Konsolen, PCs und Handys. Neben der reinen Spielentwicklung wird Unity auch zur Erstellung von 3D-Anwendungen in den Bereichen Automotive, Architecture, Aerospace und Film genutzt. Außerdem unterstützt Unity die Möglichkeit, Virtual Reality und Augmented Reality Applikationen zu entwickeln und ist durch das Nutzen von Plug-ins und das Unterstützen von Drittherstellter Software sehr flexibel. Unity konzentriert sich dabei auf die Darstellung von virtuellen Objekten und ermöglicht es durch Skripte in der Programmiersprache C# Abläufe in Echtzeit zu steuern. [17]

AR Foundation

Hierbei handelt es sich um ein von Unity Technology entwickeltes Framework, das es ermöglicht AR-Applikationen für ein breites Spektrum an Endgeräten zu konzipieren. Es beinhaltet dabei die Hauptfunktionen der AR SDKs ARKIT (Apple), ARCore (Google), Magic Leap und HoloLens. AR Foundation ermöglicht es, all diese Funktionen in einem vereinheitlichten Workflow zu nutzen. Darüber hinaus werden Aktualisierungen und Erweiterungen der einzelnen AR SDKs so verarbeitet, dass das fertige Projekt später auf allen Geräten mit sämtlichen verfügbaren Features funktionsfähig ist. [14]

Unity DOTS

DOTS steht für Datenorientierter Technologie-Stack. Er ermöglicht eine bessere Lesbarkeit, schnellere Iteration von C#-Code und zusätzlich Mehrkernprozessoren-Optimierung ohne großen Programmieraufwand. Datenorientiertes Design des Codes gestattet die thermische Kontrolle und die Akkulaufzeit auf Mobilgeräten zu optimieren. Die Auslegung für Mehrkernprozessoren eröffnet die Nutzung des Unity Burst Compilers, der eine erhebliche Performanceverbesserung bei mathematischen Operationen zur Folge hat. [18, 15]

3.2 Funktionsweise AR

ARCore

ARCore ist Googles Entwicklerplattform, um mobile Augmented Reality Anwendungen zu konzipieren. Ziel der Plattform ist es, immersive Benutzererlebnisse mit virtuellen Inhalten in der Realität zu bieten. Die Plattform wurde im Jahre 2017 vorgestellt und wird inzwischen von 1.2 Milliarden Android Geräten unterstützt. Um virtuelle Inhalte in der realen Welt authentisch zu platzieren, muss die Anwendung die Umgebung und die eigene Position genau erfassen. Dies erfolgt über die im Handy verbaute Kamera und INS Sensoren. [8]

Feature Tracking

Ein wichtiger Bestandteil von ARCore ist die Tiefen-API, diese liefert der AR-Anwendung Informationen zum Platzieren von Inhalten. Dabei ist das Feature Tracking ein wichtiger Bestandteil. Beim Feature Tracking werden markante Punkte von einem Algorithmus im Bild aufgespürt und mit den Features der anderen Bilder verglichen bzw. zugeordnet. Mit den durch Bewegung verursachten unterschiedlichen Bildern der Kamera kann mithilfe von Triangulation aus den bekannten Feature Points siehe Abbildung (3.2) der Abstand berechnet werden. [8]

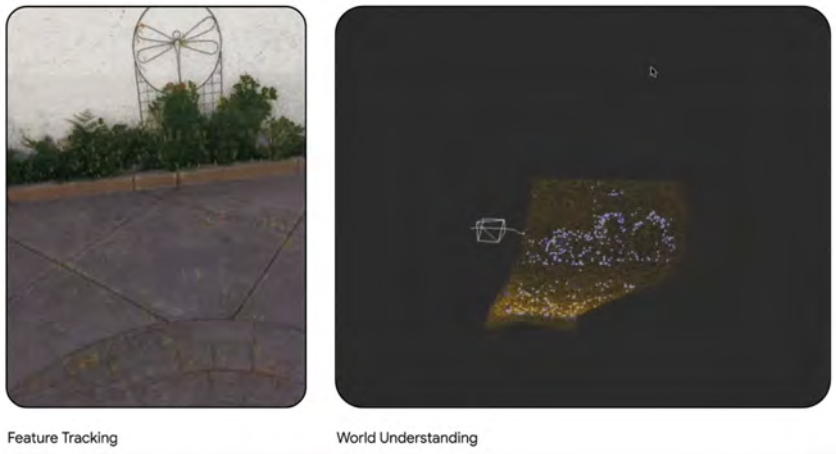

Abbildung 3.2: Feature Tracking nach [8]

Visual-Inertial Sensor Fusion

Ein großes Problem beim vorher genannten Feature Tracking sind unscharfe Kamerabilder, die durch schlechte Lichtverhältnisse oder schnelles Bewegen der Kamera entstehen. Um bei ungünstigen Kamerabedingungen eine stabile Anwendung zu ermöglichen, nutzt ARCore das Konzept Visual-Inertial Sensor Fusion. Dabei wird der INS Sensor, also die Beschleunigungs- und Drehsensoren des Handys genutzt. Bei einer typischen Kameraframerate von 30 Hz und einer INS Messrate von mehr als 100 Hz kann die Lage der Kamera zwischen den Bildern weiter aktualisiert und Unsicherheiten durch das Feature Tracking eliminiert werden. [8]

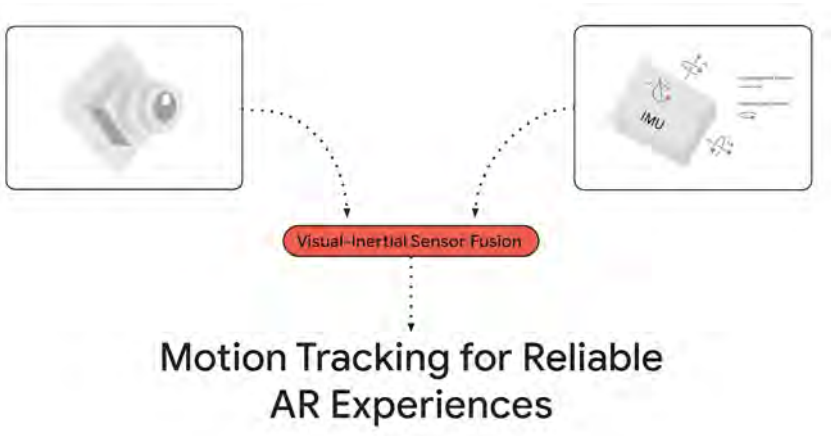

Abbildung 3.3: Visual -Inertial Sensor Fusion nach [8]

Maschine Learning based IMU Tracking

Wenn das Feature Tracking über einen längeren Zeitraum keine verlässlichen Daten zur Verfügung stellt, übernimmt weiterhin das INS System die Positionierung. Hier ergibt sich das Problem, dass sich die Fehler des INS über die Zeit aufsummieren, wodurch die Position immer ungenauer wird und so die virtuellen Objekte in der AR-Anwendung anfangen, sich zu bewegen. Dazu wurde ein neuronales Netzwerk entwickelt, das die Unsicherheiten in der INS Berechnung korrigiert. Dieses wurde mit echten Bewegungsdaten trainiert und kann die INS Messdaten in eine Positionierung umwandeln. Dies ist möglich, da menschliche Bewegungen nicht zufällig sind, sondern meistens gewissen Mustern folgen. So funktioniert die AR-Anwendung auch unter schwierigen Bedingungen beständig. [8]

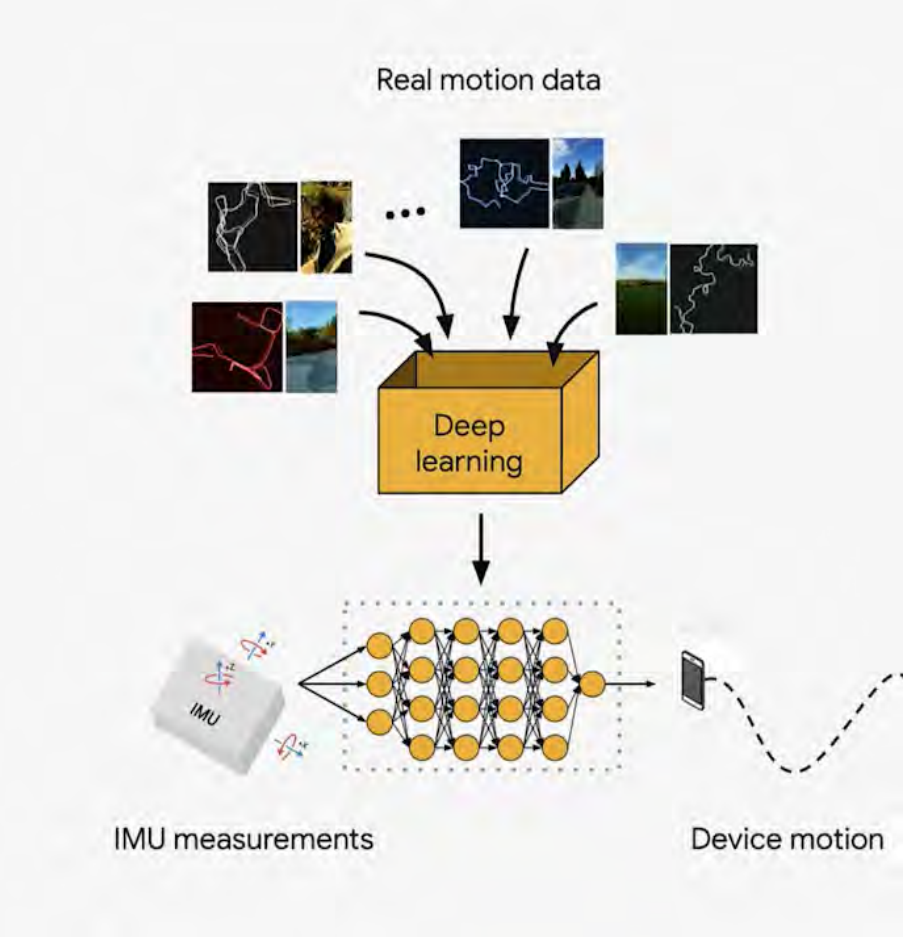

Abbildung 3.4: Maschine Learning based IMU Tracking nach [8]

