Bewegte 3D-Objekte im urbanen Raum als Herausforderung für Augmented-Reality-Systeme

Masterarbeit

im Studiengang Angewandte Informatik der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

Lehrstuhl für Medieninformatik

Verfasser: Bernhard Öder

Prüfer: Prof. Dr. Andreas Henrich

Dieses Werk ist als freie Onlineversion über das Forschungsinformationssystem (FIS; https://fis.uni-bamberg.de) der Universität Bamberg erreichbar. Das Werk steht unter der CC-Lizenz CC-BY.

Lizenzvertrag: Creative Commons Namensnennung 4.0
http://creativecommons.org/licenses/by/4.0.

URN: urn:nbn:de:bvb:473-irb-921652
DOI: https://doi.org/10.20378/irb-92165

Zusammenfassung

Die Arbeit gibt einen Überblick über verschiedene Technologien, welche im Zusammenspiel die Grundlage für Augmented-Reality-Anwendungen darstellen. Vor diesem Hintergrund wird am Beispiel der ehemaligen Straßenbahn Bambergs gezeigt, dass der komplexe Fall von bewegten 3D-Objekten im urbanen Raum für Augmented-Reality-Systeme eingeschränkt realisierbar ist. Das Modell der Straßenbahn wird dazu mittels Bilderkennung in den Stadtraum projiziert und bewegt sich über vier Streckenabschnitte entlang einer Straße. Die Anwendung läuft auf herkömmlichen Smartphones.
Als Hauptherausforderungen wurden die Bilderkennung im urbanen Raum, die präzise Objektplatzierung, die Visualisierungsstabilität der Straßenbahn gegenüber der realen Umwelt sowie das Verdeckungsverhalten erkannt. Als Faktoren, welche die Ausführung und die Visualisierungsqualität beeinflussen, wurden Belichtung, Auswahl des verwendeten Smartphones, das Nutzerverhalten und Verdeckungen erkannt. Aus den genannten Faktoren ergeben sich Einschränkungen bei der Erstellung einer solchen Anwendung, welche berücksichtigt werden müssen, um den Herausforderungen gerecht zu werden.
Die Arbeit zeigt, wie die AR-Anwendung im Detail entstanden ist. Für eine einfache und präzise Objektplatzierung kommt Bilderkennung zum Einsatz. Umgesetzt wurde sie unter der Verwendung von Unity und Vuforia. Das Modell der Straßenbahn wurde in Blender erstellt.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Kapitel 1

Motivation

Augmented Reality (kurz: *AR*) hat vielfältige Anwendungsgebiete und Augmented-Rality-Systeme entwickeln sich stetig weiter. Dieses Kapitel beschreibt in welchen Bereichen AR gewinnbringend eingesetzt werden kann und nennt verschiedene AR-Problemstellungen. Dabei wird insbesondere die Bedeutung von Augmented Reality für den Kulturerbebereich beschrieben, dem auch die ab Kapitel 3 beschriebene Anwendung zuzuordnen ist.
1997 beschreibt Azuma den Zustand von AR als „far behind Virtual Environments in maturity“ [Azuma, 1997]. Heute gibt es leistungsstarke mobile Endgeräte, z.B. Smartphones und Tablets, die einer breiten Masse an Personen zu Verfügung stehen und vielfältige Anwendungen ermöglichen. 2016 erschien mit Pokémon GO ein AR-Spiel, welches eines der erfolgreichsten Handyspiele dieser Zeit war [Paavilainen et al., 2017]. Pokémon GO war nicht die erste AR-Anwendung und auch nicht das erste AR-Spiel, welches dem Verbrauchermarkt zur Verfügung stand. Mit seinem Erfolg brachte es jedoch viele Menschen mit Augmented Reality in Berührung und trug damit zur Popularitätssteigerung von AR bei. Mit der Vielzahl an Möglichkeiten und der gesteigerten Verfügbarkeit sind auch die Erwartungen an AR gestiegen. In „Augmented reality and virtual reality displays: emerging technologies and future perspectives“ heißt es:
„VR and AR are endowed with a high expectation to revolutionize the way we interact with digital world“ [Xiong et al., 2021].

Vor diesem Hintergrund untersucht die folgende Arbeit, ob der komplexe Fall eines bewegten 3D-Objekts im urbanen Raum mittels Bilderkennung realisiert werden kann und evaluiert die Ergebnisse des gewählten Fallbeispiels bezüglich ihrer Visualisierungsqualität. Dabei wird untersucht, wie gut die Bilderkennung im urbanen Umfeld funktioniert und ob ein realistisches Verhalten des 3D-Objekts bezüglich seiner Positionierung, Stabilität im Raum und Verdeckung erzielt werden kann.

Bedeutung von AR im Kontext Kulturerbe Die folgende Arbeit behandelt eine Anwendung, bei der ein 3D-Modell der ehemaligen Straßenbahn Bambergs in die Innenstadt Bambergs projiziert werden soll. Damit lässt sich die Anwendung dem Kulturerbebereich zuordnen. Im Kulturbebereich wird mobilen AR-Anwendungen (kurz: *MAR*) eine vielversprechende Zukunft zugeschrieben:
„we conclude that MAR could be one of the answers to various problems faced by the

CH [cultural heritage, d. Verf.] area in the 21st century, such as the lack of funding, the poor impact on cultural growth, and the weak cultural cohesion of neighboring countries“ [Boboc et al., 2019].
In „A Survey of Augmented, Virtual, and Mixed Reality for Cultural Heritage“ werden bestehende Projekte analysiert und die AR- und VR-Anwendungen in fünf Anwendungsgebiete im Kulturerbebereich unterteilt. Im Bildungsbereich kann die Lernmotivation gesteigert werden. Ausstellungen können durch Beschreibungstexte, digitale Karten, einen virtuellen menschlichen Charakter oder Interaktionsmöglichkeiten erweitert werden. Explorative Anwendungen dienen der Interaktion zwischen realer und virtueller Umgebung und sollen so neue Einblicke ermöglichen. Anwendungen im Rekonstruktionsbereich können unvollständige oder nicht mehr existierende Gegenstände und Gebäude ebenso visualisieren, wie nicht greifbares Kulturerbe. Als letzter Bereich wird das virtuelle Museum als Ersatz des physischen Museums genannt. Der letzte Punkt des virtuellen Museums bezieht sich auf Anwendungen der virtuellen Realität bezieht. Die weiteren genannten Anwendungsgebiete, die sich in vielen Fällen miteinander überschneiden, zeigen das Potenzial und die sich eröffnenden Möglichkeiten von AR im Kulturerbebereich. [Bekele et al., 2018]
Augmented Reality stellt eine Möglichkeit dar, bestehende 3D-Modelle historisch relevanter Objekte in ihrem ursprünglichen Kontext darzustellen. Ein digitales Abbild eines Museumsexponats kann so an dem Ort eingeblendet werden, an dem es in Gebrauch war oder an dem es sich typischerweise einst befand. Ein Gebäude kann an einem Ort angezeigt werden, an dem es einst stand und bei bestehenden Gebäuden können frühere Zustände sichtbar gemacht werden, welche durch Zerstörung, Renovierung oder weitere Veränderungen nicht mehr bestehen. Verlorene Details können visualisiert oder vervollständigt werden.
Es gibt verschiedene technologische Möglichkeiten Exponate, aber auch Gebäude und weitere Strukturen mit hohem Detailgrad und hoher Präzision zu digitalisieren. Beispiele dafür sind Photogrammetrie sowie Laser- und Streifenlichtscanner. Nicht mehr vorhandene Objekte und Strukturen können mittels einer Modelierungssoftware anhand von Quellen nachempfunden werden. Die Qualität dieser Modelle hängt stark von der Quellenlage und der Person ab, welche die Modellierung durchführt. Eine einheitliche Qualität der Modelle in Bezug auf Detailgrad, Präzision und Vollständigkeit kann nicht gewährleistet werden und diese Modelle können nicht den Anspruch haben eine vergangene Realität abzubilden. Lücken in den Quellen werden mit Interpretationen, Übertragungen und Verallgemeinerungen aufgefüllt. Augmented Reality bietet diesen 3D-Modellen eine Plattform.
AR-Anwendungen können dazu anregen sich abseits der gewohnten Wege zu bewegen und sich genauer mit der Umgebung auseinanderzusetzen. Handelt es sich um eine ortsgebundene Anwendung, kann sie verschiedene Nutzer an bestimmten Punkten zusammenbringen und dadurch soziale Interaktion fördern. Es können neue Erfahrungen ermöglicht werden. So sagt eine Nutzerin von Pokémon GO in „The Pokémon GO Experience: A Location-Based Augmented Reality Mobile Game Goes Mainstream“: „This is probably the closest to accomplishing my childhood dream – to become a Pokémon master.“ [Paavilainen et al., 2017]
Khan et al. [2019] stellen beim Nutzen einer AR-Anwendung eine erhöhte Lernmotivation sowie erhöhte Aufmerksamkeit und Zufriedenheit fest. In „Mobile Augmented Reality for Cultural Heritage: Following the Footsteps of Ovid among Different Locations in Europe“ zeigen Boboc et al. wie die Zugänglichkeit zum immateriellen Erbe mittels AR gesteigert werden kann.

Bedeutung von AR für Smart Cities Diese Arbeit soll daher den Fall des bewegten 3D-Objekts im urbanen Raum als Herausforderung für AR-Systeme untersuchen. Dabei sollen auftretende Problemfelder untersucht, technologische Hintergründe erläutert und mögliche Lösungsansätze diskutiert werden. Die Stadt Bamberg bietet dafür die ideale Grundlage. Bamberg ist als Welterbestadt ein Besichtigungsziel für Touristen und wurde 2020 in das Smart-City-Förderprogramm aufgenommen. Hier kommen historischer Kontext mit neuen Ideen und modernen Technologien zusammen. Durch die Erweiterung des touristischen Erlebnis kann AR einen Beitrag zum *Smart Tourism* leisten [Gretzel et al., 2015]. Neben dem Tourismus ergeben sich jedoch auch weitere Anwendungsbereiche für Smart Cities. Dafür werden in „Augmented Reality (AR) and Cyber-Security for Smart Cities — A Systematic Literature Review“ die Bereiche Tourismus, Bildung und Mobilität sowie Systemüberwachung und Systemverwaltung beschrieben [Alzahrani und Alfouzan, 2022].
Für die im Folgenden beschriebene Anwendungen ergeben sich Schnittstellen zu den Smart-City-Leitlinien in den Bereichen von Tourismus, Bildung, Inklusion, Mobilität und Lebensqualität. Das 3D-Modell orientiert sich dabei an der ehemaligen Straßenbahn Bambergs und es soll sich entlang einer Straße in der Innenstadt bewegen.

In diesem Kapitel wurde gezeigt, dass Augmented Reality sowohl im Kulturerbebereich im speziellen, als auch darüber hinaus bereits eine Vielzahl von Einsatzmöglichkeiten und Vorteilen bieten kann. Dabei kommen verschiedene Arten von Visualisierungen zum Einsatz. Auf der technischen Ebene von AR-Anwendungen besteht in vielen Bereichen jedoch noch Optimierungspotenzial: „Tracking and registration in AR are far from solved“ [Panou et al., 2018]. Schlechte Performance war auch einer der Hauptkritikpunkte an Pokémon GO [Paavilainen et al., 2017]. Mit zunehmender Komplexität der Anwendungsfälle steigen auch die technischen Anforderungen. Entsprechend soll im Rahmen dieser Arbeit der komplexe Fall der ehemaligen Straßenbahn evaluiert werden, welcher dem Kulturerbebereich zugeordnet werden kann und in der Smart City Bamberg angesiedelt ist.

Kapitel 2

Grundlagen & Kontext

In dem ersten Kapitel wurde gezeigt, in welchen Anwendungsbereichen AR von Nutzen sein kann und welchen Mehrwert die Technologie bringen kann. In diesem Kapitel soll zunächst erklärt werden, was AR ist. Es folgen ein kurzer Überblick über die Geschichte von AR sowie eine Einordnung in den historischen Anwendungskontext der im Folgenden beschriebenen Anwendung.

2.1 Definition von Augmented Reality

Die Bezeichnung *Augmented Reality* (deutsch: Erweiterte Realität, kurz: *AR*) weist schon auf das Grundprinzip hin. AR-Anwendungen nutzen digitale Elemente zur Erweiterung einer realen Szene. Abbildung 1 zeigt das Kontinuum zwischen realer und virtueller Umgebung nach Milgram. Zwischen der realweltlichen Umgebung am linken Rand und der virtuellen Umgebung am rechten Rand befindet sich der Raum der gemischten Realität. Hier, im Raum der gemischten Realität, ist die Erweiterte Realität bzw. AR wiederum in der linken Hälfte zu finden. AR hat also einen starken Bezug zur realen Umgebung.

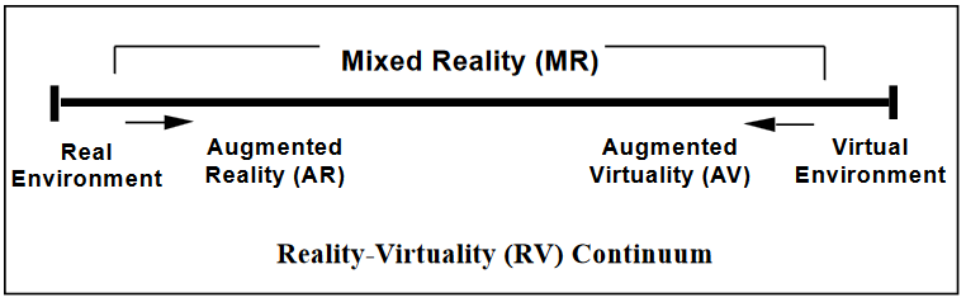


Abbildung 1: Milgrams Realitäts-Virtualitätskontinuum
Milgram et al. [1995]

Zur Erweiterung der realen Umgebung können verschiedene Medien wie Grafiken, Text oder Audio zum Einsatz kommen. Im Rahmen dieser Arbeit handelt es sich um eine grafische Erweiterung in Form eines 3D-Objekts, welches die ehemalige Straßenbahn Bambergs darstellt und zur Laufzeit auf einem Android-Smartphone angezeigt werden soll. Technologisch gibt es Parallelen zu Anwendungen der *Virtual Reality* (kurz: *VR*, deutsch: Virtuelle

Realität). AR reichert die Realität um zusätzliche Informationen an und ist im Gegensatz zu VR nicht allein im digitalen Raum verankert [Azuma, 1997]. AR erzeugt hybride Szenen aus realen und digitalen Teilen. Das Modell der ehemaligen Straßenbahn soll in diesem Projekt auf einem spezifischen Streckenabschnitt in Bamberg angezeigt werden, den die tatsächliche Straßenbahn einst befahren hat. Azuma definiert Augmented Reality 1997 folgendermaßen:

> To avoid limiting AR to specific technologies, this survey defines AR as systems that have the following three characteristics:
> 1) Combines real and virtual
> 2) Interactive in real time
> 3) Registered in 3-D
> [Azuma, 1997]

Dabei bildet 1) den essenziellen Kern von AR. In einer weiter gefassten Definition kann z.B. auch ein Video, welches mit Grafikelementen angereichert wurde, als AR gelten, obwohl diese Anreicherung nicht zur Laufzeit geschieht. Außerdem können auch zweidimensionale Grafiken in AR-Anwendungen zum Einsatz kommen.
Diese Arbeit erfüllt Azumas Definition jedoch in allen drei Punkten. Die dargestellte Straßenbahn hat einen realweltlichen Kontext, durch den sie sich bewegt. Die Anwendung ist interaktiv, da man die Perspektive zur Laufzeit ändern und die Straßenbahn wie ein reales Objekt umlaufen und untersuchen kann. Zuletzt handelt es sich bei der virtuellen Straßenbahn um ein 3D-Objekt. Das „Handbook of Augmented Reality“ bietet eine breiteren Blick auf AR:

> AR can potentially apply to all senses, augmenting smell, touch an hearing as well. AR can also be used to augment or substitute users' missing senses by sensory substitution, such as augmenting the sight of blind users or users with poor vision by the use of audio cues, or augmenting hearing for deaf users by the use of visual cues.
> [Furht, 2011]

Die Punkte 1) und 2) aus Azumas Definition werden auch hier erfüllt. Die Art der Realitätserweiterung geht jedoch über den Einsatz von 3D-Objekten hinaus und bezieht AR auf sämtliche Sinneswahrnehmungen.
AR-Anwendungen können auf verschiedenen Geräten funktionieren und angezeigt werden. Bei diesen Geräten handelt es sich beispielsweise um Smartphones, Tablets oder spezielle Brillen bzw. *Head-Mounted Displays* (kurz: *HMD*) [Siriwardhana et al., 2021]. Durch AR-Brillen mit durchsichtigen Displays lassen sich zusätzliche Informationen oder Grafiken anzeigen während die Umgebung weiterhin normal wahrgenommen werden kann. Im Rahmen dieser Arbeit kamen Android-Smartphones zum Einsatz, welche die reale Umgebung über die Kamera erfassen und das Abbild der Umgebung mit einer modellierten 3D-Grafik auf dem Display vereinen.

2.2 Geschichte von AR

Der folgende Abschnitt gibt einen kurzen Überblick über die Geschichte von AR und und die entstandenen Einsatzfelder.
Während der technologische Fortschritt mobiler Endgeräte der letzten Jahre die Einsatzmöglichkeiten von AR-Anwendungen stark gefördert hat, ist der erste Prototyp eines AR-Systems bereits 1968 entstanden. Es handelte sich dabei um ein stationäres System von Ivan Sutherland mit durchsichtigen Displays, welche bereits in der Lage waren, einfache dreidimensionale Grafiken anzuzeigen. Ab Mitte der 1960er Jahre wurden AR-Systeme zur Unterstützung von Militärpiloten entwickelt und in den 1980er Jahren kamen solche System auch in der Raumfahrt zum Einsatz. In den 1990er Jahren kamen weitere Forschungs- und Einsatzfelder hinzu wie z.B. zur Visualisierung und Kooperation im medizinischen Bereich. Zur gleichen Zeit entwickelten sich auch mobile Computersysteme. [Billinghurst et al., 2015]
Die Kombination dieser beiden Bereiche, AR-Systeme einerseits und mobiler Computersysteme andererseits, bilden die Grundlage für die heutigen mobilen AR-Anwendungen und AR-Systeme. Heute haben AR-Anwendungen sehr viele Einsatzfelder und es handelt sich nicht mehr ausschließlich um höchst spezialisierte Anwendungen. Weitere Einsatzfelder sind z.B. Industrie, Marketing, Navigation, Unterhaltung und, wie im ersten Kapitel bereits gezeigt, Bildung, Kulturerbe, Rekonstruktion und Tourismus. Ein Beispiel für die zuletzt genannten Felder findet sich in dem 2018 erschienenen Artikel „An Architecture for Mobile Outdoors Augmented Reality for Cultural Heritage“. Hier werden verschiedene Monumente des griechischen Orts Chania mittels AR in deren einstigem Zustand dargestellt. [Panou et al., 2018]

2.3 Bamberger Straßenbahn

Wie in „An Architecture for Mobile Outdoors Augmented Reality for Cultural Heritage“ [Panou et al., 2018] handelt es sich auch bei der Anwendung dieser Arbeit um ein Beispiel im Kulturerbebereich. Mit der Glassmoschee, der San Rocco Kirche und einem Stück der byzantinischen Stadtmauer handelte es sich in dem genannten Artikel jedoch um statische Bauten. Im Gegensatz dazu wird diese Arbeit bewegte 3D-Objekte im urbanen Raum behandeln. Dazu wurde die ehemalige Bamberger Straßenbahn als Beispiel gewählt. Abbildung 2 aus Wußmann [2018] zeigt sie vor dem alten Bamberger Rathaus.
Die Bamberger Straßenbahn wurde 1897 eingeweiht. Ihr Liniennetz hatte drei Hauptlinien, welche sich vom Bahnhof über die Innenstadt und in angrenzende Stadtgebiete erstreckten. Abbildung 3 zeigt das ehemalige Streckennetz mit weiteren geplanten, aber nie umgesetzten Streckenabschnitten [Brehm, 1991]. Nach knapp 25 Jahren kam ihr Betrieb im Jahr 1922 jedoch schon wieder zum erliegen und wurde letztlich permanent eingestellt. [Wußmann, 2018]
Heute, über 100 Jahre nach dem Ende des Straßenbahnbetriebs, erinnern noch einige Relikte an ihre Existenz. So sind an Fassaden teilweise noch Haken der ehemaligen Oberleitung zu sehen und andernorts ragen zu Stützpfeilern umfuktionierte Gleise aus der Erde.

Abbildung 2: Straßenbahn vor dem Bamberger Rathaus
Wußmann [2018]

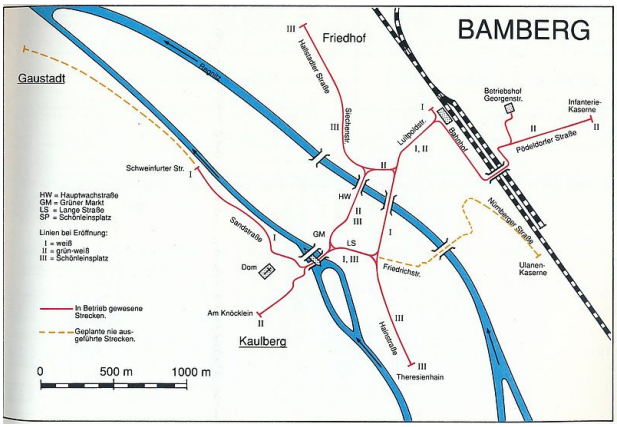


Abbildung 3: Streckennetz der Bamberger Straßenbahn
Brehm [1991]

Die vorangegangenen Abschnitte haben gezeigt, dass Azumas Definition eines AR-Systems mit einer interaktiven 3D-Anwendung, welche Realität und virtuelle Welt zur Laufzeit kombiniert, für die Anwendung dieser Arbeit gilt. Es wurde ein kurzer Überblick über die Geschichte von AR und den Hintergrund des Anwendungsbeispiel *Bamberger Straßenbahn* gegeben. Der folgende Abschnitt konzentriert sich auch die technologischen Grundlagen.

2.4 Technologische Hintergründe

Eine AR-Anwendung wird durch das Zusammenspiel verschiedener Technologien realisiert. Dieser Abschnitt geht auf die technischen Hintergründe und Methoden ein, welche als Grundlagen für eine bilderkennungsbasierte AR-Anwendung dienen können.
Die einzelnen Schritte der Bilderkennung werden als erstes gezeigt. Anschließend wird gezeigt, wie SLAM-Verfahren unabhängig von Bildankern der Positionierung dienen. Abschließend wird auf die Grundlagen der dreidimensionalen Visualisierung eingegangen. Hierbei liegt der Fokus auf der Visualisierungsreihenfolge bzw. der Objektverdeckung zwischen verschiedenen 3D-Objekten und deren Implikationen auf die Verdeckung zwischen einem digitalen Objekt einer AR-Anwendung und der realweltlichen Umgebung.

2.4.1 Bilderkennung

Mittels Bilderkennung wird die virtuelle Straßenbahn in der Umwelt platziert. Damit ist die Bilderkennung essentiell, da es von ihrem Erfolg abhängt, dass die Anwendung gestartet wird. In diesem Abschnitt wird gezeigt welche Rolle Schlüsselpunkte für die Bilderkennung spielen und wie auf algorithmischer Ebene mit den Problemstellungen der Bilderkennung im Stadtraum umgegangen werden kann.

Für die bilderkennungsbasierte Objektplatzierung erfolgen drei Hauptschritte:
1. Erkennung von markanten Punkten bzw. Schlüsselpunkten
2. Abgleich von Schlüsselpunkten aus Kamerabild und hinterlegtem Bild
3. Berechnung der Perspektive aus den Positionen der erkannten Punkte

Die drei Hauptschritte lassen sich auf unterschiedliche Weisen lösen und dem konkreten Vorgehen entsprechend in weitere Teilschritte unterteilen. Für den ersten Schritt kann zwischen der Erkennung von Eckpunkten und der Bloberkennung unterschieden werden. Für den zweiten Schritt eine Vielzahl von Ansätzen. Abbildung 4 zeigt das am Beispiel des Zustandsdiagramms von Wagner et al. [2010]. Der mit „Keypoint detection“

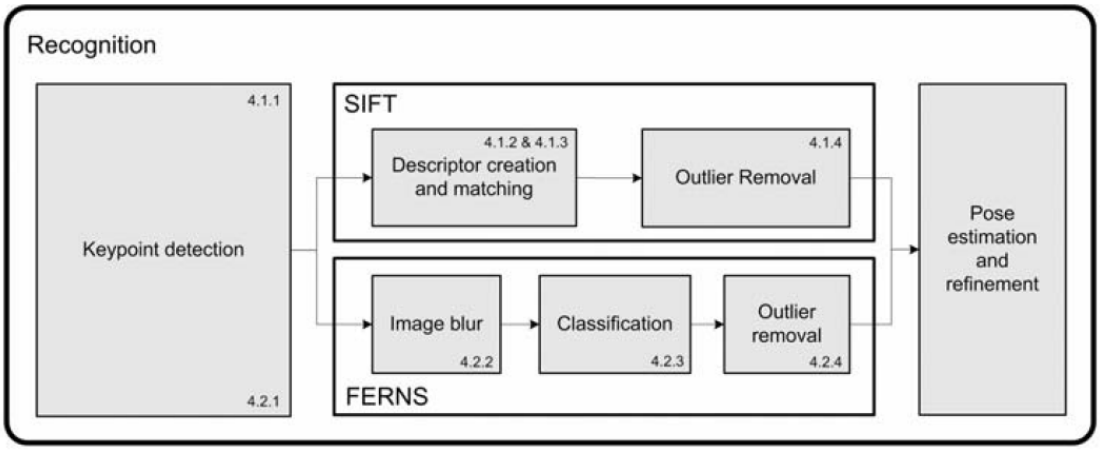


Abbildung 4: Zustandsdiagramm zur Bilderkennung
Wagner et al. [2010]

beschriftete Block entspricht dem ersten Schritt und der mit „Pose estimation and refinement“ beschriftete Block rechts dem dritten Schritt. Die beiden zentralen Blöcke, welche mit „SIFT“ und „FERNS“ beschriftet sind entsprechen beide dem zweiten Schritt des Schlüsselpunktabgleichs. Sie bestehen wiederum aus einzelnen unterschritten. Während *SIFT* die erkannten Punkte in Deskriptoren umrechnet um diese miteinander abzugleichen, löst *Ferns* die Zuordnung mit einem Klassifikationsansatz, der auf eine Vorverarbeitung des Bildes folgt. Gemeinsam haben die beiden Ansätze, dass fälschlicherweise erkannte Punkte entfernt werden.
Im Folgenden werden die genannten Schritte genauer betrachtet und am Beispiel konkreter Ansätze erklärt. Verykokou et al. [2021] stellen für den Anwendungsfall einer mobilen AR-Anwendung eine effiziente Bilderkennungsmethode vor, die sich des FAST-Detektors und des ORB-Deskriptors bedient. Entsprechend wird im ersten Abschnitt neben dem SIFT-Algorithmus auch die Eckerkennung mittels FAST und im zweiten Abschnitt ORB als Beispiel zur Verwendung von Deskriptoren vorgestellt.

