HOCHSCHULE ANHALT University of Applied Sciences

Fachbereich für Architektur, Facility Management und Geoinformation

Masterarbeit

Echtzeit-Visualisierung von Flurstücken des Liegenschaftskatasters in Augmented Reality

Prüfer: Prof. Dr. Holger Baumann
Zweitprüfer: Prof. Dr. Andreas Wichmann

Vorgelegt von:
Eike Marvin Sippel, 5063640
Geoinformationssysteme

Abgabe:
Laatzen, 23.02.2024

Bibliografische Beschreibung

Sippel, Eike Marvin: Echtzeit-Visualisierung von Flurstücken des Liegenschaftskatasters in Augmented Reality

Hochschule Anhalt, Fachbereich Architektur, Facility Management und Geoinformation, Institut für Geoinformation und Vermessung, Masterarbeit, Februar 2024

61 Seiten, 5 Tabellen, 20 Abbildungen, 2 Anlagen

Zusammenfassung: In der Arbeit wird untersucht, wie mit einem Prototyp auf Basis von Augmented Reality Bibliotheken oder Software Development Kits eine Visualisierung mit Daten aus dem Liegenschaftskataster möglich ist. Vorerst wird die Literatur herangezogen, um den Stand von Augmented Reality abzuschätzen und welche Bibliotheken und Software Development Kits für die Entwicklung eines Prototypen geeignet sind. Diese werden miteinander sowie mit Anforderungen an die Applikation verglichen. Es wird die Bibliothek AR.js untersucht, ein Protoyp entwickelt und mit Blick auf die Anforderungen beurteilt. Das Software Development Kit ARCore SDK wird ebenfalls zur Untersuchung herangezogen und ein Prototyp erstellt. Dieser Prototyp wird daraufhin untersucht und bewertet bezüglich Genauigkeit und Visualisierung. Es werden Anwendungsfälle auf Basis der Ergebnisse beschrieben. Außerdem wird erläutert, wie die Genauigkeit des Prototypen noch weiter gesteigert werden könnte. Im Ausblick werden zukünftige Szenarien beschrieben und wie diese in einer produktiven Entwicklung umgesetzt werden sollten.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass die von mir eingereichte Masterarbeit selbstständig unter Benutzung der angegebenen Literatur sowie Hinweisen der Betreuer angefertigt wurde.

Bezüglich der Masterarbeit wird der Hochschule Anhalt, insbesondere dem Institut für Geoinformation und Vermessung, und der Jade Hochschule, insbesondere dem Institut für Angewandte Photogrammetrie und Geoinformatik, ein einfaches Nutzungsrecht eingeräumt.

Laatzen, 23. Februar 2024

Eike Marvin Sippel

Danksagung

Ich bedanke mich bei dem Landesamt für Geoinformation und Landesvermessung Niedersachsen, die mir dieses Thema vorgeschlagen und die Mittel zur Bearbeitung der Masterarbeit zur Verfügung gestellt haben. Außerdem danke ich den Prüfern, die es mir ermöglicht haben, dieses Thema zu bearbeiten. Besonderen Dank widme ich Prof. Dr. Andreas Wichmann, der mich in dieser Masterarbeit betreut hat und in der Entwicklung der Prototypen viel Feedback gegeben hat. Weiterhin bedanke ich mich bei meinen Arbeitskolleginnen und Arbeitskollegen, die die Entwicklung der Prototypen verfolgt haben und mir die Zeit zur Bearbeitung der Arbeit zur Verfügung gestellt haben. Mein Dank gilt auch Familie sowie Freundinnen und Freunden, die mich aufgrund privater Angelegenheiten und eines Umzuges in den ersten Monaten der Bearbeitung sehr unterstützt haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ALKIS Amtliches Liegenschaftskatasterinformationssystem

API Application Programming Interface

AR Augmented Reality

GNSS Global Navigation Satellite System

JSON JavaScript Object Notation

LGLN Landesamt für Geoinformation und Landesvermessung Niedersachsen

RTK Real-Time Kinematic

SAPOS Satellitenpositionierungsdienst der Deutschen Landesvermessung

SDK Software Development Kit

UTM Universal Transverse Mercator

VPS Visual Positioning System

VR Virtual Reality

1 Einleitung

Liegenschaftsdaten, u. a. Flurstücke, werden aktuell nur mit einer Kartendarstellung verbunden, um eine Visualisierung herzustellen. In dieser Masterarbeit wird untersucht, wie mithilfe von Augmented Reality (AR) eine erweiterte Visualisierung des Liegenschaftskatasters möglich ist.

Flurstücke, die durch Grenzpunkte einen Raumbezug erhalten, sind ein wesentlicher Bestandteil des Liegenschaftskatasters. Viele Grenzpunkte besitzen unterirdische oder keine Abmarkungen. Das Auffinden dieser Grenzpunkte sowie die Übertragung der Flurstücksgrenzen in die Örtlichkeit, sind ohne umfangreiche Vermessungsarbeiten derzeit nicht möglich.

Mit AR können virtuelle Objekte in die Örtlichkeit übertragen werden, indem die Kamera und weitere Sensoren eines mobilen Endgerätes genutzt werden. Die Möglichkeiten der Nutzung von AR-Anwendungen zur Visualisierung von Liegenschaftsinformationen (Flurstücksgrenzen, Grenzpunkte) werden im Rahmen dieser Masterarbeit untersucht. Es erfolgt eine Literaturrecherche zum Stand der Wissenschaft der AR in Bezug auf globale Koordinatensysteme.

Aus der Literaturrecherche sind außerdem mögliche Frameworks und Bibliotheken zur Entwicklung einer AR-Anwendung zu entdecken. Zusammen mit einer Anforderungsliste werden diese gegenübergestellt. Daraus resultierend wird ein Framework oder eine Bibliothek ausgewählt, um einen Prototyp zu entwickeln.

Ein wichtiger Untersuchungspunkt ist die Verknüpfung zum amtlichen Liegenschaftskataster.
Wie können Geodaten mit einem globalen Raumbezug in eine AR-Anwendung integriert werden?

Die Visualisierung der Geodaten wird aufgrund der GNSS-Messung und der AR-Orientierung eine Abweichung zur tatsächlichen Position besitzen.
Für welche Anwendungsfälle ist die ermittelte Lagegenauigkeit mit einer Visualisierung von Liegenschaftsinformationen ausreichend?
Welche Methoden zur Verbesserung der Lagegenauigkeit sind in einer AR-Anwendung möglich?

2 Theoretische Grundlagen

2.1 Augmented Reality

AR ermöglicht eine Erweiterung der realen Welt durch virtuelle Inhalte. Die Technologie basiert auf Endgeräte mit Kameras, z. B. Smartphones, Tablets oder AR-Brillen sowie einer AR-Software. Nach Porter und Heppelmann (2017) wird mithilfe von Sensoren in dem Gerät und Auswertung der Kamerabilder eine Orientierung erzeugt, um eine Darstellung virtueller Inhalte in die reale Umgebung zu ermöglichen.

Die Abgrenzung von AR ist nach Biene u. a. (2021) unter anderem mit dem Virtualitäts-Kontinuum nach Milgram und Kishino möglich. AR ist zwischen Real Environment und Augmented Virtuality eingeordnet und ist Teil der Mixed Reality. Real Environment ist die völlige reale Welt ohne eine Erweiterung virtueller Inhalte. Bei Augmented Virtuality überwiegt der Anteil der Virtualität und kann als Teil des Virtual Reality (VR) angesehen werden, jedoch mit dem Einbezug der realen Welt. Dies bedeutet, dass eine neue virtuelle Welt erschaffen wird. Bei AR ist dies nicht der Fall. Biene u. a. (2021) beschreiben das Virtualitäts-Kontinuum im Detail.

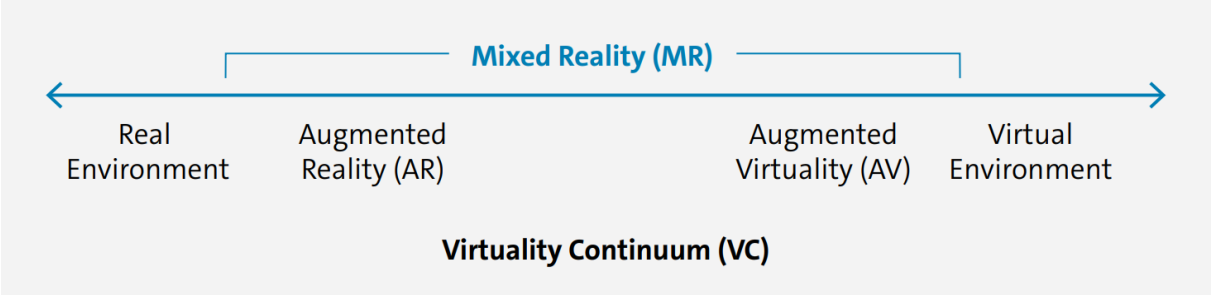

Abbildung 1: Virtualitäts-Kontinuum nach Milgram und Kishino (Biene u. a. 2021)

Eine zukünftige Technologie könnte nach Biene u. a. (2021) die AR-Cloud sein. Diese soll die reale Welt möglichst vollständig abbilden, einen Austausch von Informationen und eine vollständige Kompatibilität mit AR und VR bieten. Damit umfasst diese die Eigenschaften eines digitalen Zwillings, jedoch auf Basis der Verwendung von AR und VR.

Ein Beispiel für bestehende Use-Cases mit AR ist z. B. die Deutsche Bahn, die verschiedene Szenarien mit Einsatz von AR erprobt hat. Ein Einsatz ist die Berechnung der Orientierung über 3D-Maps und Anchors. Mithilfe von Global Navigation Satellite System (GNSS) und einem bereits bestehenden 3D-Modell des Bahnhofs wurde eine Orientierung in der AR-Anwendung ermöglicht.

Im Tourismus bestehen ebenfalls bereits Ansätze zur Verwendung von AR. Beispielsweise wurde in Soest und Lübeck ein Modellprojekt initiiert, welches Informationen zu Sehenswürdigkeiten in einer AR-Anwendung anzeigt oder zerstörte historische Sehenswürdigkeiten innerhalb der Anwendung darstellt, wie sie vor der Zerstörung durch den Zweiten Weltkrieg aussahen.

AR kann in Navigationssystemen eine immersive Darstellung bieten, um dem Nutzenden eine verbesserte Orientierung zu bieten. Vor allem Nutzenden, die mit der Bedienung oder Orientierung an einer 2D-Kartenanwendung Schwierigkeiten besitzen, hilft AR, um eine Route in die Realität einzufügen. Chung, He und Jung (2016) beschäftigen sich mit diesem Thema. Außerdem bietet nach Inman (2019) Google seit 2019 mit Google Maps eine AR-Anwendung zur Navigation.

Arbeiten in der Stadtplanung können durch AR unterstützt werden, nach B. St-Aubin u. a. (2010) und Bruno St-Aubin, Mostafavi und Roche (2012) vor allem im kollaborativen Umfeld, sodass zusammen ein eingefügtes 3D-Stadtmodell betrachtet und bearbeitet werden kann.

Im Bauwesen ist es hilfreich, unterirdische oder geplante Objekte in die Realität mithilfe von AR einzufügen. Talmaki, Dong und Kamat (2010) und Su u. a. (2013) beschäftigen sich mit der Darstellung von unterirdischen Leitungen, um z. B. Kollisionen zu vermeiden. Ratajczak, Riedl und Matt (2019) entwickelten eine Applikation zur Visualisierung von Building Information Models, um geplante Gebäude und deren Fortschritt mit AR darzustellen.

In den folgenden Bereichen, ergänzt nach GIS Geography (2023), kann AR zu einer Erweiterung oder Verbesserung von Erlebnissen oder Informationen führen:

- Spiele und Unterhaltung
- Bildung
- Industrie
- Einzelhandel
- E-Commerce
- Architektur und Bauwesen
- Stadtplanung
- Immobilien
- Tourismus
- Schulung und Simulationen
- Automobilbranche
- Werbung und Marketing
- Kultur
- Innenarchitektur
- Energie und Versorgung

Dies bedeutet, dass AR in sehr vielen Bereichen eingesetzt werden kann, um mithilfe der Erweiterung der Realität neue Möglichkeiten zu erschaffen.

Nach Chen u. a. (2019) ist ein wichtiger und kritischer Bestandteil von AR die 3D-Orientierung des Systems, um die virtuellen Inhalte mit genauer Position einzufügen. Zuerst wird eine Beziehung zwischen der virtuellen Oberfläche, den einzufügenden Modellen und den Orientierungsdaten der Kamera erzeugt. Danach wird die virtuelle Oberfläche mit den Modellen in die Realität an der festgelegten Position eingefügt. Es bestehen mehrere Möglichkeiten, wie diese Beziehung hergestellt wird. Zum Beispiel mithilfe Gerätesensoren wie Beschleunigungssensoren oder auch über die Analyse der Bilder der Kamera. Auch die Kombination der beiden Möglichkeiten wird oft angewendet, um die Genauigkeit zu erhöhen.

Es gibt verschiedene Arten, AR anzuwenden:

- Marker based: Eine für AR-konzipierte Grafik wird genutzt, um die Orienterung zu erzeugen. Die Grafik besitzt hohe Kontraste, um eine Erkennung zu vereinfachen
- Image based: Ein individuelles Bild oder Objekt wird erkannt, an dem die Kamera der AR-Anwendung die Orientierung erzeugt
- Location based: Durch eine Standortbestimmung wird eine Orientierung erzeugt, die mithilfe übergeordneter Koordinatensysteme virtuelle Objekte an Koordinaten ermöglichen

AR-Anwendungen benötigen ein Tracking-Verfahren, um virtuelle Inhalte an die richtige Position einzufügen. Das Verfahren mit einer hohen Genauigkeit zur Anzeige von virtuellen Inhalten ist das "marker based"-Verfahren. Dieses nutzt einen einzigartigen Bildcode, um einen Anker für die virtuellen Inhalte zu erzeugen. Da Flurstücke einzigartige Koordinaten besitzen und flächendeckend bestehen, ist kein kamerabasiertes Tracking möglich. Daher muss die GNSS-Positionierung des Gerätes als primäres Tracking verwendet werden.

2.2 Anforderungen an die Applikation

Die Anwendung muss zur produktiven Nutzung Anforderungen erfüllen. Dies bedeutet, dass die ausgewählte Bibliothek oder das Software Development Kit (SDK) die folgenden Punkte unterstützen muss. Der Prototyp aus der Entwicklung dieser Arbeit muss nicht alle Anforderungen erfüllen, jedoch das Potential dazu besitzen.

- Auf nahezu allen Smartphones oder Tablets ausführbar
- Einfaches Aufrufen oder einfache Installation der Anwendung
- Hohe Performance
- Open Source
- Regelmäßige und aktuelle Updates
- Hohe Skalierbarkeit
- Möglichkeit zur Integration von eigenen Geodaten
- Standortbasierte Funktionen
- Minimale Hürden zur Entwicklung und Testen der Anwendung

Die Anwendung soll auf allen aktuellen Smartphones oder Tablets ausführbar sein, um den Nutzerkreis möglichst groß zu halten. Spezielle Sensoren oder Hardware Spezifikationen, die bestimmte SDKs oder Bibliotheken benötigen, sind im Consumer-Bereich nur selten vorhanden.
Eine einfache Installation oder Aufrufen der Anwendung ist wichtig, um besonderen Aufwand für Nutzende zu vermeiden.
Eine hohe Performance ist wichtig, damit die Anwendung auf einem Smartphone mit zuverlässiger Geschwindigkeit funktioniert. AR-Anwendungen sind rechenintensiv, daher ist dieser Punkt zu beachten.
Open Source SDKs und Bibliotheken sind sofort zu konfigurieren und einzusetzen. Dies erleichtert die Entwicklung eines Prototypen enorm, da sofort und ohne Kosten die Entwicklung begonnen werden kann. Außerdem fallen für einen zukünftigen Einsatz der Anwendung weniger Kosten an. Durch Open Source

sind SDKs und Bibliotheken meist mit stetigen Updates versehen, da meist eine größere Community hinter der Entwicklung steht.
Regelmäßige und aktuelle Updates sind wichtig, um neue Versionen von Betriebssystemen von Smartphones oder Internetbrowsern zu unterstützen. Außerdem werden durchgehend Fehler korrigiert und weitere Verbesserungen durchgeführt.
Eine hohe Skalierbarkeit ist erforderlich, wenn die Anwendung über die produktive Entwicklung weitere Features erhält oder Integrationen definiert werden. Außerdem sind Portierungen zu anderen Betriebssystemen notwendig, um die Anwendung auf eine höhere Anzahl von Geräten verfügbar zu stellen.
Ein Ziel dieser Arbeit ist die Integration von Geodaten. Daher muss das SDK oder die Bibliothek eine Integration eigener Daten ermöglichen. Das Format und die Visualisierung werden in der Entwicklung diskutiert und implementiert, wenn das SDK oder die Bibliothek dies noch nicht unterstützen.
Da der Prototyp dieser Arbeit eine standortbasierte Anwendung sein soll, sind diesbezüglich Funktionen notwendig. Wenn das SDK oder die Bibliothek keine Funktionen dafür besitzen, müssten diese Funktionen erst implementiert werden. Die Implementierung solcher Funktionen erfordert einen großen Entwicklungsaufwand, dies ist nicht das Ziel der Arbeit.
Zur Entwicklung eines Prototypen in der Bearbeitungszeit dieser Arbeit sind minimale Hürden in der Entwicklung und für das Testen vorausgesetzt. Sind externe Abhängigkeiten und damit lange Wartezeiten für einen Test des Prototypen vorhanden, stört dies die Entwicklung.

2.3 Vergleich von SDKs und Bibliotheken

Die folgenden SDKs und Bibliotheken sind in der Literaturrecherche sowie nach Chen u. a. (2019), Yeeply Mobile, S.L. (2023), Acharya und Romesh (2023) und verschiedenen Repositories von GitHub, Inc. (2023) genannt.

- Apple ARKit
- Google ARCore SDK
- Vuforia
- Wikitude
- AR.js
- Kudan
- Lightship ARDK
- Unity Reflect
- MAXST
- EasyAR Sense
- Zapworks
- WebXR Geospatial
- Esri ARToolkit
- Mapbox Unity SDK, React AR, Scenekit

Die genannten SDKs und Bibliotheken sind noch nicht auf die oben genannten Anforderungen überprüft worden. Zuerst wird überprüft, ob die SDKs und Bibliotheken diese Anforderungen vollständig erfüllen. Falls keines die Anforderungen erfüllt, werden die Anforderungen angepasst, z. B. die Änderung von Open Source in kostenfreie Nutzung. Nach näherer Untersuchung sind nur die folgenden SDKs und Bibliotheken für die Entwicklung eines Prototypen nach den genannten Anforderungen möglich:

- Google ARCore SDK
- AR.js
- Esri ARToolKit (nutzt Apple ARKit oder Google ARCore SDK)

Die folgenden SDKs erfüllen nicht vollständig die Anforderungen. Dazu beschrieben ist eine Anforderung, die nicht erfüllt ist. Es ist möglich, dass ein SDK weitere Anforderungen nicht erfüllt, die nicht in der folgenden Liste geschrieben sind.

- Apple ARKit: Kein Open Source, jedoch kostenfreie Nutzung (Apple Inc. 2023)
- Vuforia: Kein Open Source, keine standortbasierten Funktionen, Schwerpunkte in Maschinenbau und Robotik (PTC, Inc. 2023)
- Wikitude: Properitäre SDK mit hohen Kosten, kein Open Source (Wikitude GmbH 2023)
- Kudan: Kein Open Source, standortbasierte Funktionen über Objekterkennung (Kudan Inc. 2023)
- Lightship ARDK: kein Open Source ((Niantic, Inc. 2023))
- Unity Reflect: Betrieb eingestellt, nur für Bestandskunden (Unity Technologies 2023a)
- MAXST: kein Open Source (MAXST Co., Ltd. 2023)
- EasyAR Sense: kein Open Source (VisionStar Information Technology (Shanghai) Co., Ltd. 2023)
- Zapworks: kein Open Source (Zappar Ltd. 2023)
- WebXR Geospatial: Keine aktuellen Updates, letztes Update 2019 (Repository von GitHub, Inc. (2023))
- Mapbox Unity SDK, React AR, Scenekit: keine aktuellen Updates, letztes Update 2020 (Repositories von GitHub, Inc. (2023))

Es ist zu erkennen, dass viele SDKs die Anforderungen nicht erfüllen, da diese nicht Open Source oder kostenfrei sind. Falls die SDKs oder Bibliotheken, die die Anforderungen erfüllen, dennoch für den Prototypen nicht geeignet sind, müssen die Anforderungen angepasst werden.

Ein Einsatz von Google ARCore SDK und AR.js wird bevorzugt, da diese die Anforderungen erfüllen. Für einen Vergleich werden die Anforderungen in Tabelle 1 gegenübergestellt. Ein "+" bedeutet, dass die Anforderung minimal erfüllt ist. "+++" hingegen bedeutet, dass die Anforderung in vollem Umfang erfüllt ist.