3.3 Geobasisdaten

Bei den Geobasisdaten handelt es sich um Geodaten, die in Bayern durch die Bayerische Vermessungsverwaltung zur Verfügung gestellt werden. Diese Geodaten kombinieren eine Sachauskunft mit einer Position oder einem geografischen Bereich der Erde. Ihnen werden durch Koordinaten eine bestimmte örtliche Lage auf der Oberfläche der Erde zugeordnet. Sie sind flächendeckend, einheitlich standardisiert und auf dem neuesten Stand. Im Folgenden werden einige Beispiele von Geobasisdaten des Freistaats Bayern nach [9] erläutert:

- *3D-Gebäudemodelle* werden in verschiedenen Varianten bzw. Genauigkeitsabstufungen LoD (Level of Detail) angeboten. Zum Beispiel das einfachere LoD1, bei dem nur die Gebäudeumrisse dargestellt werden und das LoD2 bei dem zusätzlich die Dachform enthalten ist. Die Modelle werden aus Daten im Liegenschaftskataster und durch Laserscanning-Befliegungen abgeleitet und fortlaufend durch Einmessungen von Gebäuden ergänzt. [9]
- *Digitale Orthophotos (DOP)* sind korrigierte Luftbilder, die die Erdoberfläche in Maßstab und Lage richtig wiedergeben. Sie sind in Echtfarben, aber auch im Infrarotbereich für Vegetationsanalysen erhältlich. Jährlich wird die Hälfte der digitalen Orthophotos aktualisiert. Sie sind in verschiedenen Auflösungen verfügbar. [9]
- *Luftbilder* sind zurück bis in das Jahr 1941 verfügbar. Sie dienen der Kampfmittelräumung, aber auch Historikern und Heimatforschern. Luftbilder weisen im Gegensatz zu Orthophotos noch Verzerrungen auf. [9]
- *Orientierte Luftbilder* enthalten zusätzlich Informationen zur Orientierung der Luftbildkamera während der Aufnahme, dadurch lassen sich stereoskopische Auswertungen durchführen. [9]
- *Digitales Oberflächenmodell (DOM)* wird aus Luftbildern abgeleitet und zusammen mit den Orthophotos berechnet. Es zeigt die Oberfläche der Erde und alle darauf befindlichen Objekte wie zum Beispiel Pflanzenbewuchs und Bebauung. [9]
- *Digitales Geländemodell (DGM)* bietet eine Darstellung des Geländes im 3D-Modell ohne Vegetation und Bebauung. Die erhältliche Auflösung reicht

von 1 Meter bis 200 Meter Rasterdaten. Das DGM wird aus Befliegungen mit Laserscannern abgeleitet. [9]

- *Topografische Karte* ist sowohl gedruckt als auch digital erhältlich. Hier wird der gesamte sichtbare Teil der Oberfläche der Erde grafisch dargestellt. Kartenauszüge sind in verschiedenen Maßstäben zu erhalten und zeichnen sich durch einen großen Gehalt an Informationen aus. [9]
- *Uraufnahmen und historische topographische Karten*: Bereits Anfang des 19. Jahrhunderts wurden in Bayern erste Uraufnahmen erstellt. Diese wurden im Verlauf als sogenannte Positionsblätter aufbereitet und mit topographischen Aufnahmen ergänzt. Die historischen Karten liegen heute noch als Rasterdaten und Nachdrucke vor. [9]

Weitere Geobasisdaten sind das Amtliche Liegenschaftskataster-Informationssystem (ALKIS), die Tatsächliche Nutzung (TN), die Bodenschätzungsdaten, das Digitale Landschaftsmodell (DLM), die Hauskoordinaten, die Hausumringe (HU), der Satellitenpositionierungsdienst SAPOS und die Geodateninfrastruktur Bayern. [9]

3.4 Routing

A*-Algorithmus

Der A*-Algorithmus dient dazu, in einem Graphen die kürzeste Verbindung zwischen zwei Knoten zu finden. Dabei ist wichtig, dass alle Kantengewichte positiv sind. Der A*-Algorithmus gilt als Erweiterung des Dijkstra-Algorithmus. Im Unterschied zu anderen Suchalgorithmen nutzt der A*-Algorithmus eine Schätzfunktion, um sein Ziel schneller zu finden. Diese wird als Heuristik bezeichnet. Wenn es einen Weg zwischen zwei Knoten gibt, wird immer der beste Weg gefunden. Der Algorithmus ist optimal und zeichnet sich durch Vollständigkeit aus. [19]

Der Algorithmus wählt fortwährend die Knoten aus, die wahrscheinlich am besten den kürzesten Weg zum Endpunkt haben. Für jeden bekannten Knoten wird durch eine Metrik f(x) ein Wert errechnet, der zur Bestimmung des aussichtsreichsten Knotens dient. Dabei wird geschätzt, wie lang der Weg zum Zielknoten im Optimalfall ist, außerdem werden die Kosten, um zu diesem spezifischen Knoten zu kommen, mit in die Betrachtung gezogen. [19]

$$f(x) = g(x) + h(x) \tag{3.1}$$

Die Gleichung 3.1 gibt dabei einen Wert an, der sich aus den bisherigen Kosten g(x) und der Heuristik h(x) zusammensetzt, dabei gibt h(x) eine Schätzung ab, wie hoch die Kosten bis zum Ziel sind. Es ist wichtig, dass die Heuristik die Kosten nie überschätzt. Bei der räumlichen Wegfindung eignet sich die Luftlinie zum Ziel, da diese den optimalen Weg darstellt. [19]

Bei der Durchführung des Algorithmus müssen die Knoten in drei verschiedene Kategorien unterteilt werden [19]:

- *Unbekannte Knoten*: Diese Knoten wurden vom Algorithmus noch nicht in Betracht gezogen und er weiß auch nicht, wo sie sich befinden. In der Startphase sind alle Knoten bis auf den Startknoten unbekannt. [19]

- *Bekannte Knoten*: Diese Knoten wurden bereits vom Algorithmus erkannt und in einer sogenannten Open List abgespeichert. Um jeweils den aussichtsreichsten Knoten aus dieser Liste zu nehmen, werden die zugehörigen f(x) Werte in der Liste gespeichert und nach dem Minimum sortiert. Findet sich ein besserer Weg zum Knoten, verbessert sich sein f(x). [19]
- *Abschließend untersuchte Knoten*: Bei diesen Knoten ist der beste Weg bereits bekannt. Diese Knoten werden in der sogenannten Closed List gespeichert. Diese sorgt dafür, dass ein Knoten nicht doppelt untersucht und somit die Laufzeit minimiert wird. Beim Start des Algorithmus ist die Closed List noch leer, da noch keine Knoten untersucht wurden. [19]

Jeder Knoten, der bereits bekannt ist oder abschließend untersucht wurde, bekommt seinen bisher besten Vorgänger zugeordnet. So lässt sich später der optimale Weg finden. [19]

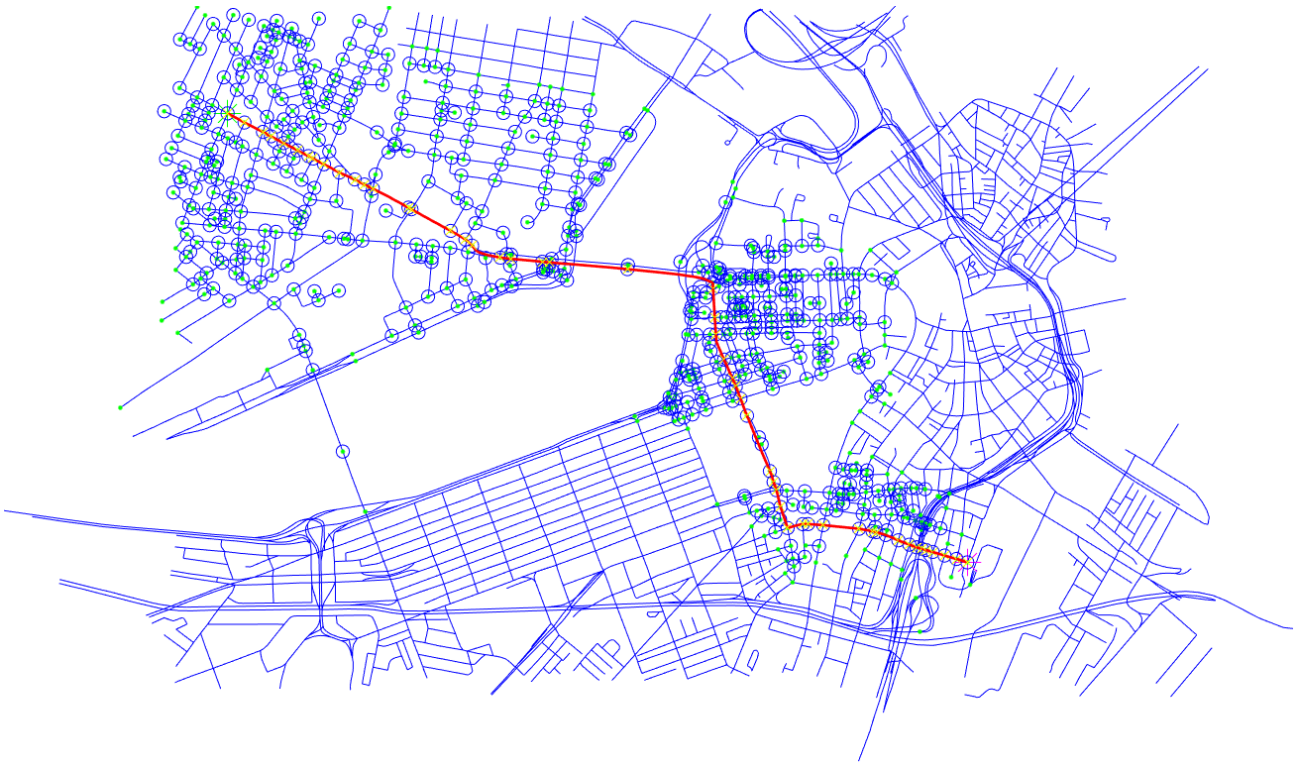

Abbildung 3.5: A*-Algorithmus

Wenn ein Knoten ausgewertet wird, werden alle Knoten mit einer Verbindung zu ihm der Open List hinzugefügt. Sie bekommen den aktuellen Knoten als