Schlüsselpunkterkennung

Bevor in diesem Abschnitt die Eckerkennung und die Bloberkennung als Erkennungsmethoden zur Extraktion von Feature- bzw. Schlüsselpunkten vorgestellt werden, werden die Schlüsselpunkte selbst thematisiert. Für den Zweck der Verankerung eines 3D-Objekts müssen diese bestimmte Qualitätseigenschaften erfüllen. Essentiell ist dabei die Wiederholbarkeit. Die erkannten Punkte eines Objekts sollen auch unter geänderten Bedingungen wiedererkannt werden. Die Wiederholbarkeit kann durch Invarianz oder Robustheit verbessert werden. Das bedeutet, dass die Transformationen der Punkte, die durch geänderte Bedingungen entstehen, zurückgerechnet werden oder durch eine Vorverarbeitung des Bildes abgefangen werden können. Ebenso ist die Unterscheidbarkeit der einzelnen Punkte für ihre korrekte Zuordnung von Bedeutung. In einem geometrischen wiederkehrenden Muster könnten Punkte zwar möglicherweise gut erkannt, jedoch nicht eindeutig zugeordnet werden, da sie jeder einzelnen Musterwiederholung zuordenbar wären. Außerdem sollen sie lokal, genau und effizient sein und in ausreichen großer Zahl vorliegen. [Tuytelaars et al., 2008]

Eckerkennung Unterschiedliche Erkennungsmethoden können die gewünschten Qualitätseigenschaften in unterschiedlichem Maße erfüllen. So ist Eckerkennung effizienter und präziser gegenüber der Bloberkennung, besitzt aber keine Skalierungsinformation. Ein klassisches Beispiel eines Detektors zur Eckerkennung ist der von Harris et al. [1988], welcher ein kombinierter Ecken- und Kantendetektor ist und und eine verbesserte Erkennungsrate gegenüber dem früheren Detektor von Moravec [1977] liefert.
Im Folgenden soll jedoch anhand *FAST*-Detektors gezeigt werden, wie Eckpunkte im Detail identifiziert werden. FAST steht für „Features from Accelerated Segment Test“, wurde von Rosten und Drummond [2006] entwickelt und gilt aufgrund seiner Effizienz als eine Erkennungsmethode, die auch mit beschränkten Ressourcen zur Laufzeit angewandt werden kann und kommt laut Herling und Broll [2011] häufig in Anwendungen für Mobiltelefone zum Einsatz. Zur Erkennung von Ecken werden die einzelnen Pixel eines Bildes auf ihre Farbwerte untersucht. Bei einer ausreichend großen Differenz der Farbwerte eines Pixels p zu dessen umliegenden Pixeln, kann es sich um eine Ecke handeln.
Ob es sich tatsächlich um eine Ecke handelt, wird mit dem folgenden Vorgehen festgestellt.

Betrachtet werden Pixel p und 16 umliegende Pixel in einem Abstand r. Es handelt sich um eine Ecke, wenn n aufeinanderfolgende umliegende Pixel sich um eine Mindestdifferenz von p unterscheiden. Rosten und Drummond [2006] beschreiben, dass r=3 und n=9 dabei die besten Ergebnisse liefern und veranschaulichen das an dem Beispiel auf Abbild 5. In der linken Hälfte ist darauf ein Foto mit zwei Spitzbögen zu sehen. In der rechten Hälfte wird eine Vergrößerung eines Ausschnitts des linken Bildes gezeigt. Um den zentralen Pixel p sind die 16 Felder der der umliegenden Pixel im Abstand r=3 markiert. Die zwölf Felder mit der weißen, gestrichelten Linie von Feld 11 bis Feld 6 haben eine Farbdifferenz zu p, welche über dem definierten Mindestwert liegt. Es handelt sich um zwölf aufeinanderfolgende Pixel entlang des Radius. Damit ist die Mindestanzahl von n=9 erfüllt und p kann als Eckpunkt identifiziert werden.
Zur Laufzeitoptimierung wird dieser Abgleich nur an Vorausgewählten Kandidaten durchgeführt. Dazu werden zunächst Pixel ausgeschlossen, bei denen es sich nicht um Ecken handeln kann. Dazu reicht es die Pixel an den Positionen 1, 5, 9 und 13 zu betrachten und mit p zu vergleichen. Wenn nicht drei dieser Pixel eine Mindestdifferenz ihres Farbwertes zu p aufweisen, wird der Pixel als Eckkandidat ausgeschlossen und der Abgleich der restlichen Pixelwerte entfällt. [Rosten und Drummond, 2006]

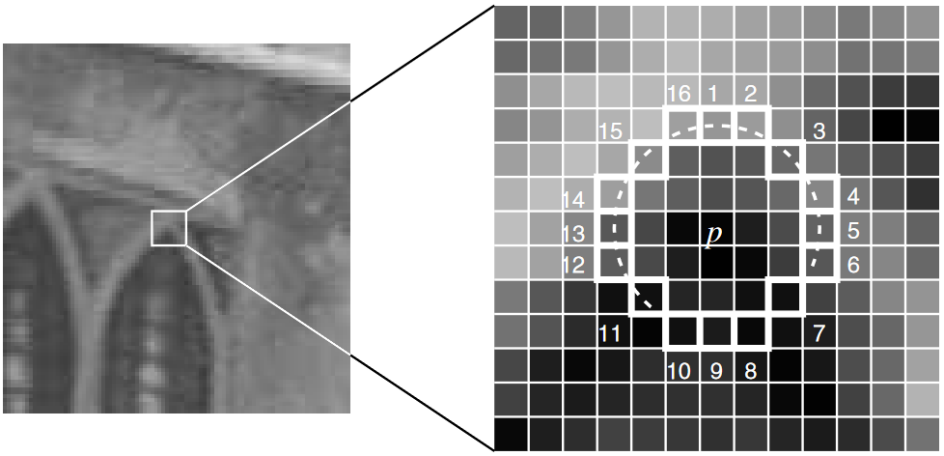


Abbildung 5: FAST - Eckerkennung
Rosten und Drummond [2006]

Werden mehrere Eckpunkte unmittelbar nebeneinander gefunden wird nur der Eckpunkt mit der stärksten Eckeigenschaft beibehalten. Dazu wird die Stärke der Eckeigenschaft aus den umliegenden Pixeln berechnet und mit einer *Non-Maximum-Suppression*-Suche (kurz: *NMS*) der stärkste Eckpunkt identifiziert. Die weiteren Eckpunkte werden verworfen. Die Funktionsweise von NMS wird beispielsweise von Neubeck und Van Gool [2006] beschrieben.
Neben der hohen Effizienz bei der Auffindung von Eckpunkten ist bezüglich FAST zu erwähnen, dass der Algorithmus keine skalierungsinvarianten Ergebnisse liefert und nicht robust gegenüber Bildrauschen ist (vgl. Herling und Broll [2011] und Rosten und Drummond [2006]).

Bloberkennung Bei der Bloberkennung wird nach Features mit Blobform wie kleinen Punkten oder größere Flächen mit ähnlicher Farbe gesucht.
Ein Vertreter dieser skalierungsinvarianten Erkennungsmethode ist der von Lowe [1999] entwickelte *SIFT*[1]-Algorithmus . Die Featurepunkte werden bei SIFT ebenfalls an Stellen lokaler Extrema gefunden. Dazu werden jedoch nicht die Pixel selbst miteinander verglichen, sondern die Pixel von Masken der Bilder. Abbildungen 6 und 7 stammen von Lowe [2004] und veranschaulichen, wie die Bildmasken entstehen und wie dort Featurepunkte identifiziert werden.

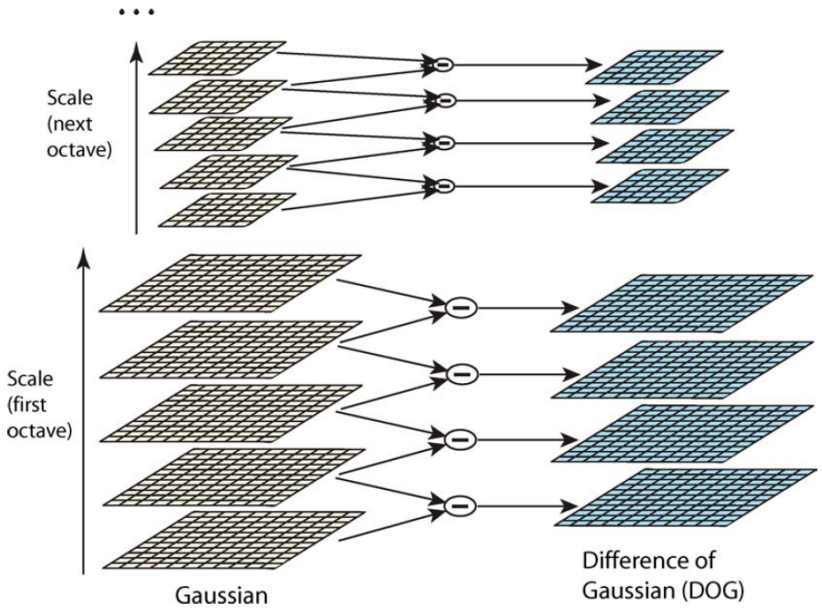


Abbildung 6: SIFT - Differenz der Gauß-Funktionen
Lowe [2004]

Die Bildmasken entstehen in zwei Schritten. Auf das Bild wird ein Gauß-Filter mehrfach angewandt (vgl. Abb. 6 links). Im zweiten Schritt werden die Differenzen der einzelnen Iterationen der Gauß-Funktion gebildet (vgl. Abb. 6 rechts). Die Masken zeigen also die Veränderungen des Bildes, welche durch die Anwendung der Gauß-Funktion entstehen. Diese beiden Schritte werden so oft wiederholt, bis eine zuvor festgelegte Anzahl erreicht ist. Anschließend wird die Abtastrate des Bildes halbiert und die beschrieben Schritte werden wiederholt. So wird eine Bildpyramide aus unterschiedlichen Auflösungen des Eingangsbildes aufgebaut.
Auf den vorliegenden Masken können nun Extrema mittels einer NMS-Suche gefunden werden. Abbildung 7 zeigt den mit einem X markierten Pixel im mittleren Skalenraum. Verglichen wird er mit den umliegenden Pixeln, die mit einem Kreis markiert sind. Dabei ist zu beachten, dass er nicht nur mit den benachbarten Pixeln des eigenen Skalenraums verglichen wird, sondern auch mit Pixeln der benachbarten Skalenräume. Auf diese Weise wird die Skalierung bei der Auffindung von Featurepunkten berücksichtigt, was folglich zu skalierungsinvarianten Featurepunkten führt.

[1]SIFT: Scale-Invariant Feature Transform

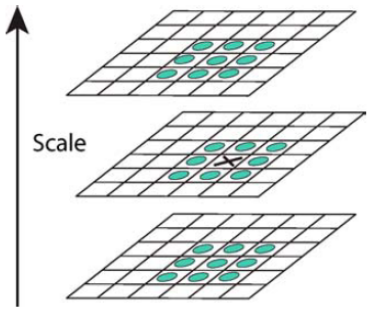


Abbildung 7: SIFT - NMS über angrenzende Pixel und Skalenräume
Lowe [2004]

Mit *SURF*[2] stellten Bay et al. [2006] eine schnellere Variante der Bloberkennung vor. SURF verwendet zur Bildfilterung im Gegensatz zu SIFT keine Gauss-Filterung sondern eine Hesse-Matrix. Auch wird keine Bildpyramide aufgebaut, da die Abtastrate des Bildes nicht neu berechnet wird. Stattdessen skaliert der Filter. SURF steigert gegenüber SIFT sowohl die Geschwindigkeit, als auch die Robustheit bei der Findung von Featurepunkten [Herling und Broll, 2011].

Unabhängig davon, welche Methode zur Erkennung von Schlüsselpunkten zum Einsatz kommt, sind für einen Abgleich von Schlüsselpunkten mindestens zwei Punktesets nötig. Entsprechend muss die Schlüsselpunkterkennung zu zwei Zeitpunkten stattfinden. Einerseits müssen die Schlüsselpunkte auf einem hinterlegten Bild erkannt werden. Das kann bereits bei der Erstellung einer Anwendung geschehen. Zur Laufzeit müssen dann die Schlüssel aus den eingehenden Kamerabildern extrahiert werden um mit den vorliegenden Schlüsseln verglichen zu werden.

Abgleich von Schlüsselpunkten

Der zweite Hauptschritt ist der Abgleich der Schlüsselpunkten. Wie viele solcher Punkte maximal gefunden werden dürfen, kann mit einem Grenzwert reguliert werden, welcher mögliche Schlüsselpunkte mit schwächeren Schlüsseleigenschaften ausschließt. Der Grenzwert ist damit ein Instrument zur Regulierung der Genauigkeit und der Performance. Zum Abgleich der neu erkannten Schlüsselpunkte mit denen aus einer Datenbank gibt es verschiedene Abgleichstrategien. Es werden die drei Strategien des Abgleichs von Bildausschnitten, des Abgleichs von Deskriptoren und der Klassifizierung vorgestellt. [Herling und Broll, 2011]

Abgleich von Bildausschnitten Schlüssel sind von Bildausschnitten mit konstanter Größe bzw. Pixelzahl umgeben und diese Ausschnitte können mit den Ausschnitten in einer Datenbank abgeglichen werden. Dies kann mit der *Summe der absoluten Differenz*

[2]SURF: Speeded Up Robust Features

(kurz: SAD) effizient berechnet werden und eignet sich somit für den Einsatz auf mobilen Geräten. Die Werte der Pixel im Umfeld der verglichenen Schlüsselpunkte werden voneinander abgezogen und die Differenzen der einzelnen Pixelwerte miteinander addiert. Der erhaltene Wert beschreibt also die Abweichung der Bildausschnitte zueinander. Ob es sich bei den verglichenen Bildausschnitten um eine Übereinstimmung handelt, kann mit einem Grenzwert bestimmt werden. Die Pixelwerte können sich je nach Betrachtungswinkel, Entfernung, Lichtbedingungen und Rotation stark ändern. Entsprechend würde sich auch die SAD stark ändern und ein Bildausschnitt kann nicht robust in Bezug auf die Perspektive und Skalierung erkannt werden.

Abgleich von Deskriptoren Deskriptoren abstrahieren die Information eines Schlüsselpunktes und können sie durch zusätzlich Information erweitern. Sie beinhalten Skalarwerte der Nachbarschaft eines Schlüssels und ermöglichen Skalierungs- und Rotationsinvarianz. Der Abgleich der Schlüsselpunkte aus der Datenbank mit den Schlüsselpunkten des Kamerabildes passiert anhand ihrer jeweiligen Deskriptoren. Schlüsseldeskriptoren können die Skalierungsinvarianz von Bloberkennungen darstellen und Punkte aus Eckerkennungen in einem zusätzlichen Schritt um Invarianz erweitern.
Ein Beispiel für Letzteres ist *ORB*. ORB steht für *Oriented FAST and Rotated BRIEF* und wurde von Rublee et al. [2011] entwickelt. Die Erweiterung von FAST um Orientierungsinformation wird mittels einer Bildpyramide erreicht. Der *BRIEF*[3]-Deskriptor beschreibt Bildausschnitte als Vektor, welcher sich aus einem paarweise durchgeführten Pixelabgleich ergibt. Der Bildausschnitt wird mit einem Gauß-Filter versehen und aus dem Abgleich der Pixelwerte des Ausschnitts werden deren Werte definiert, welche in Summe den Vektor des Bildausschnitts ergeben.
Dies ist sehr effizient, löst jedoch nicht die im vorangegangen Abschnitt beschriebene Probleme, die beim Abgleich von Bildausschnitten auftreten. ORB erweitert BRIEF um Rotationsinvarianz. Im ersten Schritt wird unter der Verwendung der Orientierungsinformation der Schlüsselpunkte eine Tabelle erstellt, welche mit vorberechneten Werten in 12°-Schritten gefüllt wird. Da durch diesen Schritt die Varianz und Unterscheidbarkeit der erhaltenen Schlüsselpunkte beeinträchtigt wird, wird diese Methode um einen weiteren Schritt ergänzt. Durch diesen weiteren Schritt wird die Varianz der Schlüsselpunkte gewahrt, indem eine Vorauswahl der Pixelpaare getroffen wird, welche den beschreibenden Vektor ergeben. Es werden nur Werte in den Ergebnisvektor aufgenommen, die einen Korrelationsgrenzwert nicht überschreiten. [Rublee et al., 2011]

Klassifizierung Zu jedem Schlüssel der Datenbank wird eine Klasse angelegt. Diese Klasse besitzt alle möglichen Erscheinungsformen des entsprechenden Schlüsselpunktes. So können verschiedene Blickwinkel, Belichtungsbedingungen usw. abgefangen werden. Zur Laufzeit gefundene Schlüsselpunkte werden ihrer entsprechenden Klasse zugeordnet und somit erkannt. Der Schlüsselabgleich kann als Suche entlang Suchbäumen sehr schnell erfolgen, ist allerdings speicherintensiv und benötigt einen hohen Aufwand in der Vorverarbeitung bzw. im Training. Der *Ferns*-Ansatz strukturiert dieses Vorgehen durch die Verwendung von Unterklassen um, und kann dadurch Speicherplatz einsparen.

[3]BRIEF: Binary Robust Independent Elementary Features

Positionsberechnung

Nach einem erfolgreichen Schlüsselabgleich und einer damit verbundenen Erkennung eines Bildes, folgt nun noch die Positionsberechnung. Für die Erkennung von Bildern ist eine Robustheit und Invarianz bezüglich unterschiedlicher Faktoren für eine möglichst hohe Wiederholbarkeit der Bilderkennung erstrebenswert. Diese Robustheit und Invarianz ermöglichen es, dass ein Bild aus unterschiedlichen Perspektiven erkannt wird. Soll mit dieser Bilderkennung ein 3D-Objekt platziert werden, ist die Perspektive auf das Bild für eine korrekte Darstellung der 3D-Objekte relevant und muss entsprechend berechnet werden. Um die Proportionen zwischen dem projizierten 3D-Objekt und der Umgebung richtig darstellen zu können, muss auch die physische Größe des Bildankers eingangs bekannt sein.
Da es bei mobilen AR-Anwendungen möglich ist, sich während der Laufzeit zu bewegen und somit die Perspektive zu ändern, müssen Schlüssel in jedem neuen Bild erkannt und mit der Datenbank abgeglichen werden. Entsprechende Deskriptoren müssen gegebenenfalls auch für jedes Bild berechnet werden. Bei ausreichend hoher Übereinstimmung kann die Kameraposition in sechs Freiheitsgraden (kurz: *6DoF*) berechnet werden.
Zur Verbesserung der Positionsberechnung müssen fälschlicherweise erkannte Schlüssel ausgeschlossen werden. Nach einer erfolgreichen Positionierung kann die vorliegende Positionsinformation als Grundlage zur weiterführenden Positionsberechnung dienen. Vorliegende Positionsinformationen können die zu durchsuchenden Deskriptoren eingrenzen, was eine effizientere Suche ermöglicht. Dies ist möglich, da für aufeinanderfolgende Bilder nur kleine Veränderungen der Kameraposition erwartet werden. Entsprechend können mögliche folgende Positionen auf der Grundlage der aktuellen Position vorhergesagt werden.
Außerdem zu berücksichtigen ist die Verzerrung des Kamerabildes. Die Verzerrung nimmt mit der Distanz eines Objekts zur Kamera zu und kann die Positionierungsgenauigkeit beeinflussen. Schlüsselpunkte sollten entsprechend entzerrt werden.
Anschließend können alle Schlüsselpunkte zur dreidimensionalen Positionsbestimmung genutzt werden. Für diese perspektivische Berechnung sind mindestens drei erkannte Schlüsselpaare zwischen Eingangsbild und dem hinterlegten Bild aus der Datenbank notwendig. Basierend auf drei Schlüsselpaaren ergeben sich vier mögliche Positionen. Für eine eindeutige Positionierung sind jedoch noch weitere Übereinstimmungen notwendig.[Herling und Broll, 2011]
Das *Perspective-n-Point*-Problem (kurz: *PnP*-Problem) beschreibt die dreidimensionale Bestimmung der Perspektive auf der Grundlage eines zweidimensionalen Bildes mit n erkannten Punkten. Lepetit et al. [2009] stellen mit *EPnP* einen effizienten Algorithmus vor, der dieses Problem löst. Sie reduzieren die Komplexität der n 3D-Punkte, indem sie diese als die Summe von vier virtuellen Kontrollpunkten ausdrücken. Das ermöglicht die Lösung des PnP-Problems mit linearem Aufwand und einer gewissen Robustheit gegenüber der Teilverdeckung von Punkten.

2.4.2 SLAM

Im vorangegangenen Abschnitt wurde gezeigt, dass die Perspektive anhand der Position von Schlüsselpunkten eines erkannten Bildes berechnet und ein 3D-Objekt entsprechend visualisiert werden kann. Die Visualisierungsstabilität soll jedoch auch dann gewährleistet sein, wenn die erkannten Schlüsselpunkte des Bildankers nicht mehr erfasst werden können, da sich z.B. die Blickrichtung oder die Position während der Nutzung ändern. An dieser Stelle können *SLAM*-Algorithmen der Positionsberechnung dienen.
SLAM steht für *Simultaneous Localization and Mapping* und nutzt keine vorab erstellte Karte oder Bildanker mit Schlüsselpunkten zur Orientierung im Raum, sondern erstellt die Orientierungsinformation zur Laufzeit. Ein SLAM-Algorithmus startet ohne eingangs vorliegende Umgebungsinformation. Während der Positionserfassung und Berechnung werden diese Informationen erst gesammelt. Es wird eine grobe Karte zum Start berechnet, welche eine initiale Positionierung ermöglicht. Diese Karte wird anschließend erweitert und die Änderung der Position berechnet. Es ergeben sich die drei Hauptelemente der Initialisierung, der Bewegungsverfolgung und der Kartierung. Hinzu kommen die Relokalisierung, die dafür sorgt, dass die Positions- bzw. Bewegungsverfolgung wieder aufgenommen werden kann, sollte sie unterbrochen werden und ein Schritt der Kartenoptimierung, welche entstehende Fehler der Kartierung minimiert. [Taketomi et al., 2017]
Wie diese Elemente umgesetzt werden, unterscheidet sich bei den unterschiedlichen SLAM-Ansätzen. Es gibt rein visuelle Ansätze und solche, die neben der visuellen Information über Sensoren wie Gyroskop und Beschleunigungssensor die Trägheit bzw. die Bewegung des Geräts erfassen und zur Positionsberechnung verwenden.
Zunächst werden die verschiedenen Ansätze der rein visuellen SLAM-Methoden (kurz: *vSLAM*) vorgestellt und für den mobilen Einsatz relevante Beispiele genannt.
Die visuellen Methoden können in drei Kategorien unterteilt werden. Methoden der ersten Kategorie nutzen, wie im Abschnitt 2.4.1 beschrieben, Schlüssel bzw. Schlüsselpunkte zur Positionsverfolgung. Eine weitere Kategorie stellen direkte vSLAM-Methoden dar, welche auf die Erstellung von Schlüsselpunkten und Deskriptoren verzichten. Das eingehende Bild wird direkt mit der erstellten 3D-Karte verglichen und auf fotometrische Konsistenz überprüft. Im Gegensatz zu den Schlüsselpunkten, die die Geometrie beschreiben, werden hier die Farbwerte abgeglichen. Methoden der dritten Kategorie generieren Tiefeninformation aus zusätzlich ausgesandtem Licht und werden als *RGB-D*-Methoden bezeichnet.
Abschließend werden visuell-inertiale Methoden vorgestellt, welche zusätzlich Bewegungssensoren verwenden.

Featurebasiertes SLAM Ein Beispiel für die erste Kategorie ist *PTAM*. PTAM steht für *Parallel Tracking and Mapping* und nutzt für die Aufgaben der Bewegungsverfolgung und der Kartierung unterschiedliche CPU-Threads. Das ermöglicht die effiziente parallele Ausführung der beiden Aufgaben. Bei der Initialisierung werden in dem ersten Bild Eckpunkte erkannt und mit Bildpunkten aus folgenden Bildern, die aus einer geänderten Position aufgenommen wurden, zu einer initialen Karte verrechnet. Dies geschieht mittels einer homografischen Projektion. Anschließend werden weitere Schlüsselpunkte hinzugefügt und die Position der Kamera wird aus den Positionsunterschieden der bekannten Schlüsselpunkte trianguliert. [Klein und Murray, 2009]

Direktes SLAM Newcombe et al. [2011] entwickelten mit *DTAM* einen Algorithmus der Kategorie der direkten vSLAM-Methoden. DTAM steht für *Dense Tracking and Mapping in Real-Time* und wurde von Ondrúška et al. [2015] für den Einsatz auf Smartphones optimiert. Die Initialisierung geschieht mittels Schlüsselpunkt-basierter Positionsverfolgung. Anschließend werden die eingehenden RGB-Bilder zu einem 3D-Modell verrechnet, welches während der Anwendung kontinuierlich erweitert wird (vgl. Abb. 8). Dazu werden Tiefenkarten volumetrisch miteinander vereint und das Modell wird aus Voxeln aufgebaut. Voxel haben einen Farbwert, einen Wert für die Distanz zur nächsten gemessenen Oberfläche und einen Gewichtungswert für die Wahrscheinlichkeit ihrer Korrektheit. Die Kameraposition wird mit einer 2,5D-Bildübereinstimmung aus dem eingehenden Bild und dem bestehenden Modell berechnet und kann optional von einer inertialen Messeinheit unterstützt werden. [Ondrúška et al., 2015]

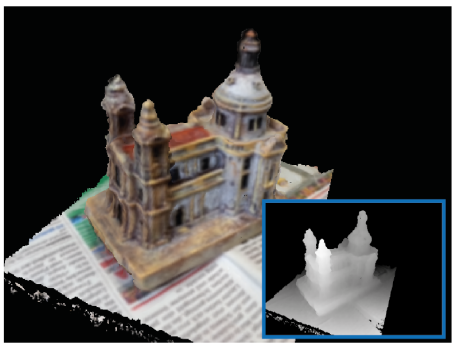


Abbildung 8: 3D-Modell aus MobileFusion
Ondrúška et al. [2015]

Eine weitere direkte Methode beschreiben Schöps et al. [2014] in „Semi-Dense Visual Odometry for AR on a Smartphone“. Ihre Methode ist deshalb „Semi-Dense“, da kein Geschlossenes 3D-Modell gebildet wird. Abbildung 9 zeigt auf der linken Seite die farbig hervorgehobenen Bereiche einer Szene, die erkannt wurden und die Tiefenkarte bilden. Die großen farblosen Flächen in der Mitte wurden dabei ausgelassen. Auf der rechten Seite ist ein dichtes Kollisionsnetz zu sehen, welches aus der Tiefenkarte berechnet wurde. Die Tiefenkarte wird mit zufälligen Tiefenwerten initialisiert und durch langsame Bewegung wird eine konsistente Tiefenkonfiguration erkannt und eine Karte gebildet.