AR.js ist nach Carpignoli und AR.js Org Community (2023) eine Bibliothek zum Erstellen einer Webanwendung. Daher ist die Kompatiblität sehr hoch, da diese in jedem aktuellen Browser funktionieren sollte. Mit dem Google ARCore SDK hingegen wird nach Google (2023a) eine zu installierende Anwendung entwickelt, entweder für Android oder für iOS. Dabei wird Android primär unterstützt, da das

Anforderung	ARCore SDK	AR.js
Kompatibilität Smartphones und Tablets	+	+++
Einfaches Aufrufen oder einfache Installation	++	+++
Hohe Performance	+++	++
Open Source	+	+++
Aktuelle Updates	++	++
Hohe Skalierbarkeit	+	+++
Integration von eigenen Daten	+	+
Standortbasierte Funktionen	++	++
Minimale Hürden in der Entwicklung	+++	+++

Tabelle 1: Vergleich von Google ARCore SDK zu AR.js

Betriebssystem ebenfalls von Google ist. Es existiert eine Möglichkeit zum Entwickeln einer Webanwendung, jedoch mit weniger Features und nur für das Android Betriebssystem.
ARCore SDK muss für die Nutzung von allen Features installiert werden. Jedoch sind eigene Applikationen für Android, im Vergleich zu z. B. iOS, einfacher zu installieren. AR.js ist eine Webanwendung, diese muss lediglich aufgerufen werden.
Da ARCore SDK eine Applikation ist, kann die gesamte Hardware des Gerätes genutzt werden, ohne Abhängigkeit einer externen Anwendung. AR.js ist von dem genutzten Browser und dem Framework für AR im Browser (z. B. WebGL) abhängig, um eine AR Webanwendung zu starten. Es unterstützt WebGL und WebRTC und bietet damit eine relativ hohe Performance für Webanwendungen.
AR.js wird vollständig von einer Community geführt, entwickelt und nutzt die MIT-Lizenz. ARCore SDK ist zwar Open Source unter der Apache 2.0 Lizenz, jedoch sind einzelne Funktionen nicht Open Source, sondern kostenfreie oder kostenpflichtige Nutzungen von Application Programming Interface (API)s von Google. Außerdem ist die Lizenz von der Plattform abhängig. Wird z. B. die Unreal Engine oder Unity Engine genutzt, sind Lizenzmodelle dieser Plattformen zu beachten.
Für ARCore SDK und AR.js sind aktuelle Updates vorhanden, die zwar nicht täglich erfolgen, sondern etwa monatlich.
Da AR.js eine Webanwendung ist, besitzt diese eine hohe Skalierbarkeit. Webanwendungen können von verschiedenen Endgeräten genutzt werden, nicht abhängig vom Betriebssystem, sondern abhängig von dem verwendeten Browser. Außerdem können Webanwendungen mit geringem Aufwand in eine Plattform integriert werden. ARCore SDK ist skalierbar, da es möglich ist, auf verschiedenen Plattformen eine Anwendung zu entwickeln. Neue Android-Versionen besitzen Abwärtskompatibilitäten von Applikationen für ältere Androidversionen. Auch die Unterstützung von WebXR für ARCore SDK soll künftig weiterentwickelt werden, sodass eine Webanwendung mit mehr Funktionen zukünftig auch möglich wäre.
Die Integration eigener Geodaten ist weder bei ARCore SDK, noch bei AR.js bereits als Funktion eingebunden. Bei ARCore SDK ist der Import von Daten von der verwendeten Platform abhängig. Bei AR.js ist dies unter anderem mit JSON möglich. Die Aufbereitung und Darstellung der Daten muss als eigene Funktionen implementiert werden.
ARCore SDK und AR.js besitzen die Möglichkeit, die eigene Position mit GNSS bestimmen. ARCore SDK besitzt die Geospatial API, die es ermöglicht, georeferenzierte Anker einzufügen. AR.js hat mehrere Beispiele mit standortbasierten Funktionen, mit denen Objekte an einem Koordinatenpaar dargestellt werden können.
AR.js kann über einen Live Server oder serverlose Plattform betrieben werden. ARCore SDK kann je nach ausgewählter Plattform eine Installationsdatei erzeugen oder über USB oder WiFi die Anwendung direkt auf dem Smartphone testen.

Die Bibliothek AR.js erfüllt die Anforderungen in einem größeren Umfang als ARCore SDK, vor allem aufgrund der Entwicklung als Webanwendung. Daher wird zuerst AR.js zur Entwicklung eines Prototypen verwendet. Im Kapitel 3.1.6 wird schlussgefolgert, dass AR.js sich nicht als Bibliothek für einen Prototypen eignet. Die Gründe sind in den nächsten Kapiteln bis dahin beschrieben, analysiert und beurteilt.

2.4 AR.js

2.4.1 Eingangsdaten

Informationen über das Liegenschaftskataster werden in Niedersachsen im Landesamt für Geoinformation und Landesvermessung Niedersachsen von den Dezernaten 3 verwaltet. Über diverse Tools können verschiedene Informationen extrahiert werden. Da Punktkoordinaten verschiedene Qualitätsangaben besitzen, muss beachtet werden, dass nur Punkte mit hohen Qualitätsangaben in der Anwendung betrachtet werden. Im Kataster existieren Punkte mit einer Abweichung von mehreren Metern, da diese ein hohes Alter aufweisen und teilweise aus Urkarten kartografisch abgegriffen worden sind. Die Daten können in den Formaten dxf, shp oder gpkg ausgegeben werden. GeoPackage ist ein neues Format, das als Nachfolger von Shape angesehen wird und viele Nachteile von Shape beseitigt. Die Daten können dann in der Software QGIS zu geoJSON konvertiert werden. Jedoch sind die Geodaten als MultiPolygone strukturiert. Es werden einzelne Linien benötigt, um diese ohne großen Entwicklungsaufwand in AR.js darzustellen. Dazu werden die MultiPolygone in MultiLineStrings und dann in LineStrings umgewandelt. Die LineStrings im geoJSON-Format sind die Daten, die der Prototyp verwenden wird. Außerdem müssen die Daten in ein anderes Koordinatensystem transformiert werden. Die exportierten Daten sind im projizierten Koordinatensystem Universal Transverse Mercator (UTM) in der Zone 32N des Ellipsoiden ETRS89, bzw. EPSG-Code 25832. AR.js nutzt das projizierte Koordinatensystem Pseudo-Mercator mit dem EPSG-Code 3857. Diese Transformation ist ebenfalls mit der Software QGIS möglich.

2.4.2 Testumgebung

Das Aufsetzen eines lokalen Servers mit dem Bestriebssystem Android verläuft ohne Probleme, jedoch ist dies mit iOS nicht möglich. Daher ist für den Prototypen eine serverlose Plattform für eine optimale Testumgebung notwendig. Dabei wird mit Docker die Webanwendung gebaut, einem einzigartigen Schlüssel vergeben und dann in eine Registry über ein “push” hochgeladen. Dort übernimmt die serverlose Plattform die Erstellung von Instanzen, die für den Webserver bereitgestellt werden. Da keine aufwändigen serverseitigen Rechenprozesse durchgeführt werden, sondern nur vorwiegend HTML- und JavaScript-Dateien bereitgestellt werden müssen, ist die geringste Konfiguration mit 0,125 virtuellen Prozessoreinheiten und 0,25 GB Arbeitsspeicher ausreichend.

2.4.3 Datenformate

Datenformate unterscheiden sich je nach eingesetzter Bibliothek. AR.js basiert u. a. auf A-Frame und kann daher folgende Datentypen importieren und darstellen:

- glTF (3D-Modell)
- OBJ (3D-Modell)
- JSON

Aufgrund des Component-Ecosystems von A-Frame können viele Komponenten aus der Open Source Community für den Import diverser Formate integriert werden. Die meisten Komponenten werden über

npm[1] veröffentlicht. Für den Prototyp dieser Arbeit wird eine Integration von Geodaten mit JSON durchgeführt, da JSON von AR.js nativ unterstützt wird.

2.5 ARCore SDK

2.5.1 Aktueller Entwicklungsstand

Der aktuelle Stand der Entwicklung von ARCore ist auf der Internetseite von Google einsehbar. ARCore ist ein SDK, mit welchem AR-Anwendungen erstellt werden können. Es bietet Multi-Platform-APIs an, um auf verschiedenen Endgeräten die Anwendungen zu ermöglichen. Außerdem sind verschiedene Entwicklungsplattformen möglich, ob direkt für Android oder iOS oder mit den Game Engines Unity oder Unreal. Auch eine Web-Anwendung ist möglich, jedoch mit wenig unterstützten Funktionen und APIs.

Folgende Punkte nach Google (2023a) sind die Kernfunktionen von ARCore ohne zusätzliche APIs wie z. B. Geospatial API für Anwendungen auf Grundlage der Standortbestimmung.

Motion Tracking: ARCore nutzt simultane Lokalisierung und Kartierung (SLAM = simultaneous localization and mapping), um eine AR-Umgebung aufzubauen und mit der Realität zu verbinden. Es werden einzigartige Merkmalspunkte im Sichtfeld der Kamera erfasst, die zusammen mit den Sensoren im Gerät verwendet werden, um eine relative Position und Ausrichtung der Kamera zu berechnen. Die Position der Kamera ist die Ausgangsposition für die AR-Umgebung, um virtuelle Inhalte einzufügen. Die virtuellen Inhalte besitzen eine relative Position zur Kameraposition und -ausrichtung.

Environmental Understanding: ARCore kann Flächen auf Basis von Merkmalspunkten erkennen, um daraus Ebenen zu berechnen und auszuwerten. Auf diesen Ebenen können z. B. die Anker von Objekten platziert werden, damit die Objekte besser mit der Realität interagieren.

Depth Understanding: ARCore kann aus RGB-Kameras Tiefenkarten erstellen. Dadurch können Entfernungen zwischen Objekten oder Flächen berechnet werden, um die Immersivität zu erhöhen.

Light Estimation: Aus den Kamerabildern können Informationen zur Beleuchtung bestimmt werden und die berechneten Intensitäten für das Einfügen von virtuellen Objekten nutzen, um die Immersivität zu steigern.

User Interaction: Über Hittests ist eine Interaktion mit der AR-Umgebung möglich, da ein Strahl vom Eingangspunkt des Touchscreens von Endgeräten in die AR-Umgebung erzeugt wird. Alle Punkte, ob Merkmalspunkte, Punkte einer erkannten Fläche oder virtuelle Inhalte auf diesem Strahl, können ausgewählt werden für weitere Interaktionen.

Oriented Points: Mithilfe der Flächenerkennung und Hittests ist es möglich, Ausrichtung und Position für virtuelle Objekte anhand der Schräge von Flächen zu bestimmen.

Anchors and Trackables: Virtuelle Objekte, die in AR angezeigt werden sollen, müssen einen Anker besitzen, der eine Beziehung zwischen dem Objekt und der AR-Session erschafft. Im Laufe der Zeit verändern sich die relativen Positionen von Objekten, da das Motion-Tracking stetig aktualisiert wird. Die Objekte werden Trackables genannt, da die relativen Objekte vom Motion Tracking abhängig sind und daher "beobachtet" werden müssen.

[1] `https://www.npmjs.com/search?q=aframe-component`, Zugegriffen im Mai 2023

Augmented Images: Um mit der Umgebung besser zu interagieren, werden in AR-Anwendungen oft Referenzgrafiken benutzt, die durch das Gerät erkannt werden, um Objekte an dieser Stelle zu platzieren. Mit ARCore ist es möglich, 2D-Bilder zu interpretieren. Z. B. Poster oder Produktverpackungen können erkannt werden, um virtuelle Inhalte an dieser Stelle oder relativ zu der Position einzufügen.

2.5.2 Geospatial API

Die Geospatial API von Google ermöglicht mit ARCore SDK die Entwicklung einer standortbestimmten AR-Anwendung. Die Kernfunktion der Geospatial API ist die Korrektur der GNSS-Standortbestimmung mit dem Visual Positioning System (VPS) von Google. Der ermittelte Standort sowie die Tiefenkarte, die durch ARCore erzeugt wird, werden mit dem VPS von Google verglichen, um die Genauigkeit der Standortbestimmung um ein Vielfaches zu erhöhen. Dies bedeutet, dass in Bereichen mit Google Streetview, das zur Erstellung des VPS verwendet wird, eine höhere Genauigkeit möglich ist und in urbanen Gebieten aufgrund der Gebäudefassaden eine noch höhere Genauigkeit.

Google besitzt mit Streetview weltweit eine hohe Dichte an Bildern auf Straßen und Wege, die als Eingangsdaten für die Berechnung einer Punktwolke dienen. Die von ARCore erstellte Tiefenkarte, die Flächen und Objektformen erkennen kann, wird mit der Punktwolke verglichen, um die Position und Ausrichtung des Geräts hochgenau zu berechnen.

Eine weitere wichtige Funktion der Geospatial API sind raumbezogene AR-Anker. Ein raumbezogener Anker kann einem Objekt zugewiesen werden, um dem Objekt eine geografische Koordinate des WGS84-Ellipsoids zuzuweisen. Es wird zwischen drei verschiedenen Arten von raumbezogenen Ankern unterschieden. WGS84-Anker werden mit einer festen 3D-Koordinate (Breiten- und Längengrad sowie Höhe über Ellipsoid) versehen und in der AR-Anwendung an dieser Stelle platziert. Terrain-Anker nutzen die Erkennung der Bodenfläche, um mithilfe 2D-Koordinaten das Objekt auf den Boden zu platzieren. Rooftop-Anker werden verwendet, um Objekte auf Gebäuden zu platzieren. Dabei wird die Höhe des Gebäudes über die zuvor erwähnte berechnete Punktwolke berechnet.

Die Entwicklung mit der Unterstützung von der Geospatial API ist mit verschiedenen Platformen möglich. Nativ mit Android-NDK mit der Programmiersprache C oder ebenfalls als Android-Applikation mit den Sprachen Kotlin oder Java. Auch die Entwicklung einer iOS-Applikation, z. B. mit Xcode, ist möglich. Außerdem werden auch die Game-Engines Unity und Unreal unterstützt, in denen sowohl Android- als auch iOS-Applikationen möglich sind. Jedoch wird die Unreal Engine nicht aktiv unterstützt und es existiert nur eine kurze Dokumentation zur ARCore SDK, außerdem keine Dokumentation zur Nutzung der Geospatial API in der Unreal Engine.

2.5.3 Geospatial Creator

Die neue Plattform “Geospatial Creator” bringt viele neue Funktionen für die Erstellung einer standortbestimmten AR-Anwendung. Die Kernfunktion ist die Integration des geografischen Koordinatensystems sowie von fotorealistischen 3D-Tiles innerhalb der Entwicklungsplatform, um Anker mit einfacher Interaktion zu platzieren. Auch das Platzieren von Ankern ohne die Kernfunktion ist mit Geospatial Creator durch weitere Funktionen intuitiver. Für die fotorealistischen 3D-Tiles ist eine kostenpflichtige API notwendig, da hier stetig ein Datenaustausch mit hohem Traffic erfolgt. Geospatial Creator ist mit der Unity Engine oder Adobe Aero möglich.

2.5.4 Unity Engine

Aufgrund der hohen Leistung einer Game Engine sowie der aktuellen Unterstützung wird die Unity Engine als Entwicklungsplattform ausgewählt, um mit der Geospatial API eine standortbasierte AR-Applikation zu entwickeln. Auch Funktionen von Geospatial Creator können verwendet werden, ohne die kostenpflichtige API für fotorealistische 3D-Tiles zu nutzen.

Nach Unity Technologies (2023b), Megha, Nachammai und Ganesan (2018) und Hussain u. a. (2020) wurde die Unity Game Engine im Jahr 2005 vorgestellt, vorerst als exklusive Game Engine für Mac OS X. Später folgte eine Ausbreitung auf viele verschiedene Plattformen. Die Game Engine wird seitdem vermehrt außerhalb der Gaming Branche genutzt, um z. B. Simulationen durchzuführen oder für Produktvorführungen, unter anderem in der Automobilbranche. Auch zur Darstellung und Interaktion mit raumbezogenen 3D-Modellen existieren bereits Ansätze, z. B. untersuchten Buyuksalih u. a. (2017) eine 3D-Modellierung für ein 3D-Solarkataster sowie für unterirdische Leitungen. Ein großer Vorteil ist die Unterstützung für mehrere Platformen. Neben Desktop-Anwendungen legt Unity einen großen Wert auf Applikationen für mobile Endgeräte, VR und AR. Andere Game Engines, z. B. die ebenfalls bekannte Unreal Engine, haben primär den Fokus auf Desktop-Anwendungen. Skripte werden mit den Programmier- bzw. Skriptsprachen C# und JavaScript erstellt und bearbeitet. Diese Sprachen besitzen einen wesentlich einfacheren Einstieg als z. B. C++. Ein Nachteil ist die Performance. Die Unity Engine hat eine etwas schlechtere Performance als andere Game Engines wie z. B. die Unreal Engine. Außerdem sind die Kosten nach einer Veröffentlichung vergleichsweise hoch. Dazu sind im nächsten Kapitel mehr Informationen.

2.5.5 Aktuelle Lizenzbedingungen der Unity Engine

Derzeit existieren nach Unity Technologies (2023b) mehrere “Unity Pläne”, um Unity zu nutzen. Der “Personal”-Plan ist kostenlos und für private Zwecke oder Schulungszwecke wie diese wissenschaftliche Arbeit sowie für Unternehmen unter 100.000 € Umsatz. Der “Pro”-Plan für derzeit 1.877 € pro Jahr ist für die kommerzielle Nutzung zur Entwicklung von Applikationen in Unternehmen mit über 100.000 € Umsatz. Außerdem existieren noch der “Enterprise”- und “Branche”-Plan, die für einen individuellen Preis mehr Features und Support beinhalten.

Ab 2024 werden die Lizenzmodelle aktualisiert. Eine große Neuerung ist eine Gebühr, die je nach Nutzung und Umsatz der Applikation berechnet wird. Ist der Umsatz der Applikation unter einer Million US-Dollar pro Jahr oder die Gesamtzahl an Erstinstallationen bzw. -downloads eine Million erreicht, besteht keine extra Gebühr. Wenn die Applikation mehr als eine Million US-Dollar pro Jahr und mehr als eine Million Erstinstallationen erreicht, dann wird eine Gebühr fällig, die jedoch maximal 2,5 Prozent dem monatlichen Umsatz der Applikation entspricht.

Alle Lizenzmodelle von Unity sind frei für die kommerzielle Verwendung, Vertrieb und Verkauf der Applikation bzw. des Spiels und lediglich an die o.g. Konditionen bzw. Gebührenmodelle gebunden. Erstellte Skripte für die Unity Engine oder sonstige Dateien sind nicht an die Lizenz gebunden und können auch als Open-Source-Lizenz veröffentlicht werden.

2.5.6 Lizenzmodelle der eingesetzten SDK und API

Die ARCore SDK ist unter der Apache License 2.0 veröffentlicht. Es ist die bekannteste Open Source Lizenz und bedeutet, dass sie frei verfügbar ist und durch eine offene Community unterstützt und weiterentwickelt wird. Lediglich zum Schutz von Patentrechten ist die Lizenz in diesem Bereich stärker

eingeschränkt als z. B. MIT-Lizenz.

Für den Prototyp wird außerdem die Geospatial API von Google genutzt. Diese API wird von Google verwaltet und ist limitiert kostenfrei. Bis 1.000 Startsessions pro Minute sowie 100.000 API-Anfragen pro Minute ist die API kostenfrei zu nutzen. Wenn diese Grenze überschritten wird, ist die Antwort auf eine API-Anfrage ein Error. Für eine zukünftige Verwendung ist ein Einsatz der Geospatial API mit einer kostenpflichtigen Erhöhung des Kontingents möglich, falls dieses überschritten werden sollte. Eine weitere Möglichkeit ist die Entwicklung einer eigenen API, um nicht nur kostenunabhängig von Google zu sein, sondern um allgemein eine Unabhängigkeit zu Google-Services zu gewährleisten. Lediglich die Nutzung des Open Source SDK von Google stellt eine Abhängigkeit von Google dar. Hier vertritt Google jedoch nur die Rolle des Maintainers für das Open Source Projekt.

2.6 Tests zur Standortbestimmung

Es existieren verschiedene Lösungen und Genauigkeiten nach Zangenehnejad und Gao (2021) und Heßelbarth und Wanninger (2020) für eine Satellitenpositionierung in mobilen Endgeräten. Es wird unterschieden zwischen Single Point Positioning (SPP), Precise Point Positioning (PPP), Precise Point Positioning with Ambiguity Resolution (PPP-AR) und Real-Time Kinematic (RTK).

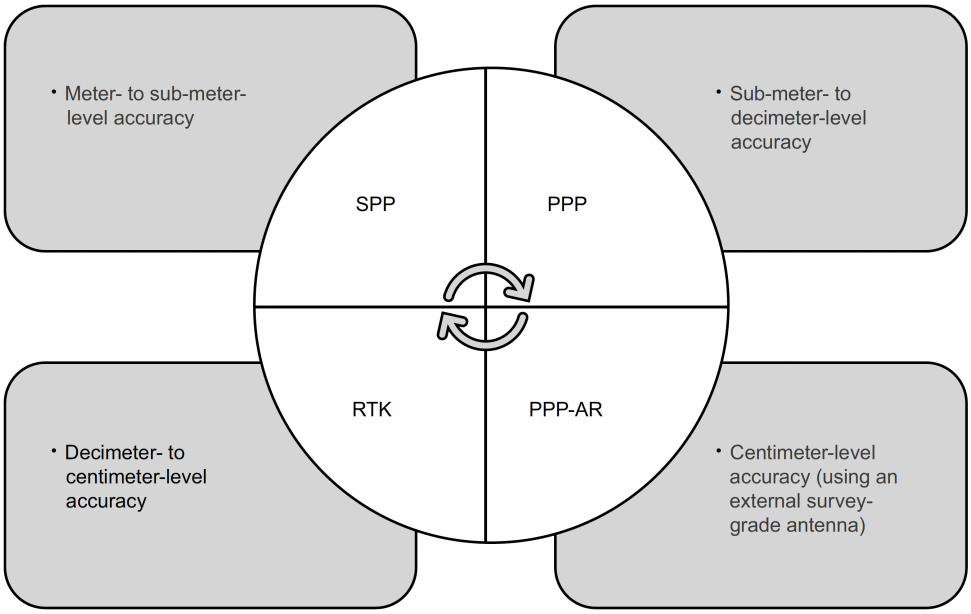

Abbildung 2: Messgenauigkeiten der GNSS-Positionierung von mobilen Endgeräten (Zangenehnejad und Gao 2021)

Dies bedeutet, dass je nach verwendetem Gerät große Unterschiede in der Genauigkeit einer GNSS-Positionierung liegen. Auch bei gleichem Verfahren kann es zu Unterschieden kommen, da innerhalb der Verfahren verschiedene GNSS-Empfänger existieren. Im Normalfall besitzen preiswerte Empfänger geringere Genauigkeiten als kostenintensive Empfänger.

Gängige Smartphones aus dem Consumer-Bereich nutzen PPP, da diese besonders preiswert sind, aber dennoch eine genügend hohe Genauigkeit für eine Positionierung für Navigationsapplikationen besitzen. Heutzutage ist Dual-Frequency GNSS in Smartphones der Standard, da hiermit PPP erreicht werden kann. Die Chipsets für die GNSS-Positionierung sind in Smartphones meist sehr preiswert und besit-

zen daher keine hohe Genauigkeit. Mit PPP wären theoretisch Genauigkeiten von unter einem Meter möglich, jedoch ist die Streuung aufgrund der sehr preiswerten Chipsets sehr hoch. Außerdem sind Korrekturdaten der Hardware nicht für jedes Gerät erhältlich, da die Geräte selbst keine Kalibrierung für die GNSS-Sensoren durchführen, sondern nur einheitlich für alle Geräte des gleichen Typs. Daher ist die Genauigkeit der GNSS-Messung mit Smartphones aus dem Consumer-Bereich über einem Meter bis zu mehreren Metern.

RTK ist nur möglich, wenn eine weitere GNSS-Messung simultan auf einem bekannten Punkt durchgeführt wird oder eine Referenzstation existiert, die zur selben Zeit an einem Ort in der Nähe gemessen wird oder eine Messung in kurzer Vergangenheit vorliegt und die Satellitenbahnen vorausberechnet werden. Dies wäre mit Smartphones jedoch möglich und in Deutschland eventuell mit dem Satellitenpositionierungsdienst der Deutschen Landesvermessung (SAPOS) ebenfalls zu realisieren.
SAPOS wird durch die Landesämter der Bundesländer betrieben, unter anderem in Niedersachsen ist die Nutzung gebührenfrei. Für diese Masterarbeit wird RTK-Messung jedoch nicht durchgeführt, da die Erhöhung der Genauigkeit nicht Thema der Arbeit ist. Im Ausblick wird dies jedoch thematisiert, da RTK eine erhebliche Verbesserung einer GNSS-Positionierung bietet.