Vorgänger zugeordnet. Der betrachtete Knoten wird der Closed List hinzugefügt. Wenn einer der neu hinzugekommenen Knoten schon auf der Closed List steht, findet er keine erneute Betrachtung. Steht er schon in der Open List, wird sein neuer f(x) Wert mit dem alten verglichen und gegebenenfalls der f(x) Wert und Vorgängerknoten aktualisiert. Dies ist der Fall, wenn der neue f(x) Wert niedriger als der alte ist. Der Algorithmus stoppt, sobald der Zielknoten analysiert wird und sucht anhand der in jedem abgeschlossenen Knoten gespeicherten Vorgänger den optimalen Weg zum Startknoten. Andernfalls sucht der A*-Algorithmus so lange, bis er keine neuen Knoten in der Open List findet und er abbrechen muss. [19]

3.5 Lokalisierung

Die Standortermittlung erfolgt über Satellitennavigationssysteme. Dabei sind unter anderem das amerikanische NAVSTAR GPS, das russische GLONASS, das chinesische BEIDOU und das europäische GALILEO vorhanden. All diese Systeme basieren auf der gleichen Grundidee mithilfe von Satellitenkonstellationen einen Standpunkt auf der Erde zu bestimmen.

Absolute Ortung

Alle modernen globalen Navigationssatellitensysteme (GNSS) arbeiten passiv. Das bedeutet, GNSS-Signale werden kontinuierlich ausgesandt und bei Bedarf von einer Empfangseinheit ausgewertet.

Grundsätzlich kann man die Funktionsweise aller Systeme auf eine Streckenmessung zwischen den Satelliten und dem jeweiligen Empfänger zurückführen. Das Messen der Strecke erfolgt, indem die Satelliten zu festen Zeiten Signale zur Erde senden. Der Empfänger kann diese Signale empfangen und anhand des Zeitstempels in den Signalen die Laufzeit zum Satelliten berechnen. Die Strecke ergibt sich aus der Ausbreitungsdauer und der Ausbreitungsgeschwindigkeit. Um jetzt die Position des Empfängers zu bestimmen, wird noch der Standort des Satelliten benötigt. Die Signale des Satelliten übertragen diese Information. Nun kann eine Gleichung aufstellt werden, mit der die Koordinaten

des Empfängers berechnen werden. [4]

$$(\Delta T_i \cdot v)^2 = S_i^2 = (X_i - X_E)^2 + (Y_i - Y_E)^2(Z_i - Z_E)^2; i = 1, 2, 3 \qquad (3.2)$$

Dabei bezeichnet:

ΔT: die gemessene Laufzeit der Satellitensignale
v: die Ausbreitungsgeschwindigkeit der Satellitensignale
X_i, Y_i, Z_i: die bekannten Satellitenkoordinaten
X_E, Y_E, Z_E: die unbekannten Empfängerkoordinaten

Die Gleichung 3.2 ist grundsätzlich lösbar, jedoch ergibt sich ein neues Problem. Die Genauigkeit, die ein Zeitmesser im Empfänger haben muss, ist enorm hoch, da sich Fehler direkt auf die Genauigkeit der Positionierung auswirken. Allerdings ist eine solch hohe Genauigkeitsanforderung für den Empfänger nicht besonders wirtschaftlich. [4]

Um Satellitennavigation trotzdem zu ermöglichen, wurde folgende Lösung entwickelt. Es wird eine Ungenauigkeit beim Messen der Zeit akzeptiert. Dieser Fehler wird nun als Δt der Gleichung hinzugefügt. Da nun die Messungen im Empfänger zu früh oder zu spät durchgeführt werden, sind auch die daraus errechneten Strecken falsch. Demnach wird in GNSS die Strecken zum Satelliten auch Pseudostrecken genannt, da sie mit einer Unsicherheit behaftet sind. Für die tatsächliche Strecke gilt $S = (\Delta T + \Delta t) \cdot v$ mit ΔT = gemessene Laufzeit und Δt = unbekannter Uhrenfehler. [4]

$$(\Delta T_i + \Delta t \cdot v)^2 = S_i^2 = (X_i - X_E)^2 + (Y_i - Y_E)^2(Z_i - Z_E)^2; i = 1, 2, 3, 4 \quad (3.3)$$

Da es nun jedoch zu jedem Messzeitpunkt vier Messunbekannte gibt, bedarf es bei jeder Messung mindestens vier Satelliten, um vier Gleichungen zu lösen und damit zu einer ausreichend genauen Position zu kommen. [4]

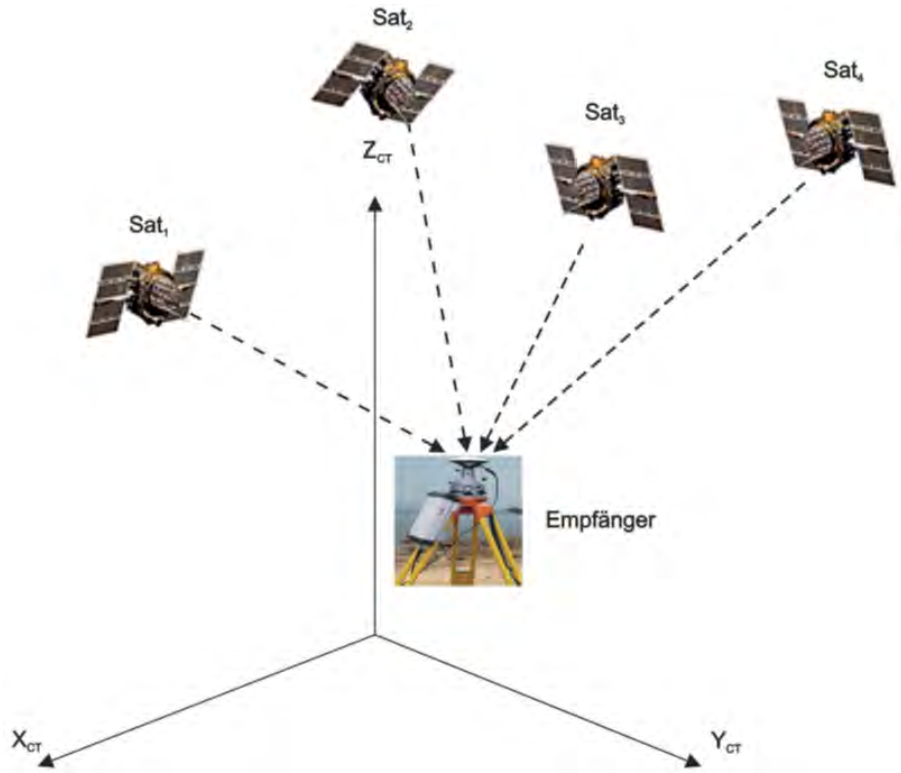

Abbildung 3.6: GNSS Schaubild nach [4]

Differenzielle Ortung

Bei dem oben genannten Verfahren kann eine Genauigkeit im „Meterbereich“ erreicht werden. Um noch bessere Messergebnisse zu erzielen, wurde das Verfahren differenzielle Ortung entwickelt.

Die letzten Fehlerquellen bei der Bestimmung der Position, die eine genauere Ortung verhindern, sind atmosphärische Störfaktoren in Thopo- und Ionosphäre sowie Fehler bei der Bestimmung der Ausbreitungsgeschwindigkeit. Um diese Fehlereinflüsse zu minimieren, wurde die differenzielle Ortung entwickelt. Dabei gibt es nun zwei Empfänger, die miteinander kommunizieren können. [4]

Der erste Empfänger wird über einem bekannten Punkt stationiert. Dauerhaft wird seine Position neu ermittelt. Diese Ist-Werte werden mit den Soll-Werten verglichen und liefern Aussagen über Ausbreitungsgeschwindigkeit und atmosphärische Einflüsse des Signals. [4]

Der andere Empfänger wird in der Nähe des Referenzempfängers aufgebaut. Durch die räumliche Nähe wird davon ausgegangen, dass die Fehlereinflüsse bei der Referenzstation und dem zweiten Empfänger, auch Rover genannt, identisch sind. Der Rover kann nun zusätzlich zu den Satellitensignalen die Daten der Referenzstation verarbeiten und in seine Ausgleichung miteinbeziehen. So lassen sich Genauigkeiten im Millimeterbereich erreichen. [4]

4 Werkzeuge

4.1 Erzeugung

Geobasisdaten

Die Beschaffung der Geodaten erfolgt über die interne Geodatenbestellung des Landesamts für Digitalisierung, Breitband und Vermessung (LDBV) und wurde von Herrn Thomas Meier (Referat 85 Luftbildmessung und Fernerkundung) zur Verfügung gestellt. Dabei muss zwischen Raster- und 3D-Daten unterschieden werden. Für die Präsentation in einer AR-Anwendung eigenen sich Rasterdaten wie digitales Orthophoto und Uraufnahme. Diese werden als TIF-Tiles mit einer Auflösung von 40 cm ausgegeben, ebenso wird ein Auszug der Web-Karte mit dem Smart-Mapping-Tool erstellt. Als 3D-Geodaten werden das LOD2 Gebäudemodell und das digitale Geländemodell herangezogen. Diese werden von den internen Servern im Esri ASCII GRID Format exportiert. Das 3D-Mesh zeigt die komplette Erdoberfläche voll texturiert. Dieses Produkt befindet sich aktuell noch in der Entwicklung und wird daher von Entwicklungsservern des dafür zuständigen Referates im OBJ-Format bezogen. Dieser Datensatz wird in das Computerprogramm 3DS Max geladen und mithilfe von Photoshop die Farben der Texturen verbessert. Das fertige 3D-Modell wird dann als FBX-Datei abgespeichert.