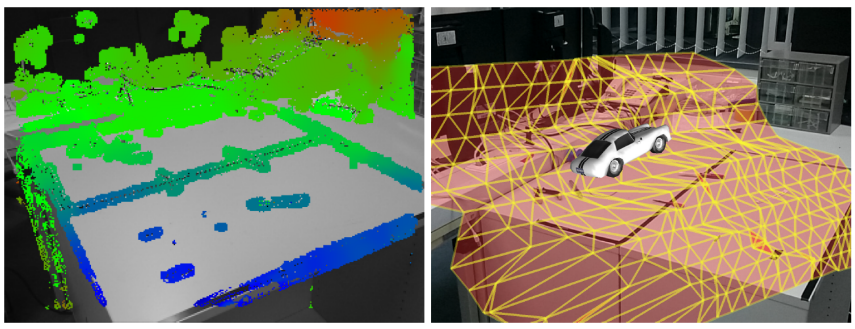


Abbildung 9: Tiefenkarte und Kollisionsnetz
Schöps et al. [2014]

RGB-D SLAM Zu der dritten Kategorie zählen Ansätze, die neben Farbbildern zusätzliche Tiefeninformation verarbeiten. Die zusätzliche Information stammt entweder von strukturiertem Licht oder von Sensoren, die zur Distanzmessung verwendet werden, welche die Lichtlaufzeit messen. Kähler et al. [2015] stellen eine SLAM-Variante mit strukturiertem Licht vor, welche auf mobilen Geräten funktioniert. Abbildung 10 zeigt die digital rekonstruierten Ergebnisse ihrer Methode. Wie bei Ondrúška et al. [2015] werden auch hier geschlossene Oberflächen aus Voxeln erstellt. Die Tiefeninformation wird dabei jedoch anders gewonnen. Die Oberflächen werden mit strukturiertem Licht beleuchtet und aus den Änderungen im Lichtmuster wird die Geometrie berechnet. Dabei kommt ein zusätzlicher Kinect-Sensor zum Einsatz, welcher nicht in herkömmlichen Smartphones verbaut ist.

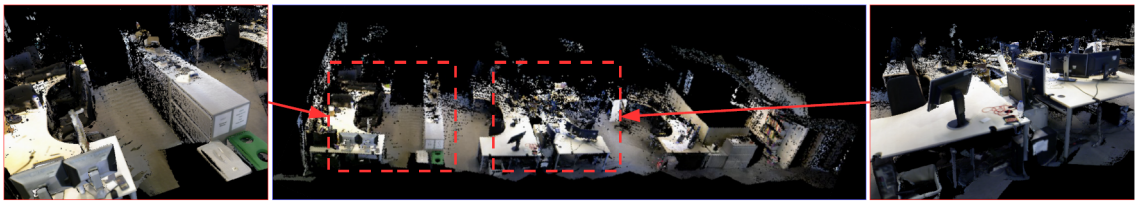


Abbildung 10: Beispiel für RBG-D vSLAM: Kinect Fusion
Kähler et al. [2015]

LiDAR[4] und *ToF*[5]-Sensoren sind hingegen in manchen Smartphones verbaut. Fan et al. [2018] beschreiben ein System, das Höhe, Position und Durchmesser von Bäumen mit einer ToF-gestützten SLAM-Methode erfasst. Dabei handelt es sich um einen Odometrie-Ansatz (vgl. Schöps et al. [2014]), bei dem mittels des ToF-Sensors Punktwolken erstellt werden.

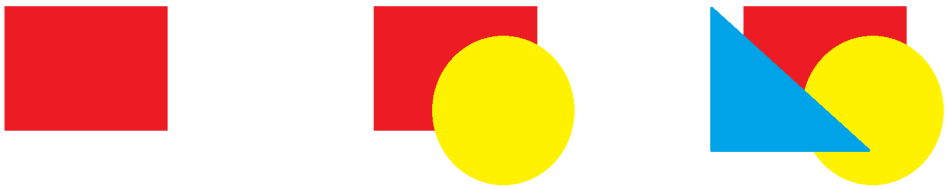
Visuell-inertiale Methoden Neben der Anreicherung um zusätzliche Tiefeninformation kann auch die Bewegung des Smartphones selbst erfasst werden. Die meisten Smartphones besitzen eine inertiale Messeinheit (kurz: *IMU*), welche aus einem Gyroscop und einem Beschleunigungssensor besteht. Das Gyroscop erfasst Rotationen und der Beschleunigungssensor erfasst die Beschleunigung bzw. Bewegung des Smartphones. Vuforia nutzt die IMU in der eigenen visuell-inertialen SLAM-Implementierung.[6]
Jinyu et al. [2019] testen verschiedene SLAM-Ansätze und erstellen vergleichende Datensätze zwischen rein visuellen und visuell-inertialen SLAM-Ansätzen (kurz: *viSLAM*) im AR-Kontext. Dabei stellen sie fest, dass die metrischen Daten der IMU helfen den Projektionsfehler zu minimieren und die absoluten Maße der Szene zu berechnen. Außerdem kann die Bewegungsverfolgung über die IMU fortgeführt werden, wenn keine visuelle Orientierung möglich ist. Zu der Erfassung der Daten nutzten sie Smartphones, zur Berechnung jedoch PCs. Entsprechend ordnen sie die Ergebnisse ein: „Since the computation power of a PC is much bigger than a mobile phone, the SLAM results could not faithfully reflect the actual SLAM effect on a mobile phone. Actually, many SLAM systems cannot perform in real-time on a mobile phone.“ [Jinyu et al., 2019]

[4] LiDAR: Light Detection and Ranging
[5] ToF: Time of Flight
[6] https://library.vuforia.com/environments/device-tracking

2.4.3 Dreidimensionale Objektverdeckung

Neben der korrekten Platzierung und der stabilen Bewegungsverfolgung ist die Objektverdeckung von dreidimensionalen Objekten im Raum ein weiteres Kernproblem von AR-Anwendungen in städtischer Umgebung. Dieser Abschnitt zeigt deshalb grundlegende Möglichkeiten der Objektverdeckung in der Computergrafik, sowie den Umgang mit Objektverdeckung für AR im speziellen.

Verdeckung in rein virtuellen Szenen Der *Painter's Algorithmus* ist eine eine einfache Variante Objektverdeckung zu realisieren. Hierbei wird der Sichtbereich von hinten nach vorne mit Objekten gefüllt. Abbildung 11 veranschaulicht das an einem zweidimensionalen Beispiel. Das rote Rechteck, das sich am weitesten vom Betrachter entfernt befindet, wurde als erstes eingefügt. Dadurch wird es durch die später eingefügten Elemente des gelben Kreises (b) und des blauen Dreiecks (c) verdeckt. Diese Elemente wurden dieser Reihenfolge nach eingefügt und über die vorherigen Elemente gelegt. Das in Abbildung 11(c) eingefügte Dreieck verdeckt sowohl das Rechteck als auch den Kreis. Im dreidimensionalen Raum muss dazu die Distanz der einzelnen Objekte zum Blickpunkt bekannt sein. Für Objektoberflächen wird eine Position anhand ihres geometrischen Zentrums berechnet, anhand derer die Distanz und letztendlich die Projektionsreihenfolge festgelegt wird. Nicht geeignet ist diese Methode für Flächen, welche sich überschneiden oder zyklische Verdeckungsreihenfolgen haben. [Newell et al., 1972]



(a) (b) (c)

Abbildung 11: Veranschaulichung Painter's Algorithm

Ein präziseres Vorgehen ist beim *Z-Puffer*-Algorithmus gegeben, bei dem für jeden Pixel ein Wert für dessen Distanz gespeichert wird. Als Entwickler des Z-Puffer-Algorithmus gelten Catmull [1974] und Straßer [1974], die sich etwa zur gleichen Zeit unabhängig voneinander mit der zweidimensionalen Projektion dreidimensionaler, gekrümmter Flächen beschäftigt und ein ähnliches Verfahren entwickelt haben. Dabei werden Flächen solange unterteilt, bis sie nur noch einem einzelnen Abtastpunkt bzw. Pixel entsprechen.
Abbildung 12 zeigt die Projektion eines dreidimensionalen Würfels in den zweidimensionalen Bildraum und das Raster der Abtastpunkte des Bildraums. Die Darstellung des Würfels ist perspektivisch verzerrt, sodass die Rückseite kleiner als die Vorderseite dargestellt wird. Das verdeutlicht, das für eine akkurate 2D-Repräsentation eines 3D-Objekts, die Distanz der einzelnen Flächen benötigt wird. Diese Distanzinformation wird auch dazu verwendet, um festzustellen ob sich ein Flächenabschnitt im Sichtbereich oder hinter der

Kamera befindet oder ob das Flächenelement verdeckt ist. Da hierbei jede Fläche bis auf Pixelgröße unterteilt wird und für jeden der entstehenden Flächenabschnitte die Sichtbarkeit einzeln geprüft werden kann, sind auch die, beim Painter's Algorithmus als ungeeignet beschriebene, Verdeckungsverältnisse korrekt visualisierbar.

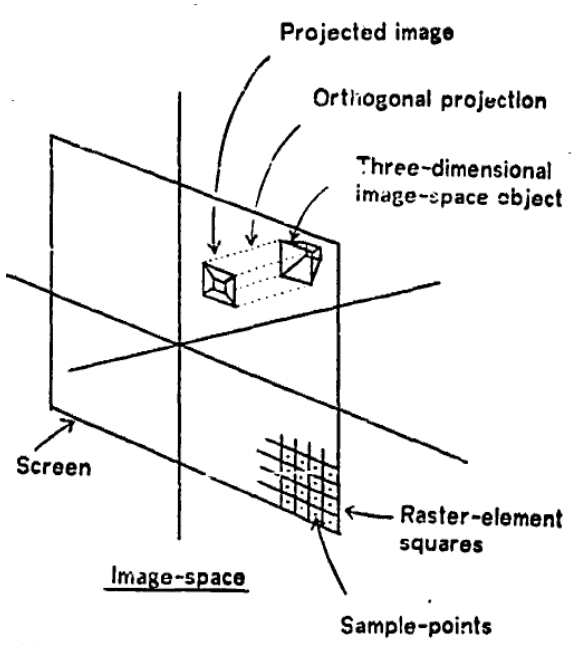


Abbildung 12: Projektion Bildraum
Catmull [1974]

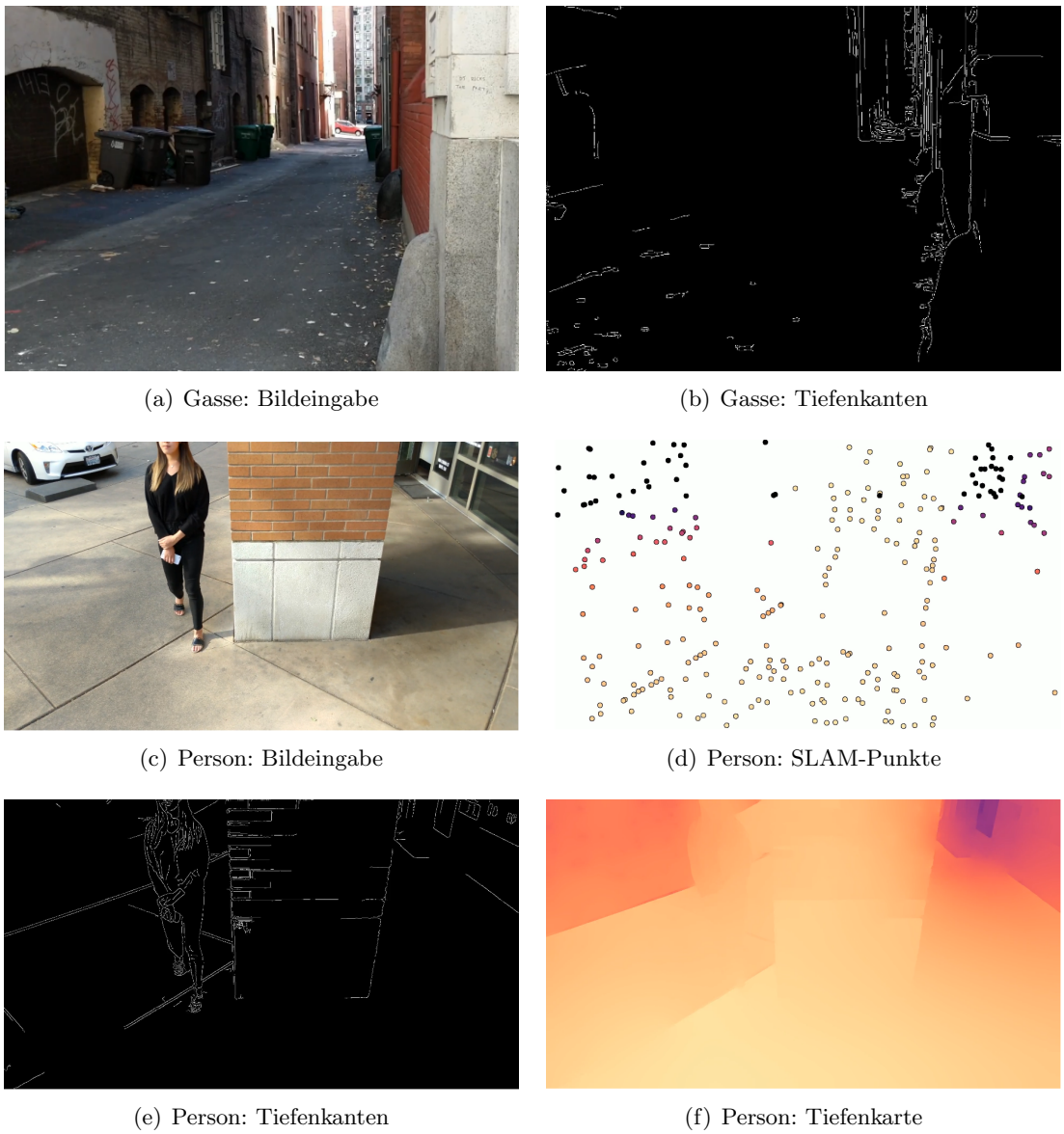
Der Painter's Algorithmus und der Z-Puffer sowie deren Optimierungen und Weiterentwicklungen haben die Gemeinsamkeit, dass die Distanz der dargestellten Objekte zu der Verdeckungsberechnung genutzt wird. In einer rein virtuellen Umgebung sind die Positionen aller Objekte bekannt und ihre Sichtbarkeit kann entsprechend berechnet werden. Auf gleiche Weise kann die Verdeckung aller virtuellen Elemente im AR-Raum stattfinden. Neben virtuellen Elementen sollten im AR-Raum auch die realen Objekte ein realistisches Verdeckungsverhalten zeigen. Das ist eine anspruchsvollere Aufgabe, da die Positionen der realen Objekte nicht von vornherein bekannt sind und für eine Distanzsortierung zunächst erfasst oder berechnet werden müssen.

Für die Verdeckung der Straßenbahn wurden im Rahmen dieser Arbeit digitale Verdeckungselemente verwendet. Die Verdeckungselemente simulieren das Verdeckungsverhalten realer Objekte bzw. Gebäude. Die Funktionsweise dieser Verdeckung entspricht damit Verdeckungsmethoden rein virtueller Szenen mit eingangs bekannten Positionen der einzelnen Objekte. Die tatsächlichen Positionen realer Objekte werden zur Laufzeit nicht erfasst. Die folgenden Abschnitte zeigen, wie der direkte Umgang mit realen Objekten zur Laufzeit realisiert werden könnte.

Nutzerauswahlbasierte AR-Objektverdeckung Für den Verdeckungsfall zwischen realen und digitalen Objekten wurden verschiedene Ansätze entwickelt. In „Real-Time Occlusion Handling in Augmented Reality Based on an Object Tracking Approach“ wurde ein Ansatz gewählt, bei dem mittels Nutzereingaben Vorder- und Hintergrund definiert wurden. Objekte und Flächen, die als Vordergrund definiert wurden, verdecken eingeblendete 3D-Objekte. Objekte im Hintergrund werden von eingeblendeten 3D-Objekten verdeckt. Diese Methode besteht aus den drei Schritten der Auswahl der Verdeckung, der Objektverfolgung und der Verdeckungshandhabung.
Im ersten Schritt wird über ein Nutzerinterface ein reales Objekt gewählt, das das virtuelle Objekt verdecken soll. Dazu bietet das Interface die Möglichkeit Pixel als Vorder- oder Hintergrund zu markieren. Alle Pixel werden entweder als Objekt oder als Hintergrund klassifiziert und der Umriss des Verdeckungsobjekts wird mittels einer graphbasierten Methode festgelegt.
Der zweite Schritt ist die Objektverfolgung. Dazu werden in einem Bild Schlüsselpunkte extrahiert, die das Objekt beschreiben. In einem folgenden Bild werden diese wiedererkannt. Die Schlüsselpunkte dienen als Grundlage zur Schätzung der Objektposition in diesem folgenden Bild. Die genaue Position wird anschließend im Umfeld der geschätzten Position festgelegt.
Im dritten Schritt werden die Pixel innerhalb des Objektumrisses erneut visualisiert um vor dem virtuellen Objekt zu erscheinen und somit die Verdeckungsreihenfolge zu realisieren. [Tian et al., 2010]

Neuronale Netzwerke zur Bestimmung von Verdeckungsobjekten Durch gefaltete Neuronale Netzwerke (kurz: *CNN*, aus dem Englischen für *Convolutional Neural Network*) können die Lage und Position von dreidimensionalen Objekten bestimmt und vorhergesagt werden. Dazu wird ein eingehendes 2D-Farbbild zunächst segmentiert und ein speziell modifiziertes neuronales Netzwerk wird zur Auffindung der Objekte angewandt. Anschließend wird ein CNN auf die extrahierten Objekte angewandt, um deren Lage genauer zu bestimmen. Das CNN ist darauf trainiert die dreidimensionale Lage eines Objekts anhand von zweidimensionaler Projektionen ihrer Hüllkörper bzw. ihrer Bounding Box zu bestimmen. [Rad und Lepetit, 2017]
Hinsichtlich der Laufzeit ist hierbei zu beachten, dass die Implementierung von Rad und Lepetit [2017] 140 ms für die Segmentierung, 130 ms für die Positionsbestimmung und 21 ms für jede Verbesserungsiteration benötigt und diese Werte auf einem Desktop-PC mit einem Intel Core i7-5820K-Prozessor und einer GeForce TITAN X-Grafikkarte erreicht wurden.

Verdichtung von Tiefeninformation Bei dieser Methode wird mit einem SLAM-Algorithmus Tiefeninformation gesammelt und auf alle Pixel übertragen. Es handelt sich um eine direkte SLAM-Methode, die, im Gegensatz zu den vorgestellten direkten Methoden (vgl. 2.4.2), Schlüsselpunkte extrahiert und keine dichten Tiefenkarten generiert [Engel et al., 2017]. Diese spärliche Tiefeninformation wird anschließend verdichtet. Nach der initialen groben Rekonstruktion mittels der genannten SLAM-Methode, werden weiche Tiefenkanten aus einer Folge von drei Bildern unter der Analyse des optischen Flusses berechnet. Mit einem modifizierten Canny-Kantendetektor werden dann die präzisen Tiefenkanten erkannt. Über eine Poisson-Gleichung wir die Tiefeninformation der SLAM-Punkte



(a) Gasse: Bildeingabe

(b) Gasse: Tiefenkanten

(c) Person: Bildeingabe

(d) Person: SLAM-Punkte

(e) Person: Tiefenkanten

(f) Person: Tiefenkarte

Abbildung 13: Verdichtung von Tiefeninformation[7]

auf alle Pixel entsprechend verrechnet. Dieses Vorgehen ermöglicht einerseits scharfe Verdeckungskanten an den Rändern von Objekten und ein gleichförmiges Verdeckungsverhalten durch deren Flächen und eignet sich auch für sich bewegende Verdeckungsobjekte. [Holynski und Kopf, 2018]
Abbildung 13 zeigt Ausschnitte aus dem Zusatzmaterial zu „Fast Depth Densification for Occlusion-aware Augmented Reality“. Abbildung 13(a) zeigt eine urbane Szene einer Gasse mit Mülltonnen und einer Straße im Hintergrund, auf der ein Auto vorbeifährt und auf (b) sind die zugehörigen erkannten Tiefenkanten zu sehen. Die Konturen der Mülltonnen

[7]Quelle: https://grail.cs.washington.edu/projects/occlusion/supplementary.html, aufgerufen: 15.06.2023

links werden hier nicht erkannt, die Konturen des Autos auf der besser belichteten Straße im Hintergrund schon.
Die Abbildungen (c)-(f) zeigen eine Szene im Freien mit einer laufenden Person. Dabei zeigt (c) das eingegebene Videobild. Auf (d) sind die entsprechenden SLAM-Punkte zu sehen. Die helleren gelben Punkte sind der Kamera näher und mit zunehmender Entfernung werden sie sie dunkler. Rechts der Bildmitte sind die gelben, nahen Punkte der Säule im Vordergrund zu sehen und die diagonal angeordneten Punkte von der linken unteren Bildecke zum Zentrum entsprechen der Fuge zwischen den Platten. Die Position der Person kann anhand fehlenden SLAM-Punkte ausgemacht werden. Abbildung (e) zeigt die Tiefenkanten, welche zusammen mit der verdichteten Tiefeninformation aus den SLAM-Punkten zu der Tiefenkarte (f) verrechnet werden. Abbildung (f) zeigt die resultierenden homogenen, klar abgegrenzten Flächen der Tiefenkarte sowie die weniger gut abgegrenzte Person.
Diese Beispiele zeigen, dass dieser Ansatz mit den Problemfelder der großen Distanz in urbanen Räumen sowie die Verdeckung durch bewegende Personen bereits in Teilen umgehen kann. Holynski und Kopf [2018] erreichen mit ihrer Implementation eine durchschnittliche Laufzeit von 48.3ms pro Bild auf einem PC mit einer Intel i7-6800K CPU bei getesteten 2-Megapixel-Videos. Sie gehen jedoch davon aus, dass eine Echtzeitanwendung auf Smartphones realisierbar ist, da 98% der Laufzeit auf die Schritte SLAM, Canny-Kantendetektion und das Lösen der Poisson-Gleichung entfallen, und es von SLAM bereits laufzeitfähige Implementationen gibt, ein optimierter Canny-Kantendetektor auf Smartphones schneller ist als ihre PC-Implementation und auch Poisson-Gleichungen mit speziellen Optimierungen zur Laufzeit gelöst werden können. [Holynski und Kopf, 2018]

Dieser Abschnitt hat die wichtigsten Grundlagen von AR-Anwendungen vorgestellt. Es wurde gezeigt, dass Schlüsselpunkte mittels Eck- und Bloberkennung gefunden werden können, dass der Abgleich der gefundenen Schlüsselpunkte über den direkten Abgleich von Bildausschnitten, über Deskriptoren oder mittels einer Klassifizierung realisiert werden kann und wie die Objektpositionierung anhand gefundener Punkte umgesetzt werden kann.
Für die Bewegungsverfolgung während der Anwendung wurden verschiedene SLAM-Methoden vorgestellt. Featurebasiertes SLAM nutzt Schlüsselpunkte zur Verfolgung der Umgebung und der entsprechenden relativen Position des Smartphones zur Umgebung. Direkte SLAM-Methoden nutzen die eingehenden Farbbilder für die Erstellung eines Umgebungsmodells, anhand dessen die Position bestimmt wird. Bei RGB-D SLAM und visuell-inertialen Methoden kommen zusätzliche Sensoren zum Einsatz, um entweder präzise Distanzen erfassen oder die Lage des Smartphones bestimmen zu können.
Bei der Objektverdeckung wurde gezeigt, wie diese im rein virtuellen Raum realisiert werden kann und im Raum der gemischten Realität mit dem Mangel an Positionsinformation realer Objekte umgegangen werden kann. Durch Nutzereingaben können Verdeckungsobjekte ausgewählt werden. Außerdem können Verdeckungsobjekte durch neuronale Netzwerke oder aufbauend auf SLAM-generierter Tiefeninformation identifiziert werden.
Das folgende Kapitel erläutert nun, wie die ehemalige Bamberger Straßenbahn zurück auf die Straße gebracht wurde und welche Technologien und Methoden dazu gewählt wurden.

Kapitel 3

Konzept

Im Rahmen dieser Arbeit wurde eine Anwendung entwickelt, welche historische Objekte, die nicht mehr im Stadtbild zu sehen sind, mittels AR wieder in das Stadtbild projiziert. Dazu wurde das Beispiel der ehemaligen Bamberger Straßenbahn gewählt. Diese soll an einem bestimmten Ort über das Display eines Smartphones angezeigt werden und sich die Straße entlang bewegen.

Der Grundablauf sieht folgendermaßen aus:
1. Die Anwendung wird vor dem Alten Rathaus gestartet.
2. Die Anwendung erkennt die Umgebung.
3. Das Modell der Straßenbahn wird auf dem Display in die Umgebung projiziert.
4. Die Straßenbahn bewegt sich entlang der Straße.

Damit dies gelingt, sind verschiedene Erstellungsschritte nötig. Dieses Kapitel zeigt verschiedene Möglichkeiten die einzelnen Schritte einer AR-Anwendung umzusetzen und geht darauf ein, welche Methoden und Technologien im Rahmen dieser Arbeit zum Einsatz kommen und warum. Da es sich um eine ortsbezogene Anwendung handelt, wird zunächst dargestellt, mit welchen Mitteln der Ortsbezug realisiert werden kann. Es wird gezeigt, wie historische 3D-Modelle entstehen können und mit welcher Software und welchen *Software Development Kits* (kurz: *SDKs*) die AR-Funktionalität realisiert werden kann.

Die Grundlagen dieser Arbeit wurden in einem Universitätsprojekt im Sommer 2022 geschaffen. Ziel dieser Projektarbeit war ebenfalls die Integration der ehemaligen Straßenbahn in die Bamberger Karolinenstraße mittels AR, welches im Rahmen dieses Projekts jedoch nicht erreicht wurde [Wagner et al., 2022]. Diese Masterarbeit baut auf der Kernidee dieses Projekts auf und realisiert eine prototypische Umsetzung.
Für das Konzept ergeben sich Parallelen bei der räumlichen Verankerung mittels Bilderkennung, der Auswahl der Quellen und der Softwareauswahl. Im Gegensatz zur Projektarbeit, welche ARCore nutzte, kommt in dieser Masterarbeit Vuforia als AR-SDK zum Einsatz. Die folgenden Abschnitte beschreiben dieses Konzept im Detail.

3.1 Räumliche Verankerung des 3D-Objekts

Das Modell der Bamberger Straßenbahn soll sich entlang der Karolinenstraße bis zur Herrenstraße bewegen. Die Karolinenstraße liegt in der Bamberger Altstadt und verbindet das alte Brückenrathaus mit dem Bamberger Dom. Dabei handelt es sich um eine relativ schmale Straße mit schmalen Gehwegen. Durch ihre zentrale Lage zwischen den genannten Bamberger Sehenswürdigkeiten ist sie neben dem Automobil- und Anwohnerverkehr auch durch Touristen frequentiert. Um die Straßenbahn mit diesem Ort zu verknüpfen, werden nun markierungslose Möglichkeiten wie die manuelle Nutzereingabe, GPS- und Sensorennutzung und markierungsbasierte Möglichkeiten wie Bilderkennung sowie dreidimensionale Umgebungserkennung erläutert und es wird erklärt, warum die Bilderkennung als Lokalisierungsmethode gewählt wurde.