Post-Processing führt dazu, die Genauigkeit nach einer Messung erheblich zu verbessern. Es werden z. B. die echten Satellitenbahnen verwendet, um die Abweichung nachträglich zu minimieren. Da AR-Anwendungen jedoch in Echtzeit messen und in Echtzeit die Daten verarbeitet werden müssen, ist eine Messung mit Post-Processing nicht möglich.

Außerdem sind durch komplexere Algorithmen zur Analyse der GNSS-Signale höhere Genauigkeiten möglich. In dieser Arbeit werden die geräteeigenen Algorithmen und Algorithmen der Bibliotheken verwendet, um eine Positionierung durchzuführen.

Nach Sadeghi-Niaraki und Choi (2020) sind vor allem bildgestützte Methoden als Korrekturdaten zur Verbesserung der Position hilfreich, um genaue Standortbestimmungen zu erhalten. Zum Nachteil werden erhöhte Rechenprozesse genannt, die vor allem bei Smartphones zu Performance-Problemen führen könnten. Auch eine Kalibirierung des Gerätes erhöht die Standortbestimmung sehr, jedoch kann dies bei Nutzenden nicht vorausgesetzt werden.

Für viele Smartphone Anwendungen ist eine Genauigkeit von wenigen Metern ausreichend, da z. B. für Navigation keine Dezimetergenauigkeit erforderlich ist. Für AR-Anwendungen wäre jedoch eine höhere Genauigkeit zum Vorteil, um die virtuellen Inhalte hochgenau zu platzieren. Chipsets mit höherer Genauigkeit sind jedoch sehr kostenintensiv und daher zu ineffizient, diese in Consumer-Smartphones einzubauen.

Paziewski (2020) nennt und bestätigt folgende Einschränkungen für hochgenaue GNSS-Messungen mit Smartphones: Zum einen liegt dies der geringen Qualität der Antennen zugrunde, die zu hoher Anfälligkeit von Multipath-Effekten und Mehrdeutigkeit von Phasenbeobachtungen führen. Zum anderen existieren keine Fehlermodelle für die Kalibrierung einzelner Antennen, sondern nur für alle Geräte des gleichen Gerätemodells.

2.6.1 Testumgebung

Die Anwendung mit ARCore SDK wird in zwei unterschiedlichen Methoden getestet. Für einen Test in direkter Umgebung des Entwicklungsrechners kann über ein USB-Kabel oder WiFi die Anwendung direkt auf das Smartphone übertragen und getestet werden. Dies ist über die Debugging Funktion von Unity möglich. Soll die Anwendung erst in der Örtlichkeit, z. B. an einer Straße, getestet werden, ist das Exportieren als APK-Datei möglich. Die APK-Datei kann auf das Smartphone übertragen und manuell installiert werden. Damit ist es möglich, die Anwendung wiederholt zu starten und zu beenden, ohne Anbindung an den Entwicklungsrechner. Eine Auswertung der Konsole ist mit dem Tool logcat[2] möglich. Entweder über das CLI-Tool oder integriert in der Software Android Studio. Mit diesem Tool können alle Logs des Smartphones beobachtet und gefiltert werden. Für Fehleranalysen kann im Unity-Skript der Befehl `console.log` verwendet werden, um Variablen auszugeben oder nachzuweisen, ob der Code bis zu diesem Befehl ausgeführt wurde.

[2] `https://developer.android.com/tools/logcat`

3 Umsetzung

3.1 Prototyp mit AR.js

Da im Kapitel 3.1.6 entschieden wird, die Entwicklung des Prototypen zu AR.js einzustellen, fällt die Beschreibung der Umsetzung kürzer aus.

AR.js bietet einige Beispiele, aus denen eigene Entwicklungen gestaltet werden können. Das Beispiel “osm-ways” in der Kategorie “new location based” ist für die Anwendung dieser Arbeit hilfreich. Es bietet eine Anzeige von Linien über einer JavaScript Object Notation (JSON) API an. Der erste Test ohne Änderungen am Code liefert ein zufriedenstellendes Ergebnis. Die Linien für Wege wurden angezeigt, jedoch mit Ungenauigkeiten in der Orientierung und der Visualisierung virtueller Inhalte. Nähere Ergebnisse und Analysen werden mit dem ersten Prototyp mit Daten aus dem Liegenschaftskataster erstellt.

3.1.1 Integration von Liegenschaftsinformationen

Das zuvor genannte Beispiel “osm-ways” ist das Ausgangsbeispiel, um darauf die Applikation mit Liegenschaftsinformationen zu entwickeln. Der Code wird so verändert, sodass keine API angesprochen wird, sondern für den Test eine lokale JSON-Datei, die nach Kapitel 2.4.1 aufbereitet wird. Außerdem wird die Erzeugung von Linien sowie deren Darstellung verändert, da die Liegenschaftsdaten eine andere Struktur und andere Attribute als Daten aus Open Street Map besitzen. Mit diesen Implementationen ist eine erste Darstellung von Liegenschaftsinformationen in AR.js möglich. In Abbildung 3 sind zwei Screenshots aus dem Prototypen mit dem Google Pixel 6 dargestellt. Die Flurstücksgrenzen sind in Weiß dargestellt, der Hintergrund ist die Kameraaufnahme.

Abbildung 3: Screenshots aus dem AR.js Prototyp mit Google Pixel 6

(Flurstücksgrenzen sind in Weiß dargestellt, im rechten Bild sind diese in der oberen Hälfte des Bildes.)

3.1.2 Erste Eindrücke des Prototypen

In Abbildung 3 sind Screenshots mit dem Google Pixel 6 zu erkennen. Im ersten Bild kann der Verlauf der Flurstücksgrenzen mit der Realität verglichen werden, jedoch mit hohen Abweichungen in der Lage, Winkel und Neigung der virtuellen Inhalte. Im zweiten Bild sind die virtuellen Inhalte mit sehr großen

Abweichungen platziert, sodass keine Übereinstimmung mit der Realität möglich ist. Diese Erfahrungen führen zu einem mangelhaften Ergebnis. Es ist zu diesem Stand des Prototypen nicht möglich, eine ausreichende Einschätzung von Liegenschaftsinformationen als virtuelle Inhalte in der Realität zu erhalten. Daher muss die Abweichung näher untersucht werden und eine Aussage treffen, ob die Abweichung korrigiert oder die Bibliothek geändert werden muss. Primär wird die Abweichung der Standortbestimmung mit GNSS untersucht, um die hohe Abweichung zu belegen.

3.1.3 Test der GNSS-Positionierung

Ein wichtiger Faktor ist die Genauigkeit der GNSS-Positionierung mobiler Endgeräte. Dies kann der Grund für die hohen Abweichungen in der Visualisierung sein. Dieser Faktor kann überprüft werden, wenn über einen bekannten Punkt das Endgerät gelegt wird.

GNSS-Applikationen bieten verschiedene Informationen über Ortungsdienste und Empfangsignale von Satelliten. Die Applikation GnssLogger für Android-Betriebssysteme bietet sehr ausführliche Informationen, z. B. Anzahl und Eigenschaften der sichtbaren Satelliten. Jedoch ist diese Applikation nicht für iOS vorhanden und es existieren keine ähnlichen Applikationen. Es existieren nur Applikationen zur Verwendung von GNSS-Empfängern zur Vermessung oder GNSS-Tracking für Sport, die nicht regelmäßig Koordinaten ausgeben. Die Applikationen für iOS sind sehr begrenzt an Informationsinhalt, meistens ist die wesentliche Funktion nur die Koordinatenangabe. Daher ist kein Vergleich mit externen Applikationen möglich, um eine Einschätzung für die Abweichung der GNSS-Standortbestimmung zu erhalten. Für die Fehleranalyse für die Visualisierung ist ein Test mit der eigenen Applikation ausreichend.

In dieser Arbeit werden zwei Geräte verwendet: Das Apple iPhone 12 Pro, das nach Apple Inc. (2020) 2020 vorgestellt wurde, und nach Osterloh (2021) das Google Pixel 6, das 2021 vorgestellt wurde. Beide Geräte nutzen nach Apple Inc. (2022) und Google (2023c) Dual-Band-GNSS mit den Satellitensystemen GPS, GLONASS, Galileo, QZSS und BeiDou (USA-Modelle des Google Pixel 6 besitzen keine BeiDou-Unterstützung).

Zuerst wurden Tests auf Grenzpunkten durchgeführt, die eine hohe Kartentreue besitzen. Dies bedeutet, dass die Koordinatenabweichung zwischen Kataster und Abmarkung minimal ist. Jedoch sind nur wenige Grenzpunkte für einen Test vorteilhaft, da viele Grenzpunkte entweder unterirdisch, durch Vegetation verdeckt oder das Flurstück nicht betretbar sind. Die Tests mit einem Grenzpunkt sind nicht aussagekräftig und wurden daher verworfen.

Vermessungspunkte wie Aufnahmepunkte und Sicherungspunkte sind für eine Genauigkeitsmessung zum Vorteil. Denn diese sind frei gelegen und in urbanen Gebieten meist auf der Straße, auf Bordstein oder Wege vermarkt. Fünf verschiedene Aufnahmepunkte bzw. Sicherungspunkte von Aufnahmepunkten wurden mit den vorhandenen Geräten gemessen, jeweils eine Messung von min. einer Minute je Gerät je Punkt. Die verwendeten Browser-Versionen für die folgenden Tests sind für das iPhone der Browser Safari mit iOS Version 16.6 sowie für das Pixel der Browser Chrome mit der Version 115.0.5790.138.

Die Applikation wurde wenige Meter entfernt geöffnet bzw. die Webanwendung aktualisiert, damit diese sich neu initialisiert. Sobald die Applikation bereit war und bereits die Koordinate gemessen hatte, wurde das Gerät mittig auf den Punkt gelegt.
Zum Schutz des Gerätes wurde eine Klarsichthülle verwendet, die zwischen dem Punkt und dem Gerät lag. Um Koordinatenpaare zu erhalten, wird eine Funktion implementiert, die es ermöglicht, das ak-

Abbildung 4: Messung der Genauigkeit mit AR.js Prototyp

tuelle Koordinatenpaar anzuzeigen sowie die Koordinatenpaare sekundenweise zu speichern und in die Zwischenablage zu kopieren. Die Darstellung dafür ist vereinfacht konfiguriert und in Abbildung 5 als Screenshot dargestellt. Kurz nach dem Hinlegen wird auf den Button 'Start' gedrückt, damit die Koordinaten in eine temporäre Datei für jede Sekunde gespeichert werden. Nach Ablauf von mindestens 1 Minute bis ca. 1:30 Minuten wird auf 'Stop' gedrückt, damit keine weiteren Daten in die temporäre Datei gespeichert werden. Daraufhin wird auf den Button 'Copy' gedrückt, damit die Messdaten in die Zwischenablage kopiert werden. Das Handy wird aufgehoben und die Zwischenablage in einer Datei gesichert.

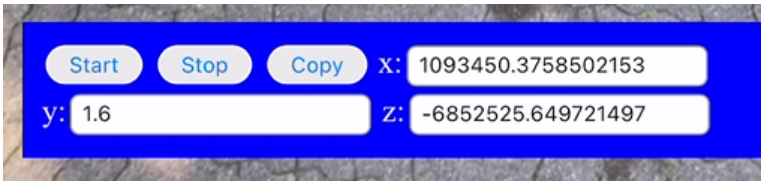

Abbildung 5: Implementation der Möglichkeit der Koordinatenanzeige sowie Aufnehmen von Koordinatenpaaren

Am Computer werden die gesammelten Daten in getrennte JSON-Dateien abgelegt, je Punkt und je Gerät, damit diese analysiert werden können. Die fünf Messreihen mit entsprechenden Wetterbedingungen sind in Tabelle 2 dargestellt.

Messreihe	Datum und Zeit	Wetterbedingungen
1	21.08.23, 15:40-16:05	Sehr sonnig, minimal bewölkt
2	21.08.23, 17:55-18:16	Sonnig, leicht bewölkt
3	22.08.23, 08:19-08:40	Sonnig, leicht bewölkt
4	22.08.23, 11:10-11:32	Bewölkt
5	22.08.23, 14:05-14:25	Sonnig, leicht bewölkt

Tabelle 2: Bedingungen für die ausgewerteten Messreihen mit AR.js

Es existieren Unterschiede zwischen den durchgeführten Messungen und der Nutzung der Anwendung. Beim Messen liegt das Smartphone ruhig auf dem Boden statt in den Händen bei ca. 1,3 bis 1,6 Meter Höhe. Die Kamera ist durch das Auflegen auf den Boden verdeckt, dabei ist diese die wichtigste Hardware

für eine AR-Anwendung. Außerdem ist bei der Nutzung der Applikation das Smartphone ständig in Bewegung und nicht still auf dem Boden. Der Ort des Chip für das Empfangen der GNSS-Daten innerhalb des Smartphones ist nicht bekannt. Dies kann mehrere Zentimeter von der Mitte des Gerätes abweichen.

Zur Auswertung der Messdaten wird die Software MatLab R2023a verwendet. Dazu werden verschiedene Skripte geschrieben, um die Daten zu verarbeiten und visuell zu betrachten. Zuerst ein Skript, das die geschriebenen JSON-Daten aus der Webanwendung in eine Matrix mit 2D-Koordinaten umwandelt. Ein weiteres Skript zeigt die Koordinaten als Punkte in einer Grafik dar, inklusive Verlauf der Messung und Sollkoordinaten. Außerdem wird die Entfernung des letzten Punktes, welcher die kleinste Entfernung der Messreihe zum Referenzpunkt darstellen sollte, berechnet. Nach Durchführung der Skripte mit allen Daten entstehen Abbildungen je Punkt und je Gerät. Außerdem sind die berechneten Entfernungen in einer Textdatei geschrieben.

3.1.4 Beurteilung der Ergebnisse

In Abbildung 6 und weitere Abbildungen im Anhang A.1 sind die Abbildungen aus dem MatLab-Skript zu sehen, geordnet nach den Referenzpunkten. In der Mitte als blauer Punkt ist der Referenzpunkt zu sehen, mit einer Ausdehnung von 10 m in alle Richtungen. Die Messungen sind als grüne Punkte dargestellt, verbunden durch Linien, um dieselbe Messreihe darzustellen. Um den Anfang mit dem Ende zu unterscheiden, sind die Startpunkte der Messung gekennzeichnet. Was als Erstes auffällt, wenn die iPhone Messungen mit denen des Pixels verglichen werden, sind fehlende Linien in nahezu allen iPhone-Messungen. Ein Blick in die Messwerte führt zu der Erkenntnis, dass die meisten Messreihen über die gesamte Zeit die gleichen Koordinaten angeben. Dies bedeutet, dass das iPhone entweder eine Minimum-Distanz besitzt, bis sich die Koordinate ändert, oder dass das iPhone primär mit Beschleunigungssensoren rechnet. Da das Smartphone still auf dem Boden lag, sind keine Bewegungen während der Messung durchgeführt worden. Damit würden Beschleunigungssensoren oder Neigungssensoren keine veränderten Daten aufzeigen.

Der Durchschnitt der Entfernungen vom letzten Messpunkt zum Referenzpunkt für alle Punkte je Gerät sind für das iPhone 4,10 m und für das Pixel 2,95 m. Die geringste durchschnittliche Entfernung vom letzten Messpunkt zum Referenzpunkt je Punkt je Gerät sind für das iPhone 2,83 m und für das Pixel 1,82 m. Das Google Pixel 6 besitzt anhand dieser Messwerte eine höhere Genauigkeit in der GNSS-Messung. Dies ist wahrscheinlich darauf zurückzuführen, dass das iPhone zu nahezu allen Messungen die erste Koordinate behält und diese nicht mehr ändert. Das Google Pixel 6 minimiert im Verlauf der Messung die Entfernung zum Referenzpunkt. Um dies besser zu verstehen, können die Entfernungen der ersten Messung betrachtet werden, um einen Mittelwert für die Messungen zu generieren. Dafür wird das MatLab-Skript umgeschrieben, indem der end-Befehl innerhalb der Matritzen in einen start-Befehl getauscht wird. Dabei ensteht für alle Messungen der Mittelwert der Entfernungen vom ersten Punkt der Messreihen zum Referenzpunkt für das iPhone mit 4,19 m, aber für das Pixel mit 5,14 m.

Für eine generelle Einschätzung sind diese Messwerte ausreichend, um aufzuzeigen, dass die Genauigkeit einer GNSS-Positionierung, ohne Post-Processing oder Echtzeit-Korrekturdaten, im einstelligen Meter-Bereich liegt. Dies erklärt die ungenau positionierten Linien im Prototyp der Webanwendung, da die Bibliothek AR.js keine Korrekturen zur Genauigkeitssteigerung durchführt.

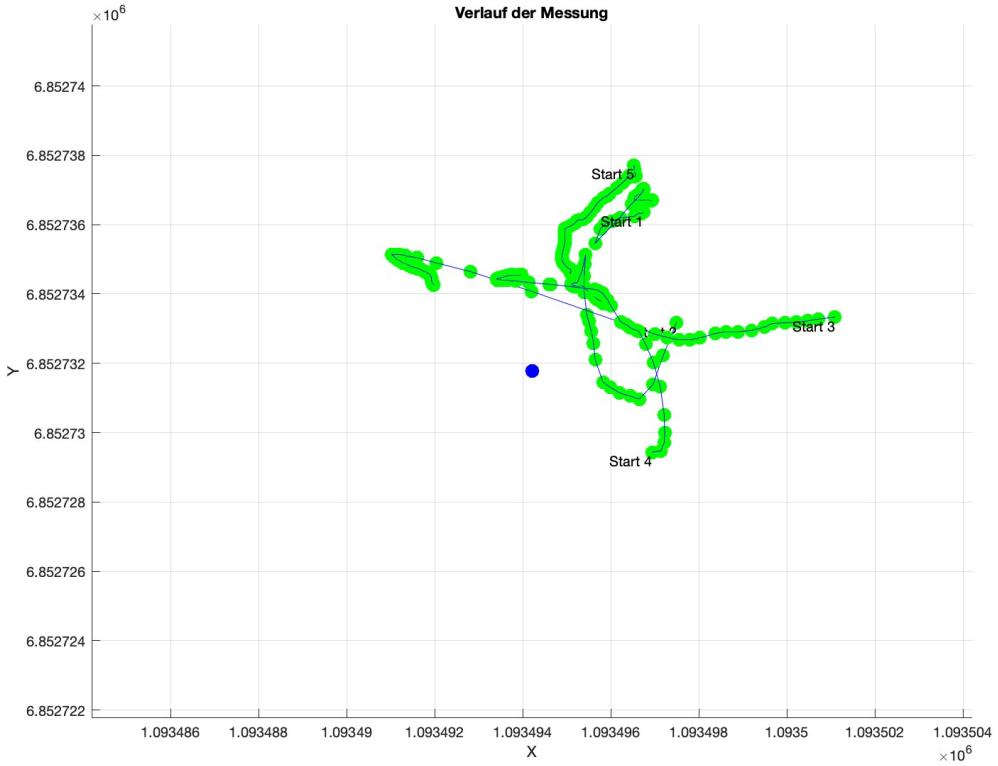

Abbildung 6: Messung des Aufnahmepunktes 210 mit AR.js und Google Pixel 6
(Einheiten der Koordinatenachsen in $\cdot 10^6 m$)

3.1.5 Weitere Tests mit dem Prototypen

Mit der neuen Erkenntnis zur Genauigkeit der GNSS-Positionierung der verwendeten Geräte können neue Tests mit dem Prototypen durchgeführt werden bzw. die Tests zur Visualisierung neu einschätzen.

Mit dem iPhone 12 Pro und dem Browser Safari oder Chrome sind keine Unterschiede erkennbar. Die Linien für Flurstücksgrenzen werden selten angezeigt. Zwischendurch werden Linien kurz visualisiert, die jedoch im Bild feststehen und nicht in der realen Welt eingefügt werden. Diese Fehler sind in Abbildung 7 zu sehen, da die virtuellen Inhalte feststehen, während das Gerät bewegt wird. Hier existiert also ein kritischer Fehler, der keine Aussagekraft zu Genauigkeiten zulässt.

Tests mit dem Pixel 6 wurden mit dem Chrome-Browser durchgeführt. Es wurden Linien zur Visualisierung der Flurstücksgrenzen angezeigt, jedoch mit größeren Abweichungen zur Realität. Während der Tests waren für kurze Zeit die Linien sehr genau auf den realen Flurstücksgrenzen, ähnlich wie bereits in Abbildung 3 in Kapitel 3.1.1 zu sehen. Jedoch sind bei Bewegung des Gerätes die Bewegungen der virtuellen Inhalte nicht passend zur Kamerabewegung.

Zusätzlich zu den Ungenauigkeiten in der GNSS-Standortbestimmung könnten die folgenden Punkte die zuvor genannten Probleme der Anwendung und Visualisierung bekräftigen. In der Dokumentation von AR.js (Carpignoli und AR.js Org Community 2023) stehen bekannte Probleme des Frameworks AR.js. Ein bekanntes Problem ist eine fehlerhafte Darstellung aufgrund der Ungenauigkeit der Kompass-Kalibrierung bei Android Smartphones. Außerdem existiert ein Bug, der Objekte, die nicht im Zentrum der Kamera lie-

Abbildung 7: Screenshots aus dem AR.js Prototyp mit iPhone 12 Pro

gen, gestreckt werden und damit eine ungenaue Positionierung hervorrufen. Die Anwendung funktioniert nur, wenn GNSS, Beschleunigungssensoren und der Magnetometer erreichbar sind. Dies kann evtl. der Fall beim iPhone sein, dass Sensoren nicht erkannt werden und damit nur die GNSS-Ortung funktioniert.

3.1.6 Fazit zur Web-Anwendung mit AR.js

Die Auswertungen haben ergeben, dass diese Lösung für eine Visualisierung von Flurstücksgrenzen nicht geeignet ist, vor allem da die Genauigkeit der GNSS-Positionierung und Stabilität der AR-Orientierung mit AR.js nicht ausreichend ist, um Flurstücksgrenzen mit der Realität zu identifizieren. Die Vorteile von AR.js sind die vollständige Open Source Bibliothek und der Fokus auf eine Webanwendung, jedoch ist aufgrund der mangelhaften Visualisierung und fehlender Standortkorrekturen eine andere Plattform zu wählen, um den Stand der Technik in dieser Arbeit aufzuzeigen.

3.2 Prototyp mit ARCore SDK

3.2.1 Entwicklung eines Prototypen

Um die Entwicklung eines Prototypen zu beschleunigen, wird ein bereits existierendes Beispiel verwendet. Hierfür eignet sich das von Google selbstentwickelte Beispiel “Geospatial Sample“. Dieses kann in Unity, nachdem die ARCore Extension über den Package Manager mit der GitHub-URL `https://github.com/google-ar/arcore-unity-extensions.git` installiert worden ist, importiert werden.

Das Beispiel hat verschiedene Features, die für einen eigenen Protoypen von großem Vorteil sind. Zum einen ist eine Szene bereits konfiguriert, sodass weniger Konfiguration für eine ARCore Anwendung mit der Geospatial API notwendig ist. Zum anderen sind viele Skripte bereits importiert, die z. B. zur Erstellung von räumlichen AR-Ankern helfen.