Shape-Files

Für die spätere Routing-Funktion wird die Lage der Straßen in Vector-Form benötigt. Dafür bieten sich die Daten des Amtlichen Topographisch-Kartographischen Informationssystem (ATKIS) an. Diese enthalten viele für das Routing nicht relevante Attribute, daher werden alle überflüssigen Attribute vorher mit Matlab entfernt. In diesem Zuge bietet es sich an, die Länge jeder Straße als Attribut abzuspeichern, um später diese nicht jedes Mal beim Start neu berechnen zu müssen. Die neu entstandenen Shape-Files werden mit ihren dazugehörigen DBF-Files abgespeichert und sind bereit für den Import in Unity.

4.2 Zusammenführung

Um alle verschiedenen Datentypen in einer Datei zu kombinieren, wird das Programm Infraworks genutzt, da es Daten georeferenziert zusammenführen kann. Dabei wird zuerst das Gelände importiert und anschließend die Rasterdaten als Textur darübergelegt. Die 3D-Daten werden einzeln hinzugefügt. Die verschiedenen gesammelten Daten können als Ganzes aber auch gesondert ausgegeben werden. Das Aufziehen einer Box siehe Abbildung (4.1) erlaubt außerdem den Export eines bestimmten Bereiches. Zur Weiterverarbeitung in Unity wird der Datensatz im FBX-Format ausgegeben.

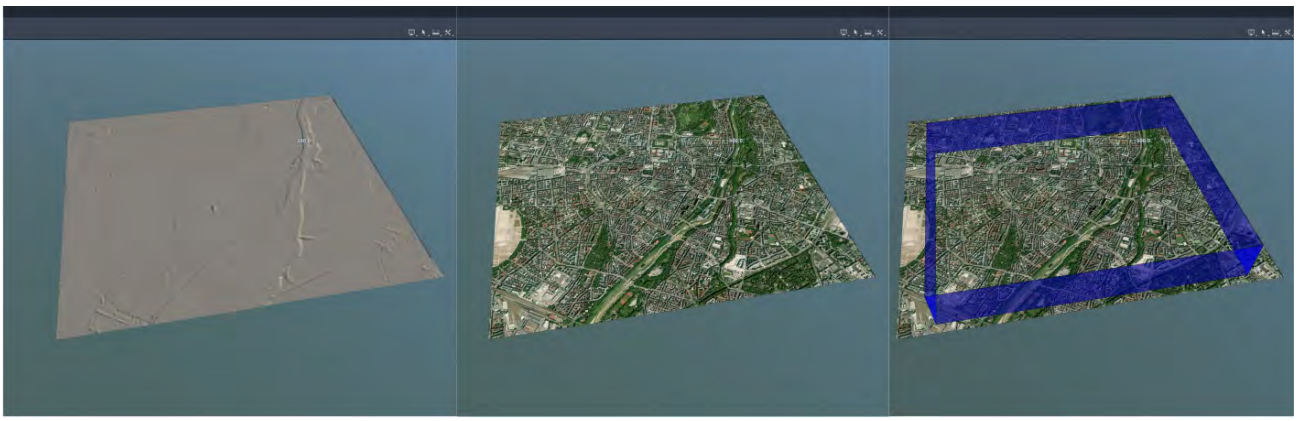

Abbildung 4.1: Workflow in Infraworks

5 Vorgehensweise

Die Grundlage für das Entwickeln einer AR-App ist das Erstellen eines Projektes in Unity. Dafür wird die Software-Version 2020.3.36f1 genutzt. Um später alle Features umsetzten zu können, müssen diverse Packages dem Projekt hinzugefügt werden. Die wichtigsten dabei sind AR Foundation in der Version 4.1.10, ARCore XR Plug-in Version 4.1.10, Unity Burst Version 1.6.6 und Unity Jobs in der Version 0.51.0-preview.32. Diese Konfiguration der Unity Game-Engine bildet die Basis für alle folgenden Implementierungen.

5.1 AR-Workflow

Der grundlegende Schritt bei der Nutzung von AR Foundation ist das Hinzufügen der Gameobjekte AR Session Origin und AR Session. Diese enthalten alle nötigen Subsysteme, um später eine gut funktionierende AR-Applikation zu ermöglichen. Die standardmäßige Unity-Kamera wird dabei durch eine eigene, auf die AR-Anwendung optimierte Kamera ersetzt, die später auch die realen Inhalte durch die physikalische Kamera des mobilen Endgerätes aufnimmt und der Anwendung zur Verfügung stellt.

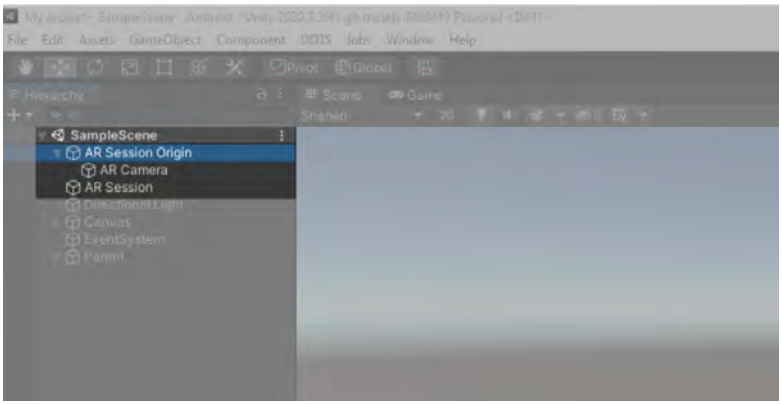

Abbildung 5.1: AR Session Grundlagen

In der späteren Anwendung sollen Bilder in der realen Welt erkannt und darauf die virtuellen Objekte dargestellt werden. Dazu muss der AR Session Origin noch um einen AR Tracked Image Manager erweitert werden, dieser stellt später die Schnittstelle zur Verfügung, mit der die digitalen Modelle platziert werden können. Dieser Manager wiederum benötigt eine sogenannte Reference Image Library . In der Bibliothek müssen alle Bilder angegeben werden, die die Anwendung erkennen soll. Für eine bessere Erkennung empfiehlt es sich, die spätere Größe des Bildes in der realen Welt anzugeben. Für diese Arbeit liegt eine physische Karte des Versuchsgebietes zugrunde.

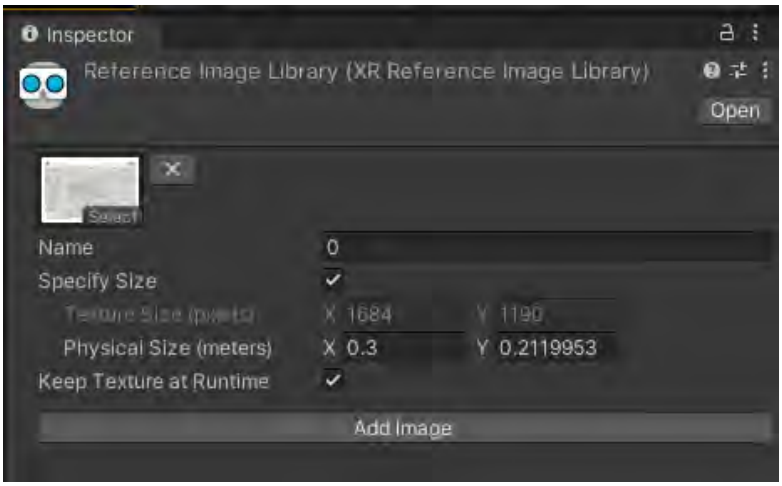

Abbildung 5.2: Reference Image Library

Damit ist das Fundament für eine AR-App mit Image Tracking gelegt. Um die virtuellen 3D-Inhalte auch richtig zu platzieren, wurde das Skript PLaceArContent geschrieben. Es wird dem AR Session Origin angefügt und es hat Zugriff auf den AR Tracked Image Manager. Die 3D-Modelle in Unity lassen sich gruppieren und hierarchisch gliedern. Das PlaceArContent Skript kann so ein einzelnes Gameobject nutzen, das alle 3D-Daten enthält. Dieses Gameobject dient auch als Mittelpunkt des Koordinatensystems, in dem sich die 3D-Modelle befinden und skaliert werden. Jedes Mal, wenn die AR-Subsysteme ein Bild aus der Reference Image Library in der echten Welt erkennen oder sich die Position eines bereits erkannten Bildes ändert, erhält das Skript eine relative Position und Rotation zwischen dem Bild und der Kamera. Bei erstmaligem Erkennen werden die 3D-Modelle sichtbar gemacht. Wenn man die relative Position und Rotation nun den virtuellen Daten zuweist, erhält man auf dem Ausgabedisplay eine Darstellung der 3D-Modelle über dem realen Bild. Sobald die AR-Subsysteme

das reale Bild verloren haben, werden auch die virtuellen Daten unsichtbar geschaltet.

5.2 Vektordatenprozessierung

Die Shape-Files nehmen einen besonderen Platz in diesem Projekt ein, da sie nicht direkt von Unity eingelesen werden können. Weil sie aber für das Routing von essenzieller Wichtigkeit sind, wurde folgende Lösung gefunden.

Allgemeines Einlesen

Im Internet wurde bereits ein Skript veröffentlicht, das den einfachen Import von Shape-Files in die Unity-Anwendung ohne Abhängigkeiten von Dependencies ermöglicht [13]. Das Einlesen der Geometrie mit dem Shape-File-Reader funktioniert sehr schnell und zuverlässig. Jedoch unterstützt das Skript nicht das Einlesen der dazugehörigen DBF-Files, in denen sich die Attribute zu den Geometrien des Shape-Files befinden. Darin sind jedoch auch die Längenangaben der Straßen enthalten, die für das Routing benötigt werden. Dies würde auch ein zukünftiges Erweitern der Anwendung um neue Funktionen stark erschweren. Um das Einlesen von DBF-Files zu ermöglichen, wird ein weiteres Skript in Form einer DLL-Datei genutzt [1]. Auch dieses Skript hat den Vorteil von keinerlei Dependencies und kann damit durch das Open Source basierte Unity genutzt werden. Der ursprüngliche Shape-File-Reader wird so erweitert, dass er nach der Shape-Datei auch die DBF-Datei einliest und die einzelnen Attributwerte ihren Shape-Geometrien anfügt. Mit diesem Verfahren lassen sich Shape-Files mit ihren Attributen in Unity-Projekte hineinladen und stehen zur weiteren Verwendung zur Verfügung.