Auslösung durch Nutzereingabe

Eine einfach zu implementierende Möglichkeit der Einblendung der Straßenbahn wäre, den Nutzer die Einblendung auslösen zu lassen. Der Nutzer befindet sich in der Karolinenstraße und drückt in der Anwendung den Knopf zur Einblendung der Straßenbahn und sie wird auf dem Display des Smartphones in der Umgebung dargestellt.
Vorteile dieser Methode wären, dass sie einfach zu implementieren ist und unabhängig von äußeren Faktoren funktioniert. Die Unabhängigkeit von äußeren Faktoren ist aber auch ein entscheidender Nachteil. Ein Ortsbezug wird nicht hergestellt. Die Straßenbahn kann an jedem beliebigen Ort eingeblendet werden.
Um einen Ortsbezug herzustellen, müssten weitere Schritte erfolgen. So könnte am entsprechenden Ort eine Benachrichtigung darauf hinweisen, dass hier der Punkt ist, an dem die Einblendung ausgelöst werden soll. Dieser Hinweis könnte physisch platziert oder über die Anwendung selbst angezeigt werden. Eine Einblendung in der Anwendung würde eine weitere Methode der Lokalisierung benötigen. Die Erstellung von physischen Markierungen geht über die reine Softwareentwicklung hinaus. Physische Markierungen können in der belebten Stadt leicht übersehen werden und lassen sich auch nicht uneingeschränkt in permanenter Form anbringen. Unabhängig davon, auf welche Art und Weise eine Benachrichtigung auf den Ort hinweist, kann die Präzision der Einblendung nicht sichergestellt werden. Je nachdem auf welcher Höhe sich das Smartphone bei der Auslösung befindet, oder in welche Richtung es zeigt, kann die eingeblendete Straßenbahn in der Luft fliegen, im Boden versinken, in eine falsche Richtung oder durch die nächste Hauswand fahren.
Um diese Problemfälle abzufangen, müssten weitere Technologien genutzt werden. Oberflächenerkennung könnte zum Abgleich mit der tatsächlichen Straßenoberfläche genutzt werden und die richtige Position und Ausrichtung des Smartphones könnte mit dem Abgleich mit GPS-Daten sichergestellt werden.

GPS

Über GPS und die Smartphoneausrichtung ließe sich die Straßenbahn auch direkt einblenden. *GPS* steht für *Global Positioning System* und berechnet die Postion einer GPS-fähigen Geräts anhand von Satellitensignalen. Dazu müssen mindestens Signale von vier GPS-Satelliten empfangen werden. Aus der Position und der Entfernung der Satelliten wird dann die Position des Empfangsgeräts ermittelt [El-Rabbany, 2002].

Mit Hilfe des Gyroskops kann auch die Richtung bestimmt werden, in welche das Smartphone zeigt. Das Gyroskop misst dazu die Rotationsbeschleunigung des Smartphones und errechnet daraus die Position [Passaro et al., 2017].
Eine GPS-basierte Objektpositionierung wird in „An Architecture for Mobile Outdoors Augmented Reality for Cultural Heritage“ von Panou et al. [2018] mit dem Ergebnis vorgestellt, dass die durchschnittliche GPS-Genauigkeit drei Meter betrug und damit die 3D-Objekte nicht präzise genug platziert werden konnten um sie passend in die Umgebung zu integrieren. In „Markerless Location Based Augmented Reality Application for Showcasing Deals“ von Khan und Soroni [2022] wird außerdem festgestellt, dass die Objekte nicht persistent dargestellt werden.
Für einige Anwendungen kann diese grobe Platzierung mit geringer Genauigkeit genügen. Im Fall der Bamberger Straßenbahn soll das Modell jedoch realistisch in die Umwelt integriert werden. Um sicherzustellen, dass sie auf der Straßenoberfläche fährt und nicht abseits davon, reicht die Genaugkeit von GPS zur Objektplatzierung jedoch nicht aus.

Lokale Sensoren

Eine präzisere Ortungsmöglichkeit bieten lokale Sensoren [Prigge und How, 2004]. Der Aufbau eines solchen Sensorennetzes ist mit hohen Kosten und Aufwänden verbunden und stellt keine allgemein anwendbare Lösung dar. Eine Softwarearchitektur die auf solch ein System aufbaut, verlangt von jedem Ort, an dem sie eingesetzt werden soll, dass dieser mit Sensoren versehen ist. In Innenräumen, wie bspw. Ausstellungen, kann das möglich und sinnvoll sein [Cheng et al., 2017]. In der Stadt erschwert diese Anforderung die inhaltliche Erweiterung und steht der Übertragbarkeit im Weg. Im öffentlichen Raum wirft das Anbringen von Ortungssensoren außerdem rechtliche und gesellschaftliche Fragen auf. Im Fall der Welterbestadt Bamberg wäre auch die physische Anbringung selbst eine Hürde, da große Teile der Innenstadt unter Denkmalschutz stehen.
Aufgrund der hohen Kosten und der Hürden und der erschwerten Übertragbarkeit kam ein solches System für diese Arbeit nicht in Frage.

Bilderkennung

Bei der Bilderkennung wird ein hinterlegtes Bild mit einem neuen Bild verglichen. Dazu wird das hinterlegte Bild vorverarbeitet. Es werden anhand der Farbwerte markante Punkte erkannt. Diese Punkte werden auch als *Schlüssel-* oder *Featurepunkte* bezeichnet. Meist kommen dazu Schwarz-Weiß-Bilder zum Einsatz, da nur die Graustufen der Bilder zur Berechnung der Schlüsselpunkte verwendet werden [Petrou und Petrou, 2010]. Bei dem neuen Bild handelt es sich um das Bild, welches mit der Smartphonekamera erfasst wird. In dem neuen Bild wird nach markanten Bildpunkten gesucht und sie werden mit den bekannten Schlüsselpunkten des hinterlegten Bildes verglichen. Stimmen diese überein, wird das Bild erkannt. Eine detailliertere Erklärung zum Ablauf der Bilderkennung folgt im Abschnitt 2.4.1.
Das mit dem *Vuforia Developer Portal*[1] erstellte Abbildung 14 zeigt am Beispiel des alten Bamberger Rathauses die markanten Bildpunkte, welche zur Erkennung dieses Bildes relevant sind. Die gelben Kreuze markieren diese Erkennungspunkte. Die Erkennung des Bildes kann nun sowohl als Auslöser für die Einblendung der Straßenbahn dienen, als auch

[1] https://developer.vuforia.com/vui/develop/

zu deren genauen Positionierung. Da ein Bild in der realen Umgebung erkannt wird, ist damit auch die Position festgelegt. Vorab wird die relative Position der Straßenbahn zu dem erkannten Bild definiert und erscheint bei erfolgreicher Bilderkennung an der entsprechenden Position in der Umgebung.
Dabei ist der Erfolg der Bilderkennung von externen Faktoren abhängig. Die Anwendung soll die Straßenbahn in die Stadt projizieren und hat somit eine kaum kontrollierbare Umgebung. Abbildung 14 verdeutlicht das daran, dass die untersten erkannten Punkte unter dem Torbogen und rechts davon zu Personen gehören, welche sich zur Zeit der Aufnahme dort befanden. Damit sind sie für die Erkennung dieses Bildes mit relevant. Dies gilt es bei der Auswahl des Bildes für die Bilderkennung zu berücksichtigen und zu vermeiden.

Abbildung 14: Markante Punkte zur Bilderkennung

Dreidimensionale Erkennung

Neben der Erkennung von Bildern ist es auch möglich dreidimensionale Objekte oder sogar ganze Areale zu erkennen. Auch diese Methode kann zur präzisen Platzierung der Straßenbahn genutzt werden und ist von externen Faktoren zur Erkennung abhängig. Ein erkanntes dreidimensionales Areal kann die Flexibilität und die Stabilität bei der Nutzung der Anwendung erhöhen. Wie bei der Bilderkennung werden auch hier Vergleichsdaten benötigt. Zum Abgleich der dreidimensionalen Umgebung wird entsprechend ein dreidimensionales Modell der Umgebung benötigt. Ein solches Modell muss vorab erstellt werden. Für die Erstellung eines solchen Umgebungs- oder Geländemodells werden Vermessungsmethoden wie Photogrammetrie oder Laserscanner benötigt.

Die im Folgenden beschriebene Anwendung realisiert die Positionierung der Straßenbahn mittels Bilderkennung. Dieser Abschnitt hat unterschiedliche Positionierungs- möglichkeiten vorgestellt. Eine individuelle Nutzereingabe als Positionierungsmethode kann dabei keine

brauchbaren Ergebnisse garantieren. GPS ist nicht präzise genug und ein lokales Postionierungssystem ist zu aufwändig und teuer. Aufwände und Kosten schließen auch die Erkennung dreidimensionaler Objekte oder Areale aus. Die Bilderkennung wurde gewählt, da es sich um eine einfache und günstige Methode handelt, die eine hohe Präzision bei der Objektpositionierung bietet und leicht auf andere Orte übertragbar ist.

3.2 Grundlagen für 3D-Modell und Animationspfad

Dieser Abschnitt zeigt, welche Probleme es mit 3D-Modellen als historische Abbildung gibt, welche Arten von Quellen die Modellierungsgrundlage darstellen können und welche Modellierungsmöglichkeiten sich daraus ergeben. Die Quellen werden dabei in vier Kategorien unterteilt und es wird gezeigt, auf welchen Quellen das Modell der Bamberger Straßenbahn und ihr Animationspfad basieren.
Zunächst ist festzuhalten, dass ein 3D-Modell eines historischen Objekts oder Gebäudes diesem grundsätzlich nicht als exakte Rekonstruktion gerecht werden kann. Zum großen Teil liegt das an lückenhaften Quellen. Hinzu kommen technologische Einschränkungen und einem Interpretationsgehalt der Person, welche das Modell erstellt. Auch ein Modell, welches auf der Digitalisierung eines vollständig erhaltenen Objekts basiert und jedes Detail des realen Objekts zeigt, kann nicht den Anspruch haben, das Objekt in seinem historischen Kontext korrekt abzubilden. Ein solches Modell zeigt den aktuellen Zustand eines Objekts mit einer technisch definierten Präzision. Das bedeutet nicht, dass das digitalisierte Objekt immer genau diesen Zustand hatte. Denkt man beispielsweise an ein mittelalterliches Haus, das mehrere Jahrhunderte alt und dennoch gut erhalten ist, erscheint es wahrscheinlich, dass es seit seinem Bau Instandhaltungsarbeiten, Renovierungen und Veränderungen daran gegeben hat. Das Haus ist noch das selbe, doch sein heutiger Zustand und sein heutiges Erscheinungsbild garantieren keine korrekte historische Abbildung.
3D-Objekte bieten jedoch einige Möglichkeiten mit der Problemstellung der historischen Rekonstruktion umzugehen, die bei einer physischen Rekonstruktion nicht, oder schwieriger umzusetzen wären. Ein 3D-Modell kann leichter erstellt und verändert werden. Außerdem kann ein 3D-Modell verschiedene Zustände haben und sein Erscheinungsbild zur Laufzeit verändern. Ein Modell kann somit erstellt werden und nachträglich noch angepasst werden. Es können auch mehrere unterschiedliche Modelle zu einem Objekt bestehen. Dadurch können beispielsweise Veränderungen des Objekts im Laufe der Zeit sichtbar gemacht werden. Auch ist es möglich verschieden Bauphasen an bestehenden Gebäuden zu visualisieren oder Modellierungen aufgrund ihrer Quellenlage optisch voneinander abzuheben. Zusätzlich lassen sich auf digitale Weise Animationen oder Interaktionsmöglichkeiten implementieren. Diese Visualisierungsmöglichkeiten können es erleichtern, die dargestellten Informationen aufzunehmen und zu verstehen und können für Transparenz zur Entstehung dreidimensionaler Rekonstruktionen beitragen. Physische Rekonstruktionen würden oftmals eine Beschädigung der Originalsubstanz bedeuten. Digitale Rekonstruktionen oder Erweiterungen lassen diese unangetastet.

Arten von Quellen

Dieser Abschnitt geht darauf ein, welche Arten von Quellen zur digitalen Rekonstruktion herangezogen werden können und welche Bedeutung die Verwendung der jeweiligen Quelle für das Resultat hat. Bestehende Objekte stellen eine der vier Kategorien dar, auf welchen Modellierungen basieren können [Hameed et al., 2022]. Die zweite Kategorie sind Bildquellen, die dritte Kategorie sind Beschreibungen und die vierte Kategorie sind kontextbasierte Quellen.

Bestehende Objekte Bestehende Objekte können mittels Photogrammetrie, Laser- und Streifenlichtscannern digital erfasst werden. Zur Photogrammetrie werden digitale Fotos von einem Objekt aus verschiedenen Perspektiven gemacht und miteinander zu einem dreidimensionalen Objekt verrechnet. Streifenlichtscanner projizieren Lichtstreifen auf Oberflächen im Nahbereich und errechnen ein 3D-Objekt aus den erfassten Konturlinien der Lichtstreifen. Laserscanner errechnen die Distanz zu einem einem Objekt aus der Zeitdifferenz zwischen dem Entsenden und dem Empfangen eines Lasers. Auf diese Weise werden einzelne Punkte vermessen. Durch eine Vielzahl von Messungen, werden Punktwolken erstellt, welche dann zu 3D-Objekten verbunden werden können.
Eingeschränkt werden diese Vermessungen durch Verdeckungen an Objekten, an denen die Objektoberfläche nicht erfasst werden kann. Auch verschiedene Oberflächeneigenschaften können die digitale Erfassung beeinträchtigen. Problematisch sind dabei spiegelnde, durchsichtige und auch schwarze Oberflächen.
Ein Leitfaden der zu berücksichtigenden Schritte bei der Digitalisierung von Objekten des Kulturerbebereichs wird im Artikel „Integrated Data Capturing Requirements For 3D Semantic Modelling of Cultural Heritage: The Inception Protocol“ [Di Giulio et al., 2017] vorgestellt. Neben der direkten digitalen Erfassung können bestehende Objekte als Anschauungsmaterial das Verständnis für das Objekt stärken und als Quelle der Modellierung dienen oder die Ausgangslage für Bilder darstellen.

Bildquellen Bildquellen können in verschiedenen Formen und Detailstufen vorliegen. Sie reichen von Skizzen und Malereien bis hin zu Fotografien und Plänen. Pläne sind besonders für die genauen Maße nützlich. Fotografien können den tatsächlichen Zustand eines Objekts zu einem bestimmten Zeitpunkt zeigen und Malereien dienen als zusätzliche Hinweise auf die Farbgebung.

Beschreibungen Wie Bildquellen, können auch Beschreibungen sehr unterschiedlich sein. Eine technische Beschreibung, die Maße, Winkel und Beschaffenheiten genau beschreibt, kann eine ähnliche Funktion wie der Schnitt aus Abbildung 15 haben. Ein Bild kann jedoch bei der Modellierung einfacher genutzt werden. Dies wird im Kapitel 4 gezeigt. Beschreibungen können auch oberflächlich und subjektiv sein.

Kontext Mit kontextbasierten Quellen sind sind solche gemeint, die darauf hinweisen, dass ein bestimmtes Objekt existiert hat, dabei aber keine direkte Quelle für das Objekt selbst darstellen. Im Kontext der Straßenbahn sind das alle Teile, die für die Funktion der Straßenbahn nötig sind, selbst aber nicht dokumentiert sind. Aus dem Wissen über dokumentierte Teilen, Funktion und vergleichbare Objekten können Lücken annäherungsweise gefüllt werden.

Quellen des Straßenbahnmodells

Für das 3D-Modell der Bamberger Straßenbahn lagen Pläne, Fotografien und Malereien vor. Abbildung 15 zeigt einen Längsschnitt der Straßenbahn mit Abmessungen. Die rechte Hälfte der Abbildung zeigt die Außenansicht, die linke die Innenansicht.

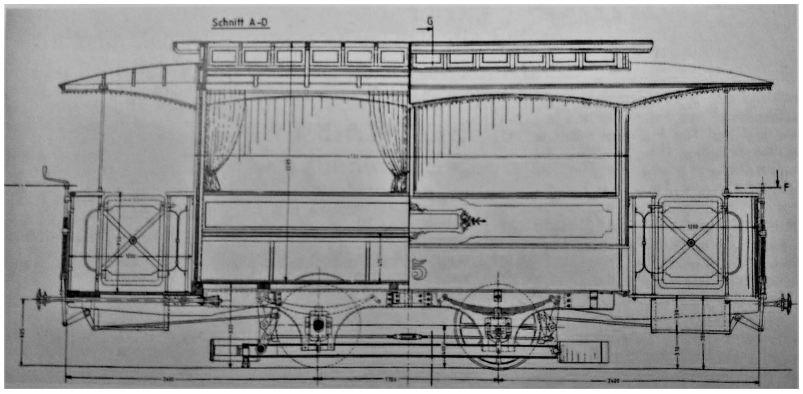


Abbildung 15: Schnitt der Straßenbahn
Brehm [1991]

Abbildung 16: Farbgebung der Straßenbahn
Wußmann [2018]

Abbildung 16 zeigt die Straßenbahn auf dem Titelbild des Buches „Die Straßenbahn kommt“ von Wußmann [2018] und diente als Grundlage der Farbgebung der Straßenbahn.
Manche Details waren durch die Bildquellen jedoch nicht vollständig abgedeckt. Es fehlten Darstellungen aus dem Innenraum, die Krümmung des Daches war nicht genau dokumentiert und mechanische Details lassen sich nur erahnen. Um ein vollständiges und geschlossenes Modell der Straßenbahn zu erhalten, wurden diese Details dennoch modelliert. Als Vergleichsobjekt diente hier die alte Nürnberger Straßenbahn, von welcher Bilder aus dem Innenraum zu finden sind.

Quellen der Animationsroute

Da sich das Straßenbahnmodell entlang der Karolinenstraße bewegen soll, musste neben der Straßenbahn selbst auch der Animationspfad erstellt werden. Um die Straßenbahn in das aktuelle Stadtbild integrieren zu können, diente die heutige Karolinenstraße als Grundlage des Animationspfades. Mögliche Änderungen zwischen dem damaligen und dem heutigen Straßenverlauf wurden nicht berücksichtigt. Die Quelle ist hier also ein bestehendes Areal. Die Erfassung eines Areals hat technologische Überschneidungen zu der Objekterfassung, bringt aber auch eigene Herausforderungen mit sich. Um aus den Straßenverlauf in einen digitalen Animationspfad zu Überführen gibt es die Möglichkeiten, auf bestehende Geodaten zurückzugreifen oder die Straße zu vermessen.

Abbildung 17: Verlauf der Karolinenstraße

Bestehende Geodaten sind online bei Karten- und Navigationsdiensten zu finden. Abbildung 17 zeigt den Verlauf der Animationsstrecke und die Problematik mit bestehenden Geodaten am Beispiel des BayernAtlas[2].
Im oberen Bereich ist der Streckenverlauf in der Draufsicht dargestellt. Am unteren Rand ist das Höhenprofil zu sehen und am linken Rand finden sich zusätzliche Informationen und Auswahlmöglichkeiten zur Fortbewegung. Aus dem Streckenverlauf und dem Höhenprofil ließe sich ein dreidimensionaler Bewegungspfad erstellen. Die zugrundeliegenden Daten sind jedoch zu ungenau um einen Bewegungspfad zu ermöglichen, der die Straßenbahn realistisch durch die Straße bewegen würde. Der Verlauf in der Draufsicht verbindet gerade Streckenabschnitte. Wo diese aufeinandertreffen entsteht eine Ecke. Der Streckenverlauf ist angenähert und es fehlen Details wie eine leichte S-Kurve im Kreuzungsbereich der Karolinenstraße und der Ringleinsgasse. Die Straßenbahn befuhr die Karolinenstraße vom Rathaus bis zur Herrenstraße, in welche sie dann abbog. Die Kreuzung zur Ringleinsgasse liegt somit außerhalb des Animationspfades, verdeutlicht aber dennoch ein grundsätzliches Problem mit zu ungenauen Daten. Auch das Höhenprofil ist unzureichend, da es sehr grob ist und dadurch wichtige Informationen fehlen. So müsste nach dem Start zunächst ein Gefälle angezeigt werden, da es vom Startpunkt am alten Rathaus zunächst nach unten geht. In diesem Straßenabschnitt soll die Straßenbahn angezeigt werden und die fehlerhaften Daten sind für eine korrekte Visualisierung nicht zu nutzen.

[2]Quelle: https://geoportal.bayern.de/bayernatlas/, aufgerufen: 09.02.2023

Eine präzise Vermessung ist mit Laserscannern möglich, kam jedoch wegen des erheblichen Aufwands und eingeschränkter Verfügbarkeit nicht in Frage. Als Grundlage für den Animationspfad sollten deshalb selbst eingemessene GPS-Daten dienen. In der Objektpositionierung ist GPS problematisch, da Objekte um mehrere Meter neben ihrer geplanten Position erscheinen können. Hier war die Idee, dass die Fehler der geringen Vermessungsgenauigkeit durch mehrfaches Messen minimiert werden können und ein konstanter Fehler bzw. eine konstante Verschiebung aller Messpunkte vernachlässigt werden kann, da nur die relativen Positionsänderungen der Messpunkte für die Animationsroute benötigt werden.

Dieser Abschnitt hat auf die eingeschränkte Aussagekraft von rekonstruierten 3D-Modellen als historisches Abbild hingewiesen und Möglichkeiten damit umzugehen gezeigt. Es wurden verschiedene Arten von Quellen vorgestellt, welche die Grundlage von 3D-Modellen darstellen können und es wurden die wichtigsten Bildquellen gezeigt, die zur Entstehung des Straßenbahnmodells herangezogen wurden. Außerdem wurde gezeigt, auf welcher Grundlage Lücken der Bildquellen bei der Modellierung aufgefüllt wurden und das GPS als Grundlage für den Animationspfad genutzt werden sollte.

3.3 Auswahl der Software

Dieser Abschnitt stellt die Software vor, welche zur Erstellung der beschriebenen Anwendung gewählt wurde. Der Schwerpunkt liegt dabei auf dem Vergleich verschiedener AR-SDKs. ARKit, ARCore und Vuforia sind die verglichenen SDKs. Es werden Spezifikationen und Ergebnisse anderer wissenschaftlicher Artikel gezeigt.

Modellierung und Animation

Zur Erstellung des Straßenbahnmodells wurde die kostenlose und quelloffene Software *Blender* verwendet. Blender ermöglicht unter Anderem Modellierung und Texturierung von Objekten. Blender lässt sich einfach erweitern, hat eine aktive Nutzergemeinde und eine gute Dokumentation[3].
Mit *Unity* wurde die Animation umgesetzt. Unity ist eine Spiel-Engine und kann 3D-Objekte mit Interaktionsmöglichkeiten und Logik, wie z.B. physikalische Eigenschaften, erweitern. Unity ist gut dokumentiert[4] und lässt sich leicht im Zusammenspiel mit Blender verwenden. Hanafi et al. [2019] beschreibt Unity als „unquestionably the most comfortable experience available with no competitor capable of such fluent and reliable tool“. Unity bietet mit ARFoundation ein eigenes AR-Framework und ermöglicht die Integration von weiterer AR-SDKs. Für die Zwecke dieser Arbeit reicht bereits die kostenlose Variante von Unity.

[3] https://docs.blender.org/, aufgerufen: 13.02.2023
[4] https://docs.unity3d.com/Manual/index.html, aufgerufen: 13.02.2023

AR-SDKs

Um aus einem modellierten und gegebenenfalls animierten Objekt eine AR-Anwendung zu kreieren, werden AR-Funktionalitäten benötigt. Diese liefern AR-spezifische *Software Development Kits* (kurz: *SDKs*).
SDKs übernehmen dabei folgende Kernaufgaben: Umgebungserkennung, Verankerung des 3D-Objekts im Raum, Postions- und Bewegungsverfolgung und die jeweils entsprechende Visualisierung des 3D-Objekts. Darüber hinaus können AR-SDKs viele weitere Aufgaben übernehmen. Der Funktionsumfang, welche Technologien verwendet werden, welche Voraussetzungen für ihren Gebrauch erfüllt sein müssen und welche Geräte unterstützt werden, variiert zwischen den verschiedenen SDKs.
Es existieren bereits einige Artikel, welche AR-SDKs miteinander vergleichen. In den vergangenen Jahren haben sich beispielsweise Amin und Govilkar [2015], Hanafi et al. [2019] und Oufqir et al. [2020] mit diesem Thema beschäftigt. Da sich AR-Technologien jedoch stetig weiterentwickeln, folgt nun eine aktualisierte Gegenüberstellung von ARKit[5], ARCore[6] und Vuforia[7].

Allgemeines	ARKit	ARCore	Vuforia
iOS	Ja	Ja	Ja
Android	Nein	Ja	Ja
Preis	Kostenlos	Kostenlos	Lizenzabhängig
Quelloffen	Nein	Ja	Nein

Tabelle 1: AR-SDKs Allgemeine Informationen

Tabelle 1 zeigt allgemeine Informationen zu den verschiedenen SDKs. Sie zeigt, dass ARCore und Vuforia zur Entwicklung für Android und iOS-Geräte genutzt werden können, das von Apple entwickelte ARKit jedoch ausschließlich für iOS. In der Folge wird noch das von Unity unterstützte Framework ARFoundation angesprochen, welches es ebenfalls ermöglicht für die beiden großen Smartphoneplattformen gleichzeitig zu entwickeln.
Weiter geht aus Tabelle 1 hervor, dass ARCore das einzige quelloffene SDK ist.
Alle drei SDKs können kostenlos genutzt werden, jedoch beinhaltet die kostenlose Version von Vuforia nicht den gesamten Funktionsumfang und beschränkt den Nutzen einzelner Funktionen.
Tabelle 2 zeigt den unterstützten Funktionsumfang der einzelnen SDKs. Die Erkennung von einzelnen oder mehreren Bildern und von Oberflächen bilden zusammen mit der simultanen Lokalisierung und Kartierung (aus dem Englischen: Simultaneous Localization and Mapping, kurz: *SLAM*) die Grundbausteine für AR-Anwendungen, welche von allen drei SDKs unterstützt werden. SLAM ermöglicht die Erfassung der Positionsänderung des Smartphones zur Laufzeit. Außerdem unterstützen alle drei SDKs Cloud-Anker, welche ein einfaches Teilen von Anwendungen ermöglicht. Die Ankerpunkte, welche die AR-Anwendung auslösen, liegen dann nicht im lokalen Speicher des Endnutzergeräts, sondern werden auf einem Server gehostet und sind somit leichter für mehrere Nutzer erreichbar.