3.2.2 Vorbereiten der Eingangsdaten

Ein wichtiger Aspekt dieser Arbeit ist die Integration von Daten aus dem Liegenschaftskataster. Flurstücksinformationen können als verschiedene Formate aus dem ALKIS (Amtliches Liegenschaftskataster-

informationssystem) exportiert werden. Zur Möglichkeit zum Export stehen die Datenformate XML über die Normbasierte Austauschschnittstelle (NAS), GeoPackage, Shape oder DXF. GeoPackage ist zwar ein aktuelles performantes Format, jedoch sind in dem Export nur die Geometrien für Flurstücke enthalten und keine weiteren Informationen zu Grenzpunkten. Daher werden die Daten im XML-Format exportiert, da diese Informationen vollständig sind.

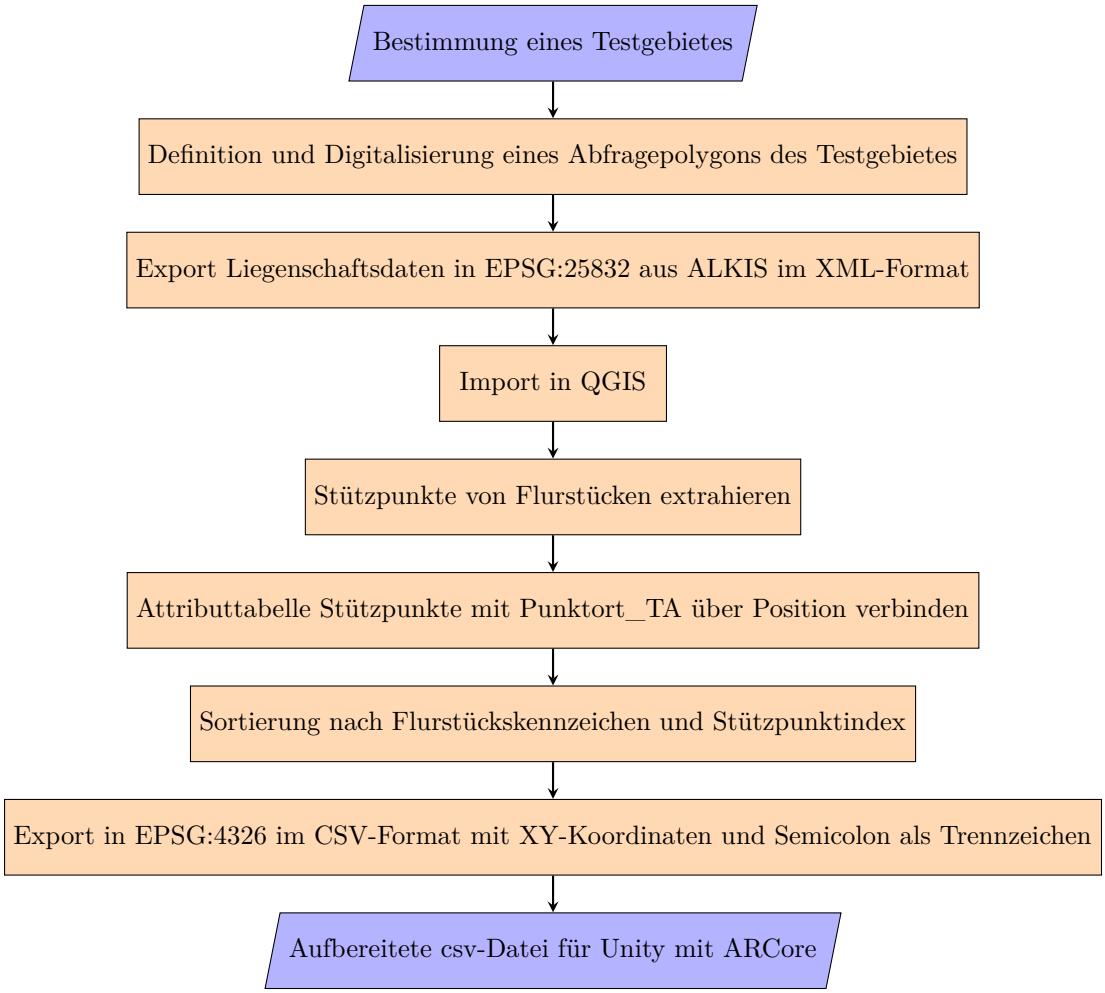

Abbildung 8: Workflow zur Vorbereitung der Eingangsdaten

In Abbildung 8 ist der Workflow dargestellt, wie die Liegenschaftsdaten aus ALKIS für die Skripte in der Unity Engine vorbereitet werden. Sind die Daten aus ALKIS exportiert, können diese in einer Geoverarbeitungssoftware verarbeitet werden. Hierfür eignet sich QGIS[3], da diese Software als Open-Source weit verbreitet ist und viele Funktionen und Erweiterungen bietet. Die Daten aus ALKIS werden sofort erkannt und visualisiert. Für die Anzeige von Flurstücken wird der Layer für Polygone der Flurstücke benötigt. Mit der Funktion 'Stützpunkte extrahieren' können die Stützpunkte von Polygone als Punkte-Layer exportiert werden. Dabei ist der erste und letzte Punkt eines Polygons an der gleichen Koordinate. Zusätzliche Attribute nach der Stützpunkterstellung sind "vertex_index" für den Index des Stützpunktes eines Polygons sowie "distance" und "angle" für Geometrieinformationen zwischen den Stützpunkten, um ein Polygon von einem einzelnen Koordinatenpaar des Startpunktes zu rekonstruieren. Für den Prototypen sind die Koordinaten und Indeces der Stützpunkte sowie die Flurstückinformationen wie der Flurstücksschlüssel wichtig.

[3] `https://www.qgis.org/de/site/`

Um Informationen zur Genauigkeit der Datenerhebung und zur Vertrauenswürdigkeit im Prototyp zu visualisieren, sind diese Informationen den Stützpunkten hinzuzufügen. Diese sind in einem separaten Layer für in der Karte dargestellten Punktorte mit der Bezeichnung "Punktort_TA" enthalten. Da die Koordinaten der Stützpunkte mit den Punktorten identisch sind, kann eine Verlinkung der Attributtabellen mit der QGIS-Funktion "Attribute nach Position verknüpfen" durchgeführt werden.
Aufgrund der zuvor verwendeten Funktion werden die Zeilen neu geordnet und nicht nach Flurstücken und Stützpunkten sortiert. Daher ist eine neue Sortierung nach Flurstückskennzeichen und dann Stützpunktindeces notwendig über das Tool "Sortieren nach Ausdruck" mit folgendem Ausdruck:
`"flurstueckskennzeichen" || ("vertex_index" + 1000)`
Der Wert 1000 wird als Offset addiert, da in der Sortierung sonst z. B. Stützpunkt 10 vor dem Stützpunkt 2 sortiert wird.
Das Koordinatensystem der Liegenschaftsdaten ist im projizierten Koordinatensystem UTM (Universal Transverse Mercator) in der Zone 32N des Ellipsoiden ETRS89, bzw. EPSG-Code 25832 gegeben. Die Geospatial API verwendet das geografische Koordinatensystem des Ellipsoiden WGS84 mit dem EPSG-Code 4326. Die Transformierung in EPSG:4326 kann mit QGIS durchgeführt werden. Diese Stützpunkte werden als CSV-Datei exportiert, da CSV ein sehr weit verbreitetes Format für Tabellen ist und Unity diese ohne Erweiterung lesen kann. Dabei werden die Datensätze mit Semicolon getrennt und die Koordinaten im XY-Format ausgegeben. In den Daten existieren String-Werte, die Kommas beinhalten. Daher ist ein Komma als Trennzeichen nicht möglich.

Die zu verwendeten Informationen aus der aufbereiteten csv-Datei sind wie folgt:

- X-Koordinate als geografische Länge in Dezimalgrad
- Y-Koordinate als geografische Breite in Dezimalgrad
- Gemarkungsnummer zur Flurstücksinformation
- Flurstückszähler zur Flurstücksinformation
- Flurstücksnenner zur Flurstücksinformation
- Flurstückskennzeichen zur Identifikation des Flurstückes
- Amtliche Fläche zur Flurstücksinformation in m^2
- Flurnummer zur Flurstücksinformation
- Stützpunktindex zur Identifizierung des Stützpunktes (zusammen mit Flurstückskennzeichen)
- Datenerhebung als Information zur Genauigkeit des Grenzpunktes
- Vertrauenswürdigkeit als Information zur Genauigkeit des Grenzpunktes

Der oben dargestellte Ablauf zur Vorbereitung der Geodaten ist automatisierbar, um den Workflow bei einem produktiven Einsatz der Applikation zu beschleunigen. Derzeit wird eine CSV-Datei erstellt, um einen Test mit einem Prototypen zu beschleunigen. Eine API-Abfrage mit JSON-Daten ist anzuzielen, z. B. wenn die Applikation landesweit eingesetzt wird. Außerdem sind in der NAS-Abgabe die Informationen zur Punktnummer ohne räumlichen Bezug und in der Attributtabelle keine Schlüsselattribute, mit denen die Attributtabelle mit den Stützpunkten verbunden werden kann. Bei einem produktiven Einsatz der Applikation sollten diese Informationen zusammenhängend gegeben sein, um diese in der Applikation zu visualisieren.

3.2.3 Vorbereiten der Unity-Szene

Die Unity-Szene des zuvor erwähnten Beispiels wird in einen neuen Ordner kopiert und umbenannt.
In dem Beispiel sind bereits Game Objects enthalten, die eine AR-Session mit der Geospatial API ermöglichen.
Es sind Konfigurationen zu tätigen, die in der Dokumentation (Google 2023a) beschrieben sind. Während der Konfiguration ist die Erstellung eines API-Schlüssels notwendig, um die Geospatial API zu verwenden. Bedingungen zur Nutzung der API sind in der API-Bibliothek (Google 2023b) von Google einzusehen und in Kapitel 2.5.6 beschrieben.

Ein leeres Game Object wird erstellt, um ein Skript für den Import und die Visualisierung von Geodaten hinzuzufügen. Die zuvor genannten verschiedenen Datenerhebung und Vertrauenswürdigkeiten in Kapitel 3.2.2 sollen unterschiedlich visualisiert werden. Mit der 3D-Grafiksoftware Blender[4] werden Grenzpunkte als 3D-Modelle gestaltet und erstellt. Die Formen der unterschiedlichen 3D-Modelle sind der Darstellung aus dem Viewer vom Landesamt für Geoinformation und Landesvermessung Niedersachsen (LGLN) entnommen, um bei einem möglichen Einsatz im LGLN bekannte Darstellungen zu verwenden. Außerdem wird in Blender die Anzeige der Flurstücksinformationen als 2D-Fläche erstellt. Die Objekte werden aus Blender exportiert und in die Unity Engine als Assets importiert.

Zur Anzeige der Flurstücksinformationen ist ein Textfeld notwendig, um die Informationen anzuzeigen. Dafür wird ein Game Object als Template erstellt, welches das erstellte 2D-Modell sowie als Children ein leeres Game Object mit der Komponente TextMeshPro zur Erzeugung eines Textfeldes enthält. Das Game Object des 2D-Modells sowie das Game Object mit der Komponente TextMeshPro werden so konfiguriert, sodass in der AR-Anwendung eine sinnvolle Visualisierung entsteht. Zusätzlich wird das Skript `LookAtCamera.cs` als Komponente auf das Parent Object gesetzt, sodass die Flurstücksinformationen auf die Kamera bzw. den Nutzenden der AR-Anwendung ausgerichtet und damit aus jeder Richtung lesbar sind. Das Parent Object wird außerdem auf inaktiv gesetzt, damit das Template ohne geografische Koordinaten nicht dargestellt wird.

3.2.4 Implementation von Skripten

Skripte werden in Unity verwendet, um die Szene mithilfe der Programmiersprache C# zu verändern. Ein Skript ist in mehrere Bereiche unterteilt. Namespaces (Namensbereiche) werden hinzugefügt, um Funktionen und Variablen aus anderen Namespaces zu nutzen. Für Unity ist der Namespace `UnityEngine`, für grundlegende Funktionen ist der Namespace `System` mit gegebenenfalls untergegliederten Namespaces hinzuzufügen. Das ARCore SDK von Google ist über den Namespace `Google.XR.ARCoreExtensions` und weitere untergegliederte Namespaces einzubinden. Im Bereich der Deklaration von Variablen werden die Variablen definiert, die im Skript verwendet werden sollen. Globale sowie innerhalb des Skriptes private Variablen sind möglich. Globale Variablen und mit `SerializeField` angegebene private Variablen können im hinzugefügten Game Object im Editor verändert oder ausgewählt werden. Code-Zeilen in der Funktion `Start` werden einmalig vor dem ersten Frame durchgeführt. Die Funktion `Update` wird vor jedem Frame durchgeführt. Eigene Funktionen können hinzugefügt werden, die innerhalb `Start` oder `Update` aufgerufen werden können.
Die geschriebenen Skripte sind auf einem externen Datenträger zu dieser Arbeit abgelegt.

[4] `https://www.blender.org/`

Definition von Variablen: Der `AREarthManager` von ARCore wird eingebunden, um Informationen über den Tracking Status der AR-Session zu erhalten. Ist die Anwendung noch nicht im Tracking-Modus, sollen noch keine AR-Inhalte visualisiert werden. Die Geodaten werden über eine private Variable mit `SerializeField` als `TextAsset` importiert, um im Editor die CSV-Datei auszuwählen, die die Geodaten beinhaltet. Die erstellten 3D-Modelle für die Grenzpunkte sowie das Game Object für die Flurstücksinformationen werden ebenfalls als private Variablen mit `SerializeField` eingebunden. Die Geodaten sind aufgrund des CSV-Formates in einer Tabelle mit Überschriften für die Spalten organisiert. Da die Spalten unterschiedlich angeordnet sein könnten, können keine Indeces für Spaltennummern verwendet werden. Daher werden die Informationen zu den Spaltennamen als private Variable mit `SerializeField` deklariert, damit die Namen in der Komponente im Unity Editor angepasst werden können. Auch bei einer anderen Datenquelle wie einer JSON-Abfrage sind Namen der Attribute wichtig, um die Daten richtig zuzuordnen. Die Anzeige der Geodaten wird über Bounding Boxes bestimmt. Die Bounding Box wird mit einem Puffer vergrößert, um die Reichweite der Visualisierung von Liegenschaften zu erhöhen. Dieser Puffer kann aufgrund der privaten Variable mit `SerializeField` verändert werden und ist im Dezimalgrad mit EPSG:4326 angegeben. Da die Grenzpunkte als 3D-Modelle importiert werden, ist keine Vorkonfiguration zur Transformation möglich bzw. nicht notwendig. Daher wird die Skalierung im Skript durchgeführt und mit einer privaten Variable mit `SerializeField` im Unity Editor veränderbar. Außerdem sind alle Variablen in diesem Bereich zu deklarieren, die in der Funktion `Start` sowie die wiederholende Funktion `Update` übergreifend verwendet werden.

Start-Funktion: In der `Start`-Funktion werden alle Code-Zeilen geschrieben, die einmalig vor dem ersten Frame ausgeführt werden sollen. Dies beinhaltet primär das Erstellen des Parent Objects für die Grenzpunkte und das Aufrufen der Funktionen zum Einlesen der CSV-Datei sowie das Erstellen der Bounding Boxes.

Import der csv-Datei: Wie zuvor erwähnt, wird die CSV-Datei über die Definition der Variablen als TextAsset eingelesen und gespeichert. Das TextAsset wird umgewandelt in ein String Array, um die Zeilen der CSV-Tabelle zu trennen. Die in der Variablendefinition definierten String-Werte für Spaltennamen werden verwendet, um den Index der jeweiligen Spalte zu bestimmen. Der erste (`Index = 0`) Wert aus dem Array wird gelesen, um die Titelzeile mit den Spaltennamen zu erhalten. Aus dieser Zeile kann nun iteriert jeder Wert gelesen werden und mit den definierten Spaltennamen verglichen werden. Stimmt der Wert mit einem Spaltennamen überein, wird der Index gespeichert.

Erstellen von Bounding Boxes: Bounding Boxes sind eine performante Möglichkeit, um zu bestimmen, ob sich ein Koordinatenpaar in oder in der Nähe eines Polygons befindet. Ist das Polygon ein regelmäßiges Rechteck, ist die Bounding Box sehr ähnlich dem eigentlichen Polygon. Ist das Polygon z. B. ein langgezogenes Straßenflurstück, kann die Bounding Box um ein Vielfaches größer sein als das eigentliche Polygon. Auch wenn dies ein Nachteil ist, werden aufgrund der Performance Bounding Boxes verwendet. Die Bounding Boxes werden einmalig bestimmt und können dann wiederholt mit einem Koordinatenpaar verglichen werden. Dies ist vor allem bei einer Berechnung pro Frame pro Punkt eine performante Möglichkeit zur Abschätzung der Lage zu einem Flurstück. Bounding Boxes sind Rechtecke, deren Seiten parallel zum Koordinatensystem sind. Dies bedeutet, dass jede Bounding Box durch nur zwei Koordinatenpaare definiert werden kann: ein Mindestwert für Längengrad sowie Breitengrad und ein Maximalwert für Längengrad sowie Breitengrad. Die Informationen werden in Arrays gespeichert. Um

eine Verbindung zu den Grenzpunkten herzustellen, werden die gleichen Indeces verwendet. Dies bedeutet, dass z. B. der Grenzpunkt mit dem Index 84 Teil der Bounding Box ist, die in den Arrays `minLat`, `minLon`, `maxLat` und `maxLon` auch unter dem Index 84 die Werte für die Bounding Box enthalten. Dies generiert einen einfachen und performanten Zugriff und Vergleich in der `Update`-Funktion.
Aus dem Array, das mit Daten aus der CSV-Datei befüllt ist, werden die geografischen X- und Y-Koordinaten entnommen. Außerdem wird die Spalte für das Flurstückskennzeichen entnommen, um einen Vergleich zu erzeugen, ob zwei Punkte zu demselben Flurstück zugehörig sind. Der erste Punkt eines Flurstücks in den Bounding Box Arrays erhält die gleichen Koordinatenwerte wie der Grenzpunkt selbst, da dieser bereits der Mindest- oder Maximalwert sein könnte. Ist der letzte Punkt und der aktuelle Punkt der Iteration dasselbe Flurstück, dann werden die X- und Y-Koordinaten der letzten Mindest- und Maximalwerte mit den tatsächlichen Koordinaten des Grenzpunktes mit aktuellem Index verglichen. Zum einen wird der niedrigere Wert als Mindestwert gespeichert, zum anderen der höhere Wert als Maximalwert. Wird der zweite und dritte Punkt eines Flurstücks verglichen und es sind neue Mindest- sowie Maximalwerte bestimmt, werden auch die Werte des ersten Grenzpunkts des Flurstücks in den Bounding Boxes Arrays ersetzt. Das gleiche Verfahren ist bei einer höheren Anzahl von Punkten der Fall, indem alle Punkte des Arrays mit dem gleichen Flurstückskennzeichen ersetzt werden. Damit ist zwar eine hohe Redundanz in den Arrays vorhanden, jedoch ist diese für den Zugriff und Vergleich mit einem Grenzpunkt in der `Update`-Funktion performanter, da kein Suchverfahren nach dem Flurstückskennzeichen für jeden Punkt pro Frame durchgeführt werden muss.

Update-Funktion: Die `Update`-Funktion ist der wesentliche Bestandteil des Skriptes und wird vor jedem Frame durchgeführt. Kernelemente in dieser Funktion sind das dynamische Erstellen, Aktualisieren und Löschen der Grenzpunkte und Flurstücksgrenzen sowie die Georeferenzierung der Objekte und das Erstellen und Löschen von Flurstücksinformationen.
Da die Objekte zum Umgehen von Fehlern erst erstellt werden sollen, wenn die AR-Session bereit ist, wird eine Abfrage erstellt, die überprüft, ob sich die AR-Anwendung im Tracking-Modus befindet. Ist dies nicht der Fall, wird die `Update`-Funktion für den aktuellen Frame übersprungen. Dies bedeutet, dass die `Update`-Funktion erst durchgeführt wird, wenn die AR-Session bereit ist.

Erstellen von Grenzpunkten: Die Grenzpunkte und Flurstücksgrenzen sollen dynamisch visualisiert werden. Ein bisheriger Versuch war die einmalige Erzeugung von Game Objects in der `Start`-Funktion, die in der `Update`-Funktion dynamisch den Status von `Aktiv` zu `Inaktiv` und andersherum gewechselt haben, je nach Nähe zum Flurstück. Jedoch ist dies ein massiver Performance-Eingriff, da die inaktiven georeferenzierten Game Objects in hoher Anzahl dennoch viel Rechenleistung benötigen. Daher müssen Game Objects vollständig gelöscht und neu erstellt werden. Auch im Hinblick auf eine mögliche Änderung des Datenimports, z. B. über eine JSON-API, ist die Methode zum dynamischen Erstellen und Löschen von Game Objects sinnvoller.
Um die Grenzpunkte dynamisch zu erstellen und zu löschen, werden die geografischen Koordinaten des verwendeten Endgerätes abgerufen. Diese werden in den folgenden Schritten benötigt, um sie mit den Bounding Boxes zu vergleichen.
Um die Performance zu erhöhen, werden nur 10 Datenreihen aus dem Datenarray pro Frame verarbeitet. Dies ist über eine einfache Iteration mit einer Zählung von 0 bis 9 möglich. Der (Render-)Index für die zu verarbeitende Datenreihe wird innerhalb der o.g. Iteration inkrementiert. Wenn der Index auf die letzte Datenreihe zeigt, wird der Index auf 1 zurückgesetzt. Der letzte Stützpunkt eines Flurstücks wird nicht dargestellt, da die Koordinaten mit dem ersten Stützpunkt identisch sind. Der Index startet bei 1,

da Index 0 die Spaltennamen beinhaltet. Diese Änderung zur Performancesteigerung ist je nach Größe der CSV-Datei anzupassen. Wenn die Datenabfrage über eine JSON-API durchgeführt wird, kann die Eingrenzung der Anzahl Punkte pro Frame entfernt oder erhöht werden, da ein Einsatz von JSON-API's für diese Fälle optimal, performant und zu empfehlen ist.