Besonderheiten unter Android

Bei Windows-Systemen funktioniert das Einlesen zunächst ohne Probleme, da die Anwendung auf die Ordner der Entwicklungsumgebung zugreifen kann, jedoch nicht auf Android-Geräten. Dies ist insofern kritisch, da die spätere AR-App für Android vorgesehen war. Um dieses Problem zu beheben, wird ein weiteres Skript erstellt. Der Kern des Problems dabei ist, dass Unity die Daten in den allgemeinen Projektordner nicht in die ausführbare Datei auf dem

Smartphone kopiert. Für dieses Problem gibt es bereits den StreamingAssets Ordner. Hierbei handelt es sich um einen speziellen Ordner, der auch später in der Anwendung vorhanden ist. Damit kann die AR-Anwendung die Shape-Files aus ihrem eigenen Ordner laden. Noch komplexer stellt sich die Lage bei Android dar. Das Handybetriebssystem ändert den Pfad der StreamingAssets mit jeder Installation. Das Einlesen mit dem Shape-File-Reader ist so nicht möglich. Um dieses neue Problem für Android-Geräte zu lösen, gibt es bereits den persistentDataPath. Die Lösung ist, die Daten der StreamingAssets durch das System selbst auszulesen und in den persistentDatapath, auf den die Anwendung selbst zugreifen kann, zu kopieren. Dieser Vorgang muss für das Shape-File aber auch für das DBF-File durchgeführt werden. Nach dieser Vorbereitung kann der Shape-File-Reader genauso wie auf einem Windowsrechner durch die Anwendung aufgerufen werden.

5.3 Routing

Initialisierung

Nachdem alle Daten wie oben beschrieben eingelesen wurden, können diese für das Routing vorbereitet werden. Da die späteren Algorithmen für das Unity Jobsystem ausgelegt sind, muss auch die Vorbereitung entsprechend erfolgen. Da die objektorientierte Programmierweise nicht unterstützt wird, müssen die Daten in Structs gespeichert werden. Für die L-Liste, die alle Knoten zur Navigation enthält, wird das Struct L-Record definiert. Dieses enthält den Hochwert, den Rechtswert, die Länge der dazugehörigen Straße, von der der Knoten entweder Start- oder Endpunkt ist und Variablen für die vier Einzelwerte, die später für die Wegfindung benötigt werden. Außerdem wird ein Struct erstellt, das später für die Open List verwendet wird. Es enthält nur die Position des Knotens in der L-Liste und die bisherige Strecke des zurückgelegten Weges.

Danach werden die Shape-File-Werte in ein Array mit L-Record Structs überführt. Die Start- und Endknoten jeder Shape-Linie werden als L-Record aufgelistet. Im nächsten Schritt wird die A-Matrix erstellt. Diese soll die Information der Topologie der Vektordaten des Shape-Files enthalten. Dafür wird die Länge der Verbindung zwischen zwei Knoten als Wert für die A-Matrix angenommen, sollte keine Verbindung vorhanden sein, wird 0 als Wert eingetragen. Die Erstellung

erfordert viele mathematische Operationen, deswegen wird hierfür das Unity Jobsystem mit dem Burst Compiler genutzt. Ein Unity Job ist einer normalen Methode sehr ähnlich. Die Besonderheit ist jedoch, dass nur einfache Datentypen insbesondere mit dem Unity Burst Compiler, unterstützt werden. Für die Erstellung der A-Matrix werden die Werte der L-Liste in ein Native-Array kopiert. Dieses wird dem Job übergeben und muss nach erfolgreicher Beendigung verworfen werden. Die A-Matrix im Job wird folgendermaßen erstellt, da die A-Matrix die Verbindungen zwischen allen Knoten enthalten muss, entspricht die Länge der Reihen und Spalten jeweils der Länge der L-Liste. In das Feld zwischen Start- und Endknoten wird die Länge der Straße geschrieben. Danach wird errechnet, welche weiteren Knoten in der Nähe liegen. Unterschreiten Sie einen gewissen Grenzwert, kann man davon ausgehen, dass die beiden Knoten auch in der Realität eine Verbindung besitzen, z. B. durch eine Kreuzung. Da Unity für die schnelle Verarbeitung in Jobs keine Matrizen unterstützt, wird die A-Matrix so in den Daten gehalten, dass jede neue Spalte an die alte angefügt wird und man einen Vektor erhält. Mit diesem Konzept kann man eine schnelle Berechnung der Grundlagen für das spätere Routing während des Starts der Anwendung garantieren.

Lokalisierung

Für die Lokalisierung kann der Standortdienst des Mobilgerätes verwendet werden. Dazu wird ein neues Skript erstellt, das auf dem Beispiel der Unity Documentation [16] basiert, dieses prüft zuerst die Berechtigungen, ob der Standortdienst verwendet werden darf. Ist die App nicht berechtigt, wird um Erlaubnis zur Nutzung des Dienstes gebeten. Wenn diese Bedingungen erfüllt sind, wird kontrolliert, ob der Standortdienst überhaupt aktiviert ist. Ist dies nicht der Fall, bricht das Programm ab und der Standort wird erst wieder bei einem erneuten Start der App abgefragt. Ist die Standortfunktion aktiviert, wird so lange gewartet, bis die Initialisierung des Dienstes abgeschlossen ist. Werden dabei 20 Sekunden überschritten oder die Initialisierung schlägt fehl, wird ebenfalls abgebrochen. Wenn die Standortfunktion des Android-Systems verfügbar ist, wird sie dem Unity-Skript in Längen- und Breitenangabe übergeben. Die amtlichen Geodaten werden jedoch in UTM-Koordinaten bereitgestellt, deswegen wird ein neues Skript zur Umrechnung von geografischen Koordinaten mit Längen- und Breitenangaben in UTM-Koordinaten erstellt. Sobald der Standort von Unity erkannt wird, wird er in UTM32-Koordinaten umgerechnet. Der Standort kann nun mit einem virtuellen 3D-Objekt über der physischen Karte dargestellt wer-

den. Dabei wird seine relative UTM-Position gegenüber dem südwestlichsten Punkt aller dargestellten 3D-Geodaten berechnet und anschließend der Maßstab mithilfe der Abmessungen der Geodaten in UTM und in den Abmessungen des eignen Koordinatensystems von Unity, berechnet. Dies ist nötig, da die spätere Nutzung der 3D-Modelle in einer AR-Umgebung ein Übereinstimmen von UTM- und Unity-Koordinaten unmöglich macht.

Wegfindung

Der erste Schritt beim Finden der kürzesten Strecke ist das Angeben des Start- und Zielpunktes. In der AR-App soll es möglich sein, auf dem Bildschirm den Start- und Zielpunkt durch Antippen des Displays zu bestimmen, dafür wird unter die 3D-Geodaten im Editor eine Ebene angelegt. Diese Ebene wird mit einem Mesh Collider-Komponenten versehen. Dieser ermöglicht es, von der Unity eigenen Physics Engine erkannt zu werden. Zum Start des Routingprozesses wird ein Button im User-Interface erstellt. Erst nach Betätigung dieses Buttons werden Eingaben verarbeitet. Einmal aktiviert, wartet der Algorithmus auf Eingaben durch das Touchdisplay. Die Bildschirmkoordinaten des ersten Berührungspunktes werden ermittelt. An diesem Startpunkt wird senkrecht zur Ebene des Bildschirms ein Raycast-Strahl entsendet. Dieser Strahl kann Objekte aus der Physics Engine erkennen und mit ihnen interagieren. Wenn er die vorher genannte Ebene trifft, erhält er die Koordinaten des Treffpunktes in Unity-Weltkoordinaten. Aus diesen Koordinaten lassen sich lokale Koordinaten errechnen, die später das Bezugssystem in der AR-Umgebung bilden. Der erste Punkt wird immer als Startpunkt angenommen und die eigentliche Routenfindung wird erst gestartet, wenn auch ein Endpunkt durch Antippen des Displays gefunden wurde.

Der eigentliche Routingprozess beginnt damit, dass die lokalen Koordinaten aus den 3D-Geodaten in UTM-Koordinaten umgerechnet werden. Dies geschieht über den Maßstab zwischen den realen Abmessungen der Daten und den Unity Internen. Damit stehen nun der Start- und Zielpunkt in UTM zur Verfügung. Im nächsten Schritt wird für beide Punkte geprüft, wo sich der nächste bekannte Knoten in den Vektordaten befindet. Dafür wird durch alle bekannten Knoten iteriert und die kürzeste euklidische Distanz zur Bestimmung des besten Knotens genutzt. Damit sind die Knoten für Start- und Zielpunkt innerhalb der Shape-File-Vektordaten bekannt. Im nächsten Schritt kann der A*-Algorithmus durchgeführt werden.

Da dieser Algorithmus sehr rechenintensiv ist, wird der Prozess mit dem Unity Jobsystem gestaltet. Dazu müssen alle Daten, die verarbeitet werden sollen, wieder in Native Arrays und simple Datentypen umgewandelt werden. Dabei wird unter anderem die vorher errechnete L-Liste und A-Matrix benötigt. Es wird ein Native Array als sogenannte Closed Liste definiert, darin wird abgespeichert, ob ein Knoten schon betrachtet wurde. Im nächsten Schritt wird die sogenannte Open Liste definiert, sie enthält alle Knoten, die für eine weitere Untersuchung in Betrachtung kommen. Da in dieser Liste ständig die Werte sortiert, hinzugefügt und gelöscht werden, erwies es sich als sinnvoll, hier eine Native List zu verwendet. Diese unterstützt alle vorher genannten Operationen. Zuletzt werden noch die Start- und Zielpunkte übergeben und ein Native Array initialisiert, in dem die Knotenreihenfolge des optimalen Wegs abgespeichert wird, danach kann der Job gestartet werden.