[5] https://developer.apple.com/documentation/arkit, aufgerufen: 14.02.2023
[6] https://developers.google.com/ar/develop?hl=de, aufgerufen: 14.02.2023
[7] https://library.vuforia.com/, aufgerufen: 14.02.2023

Funktion	ARKit	ARCore	Vuforia
Bilderkennung	Ja	Ja	Ja
Multibilderkennung	Ja	Ja	Ja
Objekterkennung	Ja	Nein	Ja
Arealerkennung	Nein	Nein	Ja
Oberflächenerkennung	Ja	Ja	Ja
Cloud-Anker	Ja	Ja	Ja
SLAM	Ja	Ja	Ja
Tiefenerkennung	Ja	Ja	Nein
LiDAR	Ja	Nein	Nein
Lichtberechnung	Ja	Ja	Ja
Multinutzer	Ja	Ja	Nein
FPS	60	30-60	30 / 60

Tabelle 2: AR-SDKs Funktionalitätsvergleich

Als Hauptunterschiede können festgehalten werden, dass Vuforia einen stärkeren Fokus auf dreidimensionale Erkennungsmethoden legt, während ARKit mit LiDAR- und Tiefenerkennungsunterstützung eine realistischere Darstellung im Raum ermöglicht.
Neben den in der Tabelle 2 zu findenden Methoden der Objekt- und Arealerkennung bietet Vuforia außerdem eine Erkennung von zylindrischen und konischen Ankern sowie die Erkennung von *VuMarks*, einer Art spezieller QR-Codes. Für den Erfolg der Erkennungsmethoden spielen die Belichtungsqualität, die Objektgröße und die Anzahl der Erkennungspunkte eine entscheidende Rolle [Hameed et al., 2022].
Die Tiefenerkennung ermöglicht es, die Distanz des Smartphones zu realen Objekten zu berechnen. Befindet sich ein virtuelles Objekt in der Szene kann somit überprüft werden, ob das virtuelle Objekt vor oder hinter dem realen Objekt ist und, wenn es sich dahinter befindet, entsprechend von dem realen Objekt verdeckt werden.
LiDAR steht für *Light detection and ranging* und dabei handelt es sich um eine Methode des Laserscannens. LiDAR-Scanner sind auch im iPhone 12 und im iPhone 13 Pro verbaut. Mit ihrer Hilfe lassen sich Distanzen noch genauer messen.
Mit ARKit und ARCore lassen sich Anwendungen erstellen, welche von mehreren Personen gleichzeitig verwendet werden. Diese Personen sehen dann zur gleichen Zeit die gleichen virtuellen Erweiterungen in ihrer Umgebung. Dies kann zu verschiedenen Interaktionen wie z.B. kooperativen Arbeiten oder Spielen genutzt werden.
FPS gibt an, wie oft das angezeigte Bild pro Sekunde aktualisiert wird (aus dem Englischen: *frames per second*. ARKit-basierte Anwendungen laufen grundsätzlich mit 60 Bildern pro Sekunde. Bei ARCore ist die Bildwiederholungsrate von dem verwendeten Smartphone abhängig und rangiert zwischen 30-60 FPS. Vuforia unterscheidet in der Bildwiederholungsrate, ob die Anwendung auf einem iOS- oder einem Androidsmartphone genutzt wird. Auf iOS-Geräten werden 60 FPS dargestellt, auf Android-basierten Geräten nur 30. Eine höhere Bildwiederholungsrate kann bei schnellen Bewegungen zu einer besseren Darstellung führen.
Nowacki und Woda [2020] stellen außerdem fest, dass ARKit gegenüber ARCore besser für Anwendungen in Innenräumen und bei schnellen Bewegungen geeignet ist, weniger Speicherplatz benötigt und Schatten automatisch erzeugt. ARCore ist hingegen besser für

Anwendungen im freien geeignet und ist besser dokumentiert. Schatten müssen selbst implementiert werden und die Leistung der Anwendung variiert geräteabhängig.
Unity bietet mit ARFoundation ein Framework, das sowohl auf ARKit als auch auf ARCore zurückgreift. Dies ermöglicht es, eine Anwendung zu erstellen, die auf Android und iOS funktioniert. Dabei sind ARKit-exklusive Funktionen jedoch iOS-Geräten vorbehalten. Für Android-Geräte wird auf Funktionen von ARCore zurückgegriffen, iOS-Geräte können von kompatiblen Funktionen beider SDKs profitieren.
Mit dem Ziel eine Anwendung zu erstellen, die von möglichst vielen Personen genutzt werden kann, wurden in dieser Arbeit Vuforia verwendet. Der Fokus liegt auf einer Anwendung für Android-Geräte. Wie aus dem Vergleich der SDKs hervorgeht, ist eine Übertragung einer solchen Anwendung von Android- auf iOS-Geräte mit Vuforia möglich. Eine iOS-spezifische ARKit-basierte Anwendung könnte hingegen nicht auf Android übertragen werden.

Dieses Kapitel hat gezeigt, mit welchen Mitteln die Anwendung erstellt werden soll. Dabei wurde insbesondere ein Fokus auf einen aktuellen Vergleich der drei SDKs ARKit, ARCore und Vuforia gelegt und gezeigt, welche Gemeinsamkeiten und Unterschiede sie bezüglich ihrer Funktionalität und Kompatibilität aufweisen. Zur Umsetzung kommen Blender, Unity und Vuforia zum Einsatz.

Kapitel 4

Umsetzung

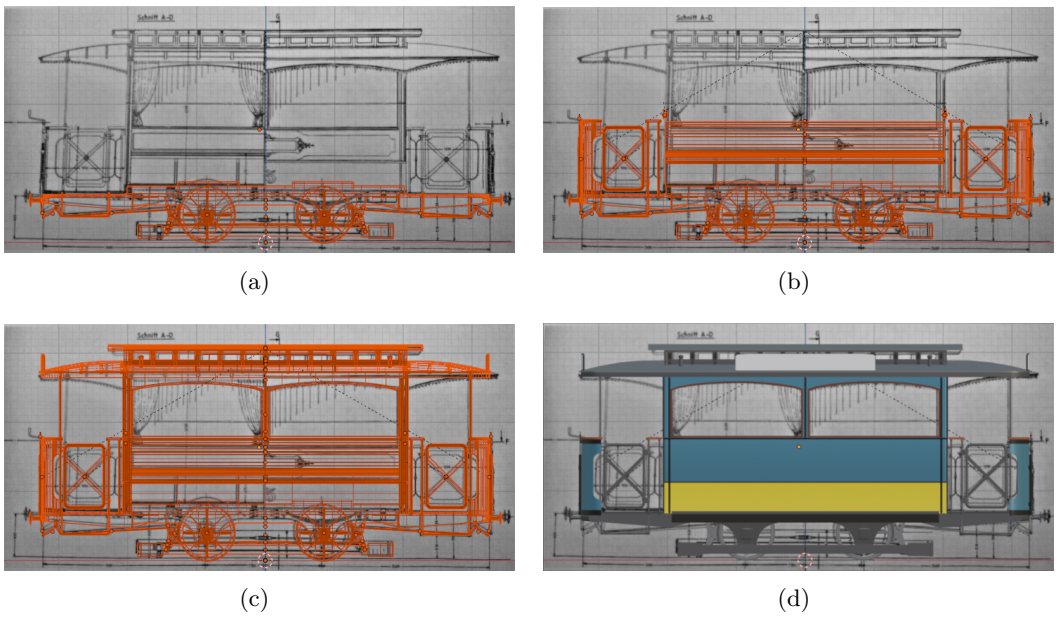
Dieses Kapitel zeigt, wie mit den in Kapitel 3 vorgestellten Technologien die auf Abbildung 18 zu sehende mobile Anwendung für Android-Smartphones erstellt wurde, welche die Karolinenstraße in Bamberg mit einem virtuellen Modell der ehemaligen Straßenbahn erweitert.

Abbildung 18: Beispielbild der entwickelten Anwendung

Abschnitt 4.1 zeigt, wie auf der Grundlage von Bildquellen ein 3D-Modell der ehemaligen Bamberger Straßenbahn geschaffen wurde. Es wird gezeigt, wie die Form der Straßenbahn anhand dieser Bildquellen auf einfache Weise nachempfunden werden konnte und wie die Texturen entstanden sind.
Abschnitt 4.2 zeigt, wie die Animationsroute erstellt und unterteilt wurde und wie versucht wurde, mittels GPS eine Grundlage für den Animationspfad zu schaffen.
Abschnitt 4.3 geht auf die Bildauswahl und die Bildvorverarbeitung zur bilderkennungsbasierten Positionierung der Straßenbahn ein.
Abschnitt 4.4 dieses Kapitels zeigt, wie mit Vuforia die Bilderkennung implementiert wurde und wie das Modell inklusive Animation damit verknüpft wurden.

4.1 Modellierung der Straßenbahn in Blender

Wie in Abschnitt 3.2 gezeigt wurde, lagen zur Modellierung der Straßenbahn Bildquellen wie Pläne und Illustrationen vor. In Blender kann ein Bild als Hintergrundbild eingefügt werden. Ein Hintergrundbild wird in Blender nur in der orthografischen Ansicht angezeigt. Die orthografische Ansicht zeigt die Szene ohne perspektivische Verschiebung. Winkel sowie Längen von Strecken werden korrekt angezeigt. Wird diese Ansicht gewählt nachdem ein Hintergrundbild eingefügt wurde, kann das Objekt dem Hintergrundbild entsprechend modelliert werden.
Abbild 19 zeigt vier Screenshots aus Blender. Im Hintergrund ist der Längsschnitt der Straßenbahn zu sehen. Er befindet sich auf der XZ-Ebene und ist somit nur aus der orthografischen Ansicht entlang der Y-Achse zu sehen. Für die Maße der Straßenbahn entlang der Y-Achse wurde das gleiche Prinzip verwendet. Auf der YZ-Ebene wurde ein Hintergrundbild eingefügt, welches in der orthografischen Ansicht entlang der X-Achse zu sehen war. Die Abbildungen 19 (a)-(c) zeigen in unterschiedlichem Umfang die Kanten des Stra-

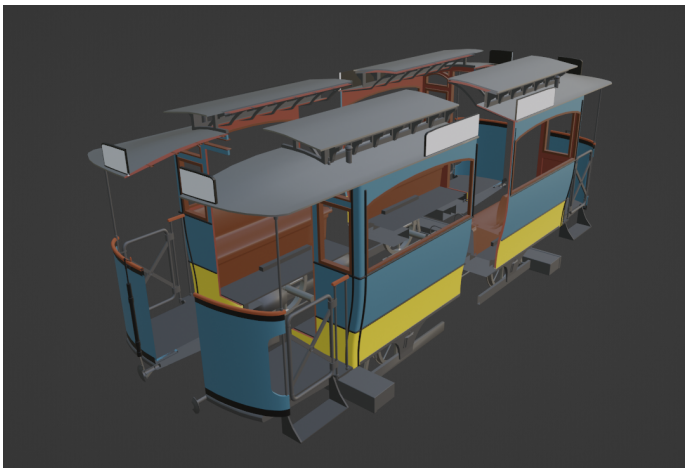


(a) (b)

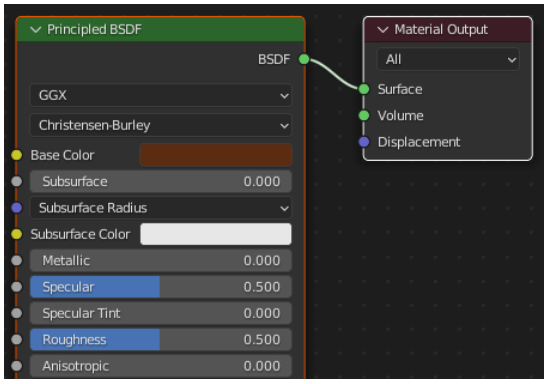
(c) (d)

Abbildung 19: Straßenbahnmodell vor Längsschnitt

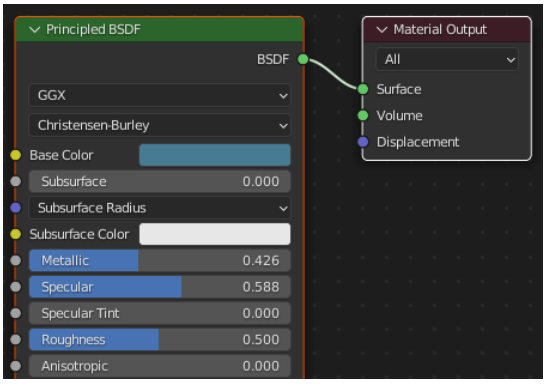
ßenbahnmodells in Orange. Abbildung 19 (d) zeigt das Straßenbahnmodell mit Texturen und den Schildern auf dem Dach, welche nicht dem Schnitt entnommen sind.
Die Grundstruktur der Straßenbahn hat zwei Symmetrieachsen (vgl. Abb. 20(a)). Das ermöglichte es, sich auf die Modellierung eines Viertels zu beschränken und dieses Viertel dann zweifach zu spiegeln um die weiteren Teile zu erhalten. Asymmetrische Details wie Türgriffe wurden im Anschluss individuell hinzugefügt.
Die Materialien wurden simpel gehalten. Abbildung 20(b) zeigt das Material, welches für die Holzbänke genutzt wurde, Abbildung 20(c) das für die blauen Karosserieteile. Für das Holz kam ein einfaches Braun zum Einsatz, bei der Karosserie und weiteren metallischen Bauteilen wurden die Parameter für metallisches Aussehen und Glanzlichter erhöht.



(a) Veranschaulichung der Symmetrieachsen



(b) Holzbänke



(c) Blau der Karosserie

Abbildung 20: Erstellung des 3D-Modells

Das Modell der Straßenbahn entstammt den Vorarbeiten aus dem Universitätsprojekt des Sommers 2022. Es verfügt über weitere Details, welche für die Umsetzung und Auswertung der Anwendung im Rahmen dieser Masterarbeit keine Rolle spielen. Eine genauere Beschreibung der Modellierung bietet der Projektbericht „Augmented Reality Timetravel - Die digitale Straßenbahn“ von Wagner et al. [2022].

4.2 Animationsroute

Auch die Erfassung des Streckenverlaufs mittels GPS und ihre Unterteilung in einzelne Animationsabschnitte stammen aus dem vorangegangenen Projekt. Zur Vermessung wurde die Strecke mit GPS-fähigen Smartphones mehrfach begangen und die GPS-Daten wurden erfasst. Abbildung 21 basiert auf einem Abbildungsausschnitt des Projektberichts von Wagner et al. [2022], die in Rot und Gelb zwei der vermessenen Routen in dem Bereich zwischen altem Rathaus und Herrenstraße zeigt. Erweitert wurde die die Abbildung durch die vier Rautensymbole, welche die Startpunkte der einzelnen Abschnitte der Animation zeigen. Am linken Bildrand ist auch zu erkennen, dass die vorgenommenen Messungen nicht dem Verlauf der ehemaligen Straßenbahnlinie in die Herrenstraße folgen, sondern der

Karolinenstraße. Diese Linien zeigen beispielhaft, dass die unterschiedlichen Messungen einerseits voneinander abweichen und andererseits meist nördlich parallel neben der Straße verlaufen. Im Bildabschnitt unten links ist eine größere Differenz zwischen den Messwerten zu erkennen. Die Daten wurden zur Fehlerminimierung gemittelt und die gemittelten Messpunkte wurden in Vektoren überführt, da für die Animation nicht die absolute Position im Raum relevant ist, sondern die relative Position der Punkte zueinander. Der Versatz der Linien bzw. der entsprechenden Messwerte in nördliche Richtung stellt damit kein Problem dar. Eine erste Annäherung konnte so zwar geschaffen werden, doch konnten die ohnehin ungenauen Höheninformationen bei der Mittlung nicht berücksichtigt werden. Da die Straßenbahn auf ihrer ehemaligen Strecke visualisiert werden sollte, konnten die erhobenen Daten keine Grundlage für die Erstellung der Animationsrouten darstellen. Die Kurve zur Herrenstraße und die Höhendaten der sonst größtenteils geradeverlaufenden Strecke wurden nicht erfasst. Die Animationsstrecken wurden letztendlich auf der Grundlage von Testläufen individuell in Unity angepasst.

Abbildung 21: GPS-Messungen in QGIS

Auf Abbildung 21 sind die Startpunkte der einzelnen Streckenabschnitte mit dem Rautensymbol markiert. An diesen Postionen befinden sich Bildanker, welche bei Erkennung den jeweiligen Streckenabschnitt starten. Die Streckenabschnitte können einzeln angesehen werden. Eine Zusammenhängende Route ergibt sich, wenn man vor dem Rathaus startet und der Straßenbahn entlang der Karolinenstraße folgt. Die Abschnitte stellen dabei zunehmend komplexere Herausforderungen für eine korrekte Visualisierung dar. Der erste Abschnitt ist nur wenige Meter lang. Er startet vor dem Rathaus und endet im Torhaus vor Informationstafeln. Der zweite Abschnitt ist länger und führt vom Torhaus über die Brücke bis vor die erste Kreuzung. Im dritten Abschnitt ändert sich die Steigung. Das Gefälle ändert sich hier zu einer leichten Steigung, was sowohl die Positionierung als auch die Rotation der Straßenbahn beeinflusst. Im letzten Abschnitt biegt die Straßenbahn in die Herrenstraße ab. Nach dem Abbiegen sollte sie hinter dem Gebäude nicht mehr sichtbar sein und die Schwierigkeit besteht in der korrekten Verdeckung der Straßenbahn. Die verwendeten Bildanker werden im folgenden Abschnitt auf Abbildung 26 gezeigt.

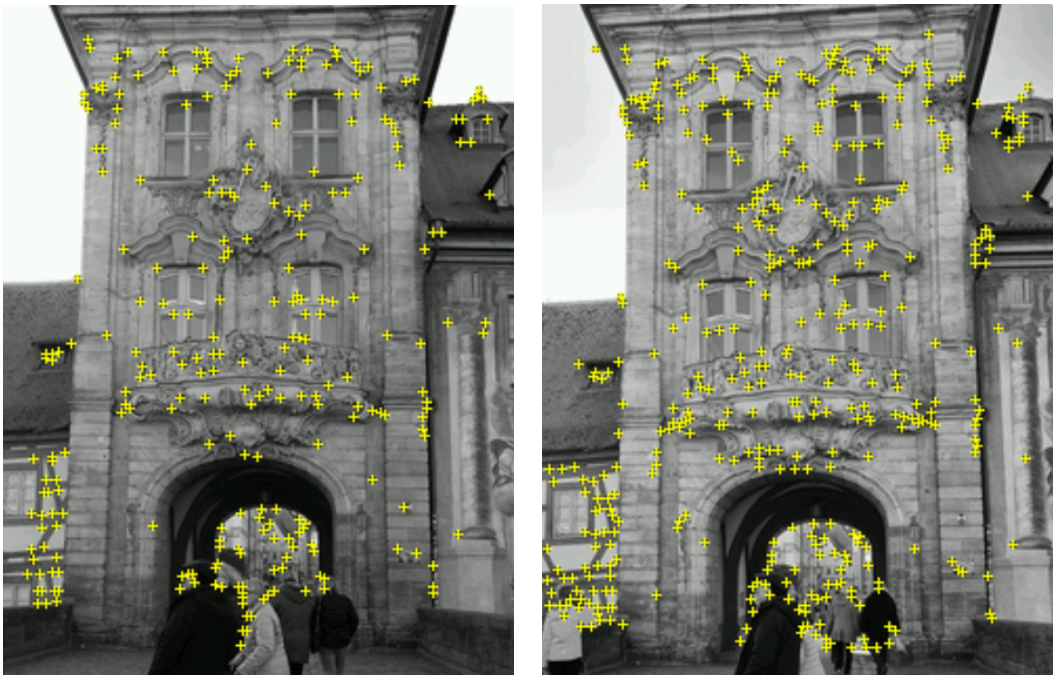
4.3 Bildauswahl für die Bilderkennung

Wie in Abschnitt 3.1 gezeigt wurde, soll zur Verankerung der Straßenbahn in der Stadt Bilderkennung zum Einsatz kommen. Dieser Abschnitt zeigt, mit welchen Maßnahmen versucht wurde den Herausforderungen dieser unkontrollierten Umgebung gerecht zu werden und was das für die Auswahl der Bilder bedeutet, welche zur Verankerung erkannt werden sollen. Es sollen Stadtszenen bzw. Bilder als Anker verwendet werden, die eine Platzierung der Straßenbahn möglichst einfach machen. Die Straßenbahn soll dazu im gleichen Bildausschnitt erscheinen, welches das Display zeigt, wenn der Bildanker erkannt wird. Dazu wurde zunächst eine Ansicht des alten Brückenrathauses gewählt und es wurden mit zwei unterschiedlichen Smartphones Bilder gemacht, welche potentiell als Bildanker verwendet werden sollten, und miteinander verglichen.
Bei den Smartphones handelt es sich um ein *Xiaomi Mi A1* und ein *Samsung Galaxy S20+*. Bei dem Xiaomi Mi A1[1] handelt es sich um ein Smartphone des günstigeren Preissegments aus dem Jahr 2017, das mit zwei Kameras ausgestattet ist, welche jeweils eine Auflösung von 12 Megapixeln bieten. Das Samsung Galaxy S20+[2] ist hingegen ein Smartphone aus dem gehobenen Preissegment und stammt aus dem Jahr 2020. Es bietet insgesamt drei Kameras. Neben zwei Kameras mit 12 Megapixeln, hat es eine dritte Kamera, die Fotos mit bis zu 64 Megapixeln macht. Zusätzlich verfügt es über einen Sensor, welcher der Distanzmessung dient. Die fertige Anwendung wird in der Folge mit diesen beiden Geräten auch getestet. Hier kamen sie zunächst für die Erstellung der Bildanker zum Einsatz.
Um vor dem Testen der Anwendung schon eine Aussage über die Eignung eines Fotos als Bildanker zur Bilderkennung machen zu können, wurde das *Vuforia Developer Portal*[3] genutzt. Hier können Bilder hochgeladen und auf ihre Eignung als Bildanker überprüft werden. Es wird eine Sternewertung vergeben. Null Sterne ist die schlechteste Wertung und ein Bild mit einer Null-Sterne-Bewertung ist nicht als Bildanker nutzbar. Die beste zu erreichende Wertung ist fünf Sterne. Relevant für diese Bewertung ist, wie viele markante Punkte auf dem Bild als Erkennungsmarker gefunden werden können. Wie Abbildung 14 zeigt, kann man sich auf den Fotos die erkannten Marker anzeigen lassen.
Abbildung 22 zeigt einen Vergleich der Fotos, die mit den beiden Smartphones gemacht wurden. Es handelt sich auf beiden Fotos um die gleiche Szene vor dem Alten Rathaus in Bamberg mit typischem Fußgängerverkehr. Auf dem Foto des älteren Xiaomi-Smartphones (vgl. 22(a)) wurden weniger Punkte erkannt als bei seinem Pendant des Samsung-Geräts (vgl. 22(b)). Entsprechend fällt die Bewertung unterschiedlich gut aus. Abbildung 23 zeigt die Sternebewertung der Bilder. Das Bild 22(a) erreicht drei Sterne, Bild 22(b) vier. Es ist auch zu erkennen, dass die markanten Punkte im allgemeinen an den gleichen Orten erkannt wurden, einzelne Punkte jedoch an anderen Orten. So fehlen dem besseren Bildanker 22(b) auch einzelne Punkte, die in 22(a) erkannt wurden. Im unteren Bereich der Bilder 22(a) und 22(b) wurden auch an Passanten markante Punkte erkannt. Sowohl die Erkennung von Passanten als auch die Eignungsqualität der gewählten Bilder (vgl. Abb. 23) sind nicht optimal für die Bilderkennung. Zur Verbesserung der beiden Probleme erfolgten zwei wesentliche Schritte. Abbildung 24 zeigt das Ergebnis der Nachbearbeitung der Bilder aus 22. Der erste Schritt war eine Anpassung des Bildausschnitts. Das

[1] https://www.teltarif.de/smartphone/xiaomi/mi-a1/, aufgerufen: 22.02.2023

[2] https://www.samsung.com/de/business/smartphones/galaxy-s/galaxy-s20-enterprise-edition-g980fds-sm-g980fzadeeb/, aufgerufen: 22.02.23

[3] https://developer.vuforia.com/vui/develop/, aufgerufen: 22.02.23

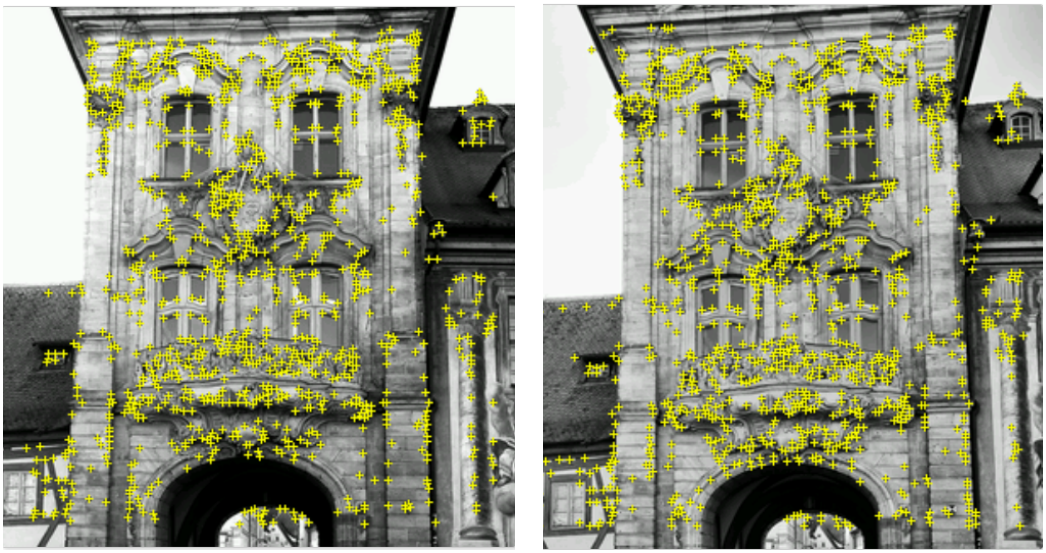


(a) Xiaomi Mi A1 (b) Samsung Galaxy S20+

Abbildung 22: Erster Bildankervergleich

Abbildung 23: Sternewertung der Bildanker

Bild wurde verkleinert und der untere Bereich entfernt. Auch abseits dieses Beispiels kann die Aussparung der befahrenen und begangenen Stadtbereiche zur Bilderkennung einige Problemfelder umgehen. In diesen Bereichen sind auch Baustellen, Märkte, Dekorationen, wechselnde Schaufenster oder Geschäfte und viele weitere variable Elemente zu finden, welche sich nachteilig auf die Bilderkennung auswirken können. Fassaden ab dem ersten Stock aufwärts sind weniger oft von Änderungen betroffen.
Diese Eingrenzung schließt auch manche Erkennungmarker aus und Erkennungsmarker sollten für einen guten Bildanker in möglichst großer Zahl vorliegen. Um die Zahl der Erkennungsmarker auf den Bildern zu erhöhen, wurden im zweiten Schritt die Kontraste angepasst. Zur Bilderkennung kommen Schwarz-Weiß Bilder zum Einsatz. In diesem Vorverarbeitungsschritt wurden die Bilder zunächst in Schwarz-Weiß Bilder überführt und die Kontraste so angepasst, dass strukturelle Unterschiede des Gebäudes sich besser voneinander abheben. Dabei lässt sich der Kontrast nicht beliebig erhöhen, da sonst Fehler wie verschwindende Konturen oder Phantomkonturen, die eigentlich nicht vorhanden sind, auftreten können. Abbildung 24 zeigt, dass auf beiden Bildern nach der Bearbeitung mehr



(a) Mi A1 (b) Galaxy S20+

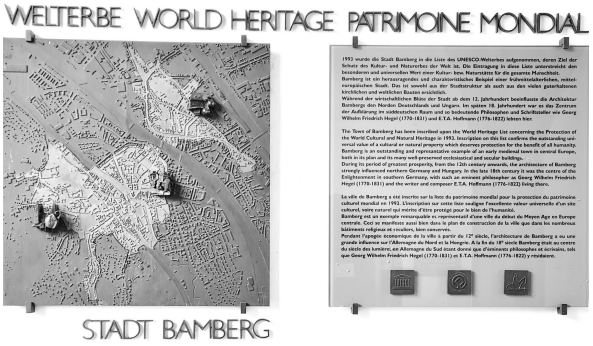
Abbildung 24: Zweiter Bildankervergleich

Abbildung 25: Sternewertung der bearbeiteten Bildanker

markante Punkte erkannt wurden. Abbildung 25 zeigt die Bewertung der Bildankereigenschaft der Bilder nach der Bearbeitung. Sowohl 24(a) als auch 24(b) erhalten nun die bestmögliche Fünf-Sterne-Wertung. Die erkannten Punkte unterscheiden sich im Detail, ein großer Unterschied in der allgemeinen Anzahl lässt sich nicht mehr feststellen.
Um die Anwendung an verschiedenen Punkten starten zu können, wurden an weiteren Orten entlang der ehemaligen Strecke der Straßenbahn Bildanker erstellt. Auch bei allen weiteren Bildankern wurden die Kontraste verstärkt, damit viele Punkte erkannt werden konnten. Das Vuforia Developer Portal bewertet sie alle mit fünf Sternen.
Abbildung 26 zeigt die vier Bildanker, welche in der Anwendung zum Einsatz kommen. Abbildung 26(a) zeigt eine nähere Aufnahme des Rathausbalkons. Im Gegensatz zu den Fotos der Abbildungen 22 und 24 wurde dieses Foto gerade und aus zentraler Position aufgenommen. Außerdem schließt das Bild mit den Mauern des Torhauses ab. Abbildung 26(b) zeigt eine Karte und einen Infotext, welche im Durchgang des Rathauses angebracht sind. Mit ihren klaren Strukturen und kontrastreichen Abgrenzungen stellen diese Tafeln inklusive Überschrift einen guten Bildanker dar. Sie sind auf einer Höhe angebracht auf der sie gut gelesen oder betrachtet werden können. Entsprechend besteht hier eine erhöhte Verdeckungsgefahr durch Passanten. Nach der Durchquerung des Rathauses gelangt man auf den letzten Brückenabschnitt und die andere Seite des Flusses. Hier finden

sich Geschäfte mit Schaufenstern, welche nicht als Bildanker herangezogen werden konnten. Als konstantes Element mit markanten Erkennungsmerkmalen wurde an dieser Stelle daher die Kombination von zwei Kanalisationsdeckeln genutzt (vgl. Abb. 26 (c)). Als letzten Bildanker zeigt Abbildung 26 (d) das Schild einer Brauerei an einer Gaststätte. Das Schild ist dank seiner Schrift gut für die Bilderkennung geeignet und ist über Kopfhöhe angebracht und somit meist unverdeckt.