Zur Erzeugung und Visualisierung von Grenzpunkten sind mehrere Schritte notwendig. Zuerst wird die Datenreihe aus dem Array für den jeweiligen Renderindex gelesen. Diese Datenreihe wird anhand der bestimmten Spaltenindeces in Einzelwerte geschrieben. Dies gilt vorerst nur für den Flurstücksschlüssel und die Stützpunktnummer. Die Werte entstehen aus einer Zeichenkette und müssen daher in passende Datentypen umgewandelt werden. Die Flurstücksschlüssel und die Stützpunktnummer bleiben im String-Format, da diese für keine Berechnung notwendig sind. Außerdem besteht der Flurstücksschlüssel aus vorangehenden Nullen und besitzt Unterstriche am Ende der Zeichenkette.
Zur Überprüfung, ob die Position des Endgerätes innerhalb einer Bounding Box ist, werden die Koordinaten mit der jeweiligen Bounding Box des Renderindex verglichen. Ist der Breitengrad zwischen der Mindest- und Maximalbreite sowie der Längengrad zwischen der Mindest- und Maximallänge, dann wird ein Booleanwert auf "true" gesetzt, um in den folgenden Bedingungen die Information bereitzuhalten, ob die Position sich innerhalb der zu betrachtenden Bounding Box befindet. Zusätzlich wird eine Puffer-Variable von den Mindestwerten subtrahiert bzw. zu den Maximalwerten addiert, um einen Rand zu bilden, um Nachbarflurstücke anzuzeigen.
Eine zweite Überprüfung, bevor verschiedene Fälle zur Erstellung, Aktualisierung oder Löschung von Game Objects angewendet werden, ist die Abfrage, ob der Punkt bereits existiert. Dies ist über die Funktion "Find" möglich. Zwar kann das Parent Object ausgewählt werden, jedoch werden Objekte mit der Komponente zur Georeferenzierung unter andere Parent Objects transformiert, sodass die Hierarchie verändert wird. Daher wird hier die Funktion "Find" auf alle Game Objects angewendet, um einen möglichen bereits existierenden Grenzpunkt zu finden. Wird kein Game Object unter dem angegebenen Flurstücksschlüssel und Stützpunktnummer gefunden, wird ein "null"-Wert ausgegeben.

Mit diesem Verhalten können nun drei verschiedene Fälle untersucht werden:

1. Fall: Die Koordinaten des Endgerätes befinden sich innerhalb der Bounding Box (+Puffer) des Flurstücks, und das Game Object des Grenzpunktes existiert nicht: Es werden weitere Daten aus der Datenreihe entnommen, um mehr Informationen über den Grenzpunkt darzustellen. Dies sind die Datenerhebung und die Vertrauenswürdigkeit. Es wird ein neues Game Object durch eine weitere Funktion erstellt. Diese Funktion ist eine static Funktion, die das zu erstellende Game Object ausgibt. Der Input sind die o.g. weiteren Informationen sowie die verschiedenen 3D-Modelle der Grenzpunkte. Je nachdem, wie die Datenerhebung und Vertrauenswürdigkeit ist, wird das nach Kapitel 3.2.3 passende 3D-Modell mit Farbe erstellt. Nach Konfigurationen zur Transformation, Name und Parent Object wird ein leeres Game Object für den Linerenderer erstellt. Dieses Game Object ist ein Child Object des Game Objects für den jeweiligen Grenzpunkt. Dem Game Object wird die Komponente "LineRenderer" hinzugefügt, die Linien zwischen Positionen erstellen kann. Neben sonstigen Konfigurationen zur Transformation und Aussehen der Linie ist das Setzen der Eigenschaft "useWorldSpace" auf "false" äußerst wichtig, damit der Linerenderer bei einer Transformation des Grenzpunktes ebenfalls transformiert wird. Aufgrund der AR-Anwendung werden die Objekte ständig pro Frame neu transformiert. Es wird vorerst nur die Linie ohne weitere Positionen erstellt, da kein weiterer Punkt in diesem Schritt bekannt ist. Nachdem das Game Object inklusive Linerenderer konfiguriert ist, wird die Komponente "ARGeospatialCreatorAnchor" hinzugefügt, um einen Terrain-Anker mit geografischen Koordinaten zu erzeugen. Dies ist der Schritt der Georeferenzierung, sodass das Game Object in der AR-Anwendung an der richtigen Stelle dargestellt

wird. Die geografischen Koordinaten werden aus der Datenreihe extrahiert, dabei ist die X-Koordinate der Längengrad und die Y-Koordinate der Breitengrad. Die Werte werden in Double umgewandelt, da eine Gleitkommazahl mit möglichst hoher Anzahl Zeichen notwendig ist. Float ist zwar auch ein Format für die Darstellung von Gleitkommazahlen, jedoch hat Float nach SAP SE (2024) nur eine Speichergröße von 4 Byte und kann damit maximal 7 Dezimalstellen anzeigen. Double besitzt eine Speichergröße von 8 Byte und kann maximal 15 Dezimalstellen anzeigen. Für die Genauigkeit einer geografischen Koordinate sind 7 Dezimalstellen nicht ausreichend, da die Position aufgrund des Wegfallens einiger Dezimalstellen zu ungenau wäre. Als Anker wird der Terrain-Anker gewählt, der bereits in Kapitel 2.5.2 beschrieben wird. Damit wird das Game Object auf der erfassten Bodenfläche dargestellt, um die Immersivität erheblich zu steigern. Als letzten Schritt in diesem Fall ist das Erzeugen von Flurstücksinformationen. Falls keine Flurstücksinformation für das Flurstück des Grenzpunktes existiert, wird die Funktion zur Erstellung dieser Information ausgeführt. Das Game Object wird aus dem Template dupliziert, das in der Definition für Variablen hinzugefügt wurde und anschließend auf den Status "aktiv" geschaltet. Die Informationen werden ebenfalls aus der Datenreihe des Arrays entnommen. Dies sind die Parameter Flurstückszähler, Flurstücksnenner, Flurnummer, Amtliche Fläche und Gemarkungsnummer. Weitere Informationen könnten hinzugefügt werden, wenn die CSV-Datei diese Informationen liefert. Das Template hat bereits den Aufbau mit der Komponente "TextMeshPro". Daher kann der Text für diese Komponente geändert werden, mit den Parametern der Flurstücksinformationen. Das Game Object wird mit einem Terrain-Anker mit der Komponente "ARGeospatialCreatorAnchor" versehen. Die geografischen Koordinaten sind die Koordinaten des Mittelpunktes der zugehörigen Bounding Box. Die Mittelpunkte werden über die Mittelwertbildung von den jeweiligen Mindest- und Maximalwerten der Koordinaten berechnet.

2. Fall: Die Koordinaten des Endgerätes befinden sich innerhalb der Bounding Box (+Puffer) des Flurstücks, und das Game Object des Grenzpunktes existiert bereits: Wenn das Game Object des Grenzpunktes bereits existiert, existieren ebenfalls die Flurstücksinformation und der Linerenderer. Nach dem ersten Durchlauf der Renderindeces sind noch keine Flurstücksgrenzen vorhanden, da die Linerenderer noch keine Positionen zum nächsten Stützpunkt enthalten. Daher ist diese Position einzutragen oder zu aktualisieren, da die Positionen im Linerenderer nicht die geografischen Koordinaten sind, sondern lokale Koordinaten, die je Frame aufgrund der AR-Umgebung den Wert verändern könnten. Da der Linerenderer an das Game Object der Grenz- bzw. Stützpunkte gebunden ist, ist diese Veränderung minimal, aber vorhanden. Zuerst wird überprüft, ob ein nächster Stützpunkt des Flurstücks existiert. Der letzte Stützpunkt eines Flurstücks, oder bei einem Fehler, darf nicht mit einem Stützpunkt eines anderen Flurstücks verbunden werden. Dies wird gewährleistet, indem der nächste Stützpunkt über den Stützpunktindex gesucht wird. Wird kein Stützpunkt gefunden, ist der Fall abgeschlossen und die Iteration wird fortgeführt. Wird ein weiterer Stützpunkt gefunden, dessen Index eine Zahl höher ist als der zu verarbeitende Stützpunkt, dann wird die Funktion zum Aktualisieren von Linien aufgerufen, mit den Game Objects des aktuellen und des nächsten Stützpunktes. Es wird die Komponente "LineRenderer" extrahiert und die Position verändert. Dabei wird die lokale Koordinate des nächsten Punktes entnommen und transformiert, um den Vektor vom aktuellen Stützpunkt zum nächsten Stützpunkt zu erhalten.

3. Fall: Die Koordinaten des Endgerätes befinden sich nicht innerhalb der Bounding Box (+Puffer) des Flurstücks, und das Game Object des Grenzpunktes existiert bereits: Besteht bereits das Game Object und das Engerät ist nicht (mehr) in der Bounding Box, wird das Game Object des Punktes gelöscht. Existiert das Game Object für die Flurstücksinformationen, wird dies ebenfalls gelöscht. Das Löschen von Game Objects wird immer erst am Ende eines Frames durchgeführt.

Ein vierter Fall, in dem die Koordinaten des Endgerätes sich nicht innerhalb der Bounding Box (+Puffer)

des Flurstücks befinden, und das Game Object des Grenzpunktes nicht existiert, wird nicht betrachtet, da in diesem Fall keine Aktionen durchgeführt werden.

LookAtCamera-Funktion: Eine weitere Funktion, die als Komponente zum Template für die Flurstücksinformation hinzugefügt wird, ist die "LookAtCamera"-Funktion. Diese ermöglicht, dass das Objekt zur Anzeige der Flurstücksinformation stetig zur Kamera, bzw. zum Nutzer zeigt. Die "Start"-Funktion identifiziert die AR-Session mit der Kamera, um in der "Update"-Funktion die Informationen der Kamera zu verwenden. In der "Update"-Funktion wird die Richtung vom Objekt zur Kamera berechnet, um darauf folgend die Eigenschaft "LookAt" der Transformation des Objektes mit dieser Richtung festzulegen. In diesem Fall wird die Richtung invertiert, da das Objekt sonst in die falsche Richtung orientiert wird.

Abbildung 9: Screenshots aus dem ARCore SDK Prototyp mit Google Pixel 6

In Abbildung 9 sind Screenshots aus der Anwendung dargestellt. Im ersten Bild ist im Vordergrund ein Grenzpunkt mit der Datenerhebung von 1300 und einer Vertrauenswürdigkeit von 1200 zu sehen. Damit ist dieser als blauer Zylinder dargestellt. Die Flurstücksgrenzen sind aktuell in grau visualisiert. Im zweiten Bild ist die Flurstücksinformation zu sehen, die aktuell als graues Schild dargestellt wird. In beiden Bildern ist zu erkennen, dass die Flurstücksgrenzen nahe dem Zaun oder der Abgrenzung zum Gehweg sind. Die von ARCore SDK berechnete Abweichung, die in Kapitel 3.2.5 beschrieben wird, ist mit 0,36 m im ersten Bild und 0,39 m im zweiten Bild angegeben.

3.2.5 Genauigkeitstests

Die Genauigkeitstests zum vorherigen Prototypen, die zur Analyse der Darstellungsfehler durchgeführt worden sind, sind in Kapitel 3.1.4 beurteilt worden. Die Genauigkeitstests für den aktuellen Prototypen mit ARCore werden aufgrund der vorliegenden Informationen auf eine andere Weise durchgeführt. Es wird die Abweichung betrachtet, die einerseits über die von ARCore integrierte Genauigkeitsangabe berechnet und angezeigt wird, andererseits wird die Darstellung virtueller Inhalte mit der Örtlichkeit gemessen und verglichen. An mehreren Orten zu unterschiedlichen Bedingungen werden die Tests durchgeführt. Primär

werden für die Genauigkeit Vermessungspunkte verwendet, um die Genauigkeit zu beurteilen. Es werden dieselben Vermessungspunkte wie in Kapitel 3.1.4 verwendet, um fünf verschiedene Vermessungspunkte in einem urbanen Gebiet zu betrachten. Es werden drei Messreihen mit je drei Messungen durchgeführt, um aussagekräftige Ergebnisse zu erhalten. Zusätzlich wird eine Messreihe in der Dunkelheit umgesetzt, um Auswirkungen nicht beleuchteter Flächen zu analysieren. Außerdem fällt eine Messung außerhalb des urbanen Gebietes, um eine Einschätzung der Abweichung zu generieren, wenn keine Informationen zum VPS vorliegen und daher keine Referenzflächen zur Korrektur der Standortbestimmung genutzt werden.

Die primären Messreihen wurden an den Daten 22.12.23, 27.12.23 und 28.12.23 durchgeführt. Am 22.12.23 ist das Wetter bewölkt, sehr stürmisch mit Regenschauern. An den Tagen 27.12.23 und 28.12.23 ist das Wetter bewölkt, windstill und ohne Niederschlag.

Die Durchführung der Messung ist wie folgt:

- Anwendung starten und auf Initialisierung des Tracking-Modus warten
- Kamera auf Referenzobjekte in der Umgebung zeigen, optimal vollständig um die eigene Achse drehen
- Kamera auf den Vermessungspunkt zeigen und den virtuellen Marker finden
- Messgerät (Messband, Gliedermaßstab) zwischen Vermessungspunkt und Spitze des virtuellen Markers legen
- Screenshot für die berechnete Abweichung erstellen, wenn der virtuelle Marker sich nicht bewegt hat
- Screenshot mit Nahaufnahme auf das Messgerät, um die gemessene Entfernung abzugreifen

In der Messung wurde ein Gliedermaßstab verwendet, da eine sehr hohe Genauigkeit aufgrund der stetig veränderten Position des virtuellen Markers nicht notwendig ist. Daher ist die gemessene Entfernung mit Millimetergenauigkeit als Momentaufnahme zu betrachten. Die stetige Veränderung der Position des Markers ist mit ca. 1-3 Zentimetern minimal. Es kann passieren, dass die AR-Session neu konfiguriert wird und der Marker um mehr als einen Meter verschoben werden kann, dies tritt jedoch selten auf. Der verwendete Gliedermaßstab besitzt nach Anhang MI-008, Kapitel I der Richtlinie 2004/22/EG[5] eine Genauigkeitsklasse von 3 und damit eine Fehlergrenze von $0,6mm + 0,4 \cdot 1m = 1,00mm$ für Messungen unter einem Meter. Diese Fehlergrenze ist ausreichend für Messungen von Abweichungen, die im Dezimeterbereich liegen. Bei der Messung mit dem Gliedermaßstab wurde darauf geachtet, dass dieser in einer Flucht zwischen Vermessungspunkt und virtuellen Markern liegt. Außerdem wurde darauf geachtet, dass der virtuelle Marker von der Seite aus durch die Kamera betrachtet wurde, da die vertikale Positionierung des Markers auf dem Gelände nicht immer exakt in der Höhe des liegenden Gliedermaßstabes liegt, sondern unterhalb oder oberhalb des Geländes in der Örtlichkeit. Zur Vorgehensweise der Messung sind Screenshots in der Abbildung 10 dargestellt.
Die Tabelle 3 zeigt die Messdaten für den Punkt 301. Im Anhang A.2 ist die vollständige Tabelle der Messdaten. Die berechnete Abweichung ist durch eine Funktion von ARCore SDK gegeben und wird in der Benutzeroberfläche der Anwendung angezeigt. In den folgenden Tabellen und Vergleichen werden die berechnete Abweichung sowie die mit dem Gliedermaßstab gemessene Abweichung angegeben und verglichen.

[5] Richtlinie 2004/22/EG des Europäischen Parlaments und des Rates vom 31. März 2004 über Messgeräte

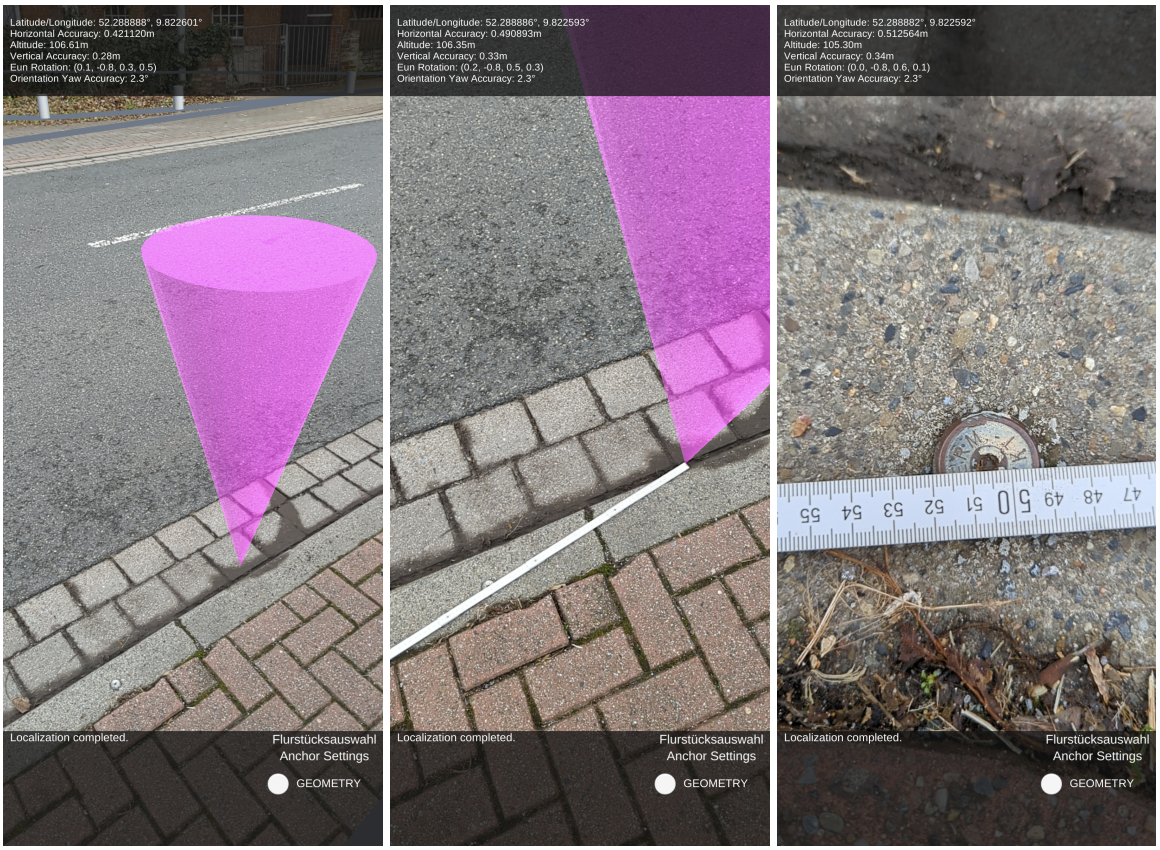

Abbildung 10: Screenshots zur Durchführung einer Messung des ARCore-Prototypen

Punkt	63-301								
Messreihe	1 (22.12.23)			2 (27.12.23)			3 (28.12.23)		
Messung	1	2	3	1	2	3	1	2	3
Uhrzeit	14:30	14:40	14:41	14:50	14:54	14:55	10:14	10:15	10:17
Abw Berechnet [m]	0,536	0,336	0,402	0,359	0,496	0,465	0,491	0,417	0,561
Abw Gemessen [m]	0,554	0,784	0,269	0,607	0,354	0,505	0,506	0,511	0,727
Ber – Gem [m]	-0,018	-0,448	0,133	-0,248	0,142	-0,04	-0,015	-0,094	-0,166
Mittelwert Ber [m]									0,451
Median Ber [m]									0,465
Mittelwert Gem [m]									0,535
Median Gem [m]									0,511
Mittelwerte Ber-Gem [m]									-0,084
Median Ber-Gem [m]									-0,046

Tabelle 3: Auszug aus den Messdaten für den Punkt 63-301

(Gem: Gemessene Abweichung. Ber: Berechnete Abweichung von ARCore SDK. Vollständige Tabelle siehe im Anhang A.2)

Um die Daten anschaulicher zu gestalten, sind in Abbildung 11 alle Messdaten in einem Diagramm dargestellt. Zum einen die berechneten abgelesten Werte in Lila, zum anderen die gemessenen Werte in Rot. Außerdem sind die Mittelwerte von allen Messdaten dargestellt. Der Mittelwert der berechneten Daten ist 0,473 m und der Mittelwert der gemessenen Daten ist 0,411 m. Die höchsten drei Differenzen zwischen berechneten und gemessenen Werten liegen bei 0,448 m, 0,391 m und 0,372 m. Die kleinsten drei Diffe-

renzen liegen bei 0,011 m, 0,015 m und 0,015 m. Auch wenn einige berechnete Werte sich stark von den gemessenen Werten unterscheiden, ist die Differenz der Mittelwerte für die berechneten und gemessenen Werte bei 0,059 m. Daher ist die Angabe der Lagegenauigkeit, die in ARCore Geospatial implementiert ist, nicht zuverlässig, jedoch für eine Einschätzung der Genauigkeit hilfreich.

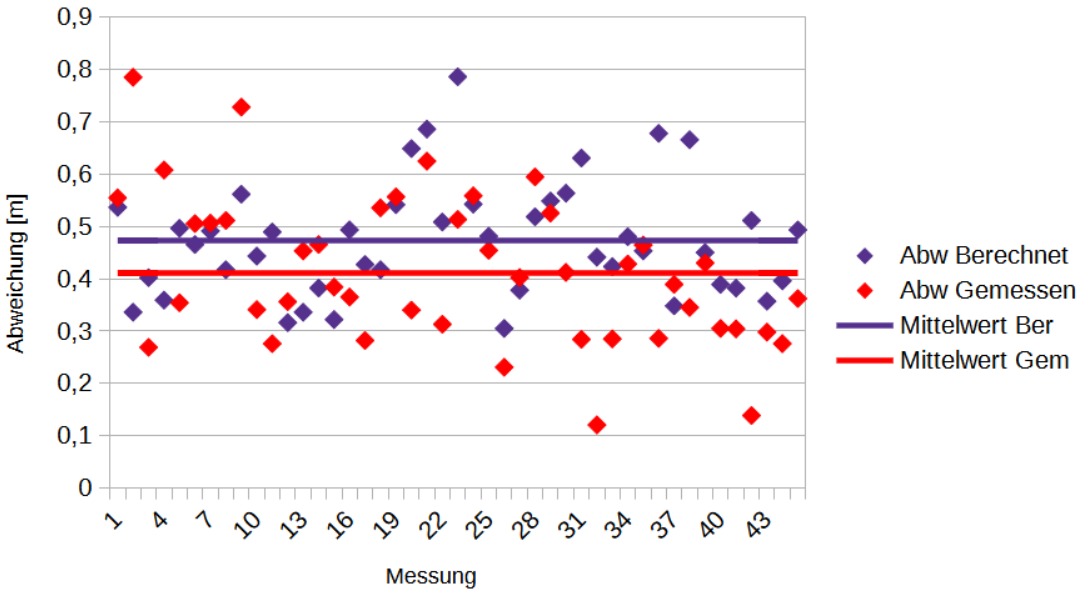

Abbildung 11: Diagramm über Messwerte zum ARCore-Prototypen

(Abw: Abweichung; Ber: Von ARCore SDK berechnete Abweichung; Gem: Gemessene Abweichung)

Die Mittelwerte und Mediane aus den Messreihen je Punkt sowie zusammengefasst der Mittelwert und Median aus allen Messungen sind in Tabelle 4 gegenübergestellt. Es ist zu sehen, dass die Mittelwerte bei allen Punkten für die berechnete Abweichung zwischen 0,403 m und 0,541 m und für gemessene Werte zwischen 0,316 m und 0,535 m betragen. Dies ist eine Spanne von 0,138 m und 0,219 m.