Für die vielen mathematischen Operationen wird zusätzlich wieder der Burst-Compiler aktiviert, der die spätere Anwendung deutlich performanter machen kann. Im eigentlichen Job wird für jeden Knoten erst die euklidische Distanz zum Zielpunkt berechnet. Diese dient als Schätzung, weist dem Algorithmus die ungefähre Richtung und wird in der H-Variable jedes Knotens abgespeichert. In die Variable F, die später die Gesamtkosten des Weges abschätzen soll, wird für jeden Knoten der Wert auf unendlich gesetzt. Der Startknoten bekommt außerdem den Wert -1000 in die P-Variable gesetzt. Bei den normalen Knoten wird hier der Vorgänger auf dem vermuteten Weg abgespeichert. Wird der Zielpunkt gefunden, kann der Algorithmus den Weg von Knoten zu Knoten zurückverfolgen und erkennt den Startpunkt dann an dem Wert -1000. Danach wird eine While-Schleife gestartet, sie wird so lange durchgeführt, bis der Zielpunkt gefunden ist oder keine neuen Knoten zur Untersuchung mehr zur Verfügung stehen. Es wird fortwährend ein einzelner Knoten genauer betrachtet. Am Anfang ist es der Startknoten und in den folgenden Iterationen immer der Knoten, der dem Algorithmus am vielversprechendsten erscheint. Für den gerade betrachteten Knoten werden alle Verbindungen zu anderen Knoten ermittelt. Für jede Verbindung wird der F-Wert des neuen Knotens verglichen. Dabei wird der F-Wert neu aus drei Komponenten berechnet. Diese Komponenten sind sein H-Wert, die Länge der Verbindung zwischen beiden Knotenpunkten aus der A-Matrix und die bisherigen Kosten des aktuell betrachteten Knotens. Ist der neue F-Wert niedriger als der bisher gespeicherte Wert, werden die Wegkosten G des neuen Knotens aus den Kosten G, des aktuell betrachteten Knotens und der Länge der Verbindung aus der A-Matrix errechnet. Der F-Wert wird durch

den vorher neu errechneten ersetzt und als Vorgängerknoten des neuen Knotens wird der aktuell betrachtete Knoten eingetragen.

Im Anschluss wird der Knoten der Open List hinzugefügt, es sei denn, er ist bereits in dieser enthalten. Wenn alle Verbindungen des aktuell betrachteten Knotens überprüft wurden, wird er in der Closed List auf den Status geschlossen gesetzt und kann später nicht mehr überprüft werden. Er wird anschließend aus der Open List entfernt. Der neue Knoten, der zur Betrachtung gezogen wird, ist der Knoten aus der Open List mit dem niedrigsten F-Wert. Mit diesem Verfahren wird immer der optimale Weg gefunden, außer es besteht keiner. Wenn der Zielpunkt gefunden wurde, wird der optimale Weg rekonstruiert anhand des besten Vorgängers jedes Knotens und in das Array geschrieben, das schließlich vom Algorithmus zurückgegeben wird.

Nachdem der A*-Algorithmus den optimalen Weg gefunden hat, müssen alle Native Arrays und Listen verworfen werden. Die Knoten des besten Weges werden davor noch in ein globales Array kopiert, um für die weitere Verarbeitung zur Verfügung zu stehen. Die Route soll in den 3D-Daten dargestellt werden. Dafür wird ein Unity LineRenderer Component genutzt. Mit diesem kann man Linien im 3D-Raum darstellen. Der LinieRenderer bekommt alle Einzelpunkte der Shape-Linien, die er darstellen soll. Dabei ergab sich das Problem, dass der LineRenderer seine Linie nicht unterbrechen konnte, dies führte zu unschönen Fehllinien, da die Endpunkte mancher Straßen nicht den Anfangspunkten anderer entsprachen. Die Lösung dafür ist, dass die Linie des LineRenderer vor und nach jeder Straße unter den dargestellten Geodaten verschwindet und erst wieder bei Beginn der nächsten Straße in den sichtbaren Bereich zurückkehrt.

5.4 User-Interface

Bei der Bedienung der App kann zwischen zwei verschiedenen Arten unterschieden werden. Zum einen gibt es die statische 2D-Oberfläche und die Interaktion mit 3D-Infosymbolen in den Geodaten.

Statische 2D-Oberfläche

Hierbei handelt es sich um das klassische Userinterface mit Buttons und Textfeldern. Es gibt einen Menü-Button, mit dem sich ein Panel mit den Namen aller unterschiedlichen Geodaten öffnen lässt. Das Drücken des jeweiligen Buttons aktiviert die Geodaten, die dann über der realen Karte angezeigt werden.

Ein weiterer Button dient zur Aktivierung des Routingprozesses. Nachdem er gedrückt wurde, wird ein Antippen in den Bereich der 3D-Geodaten als Eingabe des Start- respektive des Endpunktes gewertet. Mit jedem Drücken des Buttons wird der Prozess neu gestartet und vorherige Eingaben oder Wegfindungen verworfen.

Der letzte Button im 2D-User-Interface ist der Skalierungsbutton, er ermöglicht es, die Geodaten zu vergrößern oder auch zu verkleinern. Das dazugehörige Skript erkennt zwei gleichzeitige Touch-Eingaben. Dabei wird bei der ersten Berührung des Displays mit zwei Fingern die Länge zwischen beiden Punkten gemessen. Verändert sich der Abstand zwischen beiden durch Bewegen der Finger auf dem Display, so wird das Verhältnis aus Anfangsentfernung und aktueller Entfernung errechnet und mit dem Maßstab der 3D-Modelle multipliziert. Dies beendet einerseits die Georeferenzierung mit der realen Karte, auf der anderen Seite ermöglicht es das Vergrößern und einfachere Entdecken von Details auf der 3D-Karte.

3D-Infosymbole

Eine Besonderheit dieser Applikation soll die Interaktion mit Geodaten sein. Dies wird zum einen durch das Routing unterstrichen, zum anderen soll es auch möglich sein, Informationen über ausgewählte Sehenswürdigkeiten zu erhalten. Dafür wurden Informationen [21, 27, 24, 26, 22, 23, 25, 20] zusammengefasst und auf kleinen Tafeln dargestellt. Jeder Tafel wurde auch ein Bild der Sehenswürdigkeit angefügt. Diese wurden entweder selbst aufgenommen oder lizenzfreie Bilder aus dem Internet verwendet. Dabei ergaben sich zwei Probleme, da die Infotafeln alle statisch waren, ließen sie sich nur schwer lesen, wenn der Nutzer sich um die physische Karte herumbewegte, außerdem verdeckten die Infotafeln die eigentlichen Geodaten, wenn sie alle gleichzeitig aktiviert wurden. Um diese Probleme zu lösen, wurden zwei Skripte geschrieben. Das Skript FaceCamera beobachtet den Winkel zwischen dem Gameobjekt und der Unity-Kamera, die die Szene darstellt. Durch das Übertragen dieses Winkels auf die

Y-Drehung der Infomarker werden die Infotexte immer auf den Benutzer ausgerichtet.

Für das Verdecken der Geodaten wurde auch eine Lösung gefunden. Die Infotexte sind standardmäßig ausgeschalten. Um eine Sehenswürdigkeit zu Kennzeichnen wurden kleine transparente Symbole erstellt, die die eigentliche Sehenswürdigkeit nicht überdecken. Durch Antippen dieser Symbole ist es möglich, die Infotexte sichtbar zu machen. Dafür wurde um das Symbol herum ein Rechteck erstellt. Das Rechteck wird mit einem Box Collider ausgestattet und so konfiguriert, dass es in der späteren Anwendung nicht sichtbar ist. Dazu wurde ein passendes Skript geschrieben, das auf Touch-Eingaben wartet. Wird der Bildschirm berührt, so wird von dieser Stelle aus ein Raycast-Strahl senkrecht zur Displayoberfläche ausgesandt. Trifft er nun auf eben genannte Box, aktiviert er eine Animation, die den Infotext erscheinen lässt. Wird die Box nochmals getroffen, verschwindet der Text wieder. Außerdem wird darauf geachtet, dass immer nur eine Infotafel zu sehen ist. Wird eine neue Infotafel aktiviert, so deaktiviert sich die vorherige von selbst.

6 Ergebnisse

6.1 Versuchsfeld

Hierfür wurde die Münchner Innenstadt ausgewählt. Diese bietet sich insofern an, da viele Sehenswürdigkeiten vorhanden sind. Es existiert ein dichtes Straßennetz zur Navigation und bietet für das Gebäudemodell eine hohe Bebauungsdichte. Die Anzahl der potenziellen Nutzer ist ebenfalls sehr groß.