(a) Altes Rathaus - Balkon



(b) Informationstafeln

(c) Kanalisationsdeckel

(d) Brauereischild

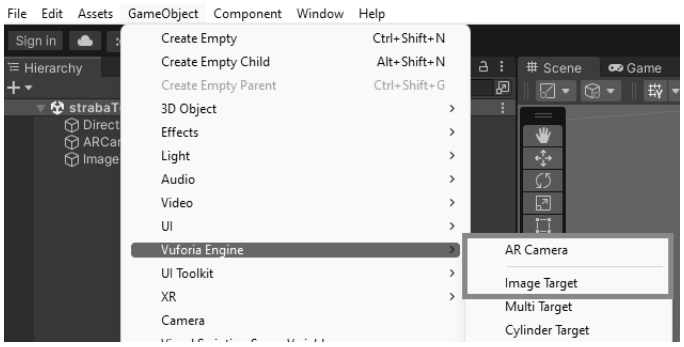
Abbildung 26: Verwendete Bildanker der vier Streckenabschnitte

Dieser Abschnitt hat gezeigt, welche Bilder als Bildanker für die Platzierung der Straßenbahn dienen und wie diese in Bezug auf die Auswahl des Bildausschnitts und die Erhöhung des Kontrasts optimiert wurden. Es wurde auch gezeigt, dass nicht für jeden Streckenabschnitt ein Bildanker gefunden werden konnte, der alle eingangs genannten Kriterien für einen guten Bildanker erfüllt. Welche Auswirkungen diese auf die Funktion der Anwendung und die Qualität der Visualisierung haben, wird im Kapitel 5 gezeigt und diskutiert.

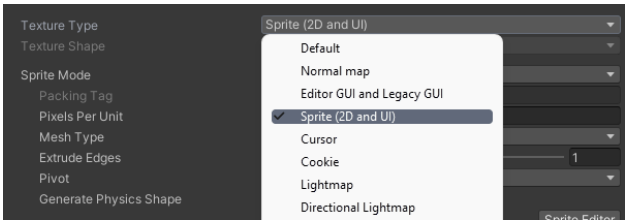
4.4 Zusammensetzung in Unity

Das Straßenbahnmodell, die Animationsroute und die gewählten Bilder zur Positionierung mittels Bilderkennung wurden in Unity unter der Verwendung von Vuforia miteinander verknüpft. Dieser Abschnitt zeigt, wie dabei vorgegangen wurde und welche Besonderheiten zu beachten waren.

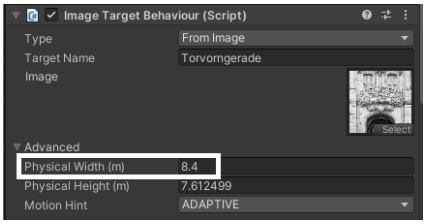
Das Vuforia SDK kann kostenlos von der Vuforia Website[4] geladen und zum Import in Unity in das entsprechende Projekt geladen werden. In Unity erscheint nun unter *GameObject* die *Vuforia Engine* mit deren Werkzeugen (vgl. Abbildung 27(a)). Daraus werden markierten ersten beiden Elemente benötigt. Die *AR Camera* stellt die Perspektive des Smartphones dar und nutzt in der fertigen Anwendung dessen Kamera. Mit dem *Image Target* kann das zur Bilderkennung gewählte Bild mit dem 3D-Objekt verknüpft werden. Diese beiden Elemente können zwar kostenlos genutzt werden, es wird jedoch ein Account auf der Website von Vuforia vorausgesetzt. Hier erhält man im Entwicklerbereich einen Lizenzschlüssel welcher in Unity unter *Vuforia Configuration* in die entsprechende Stelle kopiert werden kann.



(a) Vuforia Bausteine in Unity



(b) Texturtyp für Bildanker



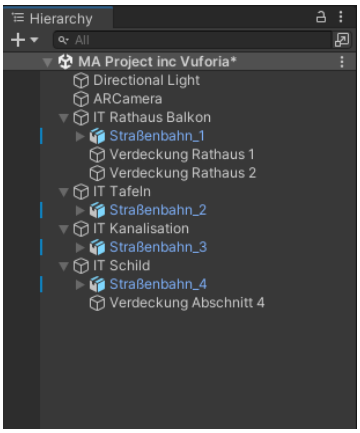
(c) Breiteeinstellung des Bildankers

Abbildung 27: Einstellungen in Unity

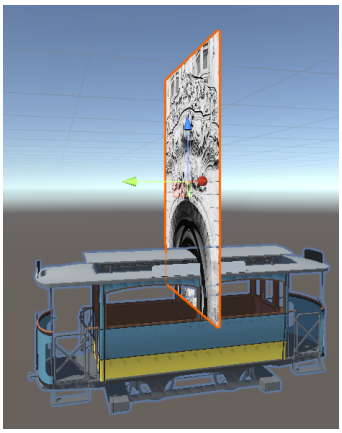
Für die Bildanker wurde ein Ordner im Projekt angelegt und die Bilder wurden dort abgelegt. Auf die Bilder kann nun in Unity direkt zugegriffen werden. Zu beachten ist, dass für ihre Funktion als Bildanker zur Bilderkennung der Texturtyp auf *Sprite* gestellt werden muss. Dazu wählt man ein Bild in Unity aus und der *Inspector* am rechten Rand zeigt die Eigenschaften des gewählten Bildes. Abbildung 27(b) zeigt die nötigen Einstellungen. An oberster Stelle ist der *Texture Type* zu finden, welcher hier entsprechend angepasst

[4]https://developer.vuforia.com/downloads/sdk, aufgerufen: 27.02.2023

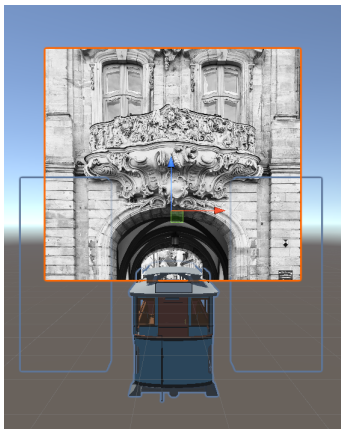
werden kann. Abbildung 27(c) zeigt das *Image Target Behaviour (Script)*, das am rechten Bildrand erscheint, sobald das entsprechende *Image Target* ausgewählt wurde. Unter den erweiterten Einstellungen wurde hier die jeweilige Breite des Bildes angegeben, das als *Image Target* verwendet wird. Dadurch werden die dargestellten Größenverhältnisse der projizierten Straßenbahn in Relation mit der Umwelt gesetzt.
Dank dem FBX-Exporter von Blender kann Unity direkt auf Blenderdatein zugreifen und ein dedizierter Schritt des Exportierens und Importierens entfällt[5]. Es wird immer die aktuelle Blenderdatei genutzt und mögliche Änderungen an dem Modell werden direkt übernommen. Dazu muss die Datei lediglich im Projekt abgelegt werden und sie muss mit Blender-Version 2.60 oder einer neueren Blender-Version erstellt worden sein.
Nachdem das *Image Target* erstellt und das Straßenbahnmodell im Projekt abgelegt wurde, folgte die Verknüpfung dieser Elemente sowie die Erstellung und Verknüpfung der Verdeckungselemente. Abbildung 28(a) zeigt, dass die Straßenbahn und die Verdeckungselemente als Kinder der *Image Targets* in die Projekthierarchie eingefügt wurden.



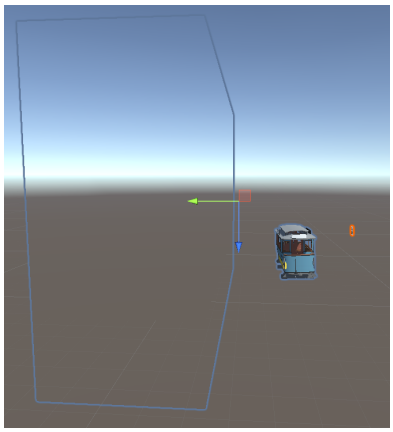
(a) Projekthierarchie



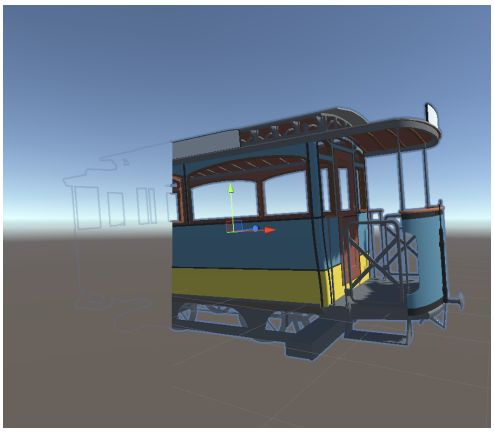
(b) Straßenbahnpositionierung



(c) Verdeckungspositionierung



(d) Letzter Streckenabschnitt



(e) Veranschaulichung der Verdeckung

Abbildung 28: Verknüpfung von Bildanker mit Straßenbahn und Verdeckung

[5]https://docs.unity3d.com/560/Documentation/Manual/HOWTO-ImportObjectBlender.html, aufgerufen: 02.03.2023

So wurden sie mit dem *Image Target* verknüpft und sie erscheinen sobald das *Image Target* erkannt wird. Nach dem Einfügen des Straßenbahnmodells wurde dessen Postion angepasst. Abbildung 28(b) zeigt die Verknüfung der Elemente für den ersten Streckenabschnitt. Die Straßenbahn wurde unter dem Torbogen positioniert. Abbildung 28(c) zeigt die Verknüpfung der Elemente für den gleichen Abschnitt aus frontaler Perspektive mit den zusätzlich eingefügten Verdeckungselementen. Auf Abbildung 28(c) und (d) werden sie mit einer blauen Kontur dargestellt. Bei den Verdeckungselementen handelt es sich um Quader, welche in Unity als Kinder des *Image Targets* eingefügt, angepasst und mit dem *Occlusion*-Material versehen wurden. Sie wurden in der Größe an die tatsächlichen Wände des Alten Rathauses angepasst und entsprechend positioniert. Das *Occlusion*-Material sorgt dafür, dass die Elemente selbst nicht dargestellt werden, andere virtuelle Objekte durch sie jedoch verdeckt werden können. Abbildung 28(e) veranschaulicht das. Sie zeigt die Straßenbahn, welche im vorderen Bereich vollständig visualisiert wird. Teilweise befindet sie sich hinter dem Verdeckungsobjekt. Im Bereich der Verdeckung wird nur noch ihre Kontur angedeutet. In der fertigen Anwendung wird auch diese Kontur nicht mehr zu sehen sein, wenn das Objekt verdeckt wird. In der Anwendung soll durch diese unsichtbaren Verdeckungselemente der Eindruck entstehen, dass die virtuelle Straßenbahn durch die tatsächlichen Gebäude verdeckt wird.
Die Animationen der einzelnen Streckenabschnitte wurden in Unity erstellt. Die mit GPS vermessenen Daten lieferten keine brauchbare Grundlage, da die Höhendaten der einzelnen Messungen sehr große Ungenauigkeiten aufwiesen und die gemittelten Werte keine Höhendaten mehr besaßen. Abgesehen von den Höhenunterschieden ist der Streckenabschnitt bis zur Herrenstraße näherungsweise eine gerade Strecke und aus den vorliegenden Daten konnte kein Nutzen gezogen werden. Stattdessen wurden die Positions- und Rotationswerte der Straßenbahn, welche die Animation definieren, in Unity iterativ angepasst und getestet. Im *Unity Animator* können die Zustände zur Animationssteuerung angelegt werden. Abbildung 29 zeigt drei Felder. „Entry“ steht für den Start der Szene, „none“ und „losfahren“ sind angelegte Animationen. Die erstellte Animation wird ausgeführt, wenn

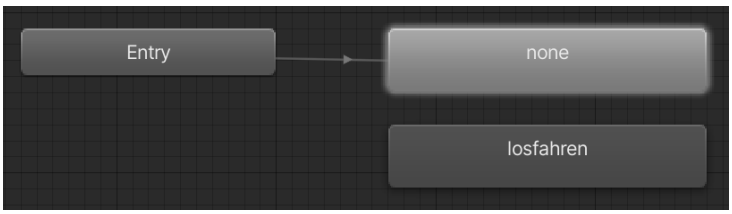


Abbildung 29: Unity Animator

„losfahren“ aufgerufen wird und „none“ dient als Platzhalter ohne hinterlegte Animation. Der Pfeil von „Entry“ auf „none“ zeigt, dass bei Start der Szene „none“ ausgeführt wird. Die Straßenbahn wird also zunächst ohne Animation visualisiert. Die Animation kann in der Anwendung durch eine Nutzereingabe gestartet und zurückgesetzt werden. Um diese Steuerung zu ermöglichen wurde ein Script angelegt. Dabei wird in der Update-Funktion abgefragt, ob es eine Nutzereingabe gibt und welcher Animationszustand aktuell vorliegt. Gibt es eine Nutzereingabe, wird der entsprechend andere Animationszustand ausgeführt. Der unten aufgeführte Pseudocode verdeutlicht diesen Steuerungsablauf. Die Update-Funktion wird bei jeder Bildaktualisierung ausgeführt.

```
void Update()
   {
      if (Nutzereingabe & Name der aktuellen Animation ist "none")
         {
            starte Animation "losfahren";
         }
      else if (Nutzereingabe & Name der aktuellen Animation ist "losfahren")
         {
            starte Animation "none";
         }
   }
```

Dieses Kapitel hat gezeigt, wie das Straßebahnmodell in Blender erstellt wurde, wie versucht wurde aus GPS-Messungen eine Datengrundlage für die Animationsroute zu erstellen und warum das nicht gelang. Außerdem wurde gezeigt, wie bei der Bildauswahl und der Vorverarbeitung der Bilder zur Bilderkennung vorgegangen wurde, wie die Bilder, das Straßenbahnmodell und die Verdeckungselemte verknüpft wurden und wie die Animationen erstellt und eingebunden wurde.
Im folgenden Kapitel werden die Ergebnisse dieser Umsetzung gezeigt und evaluiert.

Kapitel 5

Evaluierung

Dieses Kapitel gliedert sich in drei Teile. Zunächst werden die Ergebnisse der erstellten Anwendung vorgestellt und gezeigt, inwiefern sich die gewählte Architektur für den Fall des bewegten 3D-Objekts im urbanen Raum eignet. Anschließend werden die theoretischen Hintergründe erläutert, auf deren Basis die Ergebnisse dann eingeordnet und diskutiert werden.

5.1 Ergebnisse

Die Anwendung wurde zu unterschiedlichen Tageszeiten und Bedingungen mit den in Kapitel 4.3 erwähnten Smartphones getestet. Die Ergebnisse der beiden Geräte weichen stark voneinander ab. Auf dem Xiaomi Mi A1 war mit der erstellten Anwendung keine realistische Darstellung (vgl. Abb. 30) einer Straßenbahnfahrt möglich. Auf dem Samsung Galaxy S20+ läuft die Anwendung deutlich besser.
Zunächst wird gezeigt, wie zuverlässig die Bilderkennung an den einzelnen Streckenabschnitten zur Straßenbahnpositionierung funktioniert. Danach werden die Ergebnisse der Verdeckung durch Gebäude mittels statischer Verdeckungselemente vorgestellt und es wird gezeigt, wie sich das Verdeckungsverhalten von Passanten auf die Visualisierung auswirkt. Der letzte Abschnitt geht auf die Visualisierungsstabilität nach der Bilderkennung ein.

Abbildung 30: Positionierung Xiaomi Mi A1

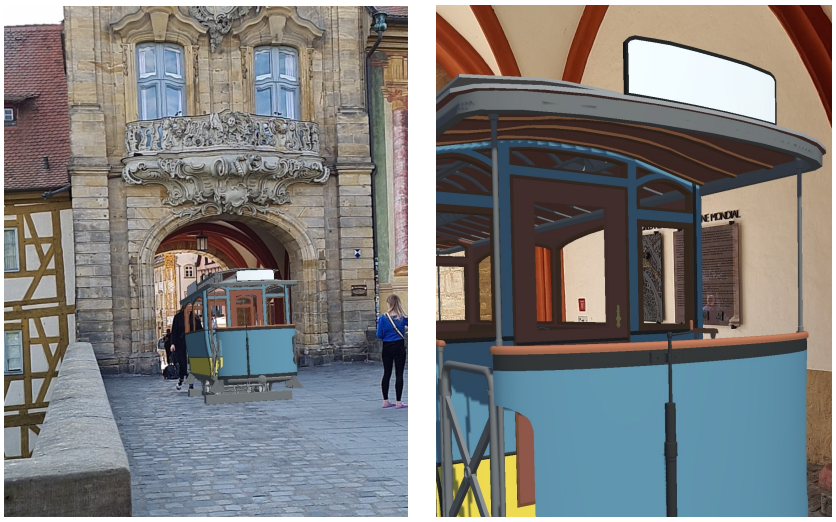
5.1.1 Positionierung der Straßenbahn

Die Straßenbahn soll in den jeweiligen Streckenabschnitten mittels Bilderkennung positioniert werden. Für eine erfolgreiche Positionierung sind zwei wesentliche Kriterien relevant. Das erste ist eine zuverlässig erfolgreiche Ausführung der Bilderkennung. In der während der Laufzeit erfassten Szene muss ein Bildanker erkannt werden. Das ist der Auslöser für die Einblendung der Straßenbahn und somit der eigentliche Start der Anwendung. Das zweite Kriterium ist die Korrekte Visualisierung des Straßenbahnmodells bezüglich seiner Position und Größe. Um einen realitätsnahen Eindruck vermitteln zu können, soll die Straßenbahn in Originalgröße auf der Straßenoberfläche angezeigt werden.

(a) Abschnitt Rathaus (b) Abschnitt Infotafeln

Abbildung 31: Positionierung Samsung Galaxy S20+, Abschnitt 1+2

Bildanker Rathausfassade Im ersten Streckenabschnitt sollte der Torbogen und der Balkon des Torhauses des Alten Rathauses erkannt werden. Der Verwendete Bildanker sowie die Bildanker aller weiteren Streckenabschnitte sind in Kapitel 4 auf Abbildung 26 zu sehen. Dieser Bildanker wurde bei Sonnenschein, Bewölkung, Regen (vgl. Abb. 31) und in der Abenddämmerung getestet. Die Erkennung funktionierte meist zuverlässig. Nur im Fall von starker seitlicher Sonneneinstrahlung oder direktem Gegenlicht durch eine tiefstehende Sonne war mit beiden Geräten keine Erkennung möglich. Sonst erkannte das Samsung-Smartphone den Bildanker unmittelbar, sobald der entsprechende Bereich des Gebäudes erfasst wurde. Abbildung 32(a) zeigt, dass der Bildanker auch vom Rand der Brücke und somit aus seitlicher Perspekive erkannt wurde. Gleiches gilt auch für unterschiedliche Distanzen in begrenztem Maß. Mit zunehmender Abweichung der Perspektive zum Bildanker nahm die Erkennungsgeschwindigkeit ab. Durch die Aussparung des unteren Bereichs des Bildankers war eine Bilderkennung trotz vieler Passanten möglich. Das Xiaomi-Gerät erreichte diese Verarbeitungsgeschwindigkeit nicht und konnte den Bildanker bei Gegenlicht nicht erkennen.

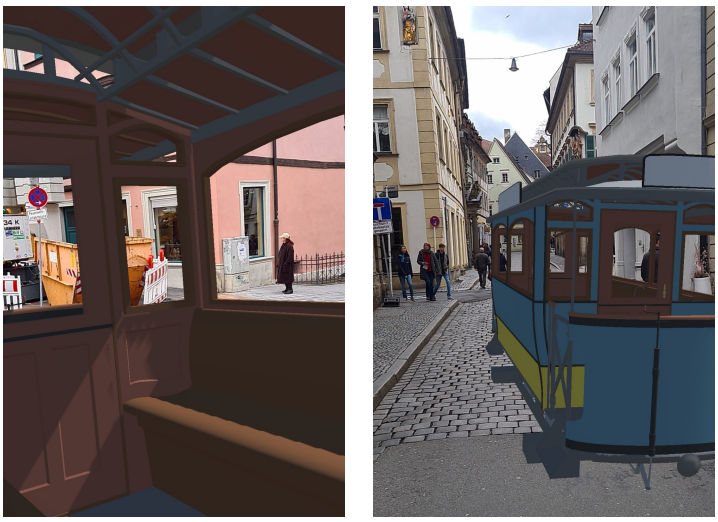


(a) Abschnitt Rathaus (b) Abschnitt Infotafeln

Abbildung 32: Positionierung Samsung Galaxy S20+, seitliche Perspektive

Auch in der Visualisierungsqualität ergaben sich Unterschiede. Abbildung 30 zeigt einen Anwendungsscreenshot des Xiaomi. Zu sehen ist die Straßenbahn vor dem Alten Rathaus. Ihre Position entspricht dabei jedoch nicht der angestrebten Position. Statt sich stabil in Relation zu ihrem Bildanker und der Szene zu verhalten, blieb ihre Position in Bezug auf das Smartphone stabil. Es war nicht möglich sich mit dem Smartphone um die Straßenbahn herumzubewegen. Abbildung 31(a) zeigt die Positionierung im selben Streckenabschnitt unter Verwendung des Samsung Galaxy S20+. Hier wird das Modell an der korrekten Position angezeigt. Nach der Bilderkennung kam es teilweise zunächst zu falschen Visualisierungen. Diese korrigierten sich jedoch innerhalb der ersten Sekunde. Auf dem Samsung-Smartphone verhält sich die Straßenbahn stabiler zur Umgebung und man kann mit dem Smartphone um das Modell herumgehen um es von allen Seiten zu besichtigen. Teilweise ist auch hier noch eine leichte Bewegung zu beobachten, wenn man sich mit dem Smartphone relativ zu Modell bewegt.

Bildanker Informationstafeln Abbildung 31(b) zeigt die Straßenbahn in ihrer Startposition des zweiten Streckenabschnitts. Der Bildanker mit den Infotafeln wurde von beiden Smartphones immer korrekt erkannt, solange er unverdeckt war und er aus mittiger Position erfasst wurde. Befanden sich einzelne Passanten vor den Tafeln, war die Erkennung mit der Xiaomi-Smartphone bereits stark eingeschränkt und die Positionierung konnte nicht zuverlässig erfolgen. Das Samsung-Smartphone konnte die Straßenbahn auch bei einer Teilverdeckung noch positionieren. Außerdem konnte es die Informationstafeln ab dem auf Abbildung 32(b) dargestellten Winkel erkennen. Dabei schwankte die Positionierungspräzision und es waren schnelle Positionsänderungen der eigentlich stehenden Straßenbahn zu beobachten. Durch die Nähe des Bildankers und die Größe der Bahn, wurde sie bei erfolgreicher Erkennung direkt vor dem Smartphone eingeblendet und war somit bei Start dieses Abschnitts nicht vollständig zu sehen.



(a) Abschnitt Knalisationsdeckel (b) Abschnitt Brauereischild

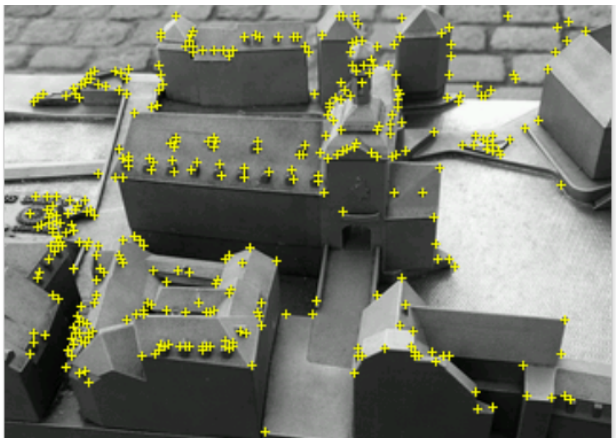
Abbildung 33: Positionierung Samsung Galaxy S20+, Abschnitt 3+4

Bildanker Kanalisationsdeckel Das Problem der Positionierung der 3D-Objekts an der Position des Smartphones wird im folgenden Streckenabschnitt noch deutlicher. Im dritten Streckenabschnitt wurden Kanalisationsdeckel erkannt und die Straßenbahn direkt darauf platziert. Da sich die Kanalisationsdeckel auf der Straße befinden, wird das Smartphone also auf den Boden gerichtet. Nach einer erfolgreichen Platzierung befindet sich das Smartphone somit im Inneren der Bahn und zeigt das graue Material des Straßenbahnbodens. Richtet man das Smartphone wieder auf, erhält man die auf Abbildung 33(a) gezeigte Perspektive. Hier ist im Vordergrund der Innenraum der Straßenbahn zu sehen und durch die Fenster sieht man im Hintergrund die Stadt. Die Erkennung dieses Bildankers funktionierte im Test am schlechtesten. Durch die Position am Boden kam es leicht zu Verdeckungen durch Passanten sowie zu Änderungen der Lichtverhältnisse durch deren Schatten. Doch auch ohne Verdeckung musste die Perspektive auf den Bildanker genau getroffen werden um die Anwendung auszulösen. Die Bildanker der anderen Streckenabschnitte bieten eine Toleranz gegenüber der Position des Smartphones bei der Bilderkennung und für eine erfolgreiche Bilderkennung ist es nicht nötig, den Bildanker aus der exakten Position zu erfassen, aus welcher er bei der Erstellung aufgenommen wurde. Die Kanalisationsdeckel wurden hingegen nur erkannt, wenn sie bei passender Belichtung im richtigen Winkel aus der richtigen Distanz erfasst wurden.