Punkt	Mittelwert Ber [m]	Median Ber [m]	Mittelwert Gem [m]	Median Gem [m]	Mittelwert Ber - Gem [m]	Median Ber - Gem [m]
63-301	0,451	0,465	0,535	0,511	-0,084	-0,046
G65-233	0,403	0,417	0,384	0,365	0,019	0,052
G65-210	0,541	0,541	0,443	0,454	0,098	0,087
G65-313	0,526	0,518	0,378	0,412	0,148	0,106
G65-203	0,443	0,396	0,316	0,305	0,127	0,091
Gesamt	0,473	0,465	0,411	0,389	0,062	0,059

Tabelle 4: Übersicht der Mittelwerte und Mediane je Vermessungspunkt und Gesamtwerte

(Gem: Gemessene Abweichung. Ber: Berechnete Abweichung von ARCore SDK. Vollständige Tabelle siehe im Anhang A.2)

Um eine weitere Einschätzung zum Prototypen zu erhalten, wurde eine Messreihe mit je drei Messungen für alle bereits beobachteten Punkte um eine Uhrzeit durchgeführt, an der kein Tageslicht vorhanden ist. In Tabelle 5 sind für den Punkt 63-301 diese Messdaten dargestellt. Der Mittelwert für die gesam-

te Messreihe mit allen Punkten beträgt für die berechnete Abweichung 0,617 m und für die gemessene Abweichung 0,642 m. Die Mittelwerte und Mediane sind in dieser Messreihe weniger zuverlässig, da zur Einschätzung nur 15 Einzelmessungen durchgeführt worden sind. Die Merssreihen mit Tageslicht bestehen aus 45 Einzelmessungen. Die Mittelwerte ohne Tageslicht sind um 0,144 m bzw. 0,231 m höher als die Mittelwerte mit Tageslicht. Eine Messung ohne Tageslicht bzw. die Nutzung der Applikation ohne Tageslicht ist jedoch erschwert. Die AR-Session wird ständig um mehrere Zentimeter oder Dezimeter verschoben. Außerdem wird die Bodenfläche schwerer erfasst, sodass der Terrain-Anker vertikal falsch eingefügt wird. Die vertikalen Abweichungen sind schwer messbar, werden jedoch durch eigene Einschätzung bis auf 30 cm geschätzt. Eine Korrektur durch Referenzobjekte ist nur von beleuchteten Objekten möglich.

Punkt	63-301			Gesamt
Messung	1	2	3	
Uhrzeit	17:08:00	17:10:00	17:11:00	
Abw Berechnet [m]	0,631	0,754	0,474	
Abw Gemessen [m]	0,653	1,691	0,182	
Ber – Gem [m]	-0,022	-0,937	0,292	
Mittelwert Ber [m]			0,620	0,617
Median Ber [m]			0,631	0,557
Mittelwert Gem [m]			0,842	0,642
Median Gem [m]			0,653	0,575
Mittelwerte Ber-Gem [m]			-0,222	-0,024
Median Ber-Gem [m]			-0,022	-0,018

Tabelle 5: Auszug der Messungen ohne Tageslicht für den Punkt 63-301

(Gem: Gemessene Abweichung. Ber: Berechnete Abweichung von ARCore SDK. Vollständige Tabelle siehe im Anhang A.2)

Zusätzlich zu den 45 Einzelmessungen mit Tageslicht und 15 Einzelmessungen ohne Tageslicht wurden weitere Messungen außerhalb von Google Streetview durchgeführt. In dieser Messreihe sollen keine Referenzdaten des VPS von Google genutzt werden, sodass keine Korrekturen über Referenzdaten durchgeführt werden. Mehrere Versuche zur Durchführung einer Messung sind gescheitert, da bei allen Versuchen die Vermessungspunkte nicht mehr vorhanden waren. Dies liegt der hohen Google Streetview-Dichte und wenigen Vermessungspunkten außerhalb urbaner Gebiete zugrunde. Es ist jedoch ausreichend, ungefähre Abweichungen einzuschätzen, da bei allen Nutzungen des Prototypen ohne VPS die Genauigkeiten im Meterbereich liegen.

Abbildung 12: Screenshot vom ARCore Prototyp ohne VPS

In der Abbildung 12 ist ein Screenshot dargestellt, in dem der Vermessungspunkt nach Aufnahmepunkt-Beschreibung innerhalb der roten Ellipse liegen soll. Dies sind ungefähr 2,5 m Entfernung zum virtuellen Marker für diesen Vermessungspunkt. Die berechnete Abweichung von ARCore SDK ist 2,007 m. Bei weiteren Versuchen an Punktgruppen an anderen Orten war die berechnete Abweichung zwischen 1,5 m und 3 m. Die ungefähren Abweichungen zwischen den Vermessungspunkten und dem jeweiligen virtuellen Marker sind ebenfalls in diesem Intervall.

4 Diskussion

Diskussionen zum Prototypen mit AR.js wurden bereits in Kapitel 3.1.4 während der Umsetzung geführt, da die Entscheidung zu einer Änderung der Entwicklungsplattform vor der Umsetzung des Prototypen mit ARCore SDK gefallen ist. Daher sind die nun folgenden Kapitel ausschließlich auf den Prototypen mit ARCore SDK bezogen.

4.1 Beurteilung der Genauigkeitstests mit ARCore SDK

In Kapitel 2.6 werden verschiedene Methoden zur Standortbestimmung mit GNSS erläutert. Da das ARCore SDK mit Geospatial API eine Punktwolke zur Korrektur verwendet, ist die Standortbestimmung mit ARCore ähnlich zu einer Messung mit Real-Time Kinematic, jedoch ohne eine simultane Messung eines anderen Punktes, sondern mit bereits vorliegenden Referenzobjekten. Da die Referenz eine Punktwolke ist und das ARCore SDK Ebenen aus Tiefeninformationen aus den aufgenommenen Bildern erzeugt, werden eine Vielzahl an Punkten und Ebenen verglichen.

Die größten Differenzen zwischen einzelnen Mittelwerten sind bei den berechneten Abweichungen 0,138 m und bei den gemessenen Abweichungen 0,219 m. Diese beiden Werte sind verglichen mit der Höhe der Abweichungen, vor allem die Minimal- und Maximalwerte der einzelnen Messungen, gering. Alle Vermessungspunkte sind zwar an verschiedenen Koordinaten, jedoch weisen diese ähnliche Bedingungen der Umgebung auf. Standortbestimmungen mit GNSS sind stark von der Umgebung betroffen, aufgrund der Sichtbarkeit der Satelliten. Außerdem sind für eine Korrektur mit Referenzobjekten nicht bewegte Objekte wie Gebäude oder Mauern wesentlich besser als beispielsweise Vegetation. Des Weiteren sollten Objekte aufgrund der Erzeugung eines Tiefenbildes mit der Kamera keine homogenen Flächen besitzen und sich nicht mit der Umgebung vermischen. Beispielweise sind weiße Wände für die Erzeugung eines Tiefenbildes zum Nachteil. Der Punkt G65-313 hat die meisten Bäume in unmittelbarer Nähe. Dies führt zu einem schlechteren Empfang von Satelliten sowie der Nutzung von ungenauen Referenzobjekten. Der Mittelwert von diesem Punkt beträgt 0,528 m bzw. 0,378 m. Diese Werte sind im Vergleich zu den Gesamtwerten nicht wesentlich schlechter. Der Mittelwert der gemessenen Werte ist sogar unter dem Gesamtmittelwert der gemessenen Werte. Dies könnte daran liegen, dass in unmittelbarer Nähe ein Garagenhof existiert, der ein gutes Referenzobjekt darstellt.
Die Gesamtmittelwerte von 0,473 m und 0,411 m sind eine erhebliche Genauigkeitssteigerung im Hinblick auf die Genauigkeit einer GNSS-Messung mit einem Smartphone, das wie in Kapitel 2.6 sowie in der Messung ohne das VPS von über einem Meter bis zu ca. fünf Meter angegeben ist. Abweichungen unter einem halben Meter ermöglichen neue Anwendungsszenarien wie z. B. das Auffinden von Vermessungs- oder Grenzpunkten. Auch eine Einschätzung zu möglichen baulichen Veränderungen von Gebäuden ist möglich. Aufgrund dieser Ergebnisse ist ein Einsatz der Applikation mit Genauigkeiten im Dezimeterbereich möglich.
Die Genauigkeit liegt weiterhin in der Genauigkeitsangabe des Precise Point Precision, das im Dezimeter- bis Submeterbereich liegt. Das Verfahren des Real-Time Kinematic liegt im Zentimeter- und Dezimeterbereich, damit fällt die eigene Messung ebenfalls in diesen Bereich. Jedoch wird nicht der Standort des Gerätes mit Korrekturdaten verglichen, sondern ein berechnetes Tiefenbild über Kamerabilder mit einer Punktwolke. Da diese Messtechnik eine wesentlich geringere Genauigkeit als eine Messung zu einem Referenzpunkt mit einem Tachymeter, Laserscanner oder ähnlichem Messgerät besitzt, können die Messungen nicht im einstelligen Zentimeterbereich liegen.

4.2 Beurteilung der Visualisierung mit ARCore SDK

Im Hinblick auf die Visualisierung virtueller Inhalte sind Beobachtungen festzustellen.

Die virtuellen Inhalte, u. a. Grenzpunkte und Flurstücksgrenzen, können sprunghaft verschoben werden, wenn der Standort und die AR-Session aktualisiert werden. Eine Aktualisierung ist notwendig, um eine höhere Genauigkeit aufgrund einer verbesserten Standortbestimmung zu gewährleisten. Dies kann entweder die Folge einer höheren Genauigkeit der GNSS-Standortbestimmung oder Erkennung neuer Referenzobjekte sein. Dieses Verhalten der Anwendung kann grundlegend nicht direkt verhindert werden, da es eine Folge der Standortbestimmung ist. “Smoothing” ist bereits in der Bibliothek als Funktion vorhanden, da virtuelle Inhalte zum neuen Standort nicht in einem Frame verschoben werden, sondern zum neuen Standort “gleiten”. Eine Möglichkeit ist die Verbesserung der “Smoothing”-Funktion der virtuellen Inhalte, um die Immersivität weiter zu erhöhen. Dabei ist zu beachten, dass eine solche Funktion nicht zu einer Verfälschung der Genauigkeit führen darf. Daher ist ein Kompromiss von Genauigkeit zu Immersivität zu treffen.

Die Integration der dynamischen Anzeige von virtuellen Inhalten je nach Position des Gerätes führt zu Performanceeinbrüchen. Hier ist noch Entwicklungsaufwand notwendig, wenn die Anwendung in der Produktion genutzt werden soll. Die dynamische Anzeige führt außerdem dazu, dass in wenigen Fällen zuvor gelöschte Objekte nicht erneut erzeugt und dargestellt werden. Dies könnte den Grund haben, dass das Löschen von Game Objects in Unity asynchron durchgeführt wird. Auch hier ist eine Verbesserung notwendig.

4.3 Vor- und Nachteile des Prototypen mit ARCore SDK

Diese Vorteile bietet der aktuelle Prototyp mit ARCore SDK und Geospatial API, u. a. im Vergleich zum vorigen Prototypen:

- Hohe Genauigkeit (Lagegenauigkeit bis zu einem halben Meter)
- Dynamische Anzeige der Liegenschaftsinformationen
- Aufbereiten der Daten automatisierbar
- Hohe Immersivität u. a. durch Geländeanker und Berücksichtigung des Lichteinfalls
- Individuelle Anzeige von Grenzpunkten durch unterschiedliche Genauigkeiten der Datenerhebung und Vertrauenswürdigkeit

Die Nachteile sind hingegen folgende:

- Entwickelt mit Unity Engine, das kostenpflichtige Lizenzmodelle für eine produktive Nutzung besitzt
- Entwicklung und Test bisher nur für Android
- Import der Daten über eine Datei und kein Service
- Abhängigkeit an Google Server aufgrund der verwendeten Geospatial API
- Daten von Google (Punktwolke) werden benutzt
- Mehrere gleiche Objekte werden pro Grenzpunkt erstellt
- Bounding Boxes sind bei unregelmäßigen Flurstücken fehleranfällig

4.4 Verbesserungen des Prototypen mit ARCore SDK

Verbesserungen der Nachteile

Die genannten Nachteile in Kapitel 4.3 sind durch Verbesserungen des Prototypen auszugleichen. Für diese Arbeit werden die Verbesserungen nicht durchgeführt, da diese eine noch intensivere Entwicklung benötigt, gegebenenfalls durch ein Entwicklungsteam.

Die Unity Engine wurde benutzt, da diese viele Funktionen wie das Erstellen von Game Objects erleichtert. Google ARCore mit Geospatial API kann ebenfalls als native Android App über C oder Java bzw. Kotlin entwickelt werden. Auch eine native iOS App mit Hilfe des ARKit ist mit ARCore und Geospatial API möglich. Dies umgeht die Abhängigkeit und die kostenpflichtigen Lizenzmodelle von Unity.

Die Tests wurden bisher nur an einem Smartphone, dem Google Pixel 6 und dem Betriebssystem Android, durchgeführt. Für eine weitere Entwicklung sind unterschiedliche Smartphones und Betriebssysteme zu verwenden. Dabei ist zu achten, dass für iOS eine Portierung notwendig ist, was zusätzlichen Entwicklungsaufwand erfordert.

Die Daten werden zurzeit über eine aufbereitete CSV-Datei eingelesen. Das Format CSV wird in Unity unterstützt und ist damit einfacher zu verwenden. Für eine spätere Verwendung der Applikation, um landesweite Daten bereitzustellen, ist ein Service erforderlich. Denkbar wäre hier eine JSON-API, die nach Eingabe von z. B. dem Standort oder Bounding Boxes die Daten zu Grenzpunkten und Flurstücken liefert. Dies wäre ebenfalls eine Performance-Verbesserung, da nicht eine Tabelle durchsucht wird, sondern eine Anfrage geschickt wird, die mit hoher Geschwindigkeit eine Antwort mit Daten liefert.

Aufgrund der Geospatial API werden Google Server angesprochen, um standortbasierte Funktionen oder hilfreiche Funktionen zur Verbesserung der Lagegenauigkeit zu nutzen. Eine Entwicklung einer eigenen API könnte hier die Abhängigkeit zu Google Server unterbinden. ARCore ist Open Source, daher kann der Quellcode verändert werden, um andere, u. a. eigene API's anzusprechen.

Die Geospatial API nutzt eine globale Punktwolke aus den Google Streetview Daten, um die Genauigkeit der Standortbestimmung wesentlich zu verbessern. Wenn eine eigene API entwickelt wird, dann können eigene Punktwolken oder andere 3D-Daten verwendet werden, um die Genauigkeit der Standortbestimmung zu erhöhen. Zu verwendende 3D-Daten könnten z. B. ein hochauflösendes Digitales Oberflächenmodell oder Gebäude aus dem Liegenschaftskataster sein.

Derzeit werden mehrere Objekte pro Grenzpunkt erstellt und dargestellt, um Flurstücke zusammenhängend darzustellen. Dies bedeutet, dass, wenn ein Grenzpunkt an drei Flurstücke grenzt, dann drei Objekte für diesen Grenzpunkt existieren, eins je Flurstück. Dies verringert die Performance sehr stark und sollte bei einer weiteren Entwicklung verbessert werden, um die Performance zu erhöhen.

Die berechneten Bounding Boxes sind zwar eine sehr performante Möglichkeit, um für die dynamische Anzeige von Liegenschaftsinformationen die Koordinaten abzugleichen. Jedoch kann dies bei unregelmäßigen Flurstücken wie langgezogenen Straßenflurstücken zu sehr großen Bounding Boxes, relativ zur Flurstücksgröße, führen. Hier sollte eine neue Möglichkeit, gegebenenfalls zusammen mit der eigenen API, entwickelt werden.

Weitere Verbesserungen und Erweiterungen

Die ermittelte durchschnittliche Abweichung von 4 bis 5 Dezimetern kann durch mehrere Ansätze weiter verbessert werden. Zum Beispiel durch den Ausbau der Messung durch Real Time Kinematic. Bisher werden nur durch die Kamera Referenzobjekte als Punktwolke ermittelt und verglichen. Eine Möglichkeit für weitere Referenzobjekte sind Vermessungspunkte, die aufgrund präziser und zuverlässiger Messungen ein Koordinatenpaar mit hoher Genauigkeit besitzen. Ist ein Vermessungs- oder Grenzpunkt in der Örtlichkeit eindeutig zu identifizieren, könnte der virtuelle Marker des Punktes auf den tatsächlichen Punkt in der Örtlichkeit in der Anwendung intuitiv verschoben werden. Dies wäre auch mit Gebäudeeckpunkten möglich, jedoch weisen diese eine schlechtere Genauigkeit als Vermessungspunkte auf.
Auch die Verwendung von SAPOS kann die Genauigkeit der Standortbestimmung ohne zusätzlichen Aufwand für den Nutzenden verbessern. Es existieren vermehrt SAPOS-Referenzstationen in ganz Deutschland, die durchgehend möglichst viele Satelliten empfangen und der Dienst liefert Korrekturdaten über Satellitenbahnen. Die Korrektur durch diese Referenzstationen kann in der Anwendung im Hintergrund implementiert werden.
Die Punktwolke von Google, die derzeit als Referenzobjekte herangezogen wird, kann durch eigene Daten ersetzt werden. Vor allem Gebäudedaten des Liegenschaftskatasters können eine höhere Genauigkeit aufweisen als die berechneten Gebäudedaten von Google, die durch Google Streetview ermittelt sind. Durch Referenzobjekte mit höherer Genauigkeit können auch höhere Genauigkeiten der Standortbestimmung realisiert werden.
Das Kalibirieren des Chips für die GNSS-Messung sowie des Kompass erhöhen ebenfalls die Genauigkeit der Standortbestimmung. Jedoch kann das Kalibirieren des GNSS-Chips nicht von allen Nutzenden gefordert werden, wenn der Nutzerkreis besonders hoch ist. Ist der Nutzerkreis gering und wird ein hoher Wert auf Genauigkeit gelegt, könnte diese Maßnahme überlegt werden.
Es ist zu prüfen, ob das Digitale Oberflächenmodell genauere Daten beinhaltet als die Punktwolke von Google. Es existieren Gebiete ohne Google Streetview, dort ist die Verwendung des Digitalen Oberflächenmodells als Referenz eine erhebliche Genauigkeitssteigerung.

Für die Möglichkeit der individuellen Darstellung von Grenzpunkten wurden die Datenerhebung und Vertrauenswürdigkeit verwendet, um Grenzpunkte in verschiedenen Formen und Farben darzustellen. Weitere Informationen, wie z. B. zur Punktnummer oder Abmarkungsart mit Tiefe bzw. Höhe, könnten entnommen werden, wenn die Eingangsdaten dies ermöglichen. Die vorliegenden Daten besitzen keine Verknüpfung der Punktnummer zum Punktobjekt, daher können diese derzeit nicht angezeigt werden.

Andere geografische Daten können zusätzlich visualisiert werden, um die Darstellung zu erweitern. Gebäude können hinzugefügt werden, was z. B. im Gebäudefeldvergleich sehr vorteilhaft wäre. Zum Auffinden von z. B. unterirdischen oder schlecht sichtbaren Vermessungspunkten können diese Informationen hinzugefügt werden, um das Auffinden zu erleichtern. Beide genannten Beispiele sind für eine Nutzung zur Hilfe bei Vermessungsarbeiten vorgesehen. Je nach Nutzungsszenario können die Geodaten sowie deren Darstellung gewählt werden. Falls die Anwendung zur Anzeige von Liegenschaftsdaten nach Grenzfeststellungen verwendet wird, dann kann die Darstellung auf die betroffenen Flurstücke bzw. Flurstücksgrenzen begrenzt werden oder mit besonderer Visualisierung hervorgehoben werden.

Die Visualisierung ist bisher so gewählt, dass Grenzpunkte nach der Datenerhebung und der Vertrauenswürdigkeit dargestellt werden, um zu zeigen, dass Attribute aus dem Liegenschaftskataster die Visualisierung verändern können. Grenzpunkte ohne dieses Attribut, meistens nicht abgemarkte Grenzpunkte ohne Vermessung, sowie die Flurstücksgrenzen werden in Grau dargestellt. Diese Visualisierung sowie

weitere Objekte können verbessert werden und auf das Anwendungsszenario angepasst werden. Für den vermessungstechnischen Außendienst können die Informationen zur Datenerhebung bleiben sowie weitere Punkte wie Vermessungspunkte und deren Vermarkung hinzugefügt werden. Bei einer Visualierung für den Eigentümer nach einer Grenzfeststellung könnten die betroffenen Flurstücke hervorgehoben werden.

Flurstücksinformationen werden in der Mitte der Bounding Box platziert. Dies bedeutet, dass Flurstücke in unregelmäßiger Form, z. B. Straßenflurstücke, dazu führen können, dass die Flurstücksinformationen außerhalb des Flurstücks angezeigt werden. Hier müssen entweder die Signaturkoordinaten der Flurstücksnummer aus ALKIS verwendet werden oder eine Methode entwickelt werden, um die Flurstücksinformationen zwingend innerhalb des Flurstücks anzuzeigen.

Die Abweichung der Standortbestimmung kann stark variieren je nach Erkennung der Umgebung, der GNSS-Messung und gegebenen Referenzdaten. Daher sollte eine Darstellung entwickelt werden, die deutlich zeigt, ob die aktuelle Abweichung gut oder schlecht ist. Zum Beispiel wie eine Ampel mit den Farben Grün, Gelb und Rot. Außerdem können Hinweise und Informationen bei verschiedenen Genauigkeiten angezeigt werden, um den Nutzenden zu helfen. Ein Beispiel ist, dass bei schlechter Genauigkeit vorgeschlagen wird, dass die Kamera um 360° geschwenkt werden soll, damit möglichst alle Referenzdaten der Umgebung genutzt werden können. Bei sehr schlechter Genauigkeit, mit z. B. einer Anzeige eines Warndreiecks, kann die Information gegeben werden, dass möglicherweise keine Referenzdaten vorhanden sind.

Die Benutzeroberfläche des Prototypen ist für die Entwicklung entsprechend konfiguriert. Je nach Nutzung der Anwendung sollte die Benutzeroberfläche geeignete Informationen und Schaltflächen beinhalten.

4.5 Anwendungsbeispiele

Die festgestellte Abweichung von unter einem halben Meter sowie der Hinblick auf mögliche Genauigkeitsverbesserungen ermöglichen die Betrachtung neuer Anwendungsbeispiele für eine produktive Nutzung der Anwendung.
Die genannten Beispiele in Abbildung 13 dienen zur Übersicht von möglichen Anwendungsbeispielen. Hier sind die Themen Liegenschaftskataster und Vermessung, öffentliche Anwendung, Landentwicklung, Stadtplanung, Immobilienmanagement und Wertermittlung, sowie sonstige Anwendungen aufgelistet.