Abbildung 6.1: Versuchsfeld

6.2 Darstellung Funktionsweise

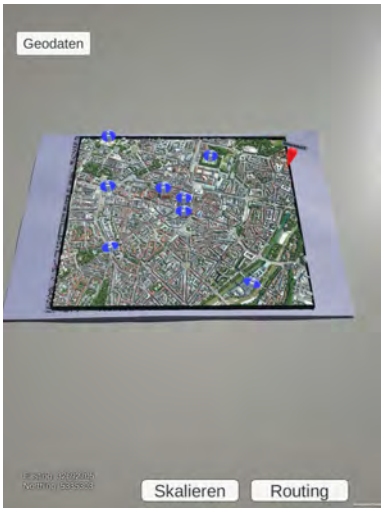

Abbildung 6.2: Start der Anwendung

Abbildung 6.3: Bewegung der Kamera

Abbildung 6.4: Vergrößern des 3D-Modells

Abbildung 6.5: Verkleinern des 3D-Modells

Abbildung 6.6: Benutzer-Menü

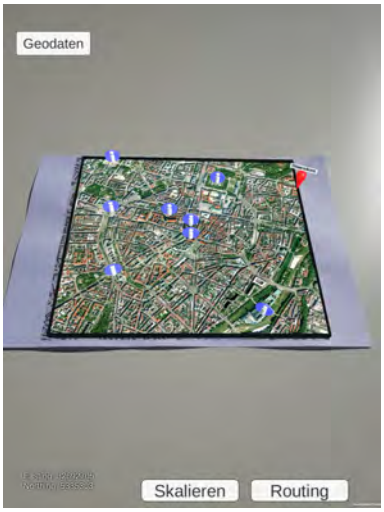

Abbildung 6.7: Orthophoto

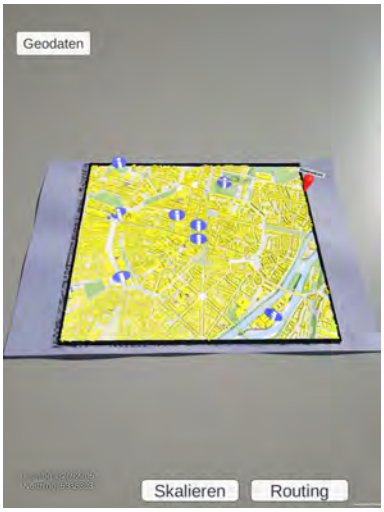

Abbildung 6.8: Gebäudemodell mit Webkarte

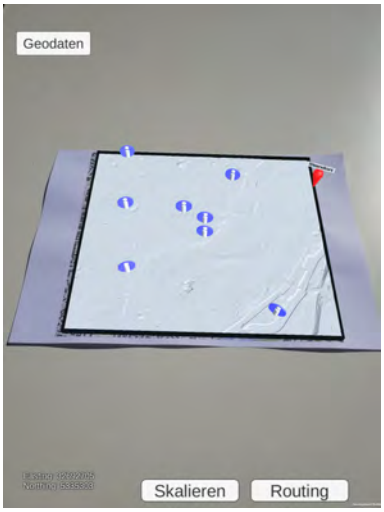

Abbildung 6.9: Digitales Geländemodell

Abbildung 6.10: Webkarte

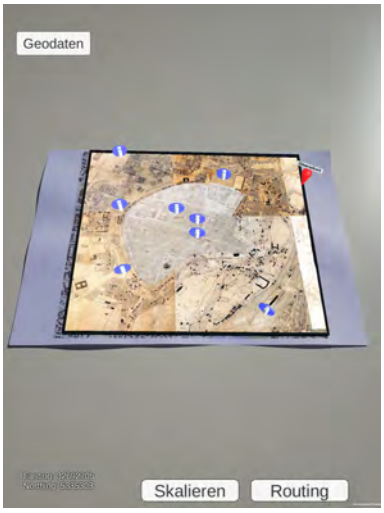

Abbildung 6.11: Uraufnahme

Abbildung 6.12: 3D-Modell mit Infotext

Abbildung 6.13: Gebäudemodell mit Infotext

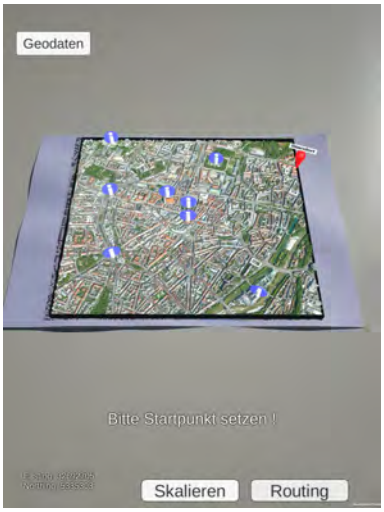

Abbildung 6.14: Starten der Routingfunktion

Abbildung 6.15: Eingabe des Endpunktes

Abbildung 6.16: Berechnete Route im 3D-Modell

Abbildung 6.17: Route mit einer Sehenswürdigkeit

6.3 Performance

Die Performance ist für eine AR-Applikation essenziell wichtig. Wenn die Berechnung der einzelnen Frames zu lange dauert, werden virtuelle Objekte für den Nutzer zu spät mit der Realität abgestimmt und es kommt zu unschönen Verzögerungen.

Die Performance dieser App wurde auf zwei verschiedenen Testgeräten untersucht. Zum einem auf einem LG V30 Handy aus dem Jahr 2017 mit dem damals sehr rechenstarken Snapdragon 835 Prozessor und einer Adreno 540 Grafikkarte. Zum anderen auf einem Samsung S3 Tablet, ebenfalls aus dem Jahr 2017, mit einem etwas leistungsschwächeren Snapdragon 820 Prozessor und einer Adreno 530 Grafikkarte.

Die Applikation läuft auf beiden Geräten flüssig mit 50 Bildern pro Sekunde und das Tracking funktioniert bei entsprechenden Lichtverhältnissen ebenfalls zuverlässig. Bei schnellen Bewegungen wird deutlich, dass das stärkere LG Handy das Tracking besser beibehalten kann. Beim Tablet ist offensichtlich, dass die Anwendung etwas langsamer läuft, wenn dynamische Situationen auftreten. Hält man das Gerät jedoch still, läuft die Anwendung auf beiden Geräten gut.

Mit dem Unity DOTS System ist es möglich, den Burst Compiler ein- und auszuschalten. Im Folgenden werden die Ladezeiten mit und ohne Burstunterstützung dargestellt.

Mit Burst-Compiler in Sekunden:

	Start	Routing
LG V30	0.01965	0.02272
Samsung Tab S3	0.02945	0.04952

Ohne Burst-Compiler in Sekunden:

	Start	Routing
LG V30	0.07352	0.03570
Samsung Tab S3	0.06523	0.07377

7 Diskussion

In dieser Bachelorarbeit soll gezeigt werden, welche Möglichkeiten es gibt, 3D-Geodaten auf mobilen Plattformen interaktiv zu präsentieren. Es ist möglich, dies im Zusammenspiel mit einer herkömmlichen Präsentation von Geodaten in Form einer Karte zu realisieren.

Die Ergebnisse verdeutlichen, dass eine vielschichtige Darstellung eines abgegrenzten Bereichs gut umsetzbar ist, außerdem ist es möglich, die Präsentation von Geodaten durch weitere Sachinformationen zu ergänzen. Die Routingfunktion unterstreicht dabei die Variabilität der Geodaten. Sie erweitert die reine Darstellung um eine funktionale Komponente, die dem Nutzer zusätzliche Interaktion ermöglicht. Die GPS-Funktion soll dem Anwender dabei seinen eigenen Bezug zu den Geodaten vermitteln. Es wird versucht, die Bedienung so anwenderfreundlich und einfach wie möglich zu halten.

Unity erwies sich insgesamt als eine sehr gute Basis für die Applikation. Durch die hohe Flexibilität mit einem großen Umfang an spezialisierten Packages und Dritthersteller-Plug-ins ist es einfach möglich, ein zuverlässiges Grundgerüst für die spätere App zu erstellen. Die Kombination von Unity und AR Foundation erlauben es außerdem, sowohl für Android als auch für IOS zu entwickeln.

Durch die Vorbearbeitung der Daten verlief das Einlesen und Verarbeiten der 3D-Geodaten in Unity problemlos. Schwierigkeiten ergaben sich eher bei den Straßen-Vektoren der Shape-Files. Es dauerte einige Zeit, bis die Shape-Files auch in der gewünschten Form eingelesen wurden. Die Darstellung mit dem LineRenderer war schwieriger als anfangs vermutet. Ursprünglich war an dieser Stelle angedacht, die Straßen über Draw-Calls im Open GLS Renderer zu zeichnen. In einer 2D-Testapplikation für Android erwiesen sich sowohl LineRenderer als auch Draw-Calls als probates Mittel. Bei der Umsetzung des Ganzen als 3D-Augmented Reality Anwendung zeigte sich jedoch, dass die Draw Calls eine sehr große Performanceeinbuße zur Folge hatten und deswegen ungeeignet waren. Daher musste ein Workaround für die Limitierungen des LineRenderers

entwickelt werden. Im ersten Versuch wurde für jedes Straßenstück ein einzelner LineRenderer initialisiert. Bei längeren Strecken wirkte sich das jedoch stark auf die Performance der App aus und verursachte bei weniger leistungsfähigeren Testgeräten einen Absturz der Anwendung. Das Konzept mit einem einzelnen LineRenderer, der nach jeder Straße unter den 3D-Geodaten verschwindet, erwies sich als beste Lösung. Lediglich bei der Nutzung der Routingfunktion zusammen mit dem Mesh lässt sich manchmal das Verschwinden der Linie unter die Geodaten beobachten. Durch das Vergrößern des Liniendurchmessers konnte dies jedoch stark reduziert werden.

Es ergaben sich trotzdem auch einige Limitierungen in der fertigen Applikation. Diese sind zum Teil auf die AR-Subsysteme zurückzuführen. So ergibt sich manchmal das Problem, dass der Marker von der Applikation nicht erkannt wird. Dies ist größtenteils bei schlechten Lichtverhältnissen der Fall. Ein weiteres Problem ist das Verschwinden der Geodaten, wenn das Mobil-Gerät zu nah an die physische Karte kommt. Dies ist für die Erfahrung der AR-Applikation hinderlich, da es ein Betrachten von Details verhindert. Um dieses Defizit auszugleichen, wurde das Skalieren durch den Benutzer hinzugefügt. Ein weiteres Thema ist die Größe der späteren Applikation, obwohl es sich noch um ein relativ kleines Gebiet handelt, hat diese schon eine Größe von ca. 600 MB. Das ist ausschließlich auf die Geodaten zurückzuführen. Dies ist sehr hinderlich, da es eine Skalierung der App sehr schwierig macht. Die Aktualität der Geodaten kann nicht gewährleistet werden. Um die Daten auf den neuesten Stand zu bringen, wäre es jedes Mal nötig, diese im Editor einzulesen und einen komplett neuen Build der App zu erstellen.

Die Performance der Applikation war auf beiden Testgeräten zufriedenstellend. In Anbetracht der sehr rechenaufwendigen Prozesse während der Laufzeit der App ist auch der Stromverbrauch nicht zu hoch. Die Analyse des Burst Compilers zeigt, dass er sich für die Verarbeitung von Vektordaten sehr gut eignet. Mit einer bis zu dreimal schnelleren Verarbeitung bietet er große Chancen für Anwendungen mit größeren Datenmengen.