Bildanker Brauereischild Die Positionierung im letzten Streckenabschnitt ist auf Abbildung 33(b) zu sehen. Dazu wurde ein Schild einer Brauerei an der Fassade eines Gebäudes erkannt. Wie auf dem Bild zu erkennen ist, wurde die Straßenbahn nicht dem Straßenverlauf entsprechend positioniert. Dies ist jedoch nicht auf eine technische Positionierungsungenauigkeit zurückzuführen. Bei der Erstellung wurde übersehen, dass die Wand, an der das Brauereischild angebracht ist, nicht parallel zur Straße steht. Die Erkennung des Bildankers selbst funktionierte zuverlässig bei verschiedenen Belichtungsbedingungen und Aufnahmewinkeln, insofern er aus der Nähe erfasst wurde. Da er mit ca. 40cm Breite relativ

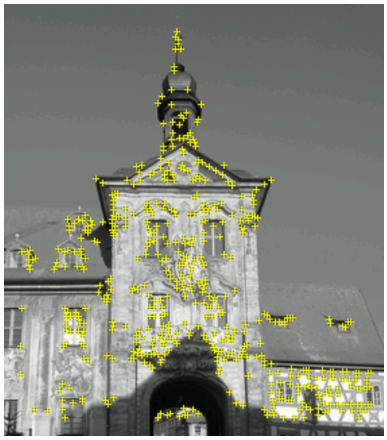
klein ist, konnte er nicht aus großer Distanz erkannt werden. Idealerweise sollte er von dem gegenüberliegenden Gehweg erkannt werden, damit die Straßenbahn zwischen Smartphone und Bildanker erscheint. Das gelang jedoch nur im Einzelfall. Wird der Bildanker von einer Position auf der Straße erkannt, ergibt sich gleiche Problem der Positionierungsnähe wie bei dem zweiten und dritten Streckenabschnitt. Außerdem befindet sich dieser Bildanker an einer von Autos befahrenen Straße, was das Begehen der Straße einschränkt. Wird das Schild aus unmittelbarer Nähe zur Straßenbahnpositionierung aufgenommen, wird sie hinter dem Smartphone und somit außerhalb des Sichtbereichs platziert und es ist nicht unmittelbar ersichtlich, ob die Bilderkennung erfolgreich war.

Zwischenergebnisse verworfener Bildanker Vor der Festlegung der einzelnen Streckenabschnitte und der entsprechenden Bildanker wurden weitere Bilder getestet, die sich aus unterschiedlichen Gründen nicht als Bildanker eigneten. Es folgen die Zwischenergebnisse von zwei Bildankern, welche in der finalen Anwendung nicht zum Einsatz kamen. Sie dienen als Negativbeispiele und verdeutlichen weitere Kriterien bei der Bildauswahl für Bildanker.
Abbildung 34(a) zeigt das Modell des Bamberger Rathauses und seiner Umgebung. Es befindet sich vor dem Alten Rathaus auf der Oberen Brücke. Um diesen Bildausschnitt zu erfassen, kann man direkt an das Modell herantreten. Auch wurden genügend viele markante Punkte erkannt, um dem Bild eine Fünf-Sterne-Wertung im Vuforia Developer Portal einzubringen. Im Gegensatz zu den finalen Bildankern der Anwendung (vgl. Abb. 26), ist das Bild von 34(a) nicht in horizontaler oder vertikaler Lage entstanden sondern in schräger Lage. Abbildung 34(b) zeigt die Positionierung der Straßenbahn, wenn der Winkel dieser Schräglage nicht berücksichtigt wird. Neben der Positionierungsproblematik ist auch die zuverlässige Erkennung dieses Motivs, trotz vieler Erkennungspunkte, nur aus der exakten Perspektive von 34(a) gegeben. Die Erkennungspunkte sind im dreidimensionalen Raum in verschiedenen Distanzen erkannt worden. Durch eine Bewegung des Smartphones ändert sich somit ihre Relation zueinander stärker als bei den verwendeten Bildankern, bei welchen die Erkennungspunkte jeweils näherungsweise auf einer Ebene angebracht sind.
Abblidung 34(c) zeigt die Südwestseite des Rathauses. Es werden viele Punkte erkannt. Zu beachten ist jedoch der Schatten, der von einem benachbarten Gebäude auf das Rathaus geworfen wird. Im Bereich des Schattens selbst werden keine Punkte erkannt. Stattdessen werden an der Kante des Schattens Punkte erkannt, da sich die Pixelwerte hier stark unterscheiden. Diese erkannten Punkte sind jedoch nicht in der permanenten Struktur des Gebäudes begründet und somit zu vermeiden. Dieses Bild wurde im Test von dem Samsung-Smartphone teilweise erkannt, von dem Xiaomi-Smartphone grundsätzlich nicht.
Um das Alte Rathaus vollständig zu erfassen, wurde das Bild mit geneigtem Smartphone aufgenommen und somit ist auch dieses Bild von der Problematik des Neigungswinkels betroffen. Abbildung 34(d) zeigt, wie in Unity versucht wurde mit der Neigungsproblematik umzugehen. Der Bildanker wurde, mit dem Versuch den Neigungswinkel des Smartphones zum Zeitpunkt der Aufnahme zu treffen, schief platziert. Da dieser Winkel nicht dokumentiert war, handelte es sich um eine annähernde Schätzung.
Für eine gute Erkennbarkeit und einfache Objektpositionierung sollten Fotografien, welche als Bildanker dienen sollen, in einem rechten Winkel auf eine Fläche mit Erkennungspunkten aufgenommen werden. Die Erkennungspunkte sollten einer permanenten Struktur oder Farbgebung entstammen und nicht durch aktuelle Belichtungsverhältnisse bedingt sein.

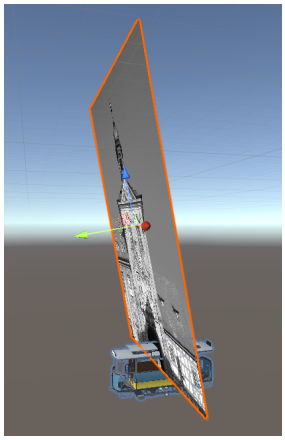


(a) Modell des Rathauses und Umgebung

(b) Falsche Visualisierung



(c) Rathausrückseite mit Schatten

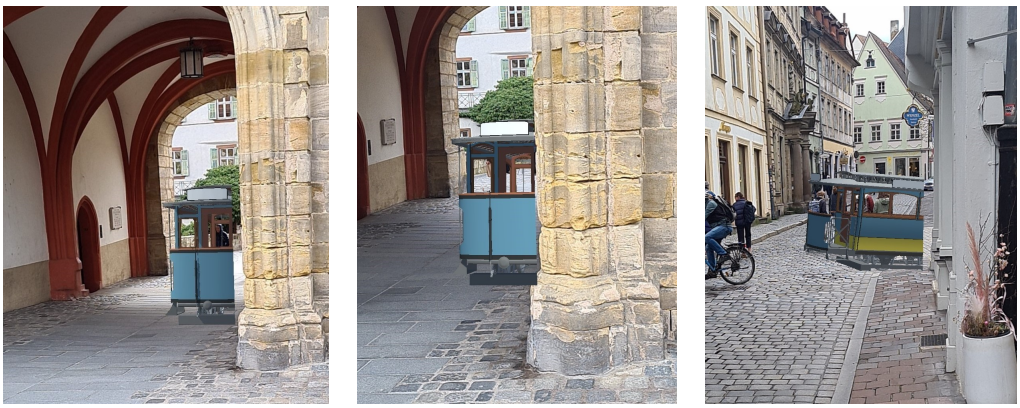


(d) Foto mit Winkel

Abbildung 34: Negativbeispiele

5.1.2 Verdeckung der Straßenbahn

Dieser Abschnitt beschreibt das Verdeckungsverhalten der Straßenbahn. Dabei werden zwei Fälle unterschieden. Einerseits kann es zu Verdeckung mit statischen Elementen wie Gebäuden kommen und andererseits können sich mobile Hindernisse wie Passant:innen oder Fahrzeuge vor die Straßenbahn bewegen.
Der Abschnitt 4.4 zeigt, wie für die Verdeckung von Gebäuden in Unity spezielle Verdeckungselemente eingefügt wurden. Abbildung 35 zeigt die Ergebnisse dieses Ansatzes. Die Abbildungen 35(a) und (b) zeigen die Straßenbahn im ersten Streckenabschnitt nachdem sie in das Torhaus gefahren ist. Dabei wird sie nicht vollständig visualisiert, da sie sich teilweise hinter dem Verdeckungselement befindet. Das Verdeckungselement selbst ist dabei nicht sichtbar und der Blick auf die Umgebung wird dadurch nicht eingeschränkt. Die Verdeckung findet also nur in Bezug auf die Projektion der Straßenbahn statt. Die beiden Bilder unterscheiden sich durch ihre Perspektive. Bei (a) ist die Distanz zur Gebäudewand und zur Straßenbahn etwas größer als bei (b). Während die Verdeckung bei (b) reltativ genau mit dem Gebäude übereinstimmt, ist bei (a) ein deutlicher Versatz erkennbar.



(a) (b) (c)

Abbildung 35: Straßenbahnverdeckung durch Gebäude

Das zeigt, dass die Position des Verdeckungselements nicht genau mit Position der Wand übereinstimmt. Dabei ist zu beachten, dass hier ein einfacher Quader als Verdeckungselement eingefügt wurde und das Verdeckungselement somit nicht genau der Kontur der Wand entspricht.
Abbildung 35(c) zeigt das Verdeckungsergebnis des letzten Streckenabschnitts. Zu sehen ist die Straßenbahn, die gerade in die Herrenstraße abbiegt und ebenfalls halb verdeckt ist. Auch hier ist ein Versatz zwischen dem Verdeckungsbeginn und der tatsächlichen Hauswand zu beobachten. Die Visualisierungspräzision leidet in diesem Streckenabschnitt unter der auf Abbildung 33(b) gezeigten Fehlpositionierung. Diese Fehlpositionierung hat zur Folge, dass nicht nur die Straßenbahn schräg zur Straße angezeigt wird, sondern die gesamte Szene ebenso schief initialisiert wird. So verläuft auch die Animationroute schräg über die Straße und das Verdeckungsobjekt wurde ebenfalls schräg und zu weit links platziert.

(a) Ohne Verdeckung (b) Verdeckung durch Passanten

Abbildung 36: Verdeckung und Personen

Neben statischen Gebäuden kann es in einer belebten Innenstadt auch zu vielen weiteren Verdeckungssituationen kommen. Verdeckungen durch Personen, Fahrzeuge oder weitere mobile Elemente können nicht vorab mit statischen Verdeckungselementen versehen wer-

den und können den Eindruck einer realistischen Visualisierung mindern. Abbildung 36 zeigt dies am Beispiel von Passant:innen. Abbildung 36(a) zeigt einen freien Blick auf die Straßenbahn und erweckt den Eindruck, dass sie in einiger Entfernung auf der Straße steht. Bei Abbildung 36(b) befinden sich Passant:innen im Vordergrund. Da diese die Straßenbahn jedoch nicht verdecken, sondern von der Straßenbahn verdeckt werden, entsteht der Eindruck, die Straßenbahn befinde sich vor den Personen und nicht dahinter. Daraus resultiert außerdem der Eindruck, sie wäre kleiner und würde schweben.
Die im Rahmen dieser Arbeit erstellte Anwendung bietet keine Lösung für derartige Verdeckungsszenarien. Dafür müssten Hindernisse während der Laufzeit erkannt und ihre Visualisierung entsprechend angepasst werden (vgl. Abschnitt 2.4.3).

5.1.3 Visualisierungsstabilität

Dieser Abschnitt beschreibt das Verhalten der projizierten Straßenbahn nach der Platzierung, wenn der Bildanker nicht mehr erfasst wird. Es wird gezeigt, wie sich das Straßenbahnmodell vor und während der Animation verhält und wie es von der Bewegung des Smartphones beeinflusst wird.
Auf dem Samsung Galaxy S20+ ist im ersten Streckenabschnitt nach der Straßenbahnplatzierung und vor dem Start der Animation teilweise eine leichte Bewegung der Straßenbahn in der Bewegungsrichtung des Smartphones festzustellen. Insgesamt erscheinen Position und Ausrichtung des Modells jedoch stabil. Gleiches gilt für das Verhalten während und nach der Animation. Im zweiten Streckenabschnitt war das Verhalten vor der Animation stabil und es waren keine Abweichungen zu beobachten. Nach der Animation war ein leichtes Schweben in die Bewegungsrichtung zu beobachten. In den verbleibenden Abschnitten konnten keine Abweichungen beobachtet werden. Für alle Streckenabschnitte gilt jedoch, dass Animationsverlauf und Endposition der Straßenbahn zwar annäherungsweise gleich blieben, aber nicht immer exakt reproduzierbar waren.
Auf dem Xiaomi Mi A1 war kein stabiles und realistisches Verhalten der Straßenbahn zu beobachten. Sobald der Bildanker nicht mehr im Fokus war bewegte sich das Modell nicht nur mit dem Smartphone weiter, sondern zeigte auch darüber hinaus ein unerwartetes Verhalten. Die Position wurde grob beibehalten, die Ausrichtung änderte sich stark. Das Modell drehte sich so stark, dass es nicht mehr wirkte, als sei die Straßenbahn auf der Straße positioniert. Von dieser Rotation war auch die gesamte hinterlegte Szene betroffen. Entsprechend verlief die Animationsroute ebenfalls nicht mehr der Straße entsprechend und die Verdeckungselemente wurden ebenfalls falsch platziert. Abbildung 37 zeigt die Straßenbahn während der Fahrt entlang der rotierten Animationsroute. Auf Abbildung 37(a) ist die Obere Brücke zu sehen, welche die Straßenbahn im zweiten Streckenabschnitt entlang fahren soll. Die Szene ist jedoch um mehrere Achsen rotiert und die Straßenbahn bewegt sich weder entlang des Streckenverlaufs, noch ist ihre Ausrichtung der Oberflächenneigung entsprechend. Abbildung 37(b) zeigt das gleiche Problem an der Kreuzung von Karolinen- und Herrenstraße im vierten Streckenabschnitt. Die Straßenbahn scheint hier nicht um die Kurve zu fahren, sondern durch die Luft zu fliegen. Das Zusammenspiel von Umgebung und digitaler Erweiterung funktioniert hier nicht im Sinne der Anwendung, da der Umgebungsbezug verlorengeht.

(a) Zweiter Streckenabschnitt (b) Vierter Streckenabschnitt

Abbildung 37: Fehlende Visualisierungsstabilität des Xiaomi Mi A1

5.1.4 Nutzerfeedback

Die Anwendung wurde sieben Personen zum Testen zur Verfügung gestellt. Sie nutzten dazu jeweils das eigene Smartphone. Um eine externe Perspektive auf die erstellte Anwendung zu erhalten, wurden diese Personen anschließend um eine Rückmeldung zur Nutzung der Anwendung gebeten. Abgefragt wurden der Gesamteindruck, die Bewertung der Bedienbarkeit sowie eigene Anmerkungen.
Mit jeder Testperson wurde die Strecke mit den vier Startpunkten zweimal abgegangen. Dabei wurden die vorangegangenen Testergebnisse in großen Teilen bestätigt, es gab jedoch auch Ergebnisse, die in den eigenen Tests nicht zu beobachten waren. In der Folge werden die erhobenen Ergebnisse vorgestellt. Um Aufschlüsse darüber zu erhalten, ob eine Steuerung des Verhaltens während der Nutzung die Qualität der Darstellung der Straßenbahn und des entsprechenden Erlebnisses bei der Nutzung verbessert, erhielten sie dabei unterschiedlich viele Informationen. Im ersten Durchlauf wurde den Personen lediglich gesagt, dass es vier Stationen gibt, die mit dem Smartphone zur Positionierung erfasst werden müssen und die vier Stationen wurden genannt. Welche Bildausschnitte genau als Bildanker dienen und wie diese am besten zu erfassen sind, wurde erst im zweiten Durchlauf genauer erklärt. Dazu wurden die Bildanker gezeigt und erklärt, dass diese idealerweise aus der gleichen Perspektive zu erfassen sind, aus der sie aufgenommen wurden.
Der Gesamteindruck wurde trotz unterschiedlicher Probleme insgesamt als gut bewertet. Ein Teil der Probleme konnten durch die Nutzersteuerung in der zweiten Testrunde minimiert werden. Der Umgang mit dem digitalen Objekt in der realen Umgebung war für alle Testpersonen intuitiv und regte sie zu Ideen bezüglich inhaltlicher Erweiterung und möglichen Einsatzgebieten an. Im Detail ergaben sich folgende Ergebnisse:

Bilderkennung Die Erkennung der Bildanker klappte in allen Testläufen sehr gut. In den ersten beiden Abschnitten klappte diese auch bei Teilverdeckung des Bildankers zuverlässig. Abbildung 38(a) zeigt das Ergebnis einer erfolgreichen Erkennung und Positionierung trotz der teilweisen, beflaggungsbedingten Verdeckung des Bildausschnitts, welcher als Bildanker dient. Die Ausnahmen dazu ergaben sich bei starker Sonneneinstrahlung.

Dadurch wurde in den meisten Fällen die Erkennung der Kanalisationsdeckel des dritten Streckenabschnitts verhindert oder zumindest erschwert und in einem Fall war die Erkennung der Rathausfassade des ersten Abschnitts nicht möglich. Der Bildanker der Kanalisationsdeckel konnte bei Sonneneinstrahlung, wenn überhaupt, erst im zweiten Testdurchlauf erkannt werden. In einem Test lag der Bildanker im Schatten und konnte aus der Perspektive erkannt werden, welche auch Abbildung 38(b) zu sehen ist. Zur Bilderkennung wurde angemerkt, dass die Auffindung der Position, aus der ein Bildanker erkannt werden kann, im ersten Testlauf schwerfiel. Im angeleiteten zweiten Testlauf gelang das einfacher.

Objektpositionierung Im Normalfall ging mit einer erfolgreichen Bilderkennung auch eine korrekte Objektplatzierung einher. In wenigen Fällen trat nach der Bilderkennung eine deutlich fehlerhafte Objektpositionierung auf. Dies ist auf Abbildung 38(c)+(d) zu sehen. Im Fall von (c) steht die Straßenbahn nicht auf dem Boden sondern schwebt schräg in der Luft. Dazu wurde angemerkt, dass die Immersion darunter leidet, da es nicht wie ein realistisches Objekt platziert wird und sich nicht realistisch verhält. Dies war ein Ergebnis aus einem ungeleiteten Testlauf und trat im folgenden geleiteten Testlauf nicht auf. In weiteren Fällen traten kleinere Fehlplatzierungen, wie eine falsche Platzierungshöhe, während des ersten Testlaufs auf. Auch diese konnten jeweils im zweiten Durchgang vermieden werden. Im Einzelfall von (d) steht die Straßenbahn zwar auf dem Boden, ist aber um ca. 90° gedreht und zeigt in die falsche Richtung. Wurde die Animation gestartet, fuhr sie auch in diese falsche Richtung. Diese Fehlplatzierung im dritten Abschnitt (d) ließ sich auch durch Anleitung nicht beheben. Kleinere Unterschiede in der Positionsausrichtung führten zu leicht abweichenden Fahrtstrecken der Straßenbahn. Der Fehler war dabei jedoch meist vernachlässigbar und führte nur dann zu einem merkbar schlechteren Ergebnis, wenn es sich um einen Streckenabschnitt mit Verdeckungselementen handelte.

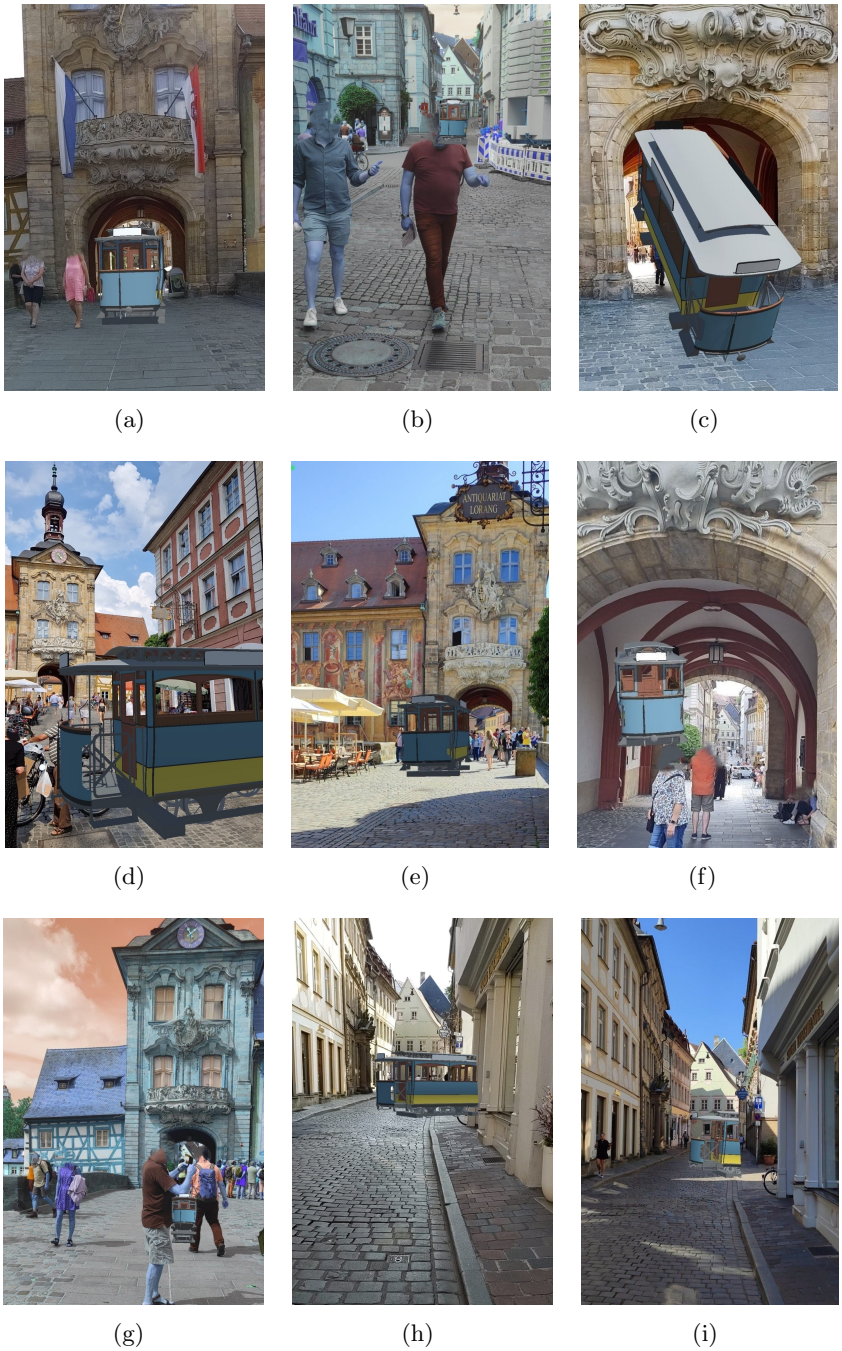
Objektstabilität Die Objektstabilität bzw. das Verhalten der Straßenbahn im Kontext der realen Umgebung wurde am häufigsten positiv kommentiert. Ein Beispiel für die gute Stabilität zeigt Abbildung 38(e). Hier ist die Straßenbahn auf dem zweiten Streckenabschnitt zu sehen. Die Aufnahme wurde jedoch von dem Startpunkt des dritten Abschnitts gemacht. Der Bildanker des zweiten Abschnitts wurde im Tor des Rathauses erkannt und die Testperson hat sich bis zu dem Startpunkt des dritten Abschnitts bewegt und blickt von dort zurück auf die fahrende Straßenbahn. Dabei wurde die auf (e) zu sehende Strecke zwischen Aufnahmeposition und Rathaus zurückgelegt. Trotz der großen Distanz zwischen der Aufnahmeposition und der Initialisierung der digitalen Inhalte, werden die Straßenbahn und ihre Animationsroute korrekt dargestellt. In zwei Testfällen war jedoch keinerlei Stabilität gegeben. Das hatte zur Folge, dass die Straßenbahn zitternd an ihrem Startpunkt angezeigt wurde, solange der Bildanker im Fokus war. Nach dem Start der Animation konnte der Bildanker meist nicht weiter erfasst. Ohne erfassten Bildanker ging in diesen Fällen auch der Bezug zur realen Umgebung und der Kontakt zum Boden verloren (vgl. Abb. 38(f)). Abbildung 38(g) zeigt ein weiteres Beispiel, bei dem die Straßenbahn nach der Animation nicht mehr korrekt dargestellt wird. Die Stabilität ist nicht wie im zuvor beschriebenen Fall komplett verloren, war jedoch auch hier nicht in ausreichendem Maß gegeben und die Straßenbahn scheint sich unter dem Boden zu befinden. Beim mehrfachen Testen gleicher Streckenabschnitte wurde festgestellt, dass sich die Straßenbahn nicht immer exakt auf der selben Strecke bewegt.

Verdeckungsverhalten Das Verdeckungsverhalten wurde gemischt bewertet. Das gilt sowohl in Bezug auf die implementierte Verdeckung durch statische Elemente als auch in Bezug auf das Verhalten gegenüber Passanten und weiteren nicht permanenten Objekten im Stadtraum. Dass die Straßenbahn hinter Gebäuden verschwand, wenn sie sich dahinter bewegte, wurde positiv bewertet. Ein Teil der Befragten störte sich auch nicht an dem Versatz zwischen Gebäude und Verdeckungselement. Ein anderer Teil bewertete diese ungenaue Verdeckung negativ. Ein Tester merkte an: „Objektverdeckung ist cool, der User muss aber am perfekten Standpunkt stehen, damit es gut aussieht“. Dieses Verhalten tritt auf, sofern die Position des digitalen Verdeckungsobjekt nicht exakt mit dem realen Objekt, in diesem Fall der Hauswand, übereinstimmt. Das kann durch Ungenauigkeit bei der Erstellung bedingt sein. Auf den Abbildungen 38(h)+(i) wird deutlich, dass die Szene und die darin enthaltenen Verdeckungselemente bei der Bilderkennung nicht immer an der gleich Position initialisiert werden und sich der Fehler der Verdeckungsposition auch unabhängig von Erstellungsfehlern und der Aufnahmeposition verändert. Zur Verdeckung bezüglich Passanten und mobiler Hindernisse wurde sowohl kritisiert als auch positiv bewertet, dass die Straßenbahn diesen gegenüber stets im Vordergrund bleibt. Einerseits leidet der Eindruck, ein reales Objekt zu betrachten, wenn dieses nicht hinter anderen Objekten verdeckt wird, andererseits wird so ermöglicht, dass die Straßenbahn auch bei regem Verkehr betrachtet und verfolgt werden kann.