Im Liegenschaftskataster und zur Vermessung kann eine AR-Anwendung unterstützen und neue Anwendungsfelder ermöglichen. Die Visualisierung von Flurstücksgrenzen in AR kann nach einer Zerlegung zur Grenzanzeige oder im Nachhinein zu einem besseren Verständnis des Flurstücksverlaufs für die Eigentümerin oder den Eigentümer führen. Um Vermarkungen von Vermessungspunkten oder Abmarkungen von Grenzpunkten schnell und einfach zu finden, kann die AR-Anwendung genutzt werden. Aufgrund der festgestellten Genauigkeit liegt in einem Umkreis bis zu einem halben Meter des virtuellen Markers die Vermarkung oder Abmarkung. Auch beim Setzen von unterirdischen Vermarkungen und Abmarkungen kann eine solche Anwendung hilfreich sein, um im Zusammenhang mit einem Leitungskataster mögliche unterirdische Leitungen in der Nähe aufzudecken. Werden Geodaten mit niedrigen Genauigkeiten entsprechend der Genauigkeit der AR-Anwendung geführt, kann die Anwendung auch zur Erfassung von Geodaten dienen. Wenn die integrierten Liegenschaftsdaten Gebäude enthalten, können vor Ort in der Realität die eingetragenen Gebäude des Liegenschaftskatasters mit denen der Realität verglichen werden, um neue Gebäude oder Gebäudeerweiterungen zu erkennen.

Abbildung 13: Mindmap zu Anwendungsbeispielen für AR-Anwendungen mit Liegenschaftsdaten

Als öffentliche Anwendung kann eine AR-Anwendung vor allem den Bürgerinnen und Bürgern zur Verfügung stehen. Beispiele sind eine immersive Stadtkarte, um Informationen über Point of Interests oder andere nützliche Geodaten einer Kommune zu erhalten, oder Crowdsourcing, um Daten mit hoher Quantität durch die Bevölkerung zu generieren. Dabei sind z. B. die zuvor genannten Erfassungen von Geodaten niedriger Genauigkeit zu nennen. OpenStreetMap besteht aus Geodaten, die durch eine Community eingetragen werden. Solche Erfassungen können direkt, intuitiv und einfach in der AR-Anwendung erfolgen.

In der Landentwicklung, vor allem bei Durchführung von Flurbereinigungen, kann die Anwendung Entwürfe sowie festgestellte Pläne in der Örtlichkeit als virtuelle Inhalte integrieren. Die Flurstücke sind in der Landwirtschaft oft sehr groß und werden in einer Flurbereinigung meist vollständig verändert. Daher bestehen kaum Anhaltspunkte zur Lokalisierung von zukünftigen Flurstücksverläufen vor der Zuteilung von Flurstücken vor Ort. Die Informationen liegen nur in einem Entwurf als Karte vor. Hier unterstützt die AR-Anwendung die Visualisierung der neuzugeteilten Flurstücke sowie des Wege- und Gewässerplanes. Einerseits ist dies hilfreich für Mitarbeiter in der Planung, andererseits für Beteiligte der Flurbereinigung wie z. B. die Eigentümerinnen und Eigentümer der betroffenen Flurstücke.

In der Stadtplanung ähneln die Funktionen der Anwendung denen von der Landentwicklung. Planungen der Bauleitpläne sowie von Umlegungsverfahren können zum einen im Planungsstand und zum anderen im festgestellten Stand visualisiert werden. Festgestellte Bauleitpläne sind öffentlich zugänglich und können daher auch Bestandteil einer öffentlichen Anwendung sein. Baugrenzen und weitere Informationen aus Bebauungsplänen können als virtuelle Inhalte in der Anwendung zu einem besseren räumlichen Verständnis führen, an welcher Stelle eine Bebauung getätigt werden könnte. Geplante Bauvorhaben können innerhalb der Baugrenzen visualisiert werden, um die Ausmaße der Flurstücke zusammen mit der zukünftigen Bebauung immersiv zu betrachten. Entfernungen wie z. B. der Abstand zwischen einer Flurstücksgrenze und einer Gebäudeseite können in AR nicht nur besser visualisiert, sondern auch in der erweiterten Realität angezeigt werden.

Im Immobilienmanagement und der Wertermittlung kann die AR-Anwendung ebenfalls als Unterstützung herangezogen werden. Bei einem Immobilienkauf können Kaufinteressenten den Flurstücksverlauf, vor allem bei großen Flurstücken, zur Immobilienbegehung mit der AR-Anwendung ansehen. Verkäufer können zusätzlich Informationen mit Raumbezug bereitstellen, z. B. wenn die Immobilie mehrere Gebäude beinhaltet. Zur Erstellung von Verkehrswertgutachten können von dem Objekt Bilder mit Informationen exportiert werden. Auch wäre es möglich, Informationen zu z. B. Außenwänden oder Dachformen intuitiv mit dem integrierten Gebäudemodell zu erfassen. Sind Gebäude oder Gebäudeteile nicht im Liegenschaftskataster eingetragen, können diese ebenfalls mit der Anwendung und integrierten Gebäudedaten erkannt werden.

Es sind noch viele weitere Anwendungsbeispiele möglich, die hier nicht aufgeführt werden. Zum Beipiel ist dies das bereits genannte Leitungskataster, um ungefähr abzuschätzen, wo unterhalb des Bodens Leitungen platziert sind. Dies ist nicht nur bei Vermessungsarbeiten hilfreich, sondern auch zur Planung und Durchführung von Maßnahmen zur Erschließung von Grundstücken. Viele weitere raumbezogene Informationen können in solch eine Anwendung integriert werden. Z. B. Informationen zur Bodenschätzung können Auskunft über die Bewertung landwirtschaftlicher Grundstücke geben, unter anderem hilfreich bei einem Kauf oder Verkauf von landwirtschaftlichen Flächen.

5 Ausblick

Für eine produktive Nutzung des Prototypen ist eine weitere Entwicklung notwendig. Die einzelnen Punkte zur Verbesserung des Prototypen sind in Kapitel 4.4 beschrieben. Werden diese Verbesserungen umgesetzt, ist der Prototyp nahe einem produktiven Einsatz.

In der Diskussion im Kapitel 4.1 wird erwähnt, dass mit ARCore SDK Abweichungen von unter einem halben Meter erreicht werden, was eine erhebliche Steigerung der Genauigkeit einer GNSS-Standortbestimmung mit einem Smartphone darstellt. Hiermit sind bereits eine Vielzahl neuer Anwendungsszenarien möglich. Zukünftig ist zu erwarten, dass die Genauigkeit weiter verbessert wird, z. B. durch die Technik in Smartphones oder Quantität und Qualität an Referenzdaten. Damit wären viele weitere Anwendungsszenarien, zusätzlich zu den genannten Beispielen in Kapitel 4.5, möglich. Als Beispiel könnten weitere Vermessungen mit der Anwendung durchgeführt werden, wenn die Genauigkeit ausreicht. Auch das Ermitteln der Position zum Graben eines Loches zur Freilegung unterirdischer Vermarkungen und Abmarkungen kann bei hoher Genauigkeit vollständig durch die Anwendung realisiert werden.

Nicht nur die zukünftige Verbesserung der GNSS-Messung kann die Genauigkeit der Standortbestimmung verbessern, sondern auch die Verbesserung von AR-Technologien oder ARCore SDK. Auf Veränderungen der Technologien oder des SDK muss die Anwendung kontinuierlich angepasst werden.

Im Zuge der fortschreitenden Digitalisierung können viele Informationen bereits digital abgerufen werden und zukünftig weitere Informationen. Die Anwendung kann auf einer Plattform integriert werden, sodass Nutzerinnen und Nutzer einen einfachen Zugriff auf diese Anwendung besitzen. Ob die Anwendung für den vermessungstechnischen Außendienst, für Kundinnen und Kunden oder auch die gesamte Bevölkerung ist, spielt hier keine Rolle, da der Plattformgedanke jeden Nutzerkreis einschließen oder beschränken kann.

In einer produktiven Entwicklung ist der Austausch mit Stakeholdern, z. B. die zukünftigen Nutzerinnen und Nutzer der Anwendung, wichtig. Jedes Feedback kann diskutiert und eingearbeitet werden, um die Anwendung zu verbessern. Es ist vorteilhaft, die Entwicklung in ein Entwicklungsteam zu tragen, um die Vielzahl an individuellen Kompetenzen zu nutzen. Zum Beispiel ein Team mit agilen Arbeitsmethoden, um regelmäßig Ergebnisse zu erzielen sowie kurzfristig auf Feedback einzugehen.

Zukünftig könnten Themen bezüglich der Sicherheit und des Datenschutzes schärfer reguliert werden. Auf diese Veränderungen muss die Anwendung angepasst werden. Es könnte der Zugriff oder die Verwendung der Kamera oder des Standortes unter besonderem Schutz stehen.

Barrierefreiheit ist aktuell bereits ein wichtiges Thema in der Gesellschaft und kann in Zukunft noch stärker im Fokus sein. Daher kann es vorkommen, dass die Anwendung in der Zukunft barrierefrei sein muss. Wie dies mit einer AR-Anwendung möglich ist, muss zukünftig erarbeitet werden.

Es ist kontinuierlich zu beobachten, wie andere AR-Anwendungen durch neue Technologien Fortschritte erzielen. An diesen Fortschritten kann auch die Anwendung dieser Arbeit profitieren.

6 Fazit

Wie können Geodaten mit einem globalen Raumbezug in eine AR-Anwendung integriert werden?

Es existieren mehrere Möglichkeiten, Geodaten in einer AR-Anwendung zu integrieren. In dieser Arbeit wurden die Bibliothek AR.js und das SDK ARCore SDK untersucht. In beiden Fällen können Funktionen zur Integration eigener Daten entwickelt und implementiert werden. Viele Datenformate von Geodaten sind möglich, wenn die Bibliothek oder das SDK diese Formate unterstützt. Die AR-Anwendung muss standortbestimmte Funktionen besitzen, um die eigene Position zu berechnen. Ohne die Bestimmung der Position des Gerätes können keine Geodaten mit Raumbezug an den richtigen Koordinaten visualisiert werden. Die Untersuchungen mit AR.js haben zu einem nicht zufriedenstellenden Ergebnis geführt, da die Geodaten mit der Realität aufgrund großer Abweichung nicht verglichen werden konnten. Mit ARCore SDK konnten Genauigkeiten und eine Immersivität erzielt werden, die es ermöglichen, Geodaten als virtuelle Inhalte in der Art einzufügen, dass diese in der Realität wieder zu erkennen sind. Auch Metadaten können zur unterschiedlichen Visualisierung von virtuellen Inhalten oder als Information in Textform integriert werden. Die Abweichung von virtuellen Markern zu Vermessungspunkten in der Realität konnte mit durchschnittlich 4 bis 5 Dezimetern ermittelt werden.

Für welche Anwendungsfälle ist die ermittelte Lagegenauigkeit mit einer Visualisierung von Liegenschaftsinformationen ausreichend?

Aufgrund der relativ hohen Genauigkeit der Standortbestimmung von unter 5 Dezimeter im Vergleich zu einer GNSS-Messung mit einem Smartphone ohne Referenzdaten können weitere Anwendungsfälle untersucht werden. In verschiedenen Themenfeldern könnte diese AR-Anwendung eine Nutzung erbringen, z. B. Liegenschaftskataster und Vermessung, öffentliche Anwendung, Landentwicklung, Stadtplanung, Immobilienmanagement und Wertermittlung. Konkrete mögliche Anwendungsfälle sind die Unterstützung zum Auffinden von Vermarkungen oder Abmarkungen, Einbindung des Leitungskatasters oder der Bodenschätzung, intuitives Eintragen von Geodaten niedriger Genauigkeitsanforderungen zusammen mit Crowdsourcing, eine immersive Stadtkarte, Visualisierung von Planungen wie Flurbereinigungen, Umlegungen, Bauleitplänen oder Bauvorhaben, Unterstützung beim Immobilien(ver)kauf und die Erstellung von Verkehrswertgutachten.

Je höher die Genauigkeit der Standortbestimmung ist, desto höher ist die Anzahl von möglichen Anwendungsfällen. Daher ist zu erwarten, dass bei einer produktiven Entwicklung der AR-Anwendung Genauigkeitsverbesserungen durchgeführt werden und damit weitere Anwendungsfälle untersucht werden können. Neben der Erweiterung der Realität kann eine AR-Anwendung intuitive Funktionen beinhalten, um mit Objekten der Realität und virtuellen Inhalten zu interagieren. Dies schafft neue Anwendungsfälle, die über eine reine Visualisierungskomponente hinausgehen. Aber auch die Visualisierungskomponente bietet aufgrund der Immersivität von AR neue Möglichkeiten.

Welche Methoden zur Verbesserung der Lagegenauigkeit sind in einer AR-Anwendung möglich?

Im Prototyp dieser Arbeit wird mit ARCore SDK eine Punktwolke, die dem Digitalen Oberflächenmodell ähnelt, als Referenz verwendet, um die GNSS-Standortbestimmung mit Korrekturdaten zu verbessern. Weiterhin ist es möglich, weitere Referenzdaten zu verwenden, z. B. Gebäudedaten aus dem Liegenschaftskataster, Vermessungspunkte oder das Digitale Oberflächenmodell. Auch die Verwendung von SAPOS kann die Genauigkeit erhöhen. Das Kalibrieren des Antennenchipsets für die GNSS-Messung kann die Genauigkeit erhöhen, jedoch ist dies für jedes verwendete Gerät notwendig. Ebenfalls das Nutzen von höherwertigen Antennenchipsets mit geringerer Streuung erhöht die Genauigkeit.

Literatur

Acharya, Durga Prasad und Usha Romesh (Nov. 2023). *9 Best Augmented Reality SDKs to Build Creative Apps*. Zuletzt zugegriffen im November 2023. URL: https://geekflare.com/best-augmented-reality-sdk/.

Apple Inc. (Okt. 2020). *Apple stellt iPhone 12 Pro und iPhone 12 Pro Max mit 5G vor*. Zuletzt zugegriffen im Juli 2023. URL: https://www.apple.com/de/newsroom/2020/10/apple-introduces-iphone-12-pro-and-iphone-12-pro-max-with-5g/.

— (Apr. 2022). *iPhone 12 Pro - Technische Daten*. Zuletzt zugegriffen im Juli 2023. URL: https://support.apple.com/kb/SP831?locale=de_DE.

— (2023). *Apple ARKit*. Zuletzt zugegriffen im Mai 2023. URL: https://developer.apple.com/augmented-reality/arkit/.

St-Aubin, B. u. a. (2010). „A 3D Collaborative Geospatial Augmented Reality System For Urban Design And Planning Purposes“. In: URL: https://api.semanticscholar.org/CorpusID:18120948.

St-Aubin, Bruno, Mir Abolfazl Mostafavi und Stéphane Roche (Juni 2012). „Development of a collaborative geospatial augmented reality system in support of urban design practice“. In: *Revue internationale de géomatique* 22, S. 307–330. DOI: 10.3166/rig.22.307-330.

Biene, R. u. a. (2021). *Augmented und Virtual Reality*. Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.

Buyuksalih, I. u. a. (Okt. 2017). „3D Modelling and Visualization Based on the Unity Game Engine - Advantages and Challenges“. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-4/W4, S. 161–166.

Carpignoli, N. und AR.js Org Community (2023). *AR.js Documentation*. Zuletzt zugegriffen im Juli 2023. URL: https://ar-js-org.github.io/AR.js-Docs/.

Chen, Yunqiang u. a. (Juni 2019). „An overview of augmented reality technology“. In: *Journal of Physics: Conference Series* 1237, S. 022082. DOI: 10.1088/1742-6596/1237/2/022082.

Chung, C. O., Y. He und H. K. Jung (Feb. 2016). „Augmented Reality Navigation System on Android“. In: *International Journal of Electrical and Computer Engineering (IJECE)* 6.1, S. 406 412.

Community, QGIS (2023). *QGIS Documentation*. Zuletzt zugegriffen im September 2023. URL: https://www.qgis.org/en/docs/index.html.

GIS Geography (2023). *Augmented Reality Applications in GIS*. Zuletzt zugegriffen im Mai 2023. URL: https://gisgeography.com/augmented-reality-applications-gis/.

GitHub, Inc. (2023). *GitHub Repositories*. Zuletzt zugegriffen im Dezember 2023. URL: https://github.com/.

Google (2023a). *AR Core Documentation*. Zuletzt zugegriffen im September 2023. URL: https://developers.google.com/ar/develop.

— (2023b). *Google API-Bibliothek*. Zuletzt zugegriffen im Dezember 2023. URL: https://console.cloud.google.com/apis/library.

— (2023c). *Google Pixel - technische Daten zur Hardware*. Zuletzt zugegriffen im Juli 2023. URL: https://support.google.com/pixelphone/answer/7158570?hl=de#zippy=%2Cpixel.

Heßelbarth, Anja und Lambert Wanninger (2020). „Towards centimeter accurate positioning with smartphones“. In: *2020 European Navigation Conference (ENC)*, S. 1–8. DOI: 10.23919/ENC48637.2020.9317392.

Hussain, Afzal u. a. (Okt. 2020). „Unity Game Development Engine: A Technical Survey“. In: *University of Sindh Journal of Information and Communication Technology* 4.

Inman, Rachel (Aug. 2019). *Take off to your next destination with Google Maps*. Zuletzt zugegriffen im September 2023. URL: https://blog.google/products/maps/take-your-next-destination-google-maps/.

Kudan Inc. (2023). *Kudan Technology*. Zuletzt zugegriffen im Mai 2023. URL: https://www.kudan.io/our_technology/.

MAXST Co., Ltd. (2023). *MAXST Developer*. Zuletzt zugegriffen im Mai 2023. URL: https://developer.maxst.com/.

Megha, Pa., L. Nachammai und T.M. Senthil Ganesan (2018). „3D Game Development Using Unity Game Engine“. In: *SSRG International Journal of Computer Science and Engineering* 5.4, S. 15–18. DOI: 10.1186/2213-7459-1-2.

Niantic, Inc. (2023). *Lightship ARDK*. Zuletzt zugegriffen im Mai 2023. URL: https://lightship.dev/products/ardk?hl=en.

Osterloh, Rick (Aug. 2021). *Neu in Pixel 6 diesen Herbst: Google Tensor*. Zuletzt zugegriffen im Juli 2023. URL: https://blog.google/intl/de-de/produkte/hardware/neu-pixel-6-diesen-herbst-google-tensor/.

Paziewski, Jacek (Juni 2020). „Recent advances and perspectives for positioning and applications with smartphone GNSS observations“. In: *Measurement Science and Technology* 31.9, S. 091001. DOI: 10.1088/1361-6501/ab8a7d. URL: https://dx.doi.org/10.1088/1361-6501/ab8a7d.

Porter, M. E. und J. E. Heppelmann (Nov. 2017). *How Does Augmented Reality Work?* Zuletzt zugegriffen im Mai 2023. URL: https://hbr.org/2017/11/how-does-augmented-reality-work.

PTC, Inc. (2023). *Vuforia Engine Dokumentation*. Zuletzt zugegriffen im Mai 2023. URL: https://developer.vuforia.com/library/getting-started/vuforia-features.

Ratajczak, Julia, Michael Riedl und Dominik T. Matt (2019). „BIM-based and AR Application Combined with Location-Based Management System for the Improvement of the Construction Performance“. In: *Buildings* 9.5. ISSN: 2075-5309. DOI: 10.3390/buildings9050118. URL: https://www.mdpi.com/2075-5309/9/5/118.

Sadeghi-Niaraki, A. und S. Choi (2020). „A Survey of Marker-Less Tracking and Registration Techniques for Health - Environmental Applications to Augmented Reality and Ubiquitous Geospatial Information Systems“. In: *Sensors* 20.10.

SAP SE (2024). *SAP Help Portal (Documentation)*. Zuletzt zugegriffen im Februar 2024. URL: https://help.sap.com/docs/.

Su, Xing u. a. (Dez. 2013). „Uncertainty-aware visualization and proximity monitoring in urban excavation: a geospatial augmented reality approach“. In: *Visualization in Engineering* 1. DOI: 10.1186/2213-7459-1-2.

Talmaki, Sanat, Suyang Dong und Vineet Kamat (Mai 2010). „Geospatial Databases and Augmented Reality Visualization for Improving Safety in Urban Excavation Operations“. In: DOI: 10.1061/41109(373)10.

Unity Technologies (2023a). *FAQ Unity Reflect*. Zuletzt zugegriffen im Mai 2023. URL: https://unity.com/products/unity-reflect#frequently-asked-questions--2.

— (2023b). *Unity Engine*. Zuletzt zugegriffen im Dezember 2023. URL: https://unity.com/de/products/unity-engine.

VisionStar Information Technology (Shanghai) Co., Ltd. (2023). *EasyAR Product Overview*. Zuletzt zugegriffen im Mai 2023. URL: https://www.easyar.com/price.html.

Wikitude GmbH (2023). *Wikitude Plans and pricing*. Zuletzt zugegriffen im Mai 2023. URL: https://www.wikitude.com/store/.

Yeeply Mobile, S.L. (2023). *Top 6 Frameworks and Augmented Reality SDKs for Android and iOS*. Zuletzt zugegriffen im Mai 2023. URL: https://www.yeeply.com/en/blog/frameworks-sdks-augmented-reality-app/.

Zangenehnejad, Farzaneh und Yang Gao (Dez. 2021). „GNSS smartphones positioning: advances, challenges, opportunities, and future perspectives“. In: *Satellite Navigation* 2. DOI: 10.1186/s43020-021-00054-y.

Zappar Ltd. (2023). *Zapworks Plans*. Zuletzt zugegriffen im Mai 2023. URL: https://zap.works/pricing/.