Die Anwendung bietet auch noch Potenzial für zukünftige Erweiterungen und Verbesserungen. So ist das aktuelle Tracking nur auf vorher ausgewählte Bilder bzw. Kartenausschnitte begrenzt. Eine sinnvolle Erweiterung wäre hier das dynamische Tracken von Ausschnitten aus Karten, die in einer applikationseignen Datenbank gespeichert sind. Dies würde die Anwendung deutlich flexibler machen. Dementsprechend müsste die Anwendung, aber auch die Position des

getrackten Ausschnittes in der Datenbank ermitteln, um anschließend die 3D-Geodaten so auszuschneiden, dass eine Georeferenzierung entsteht. Die Schwierigkeiten bestünden hier im Tracking und im Anlegen einer Datenbank, die allen Anforderungen gerecht wird und trotzdem nicht zu viel Speicherplatz belegt. Dies gilt auch für die 3D-Geodaten, diese müssten entweder komprimiert werden, wenn sie einen größeren Bereich abdecken oder über einen Webserver On-Demand von der Anwendung zur Darstellung heruntergeladen werden. Hier wäre allerdings auch eine Komprimierung notwendig, da der Übertragungsprozess sonst zu viel Zeit in Anspruch nehmen würde und bei schlechten Internetverbindungen gar nicht durchführbar wäre.

8 Fazit

Abschließend lässt sich sagen, dass eine Augmented Reality Anwendung zu einer neuartigen Präsentation von Geodaten erstellt werden kann, in der es nicht nur um die reine Darstellung, sondern auch um eine Interaktion mit den Daten geht. Dies wurde mithilfe von Unity, AR Foundation und Unitys diverser „Plug-ins“ realisiert.

Hierbei entstand ein lauffähiger Prototyp einer App für interaktive Präsentationszwecke von Geodaten. Die Kombination und Integration verschiedenster Geodaten in einer AR-Umgebung sind gut umsetzbar. Durch die App, in der die Münchner Innenstadt dargestellt wird, zeigt sich, wie gut die amtlichen Geodaten für die Endnutzung geeignet sind. Die Zusammenführung aller Daten erfordert eine vielschichtige Vorbearbeitung durch Softwareprogramme, insbesondere bei den Shape-Files für die Routenplanung. In Zukunft wäre es aber möglich, die Methodik dieser App auf andere Städte anzuwenden. Die prototypartige App lässt sich auf neueren Smartphones und Tablets mit ausreichender Rechenleistung ausführen, sodass eine große Basis zur Nutzung durch den Endverbraucher gegeben ist.

Der Prototyp der Augmented Reality App könnte, wie bereits in der Diskussion angesprochen durch dynamisches Tracken von Kartenausschnitten, die aus bereits bestehenden Datenbanken erkannt werden, in Zukunft noch verbessert werden. Ferner wäre eine Internetverbindung denkbar, mit der die Geobasisdaten fortwährend aktualisiert und bei Bedarf verändert werden könnten.

Danksagung

An dieser Stelle möchte ich mich bei allen Personen, die auf unterschiedliche Art und Weise zum Gelingen dieser Arbeit beigetragen haben, bedanken.

Mein erster Dank gilt dabei meinem Betreuer Herrn Prof. Dr. Thomas Abmayr für die hilfreichen Anregungen während des gesamten Betreuungszeitraums. Die regelmäßigen Meetings halfen sehr bei dem Entwicklungsprozess dieser Arbeit.

Ich bedanke mich außerdem nachdrücklich bei Herrn Thomas Meier und dem LDBV für die Unterstützung. Die Fachexpertise war stets sehr hilfreich und die zur Verfügung gestellten Geodaten bilden die Grundlage dieses Projekts.

Literaturverzeichnis

[1] E. P. Ahmed Lacevic, Ferdinando Santacroce, "Fastdbf," Github, Jun. 2018, [Online; Stand 20. Mai 2022]. [Online]. Available: https://github.com/SocialExplorer/FastDBF

[2] C. Bade, "Untersuchungen zum einsatz der augmented reality technologie für soll/ist-vergleiche von betriebsmitteln in der fertigungsplanung," Ph.D. dissertation, Otto-von-Guericke-Universität Magdeburg, 2012, [Online; Stand 5. August 2022]. [Online]. Available: https://d-nb.info/1053914377/34

[3] D. Balla, M. Zichar, R. Toth, E. Kiss, G. Karancsi, and T. Mester, "Geovisualization techniques of spatial environmental data using different visualization tools," *Applied Sciences*, vol. 10, no. 19, 2020, [Online; 20 Mai 2022]. [Online]. Available: https://www.mdpi.com/2076-3417/10/19/6701

[4] M. Bauer, *Vermessung und Ortung mit Satelliten: globale Navigationssatellitensysteme (GNSS) und andere satellitengestützte Navigationssysteme.* Wichmann Heidelberg, 2018, [Online; 4. Juni 2022]. [Online]. Available: https://www.ebook.de/de/product/30014144/manfred_bauer_vermessung_und_ortung_mit_satelliten.html

[5] W. Broll, "Augmentierte realität," in *Virtual und Augmented Reality (VR/AR)*, R. Dörner, W. Broll, P. Grimm, and B. Jung, Eds. Springer, 2013, pp. 248–250.

[6] H. Buchholz, C. Brosda, and R. Wetzel, "Science center to go a mixed reality learning environment of miniature exhibits," 01 2010.

[7] M. Christen, U. Clement, and A. Meyer, "Mixed reality anwendungen mit 3d-stadtmodellen," *Wissenschaftlich-Technische Jahrestagung der DGPF: München, Germany*, pp. 328–340, 2018.

[8] L. B. Konstantine Tsotsos, "A deep dive into arcore," Jul. 2022, 20. [Online]. Available: https://www.youtube.com/watch?v=MM786WD-Gv8&t=362s

[9] B. u. V. Landesamt für Digitalisierung, "Die amtlichen geobasisdaten der bayerischen vermessungsverwaltung," Mar. 2019, [Online; Stand 22. Juli 2022]. [Online]. Available: https://www.ldbv.bayern.de/file/pdf/4628/Produkt%C3%BCbersicht_DIN%20A%204.pdf

[10] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," *IEICE Trans. Information Systems*, vol. vol. E77-D, no. 12, pp. 1321–1329, 12 1994.

[11] A. Morrison, A. Oulasvirta, P. Peltonen, S. Lemmelä, G. Jacucci, G. Reitmayr, J. Näsänen, and A. Juustila, "Like bees around the hive: A comparative study of a mobile augmented reality map," 04 2009, pp. 1889–1898.

[12] L. Santopietro, G. Faruolo, F. Scorza, A. Rossi, M. Tancredi, A. Pepe, and M. Giordano, "Geovisualization for energy planning," in *Computational Science and Its Applications – ICCSA 2020*, O. Gervasi, B. Murgante, S. Misra, C. Garau, I. Blečić, D. Taniar, B. O. Apduhan, A. M. A. C. Rocha, E. Tarantino, C. M. Torre, and Y. Karaca, Eds. Cham: Springer International Publishing, 2020, pp. 479–487.

[13] Suntabu, "Unity shape file importer by esri," Online, Dec. 2018, [Online; Stand 05. Mai 2022]. [Online]. Available: https://www.suntabu.com/post/unity-shape-file-importer-by-esri/

[14] U. Technology, "Ar foundation," Online, Aug. 2022, [Online; Stand 02 August 2022]. [Online]. Available: https://unity.com/de/unity/features/arfoundation

[15] ——, “Burst user guide,” Aug. 2022, [Online; Stand 02 August 2022]. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.burst@1.0/manual/index.html#burst-user-guide

[16] ——, “Documentation locationservice.start,” Online, Aug. 2022, [Online; Stand 03. August 2022]. [Online]. Available: https://docs.unity3d.com/ScriptReference/LocationService.Start.html

[17] ——, “Our company,” Online, Aug. 2022, [Online; Stand 02 August 2022]. [Online]. Available: https://unity.com/our-company

[18] ——, “Performance by default,” Online, Aug. 2022, [Online; Stand 02 August 2022]. [Online]. Available: https://unity.com/dots

[19] Wikipedia, “A*-algorithmus — wikipedia, die freie enzyklopaedie,” 2022, [Online; Stand 19. Juli 2022]. [Online]. Available: https://de.wikipedia.org/w/index.php?title=A*-Algorithmus&oldid=219778390

[20] ——, “Deutsches museum — wikipedia, die freie enzyklopädie,” 2022, [Online; Stand 27. August 2022]. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Deutsches_Museum&oldid=225543179

[21] ——, “Frauenkirche (münchen) — wikipedia, die freie enzyklopädie,” 2022, [Online; Stand 27. August 2022]. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Frauenkirche_(M%C3%BCnchen)&oldid=225486911

[22] ——, “Hofgarten (münchen) — wikipedia, die freie enzyklopädie,” 2022, [Online; Stand 27. August 2022]. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Hofgarten_(M%C3%BCnchen)&oldid=222662036

[23] ——, “Königsplatz (münchen) — wikipedia, die freie enzyklopädie,” 2022, [Online; Stand 27. August 2022]. [Online]. Available: https://de.wikipedia.org/w/index.php?title=K%C3%B6nigsplatz_(M%C3%BCnchen)&oldid=225475654

[24] ——, “Marienplatz (münchen) — wikipedia, die freie enzyklopädie,” 2022, [Online; Stand 27. August 2022]. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Marienplatz_(M%C3%BCnchen)&oldid=224666690

[25] ——, “Sendlinger tor — wikipedia, die freie enzyklopädie,” 2022, [Online; Stand 27. August 2022]. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Sendlinger_Tor&oldid=221591850

[26] ——, “St. peter (münchen) — wikipedia, die freie enzyklopädie,” 2022, [Online; Stand 27. August 2022]. [Online]. Available: https://de.wikipedia.org/w/index.php?title=St._Peter_(M%C3%BCnchen)&oldid=224796800

[27] ——, “Stachus — wikipedia, die freie enzyklopädie,” 2022, [Online; Stand 27. August 2022]. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Stachus&oldid=224065425