Weitere Ergebnisse Der Umgang und die Interaktion mit der digitalen Straßenbahn gelang allen Tester:innen ohne Probleme und bedurfte keiner weiteren Erklärung.
In zwei Testläufen wurden die Farben des eingehenden Kamerabildes falsch dargestellt. Zu sehen ist dies auf den Abbildungen 38(b)+(g), welche einem Smartphone der Reihe Redmi Note 9 entstammen.
Bei einem Testlauf an einem heißen, sonnigen Tag kam es zu einer Überhitzung des verwendeten Smartphones, welches sich daraufhin ausschaltete und ein paar Minuten abkühlen musste, bevor es weiter verwendet werden konnte.
Weitere Anmerkungen waren, dass der Umgang Spaß gemacht habe, die Animation der Straßenbahnfahrt in der Straße realistisch wirke und die Anwendung für Stadtführungen oder zur Vermittlung historischer Fakten an Schüler:innen geeignet sei. Auch wurde der Wunsch nach einer inhaltlichen Erweiterung geäußert. Genannt wurden weitere Modelle, wie Schienen und Passagiere, sowie die Verknüpfung der Anwendung mit Informationen zu historischen Hintergründen.

Die Ergebnisse im Abschnitt 5.1 haben gezeigt, dass die Verwendung von Bilderkennung zur Objektpositionierung grundsätzlich möglich ist, die Auswahl des Bildes und die äußeren, variierenden Bedingungen jedoch Auswirkungen auf den Visualisierungserfolg und die Visualisierungsqualität der Anwendung haben.
Ein Beispiel für die Abhängigkeit der Qualität vom Bildanker und dessen Positionierung in der Szene wurde bei der Verdeckung der Straßenbahn gezeigt. Hier wurde eine einfache Methode vorgestellt, die Objektverdeckung umzusetzen. Der Eindruck, die Straßenbahn würde von einem Gebäude verdeckt werden, ist dabei stark von der präzisen Positionierung der Verdeckungselemente abhängig. Außerdem kann das Verdeckungsverhalten gegenüber Passanten und anderen mobilen Hindernissen einen realistischen Eindruck mindern.

Im Bereich der Visualisierungsstabilität, konnten geräteabhängig große Unterschiede festgestellt werden. Auch wenn das Ausführen der Anwendung auf dem älteren Xiaomi Mi A1 grundsätzlich möglich ist, lassen sich damit keine akzeptablen Ergebnisse erzielen.
Die Testdurchläufe mit Testproband:innen bestätigten die vorherigen Ergebnisse, zeigten jedoch auch, dass eine AR-Darstellung historischer Objekte auf Interesse stößt und trotz verschiedener Probleme insgesamt positiv bewertet wird.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Abbildung 38: Ergebnisse von Testproband:innen

5.2 Einordnung

Dieser Abschnitt ordnet die gezeigten Ergebnisse des vorangegangenen Abschnitts ein und zeigt, welche Faktoren für eine erfolgreiche Ausführung und eine realistische Darstellung von Bedeutung sind.
Als Hauptfaktoren für Ergebnisqualität konnten die Belichtung bzw. die Sonneneinstrahlung und die Wahl des verwendeten Smartphones identifiziert werden. Sie gelten als Hauptfaktoren, da sie nicht nur bessere oder schlechtere Ergebnisse bedingen, sondern eine sachgemäße Verwendung gänzlich verhindern können. Als weitere Faktoren kommen das Nutzerverhalten und Verdeckungen in der Szene zum Tragen.

Sonneneinstrahlung Bei starker Sonneneinstrahlung entstanden starke Schlagschatten am Balkon der ersten Station und der Bildanker konnte nicht erkannt werden. Auch bei der dritten Station wurden die Kanalisationsdeckel selten erkannt, wenn sie direktem Sonnenlicht ausgesetzt waren. Bei bewölktem Wetter oder einem Sonnenstand, bei dem die Bildanker keinem direkten Sonnenlicht ausgesetzt waren, funktionierten sie gut.
Eine Erklärung dazu kann in der Methode zur Auffindung von Schlüsselpunkten gefunden werden (vgl. Abschnitt 2.4.1). Der Abgleich gefundener Schlüsselpunkte mit den hinterlegten Schlüsselpunkten eines Bildankers ist grundlegend für die Bilderkennung. Für die Auffindung von Schlüsselpunkten werden die Pixelwerte des Bildes betrachtet. Vuforia verwendet Eckerkennung zur Identifizierung von Schlüsselpunkten und Ecken werden an Punkten mit starken Pixelkontrasten gefunden. Zeigt das eingehende Bild starke Schlagschatten, sind diese Kontraste am Rand des Schattens zu finden und nicht in der permanenten Struktur begründet, wie die Schlüsselpunkte des hinterlegten Bildankers.
Die Bildanker der letzten beiden Streckenabschnitte zeigen auch ohne Schlagschatten ein unterschiedliches Verhalten bei Sonneneinstrahlung. Das Brauereischild funktioniert weiterhin problemlos. Die Kanalisationsdeckel werden als Bildanker meist nicht mehr erkannt. Bildanker können unterschiedlich stabil gegenüber Sonneneinstrahlung sein.

Smartphone-Modell Im Vergleich der verwendeten Smartphones konnte festgestellt werden, dass die Anwendung auf aktuellen Smartphones gut läuft. Die Bilderkennung funktioniert schnell, die Objektplatzierung ist trotz kleinerer Abweichungen meist präzise und schnell. Das 3D-Modell verhält sich sehr stabil gegenüber der Umgebung. Auf älteren, schwächeren Smartphones funktioniert die Bilderkennung noch gut. Das 3D-Modell erscheint meist auch an der richtigen Stelle, wo es jedoch nicht stabil der Umgebung gegenüber bleibt. Es ist keine Visualisierungsstabilität jenseits des Bildankers mehr gegeben. Die Bewegungsverfolgung funktionierte nicht in dem Maße, als dass die angedachte Nutzung möglich gewesen wäre.
Im Rahmen dieser Arbeit wurde dabei nicht geklärt, ob die fehlende Objektstabilität auf älteren Smartphones deshalb nicht funktioniert, da der verwendete SLAM-Allgorithmus für diese Geräte zu rechenintensiv ist, oder ob andere Gründe die Objektstabilität verhindern. Eine mögliche Erklärung liegt in den verschiedenen SLAM-Methoden, die von Vuforia verwendet werden. Vuforia verwendet je nach Gerät die bestmögliche, kompatible SLAM-Methode.[1] Es können drei unterschiedliche SLAM-Methoden zum Einsatz kommen und zu unterschiedlichen Ergebnissen führen.

[1] https://library.vuforia.com/environments/vuforia-fusion, aufgerufen: 03.07.23

Nutzerverhalten Das Nutzerverhalten spielte für die Auffindung der Bildanker und die Präzision der Positionierung der Straßenbahn eine Rolle. In ungeleiteten Testläufen mussten Testproband:innen zunächst nach der Position suchen, von welcher aus der Bildanker erkannt werden konnte. Wird der Bildanker nicht vollständig erfasst, kann er trotzdem erkannt werden. Gegenüber einer vollständigen Erfassung liegen erkannte Schlüsselpunkte dann in geringerer Zahl vor und es können nur die erkannten Punkte zu einer Positionierung verwendet werden. In der Folge kam es teilweise zu unpräzisen oder falschen Platzierungen der Straßenbahn. Durch eine zusätzliche Anleitung der Tester:innen konnten sie den Bildanker schnell und zielgerichtet erfassen, was die Bilderkennung beschleunigte und die Positionierung genauer machte.

Verdeckung Die Verdeckung spielt in zwei Bereichen eine Rolle für die Ausführung der Anwendung. Verdeckungen von Bildankern können die Bilderkennung beeinträchtigen und das Verdeckungsverhalten zwischen 3D-Objekt und Umwelt beeinflusst, wie realistisch sich das 3D-Objekt in die Realität einfügt.
Bei einer vollständigen Verdeckung eines Bildankers kann dieser nicht erkannt werden, Teilverdeckungen stellten in den Testläufen meist keine Probleme für die Erkennung dar. Die Positionierung und Bewegungsverfolgung wurde jedoch auf Smartphones, auf denen Positionierung und Bewegungsverfolgung ohnehin Probleme darstellten, zusätzlich negativ durch Passant:innen und deren Bewegungen beeinflusst.
Die Gefahr der Bildankerverdeckung sollte bei der Erstellung berücksichtigt und Bildanker so gewählt werden, dass das Verdeckungsrisiko minimiert wird. Während im Rahmen dieser Arbeit keine Methode für den Umgang mit mobilen Verdeckungen implementiert wurde, hängt die Präzision der statischen Verdeckung, welche durch Verdeckungselemente implementiert wurde, von zwei Faktoren ab. Diese Faktoren sind die Positionierungspräzision der digitalen Szene bei der Initialisierung und die Präzision bei der Erstellung der digitalen Verdeckungen. Eine realistische Darstellung der Verdeckung entsteht durch die lokale Übereinstimmung der digitalen Verdeckungselemente mit realen Objekten z.B. Wänden, deren Verdeckungsverhalten simuliert werden soll.
Wird auf eine technische Vermessung der Szene verzichtet, erschweren die Positionsabweichungen der digitalen Szene eine iterative Positionsannäherung der digitalen Inhalte an die realweltliche Umgebung. Insbesondere im vierten Streckenabschnitt ist zu beobachten, dass die Bahn nicht parallel zum Gehweg die Straße entlangfährt und die Verdeckung einen Versatz zum realen Gebäude hat. Zurückzuführen ist dieses Verhalten auf die Ausrichtung der Fassade, an welcher der Bildanker des vierten Streckenabschnitts angebracht ist. Die Fassade ist ebenfalls nicht exakt parallel zur Straße, was bei genauer Beobachtung auch mit bloßem Auge erkennbar aber nicht unmittelbar offensichtlich ist. Die Differenzen der Verdeckungspositionierung im vierten Streckenabschnit (vgl. Abb. 33 und Abb. 38(h)+(i)) verdeutlichen die abweichende Initialisierung der Szene und die Schwierigkeit der iterativen Positionsannäherung, welche auf Testergebnissen aufbaut.

Kapitel 6

Fazit

Diese Arbeit hat den Fall von bewegten 3D-Objekten im urbanen Raum als Herausforderung von Augmented-Reality-Systemen am Beispiel der ehemaligen der ehemaligen Straßenbahn Bambergs untersucht. Abschließend kann festgestellt werden, dass dieser Anwendungsfall möglich und mit Bilderkennung realisierbar ist, dabei jedoch einigen Einschränkungen unterliegt.
Als Maßstab zur Bewertung des Anwendungsfalls wurde die Bewältigung der Herausforderungen im Bereich der Bilderkennung, der Objektpositionierung, der Objektstabilität und der Verdeckung herangezogen. Bedingt wird die Bewältigung der Herausforderungen durch die Faktoren der Sonneneinstrahlung, der Wahl des verwendeten Smartphones, dem Nutzerverhalten und Verdeckungen.

Die Umsetzung der Straßenbahn, welche über vier Stationen von der Oberen Brücke über die Karolinenstraße in die Herrenstraße fährt, hat trotz auftretender Probleme in all diesen Bereichen gezeigt, dass sie zu bewältigen sind:

- Bilderkennung kann im Stadtraum funktionieren.
- Es können nicht flache Objekte (z.B. Rathausbalkon) als Bildanker zur Bilderkennung und Objektpositionierung dienen.
- Objekte können realistisch positioniert werden.
- Objekte können sehr stabil gegenüber der Umgebung visualisiert werden und sich realistisch bewegen.
- Digitale Verdeckungsobjekte können das Verdeckungsverhalten von Gebäuden simulieren(vgl. Abb.35(b)).

Die Ergebnisse des vorangegangen Kapitels zeigten, dass die genannten Bereiche nicht immer erfolgreich bewältigt wurden und von den genannten Faktoren abhängen. Aus den Herausforderungen und Faktoren ergeben sich Konsequenzen für die Erstellung und sie zeigen, in welchen Bereichen andere Ansätze und weitere Forschung die Ergebnisse verbessern könnten.
Für die Bilderkennung ergibt sich daraus, dass sie zwar möglich ist, jedoch nicht überall. Die Anforderungen an Bildanker im urbanen Raum sind höher als die Anforderungen an Bildanker, die unter kontrollierten Bedingungen erfasst werden können. Ihre Position muss

gut sichtbar und erfassbar und ihre Schlüsselpunkte müssen stabil gegenüber Sonneneinstrahlung sein. Bei der Anwendung dieser Arbeit trifft das auf die Bildanker 26(b)+(d) zu. Die Informationstafeln des zweiten Abschnitts waren stets vor direkter Sonneneinstrahlung geschützt, das Brauereischild wurde unabhängig von der Sonne erkannt. Bildanker, die nicht flache Objekte zeigen, können bei konstanten Belichtungsverhältnissen gut erkannt werden. Variieren die Belichtungsverhältnisse, erhöht die Struktur des abgebildeten Objekts die Gefahr von Schlagschatten. Für die Erkennung von Bildankern in unkontrollierten Umgebungen kann weitergehend untersucht werden, ob alle möglicherweise auftretenden Zustände für eine Erkennung hinterlegt werden können oder ob Kombinationen mit anderen Technologien die negativen Umwelteinflüsse minimieren können.
Für die Positionierung ergeben sich gute Ergebnisse, wenn Nutzer:innen mit ausreichend Informationen zur Nutzung der Anwendung versorgt werden. Die iterative Annäherung während der Erstellung reichte für eine überzeugende Positionierung und Animation der Straßenbahn aus. Für die Positionierung von Verdeckungselementen werden jedoch präzise Daten benötigt. Hinsichtlich der Positionsabweichungen des letzten Streckenabschnitts könnte der Einsatz eines größeren Bildankers für mehr Winkel- bzw. Positionierungsstabilität untersucht werden.
Die Stabilität des 3D-Modells im Raum war in den meisten Fällen sehr überzeugend. Dies gilt jedoch nicht für alle getesteten Smartphones und es ergibt sich die Einschränkung, dass eine sinngemäße Nutzung nur mit kompatiblen Smartphones möglich ist.
Die Qualität der Verdeckung des 3D-Modells durch statische Objekte hängt unmittelbar von der Positionierungspräzision ab. Für die Erstellung dieser Verdeckungen sollte die Umgebung präzise vermessen werden. Bereits kleine Abweichungen sind sonst, abhängig von der Nutzerposition, deutlich sichtbar. Für die Positionierung von Verdeckungselementen könnten künftig außerdem Kombinationen mit anderen Technologien, wie Flächen- oder Objekterkennung, untersucht werden. Die Verdeckung mobiler Hindernisse wurde diskutiert, im Rahmen dieser Arbeit wurde aber keine Methode dafür implementiert. Im Abschnit 2.4.3 wurden Methoden vorgestellt, die zur Laufzeit für die Verdeckungsberechnung genutzt werden können und die keine vorab erstellten Verdeckungen benötigen. Die vorgestellten Arbeiten gehen dabei auf die Verwendung von Smartphones ein, nutzen aber selbst PCs zur Berechnung. Auch hier könnte weiterführend untersucht werden, welche Ergebnisse tatsächlich auf Smartphones mit diesen Methoden erzielt werden können.

Diese Arbeit hat gezeigt, dass mit Augmented Reality bereits anspruchsvolle Szenarien visualisiert werden können. Herausforderungen wie Bilderkennung, Positionierung und Stabilität von 3D-Objekten sowie statische Verdeckungen sind weitestgehend gelöst und können von aktuellen Smartphones bewältigt werden. Für die Verdeckungsberechnung zur Laufzeit existieren vielversprechende Ansätze.
Ein in dieser Arbeit nicht betrachteter Aspekt der Visualisierungsqualität ist der, der Oberflächen- bzw. Texturdarstellung. Die Erscheinung eines realen Objekts variiert je nach Material mehr oder weniger stark abhängig von Umgebung, Belichtung und Witterung. Auch der Schattenwurf ist abhängig von Richtung und Stärke der Belichtung. Grundlagen für Lösungsansätze zu diesen Problemen sind z.B. bei Rohmer et al. [2014], Kan und Kaufmann [2012] und Dos Santos et al. [2012] zu finden und werden in Teilen von aktuellen AR-SDKs schon unterstützt.

Literaturverzeichnis

Alzahrani, N. M. und Alfouzan, F. A. (2022). Augmented Reality (AR) and Cyber-Security for Smart Cities—A Systematic Literature Review. *Sensors*, 22(7):2792.

Amin, D. und Govilkar, S. (2015). Comparative study of augmented reality SDKs. *International Journal on Computational Science & Applications*, 5(1):11–26.

Azuma, R. T. (1997). A survey of augmented reality. *Presence: teleoperators & virtual environments*, 6(4):355–385.

Bay, H., Tuytelaars, T., und Van Gool, L. (2006). Surf: Speeded up robust features. *Lecture notes in computer science*, 3951:404–417.

Bekele, M. K., Pierdicca, R., Frontoni, E., Malinverni, E. S., und Gain, J. (2018). A survey of augmented, virtual, and mixed reality for cultural heritage. *Journal on Computing and Cultural Heritage (JOCCH)*, 11(2):1–36.

Billinghurst, M., Clark, A., und Lee, G. (2015). A survey of augmented reality.

Boboc, R. G., Duguleană, M., Voinea, G.-D., Postelnicu, C.-C., Popovici, D.-M., und Carrozzino, M. (2019). Mobile augmented reality for cultural heritage: Following the footsteps of Ovid among different locations in Europe. *Sustainability*, 11(4):1167.

Brehm (1991). *1897 - 1991, Öffentlicher Personennahverkehr in Bamberg : die Geschichte in Texten, Bildern und Dokumenten*. Stadtwerke Bamberg.

Catmull, E. E. (1974). *A subdivision algorithm for computer display of curved surfaces*. The University of Utah.

Cheng, J. C., Chen, K., und Chen, W. (2017). Comparison of marker-based AR and markerless AR: A case study on indoor decoration system. In *Lean and Computing in Construction Congress (LC3): Proceedings of the Joint Conference on Computing in Construction (JC3)*, Seiten 483–490.

Di Giulio, R., Maietti, F., Piaia, E., Medici, M., Ferrari, F., und Turillazzi, B. (2017). Integrated data capturing requirements for 3D semantic modelling of cultural heritage: the INCEPTION protocol. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:251.

Dos Santos, A. L., Lemos, D., Lindoso, J. E. F., und Teichrieb, V. (2012). Real time ray tracing for augmented reality. In *2012 14th Symposium on Virtual and Augmented Reality*, Seiten 131–140. IEEE.

El-Rabbany, A. (2002). *Introduction to GPS: the global positioning system*. Artech house.

Engel, J., Koltun, V., und Cremers, D. (2017). Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625.

Fan, Y., Feng, Z., Mannan, A., Khan, T. U., Shen, C., und Saeed, S. (2018). Estimating tree position, diameter at breast height, and tree height in real-time using a mobile phone with RGB-D SLAM. *Remote Sensing*, 10(11):1845.

Furht, B. (2011). *Handbook of augmented reality*. Springer Science & Business Media.

Gretzel, U., Reino, S., Kopera, S., und Koo, C. (2015). Smart tourism challenges. *Journal of Tourism*, 16(1):41–47.

Hameed, Q. A., Hussein, H. A., Ahmed, M., und Omar, M. B. (2022). Development of Augmented Reality-based object recognition mobile application with Vuforia. *Journal of Algebraic Statistics*, 13(2):2039–2046.

Hanafi, A., Elaachak, L., und Bouhorma, M. (2019). A comparative study of augmented reality SDKs to develop an educational application in chemical field. In *Proceedings of the 2nd International Conference on Networking, Information Systems & Security*, Seiten 1–8.

Harris, C., Stephens, M., u a. (1988). A combined corner and edge detector. In *Alvey vision conference*, Band 15, Seiten 10–5244.

Herling, J. und Broll, W. (2011). Markerless tracking for augmented reality. *Handbook of augmented reality*, Seiten 255–272.

Holynski, A. und Kopf, J. (2018). Fast depth densification for occlusion-aware augmented reality. *ACM Transactions on Graphics (ToG)*, 37(6):1–11.

Jinyu, L., Bangbang, Y., Danpeng, C., Nan, W., Guofeng, Z., und Hujun, B. (2019). Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality. *Virtual Reality & Intelligent Hardware*, 1(4):386–410.

Kähler, O., Prisacariu, V. A., Ren, C. Y., Sun, X., Torr, P., und Murray, D. (2015). Very high frame rate volumetric integration of depth images on mobile devices. *IEEE transactions on visualization and computer graphics*, 21(11):1241–1250.

Kan, P. und Kaufmann, H. (2012). High-quality reflections, refractions, and caustics in augmented reality and their contribution to visual coherence. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Seiten 99–108. IEEE.

Khan, M. M. und Soroni, F. (2022). Markerless Location-Based Augmented Reality Application for Showcasing Deals. In *Topical Drifts in Intelligent Computing: Proceedings of International Conference on Computational Techniques and Applications (ICCTA 2021)*, Seiten 127–136. Springer.

Khan, T., Johnston, K., und Ophoff, J. (2019). The impact of an augmented reality application on learning motivation of students. *Advances in Human-Computer Interaction*, 2019.

Klein, G. und Murray, D. (2009). Parallel tracking and mapping on a camera phone. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, Seiten 83–86. IEEE.

Lepetit, V., Moreno-Noguer, F., und Fua, P. (2009). EPnP: An accurate O(n) solution to the PnP problem. *International journal of computer vision*, 81:155–166.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, Band 2, Seiten 1150–1157. IEEE.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110.

Milgram, P., Takemura, H., Utsumi, A., und Kishino, F. (1995). Augmented reality: A class of displays on the reality-virtuality continuum. In *Telemanipulator and telepresence technologies*, Band 2351, Seiten 282–292. Spie.

Moravec, H. P. (1977). Towards Automatic Visual Obstacle Avoidance. In *International Joint Conference on Artificial Intelligence*.

Neubeck, A. und Van Gool, L. (2006). Efficient non-maximum suppression. In *18th international conference on pattern recognition (ICPR'06)*, Band 3, Seiten 850–855. IEEE.

Newcombe, R. A., Lovegrove, S. J., und Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, Seiten 2320–2327. IEEE.

Newell, M. E., Newell, R., und Sancha, T. L. (1972). A solution to the hidden surface problem. In *Proceedings of the ACM annual conference-Volume 1*, Seiten 443–450.

Nowacki, P. und Woda, M. (2020). Capabilities of arcore and arkit platforms for ar/vr applications. In *Engineering in Dependability of Computer Systems and Networks: Proceedings of the Fourteenth International Conference on Dependability of Computer Systems DepCoS-RELCOMEX, July 1–5, 2019, Brunów, Poland*, Seiten 358–370. Springer.

Ondrúška, P., Kohli, P., und Izadi, S. (2015). Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones. *IEEE transactions on visualization and computer graphics*, 21(11):1251–1258.

Oufqir, Z., El Abderrahmani, A., und Satori, K. (2020). ARKit and ARCore in serve to augmented reality. In *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, Seiten 1–7. IEEE.

Paavilainen, J., Korhonen, H., Alha, K., Stenros, J., Koskinen, E., und Mayra, F. (2017). The Pokémon GO experience: A location-based augmented reality mobile game goes mainstream. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, Seiten 2493–2498.

Panou, C., Ragia, L., Dimelli, D., und Mania, K. (2018). An architecture for mobile outdoors augmented reality for cultural heritage. *ISPRS International Journal of Geo-Information*, 7(12):463.

Passaro, V. M., Cuccovillo, A., Vaiani, L., De Carlo, M., und Campanella, C. E. (2017). Gyroscope technology and applications: A review in the industrial perspective. *Sensors*, 17(10):2284.

Petrou, M. M. und Petrou, C. (2010). *Image processing: the fundamentals*. John Wiley & Sons.

Prigge, E. A. und How, J. P. (2004). Signal architecture for a distributed magnetic local positioning system. *IEEE sensors journal*, 4(6):864–873.

Rad, M. und Lepetit, V. (2017). Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE international conference on computer vision*, Seiten 3828–3836.

Rohmer, K., Büschel, W., Dachselt, R., und Grosch, T. (2014). Interactive near-field illumination for photorealistic augmented reality on mobile devices. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Seiten 29–38. IEEE.

Rosten, E. und Drummond, T. (2006). Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, Seiten 430–443. Springer.

Rublee, E., Rabaud, V., Konolige, K., und Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision*, Seiten 2564–2571. Ieee.

Schöps, T., Engel, J., und Cremers, D. (2014). Semi-dense visual odometry for AR on a smartphone. In *2014 IEEE international symposium on mixed and augmented reality (ISMAR)*, Seiten 145–150. IEEE.

Siriwardhana, Y., Porambage, P., Liyanage, M., und Ylianttila, M. (2021). A survey on mobile augmented reality with 5G mobile edge computing: architectures, applications, and technical aspects. *IEEE Communications Surveys & Tutorials*, 23(2):1160–1192.

Straßer, W. (1974). *Schnelle kurven-und flächendarstellung auf grafischen sichtgeräten*. Dissertation.

Taketomi, T., Uchiyama, H., und Ikeda, S. (2017). Visual SLAM algorithms: A survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):1–11.

Tian, Y., Guan, T., und Wang, C. (2010). Real-time occlusion handling in augmented reality based on an object tracking approach. *Sensors*, 10(4):2885–2900.

Tuytelaars, T., Mikolajczyk, K., u a. (2008). Local invariant feature detectors: a survey. *Foundations and trends® in computer graphics and vision*, 3(3):177–280.

Verykokou, S., Boutsi, A.-M., und Ioannidis, C. (2021). Mobile Augmented Reality for Low-End Devices Based on Planar Surface Recognition and Optimized Vertex Data Rendering. *Applied Sciences*, 11(18):8750.

Wagner, A., Panzer, L., und Löffler, P. (2022). Augmented Reality Timetravel - Die digitale Straßenbahn. *Unveröffentlichtes Manuskript*. Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik.

Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., und Schmalstieg, D. (2010). Real-Time Detection and Tracking for Augmented Reality on Mobile Phones. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):355–368.

Wußmann, Müller, S. (2018). *Die Straßenbahn kommt: Die Geschichte der Bamberger Tram von 1897 bis 1922*. Heinrichs-Verlag GmbH.

Xiong, J., Hsiang, E.-L., He, Z., Zhan, T., und Wu, S.-T. (2021). Augmented reality and virtual reality displays: emerging technologies and future perspectives. *Light: Science & Applications*, 10(1):1–30.