A Anhang

A.1 Diagramme der Messdaten des AR.js Prototypen

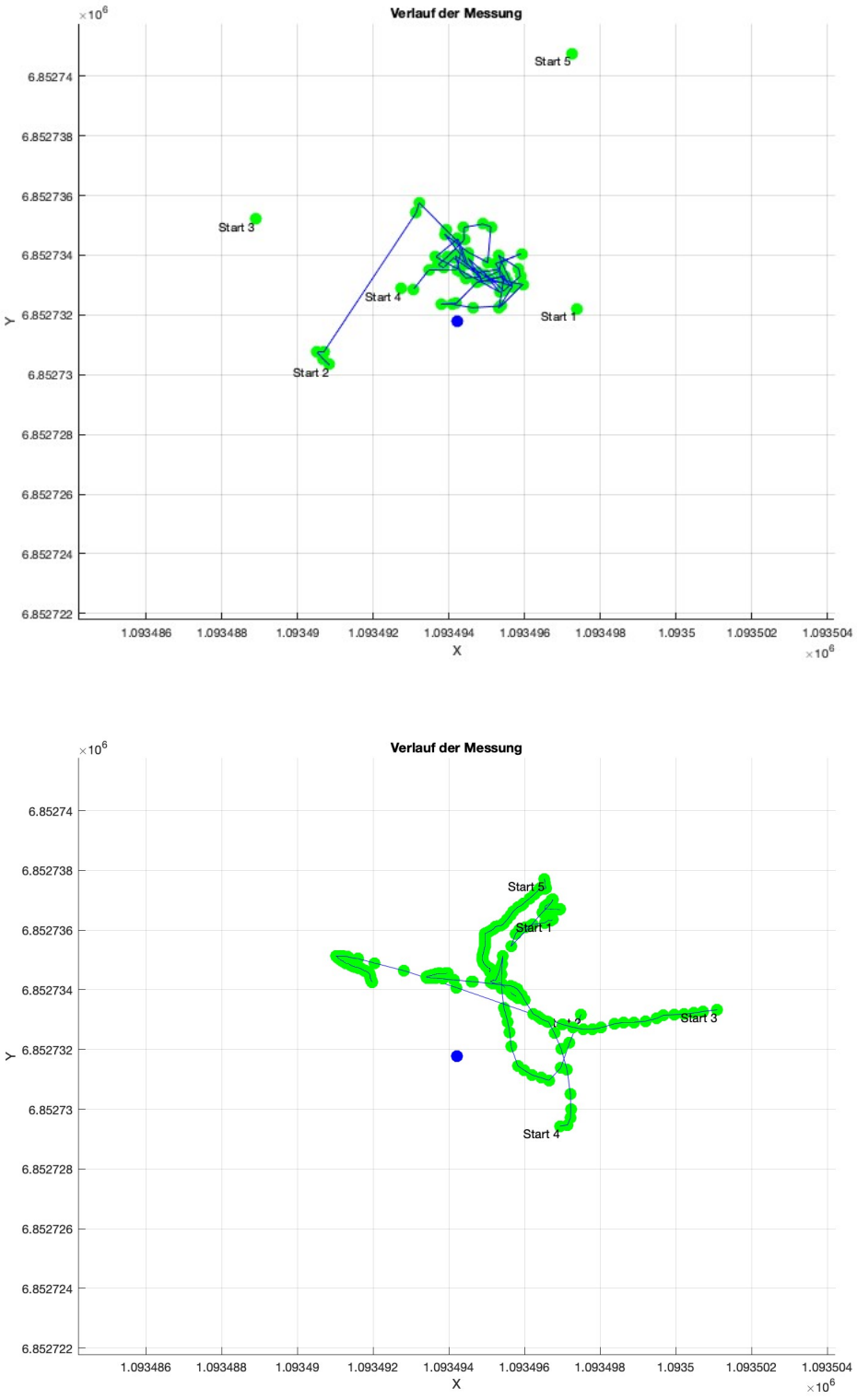

Abbildung 14: AP210 - o. Apple iPhone 12 Pro, u. Google Pixel 6

(Einheiten der Koordinatenachsen in $\cdot 10^6 m$)

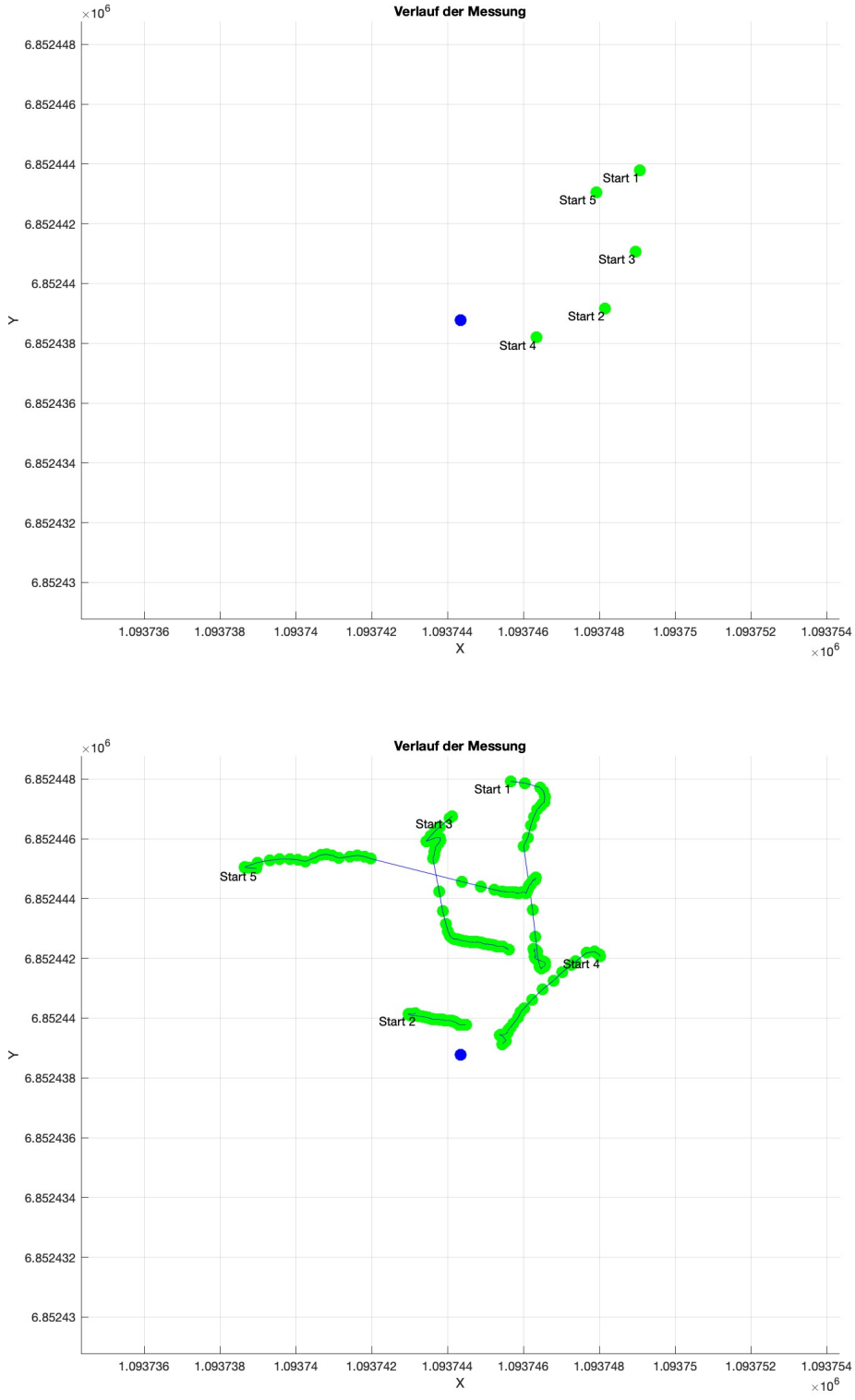

Abbildung 15: SP203 - o. Apple iPhone 12 Pro, u. Google Pixel 6

(Einheiten der Koordinatenachsen in $\cdot 10^6 m$)

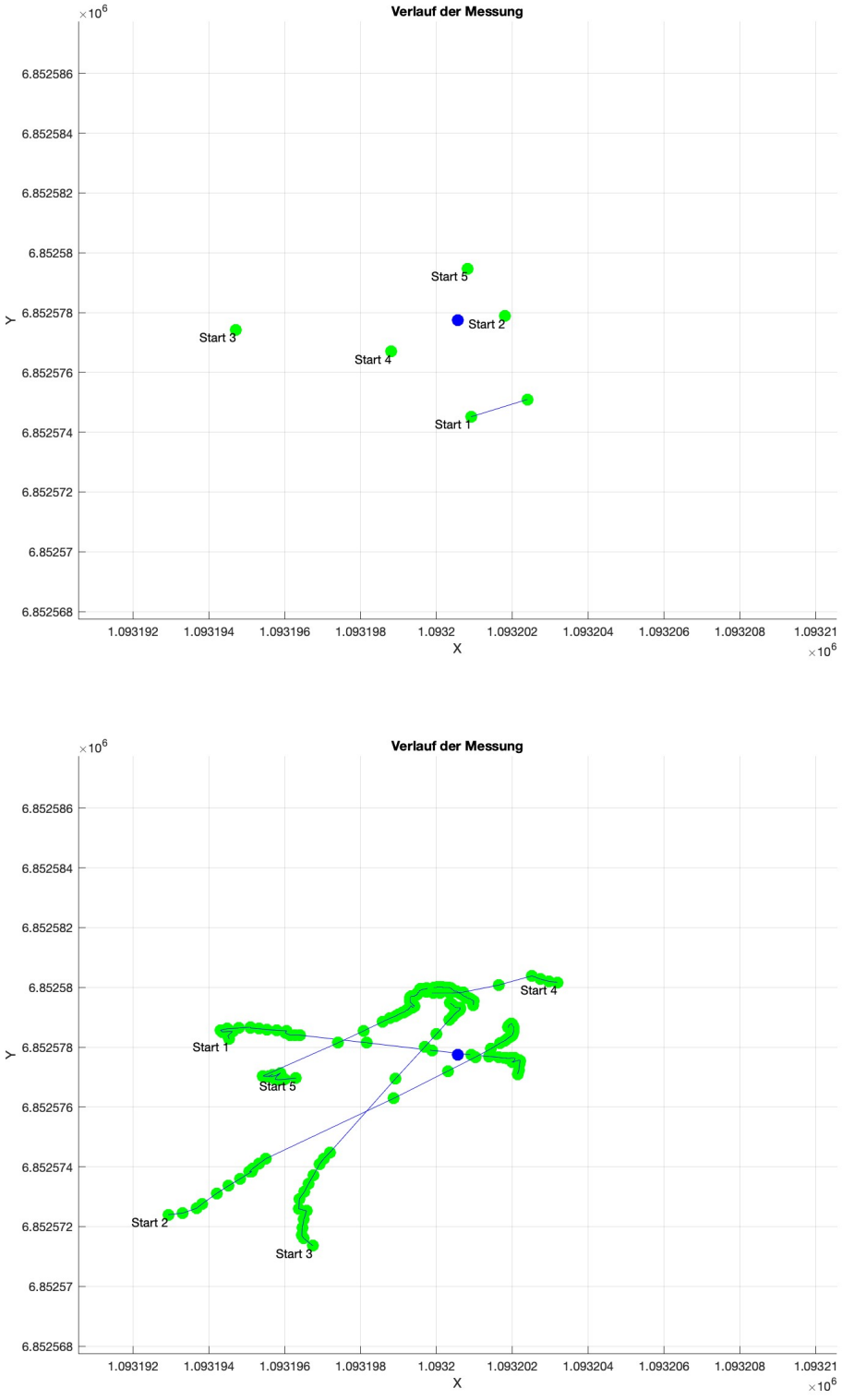

Abbildung 16: SP233 - o. Apple iPhone 12 Pro, u. Google Pixel 6

(Einheiten der Koordinatenachsen in $\cdot 10^6 m$)

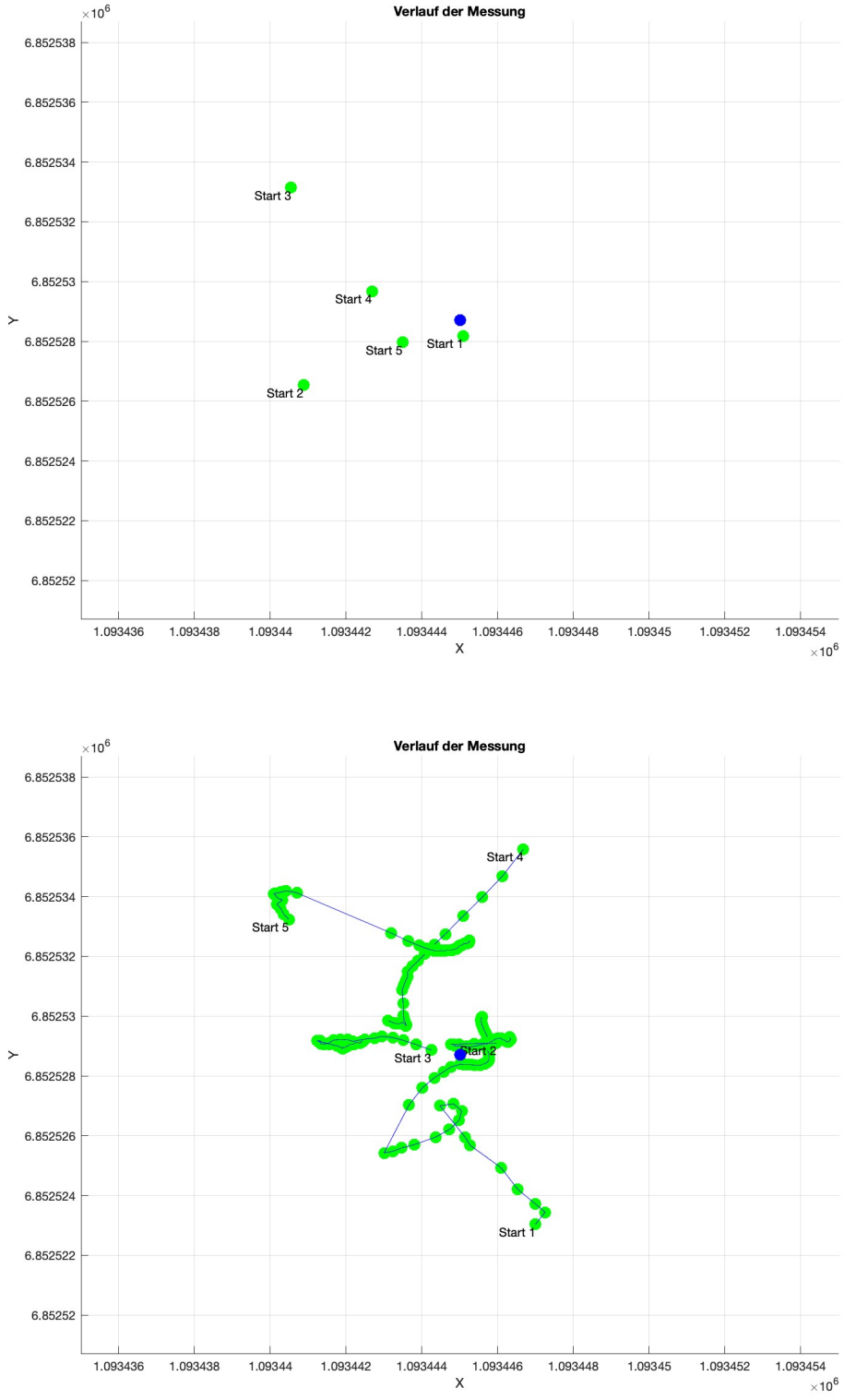

Abbildung 17: SP301 - o. Apple iPhone 12 Pro, u. Google Pixel 6

(Einheiten der Koordinatenachsen in $\cdot 10^6 m$)

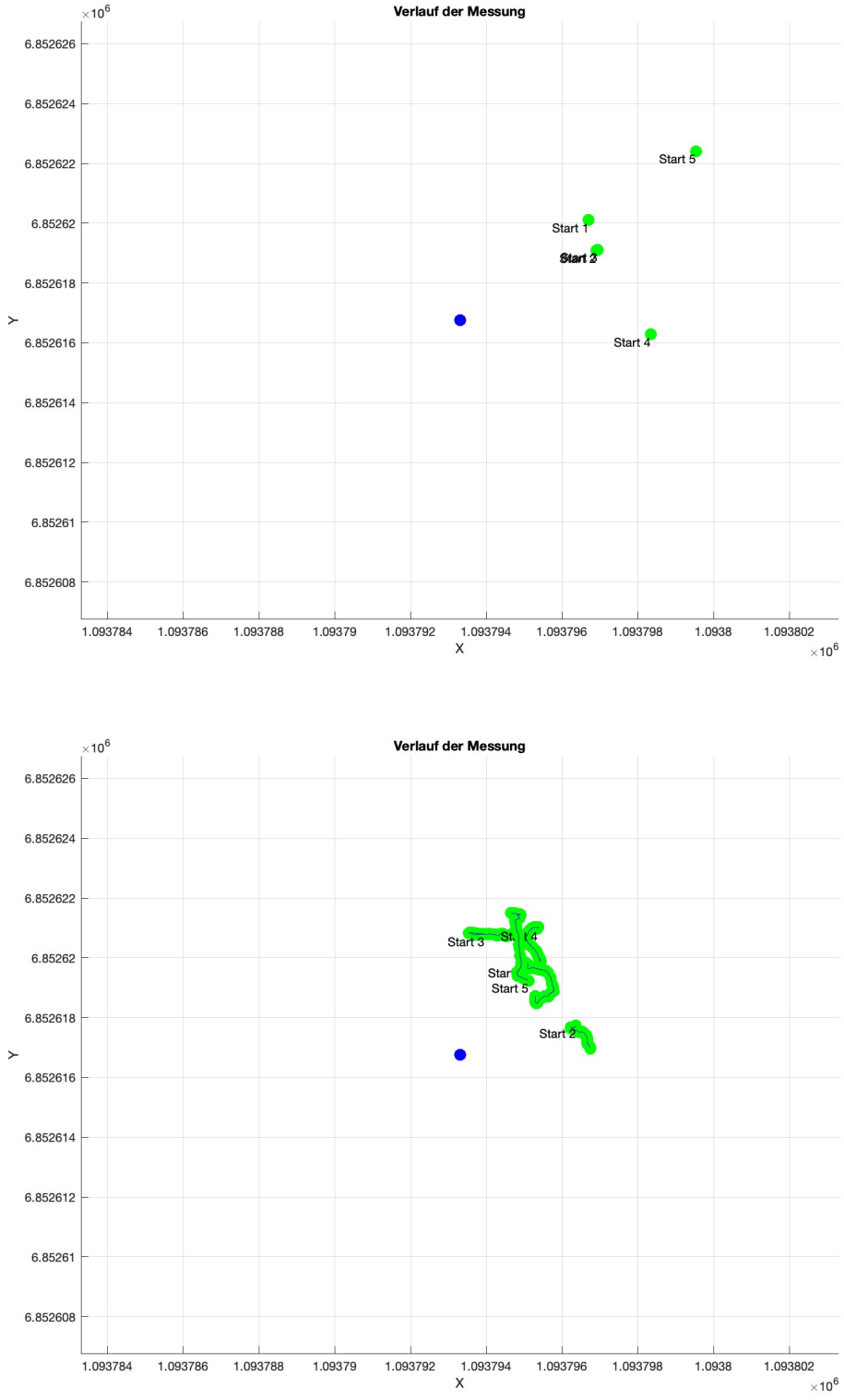

Abbildung 18: SP313 - o. Apple iPhone 12 Pro, u. Google Pixel 6

(Einheiten der Koordinatenachsen in $\cdot 10^6 m$)

A.2 Messdaten des ARCore SDK Prototypen

Abbildung 19: Messungen mit ARCore SDK mit Google Pixel 6

Messdaten mit AR-Core Prototyp

Punkt	Messreihe	Messung	Zeit [hh:mm]	Abw Berechne [m]	Abw Gemesse [m]	Ber – Gem [m]	Mittelwert Ber [m]	Median Ber [m]	Mittelwert Ge [m]	Median Gem [m]	Mittelwerte B [m]	Median Ber-G [m]
63-301	1 (22.12.23)	1	14:30	0,536	0,554	-0,018						
		2	14:40	0,336	0,784	-0,448						
		3	14:41	0,402	0,269	0,133						
	2 (27.12.23)	1	14:50	0,359	0,607	-0,248						
		2	14:54	0,496	0,354	0,142						
		3	14:55	0,465	0,505	-0,04						
	3 (28.12.23)	1	10:14	0,491	0,506	-0,015						
		2	10:15	0,417	0,511	-0,094						
		3	10:17	0,561	0,727	-0,166	0,451	0,465	0,535	0,511	-0,084	-0,046
G65-233	1 (22.12.23)	1	14:34	0,443	0,341	0,102						
		2	14:35	0,489	0,276	0,213						
		3	14:38	0,316	0,356	-0,04						
	2 (27.12.23)	1	14:58	0,336	0,453	-0,117						
		2	14:59	0,382	0,465	-0,083						
		3	15:00	0,322	0,384	-0,062						
	3 (28.12.23)	1	10:20	0,493	0,365	0,128						
		2	10:21	0,427	0,282	0,145						
		3	10:22	0,417	0,535	-0,118	0,403	0,417	0,384	0,365	0,019	0,052
G65-210	1 (22.12.23)	1	14:44	0,541	0,556	-0,015						
		2	14:45	0,648	0,34	0,308						
		3	14:46	0,685	0,624	0,061						
	2 (27.12.23)	1	15:04	0,508	0,313	0,195						
		2	15:05	0,785	0,513	0,272						
		3	15:06	0,542	0,558	-0,016						
	3 (28.12.23)	1	10:26	0,481	0,454	0,027						
		2	10:27	0,305	0,231	0,074						
		3	10:29	0,378	0,402	-0,024	0,541	0,541	0,443	0,454	0,098	0,087
G65-313	1 (22.12.23)	1	14:49	0,518	0,594	-0,076						
		2	14:50	0,548	0,525	0,023						
		3	14:51	0,563	0,412	0,151						
	2 (27.12.23)	1	15:10	0,63	0,284	0,346						
		2	15:11	0,441	0,121	0,32						
		3	15:12	0,423	0,285	0,138						
	3 (28.12.23)	1	10:33	0,48	0,428	0,052						
		2	10:34	0,453	0,464	-0,011						
		3	10:35	0,677	0,286	0,391	0,526	0,518	0,378	0,412	0,148	0,106
G65-203	1 (22.12.23)	1	14:53	0,348	0,389	-0,041						
		2	14:54	0,665	0,345	0,32						
		3	14:56	0,45	0,43	0,02						
	2 (27.12.23)	1	15:14	0,389	0,305	0,084						
		2	15:15	0,382	0,304	0,078						
		3	15:17	0,511	0,139	0,372						
	3 (28.12.23)	1	10:37	0,357	0,298	0,059						
		2	10:38	0,396	0,276	0,12						
		3	10:39	0,493	0,362	0,131	0,443	0,396	0,316	0,305	0,127	0,091
Gesamt							0,473	0,465	0,411	0,389	0,062	0,059

Messdaten mit AR-Core Prototyp ohne Tageslicht

Punkt	Messung	Zeit(27.12.23) [hh:mm]	Abw Berechne [m]	Abw Gemesse [m]	Ber – Gem [m]	Mittelwert Ber [m]	Median Ber [m]	Mittelwert Ge [m]	Median Gem [m]	Mittelwerte B [m]	Median Ber-G [m]
63-301	1	17:08	0,631	0,653	-0,022						
	2	17:10	0,754	1,691	-0,937						
	3	17:11	0,474	0,182	0,292	0,62	0,631	0,842	0,653	-0,222	-0,022
G65-233	1	17:15	0,767	0,701	0,066						
	2	17:16	0,49	0,667	-0,177						
	3	17:19	0,708	0,575	0,133	0,655	0,708	0,648	0,667	0,007	0,066
G65-210	1	17:26	0,481	0,981	-0,5						
	2	17:27	0,814	0,473	0,341						
	3	17:28	0,55	0,271	0,279	0,615	0,55	0,575	0,473	0,04	0,279
G65-313	1	17:34	0,871	0,942	-0,071						
	2	17:35	0,557	0,507	0,05						
	3	17:41	0,813	0,472	0,341	0,747	0,813	0,64	0,507	0,107	0,05
G65-203	1	17:44	0,391	0,383	0,008						
	2	17:45	0,451	0,548	-0,097						
	3	17:46	0,51	0,583	-0,073	0,451	0,451	0,505	0,548	-0,054	-0,073
Gesamt						0,617	0,557	0,642	0,575	-0,024	-0,018

Abbildung 20: Messungen mit ARCore SDK mit Google Pixel 6 ohne Tageslicht