DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

A Framework for location-based Augmented Reality Content on Mobile Devices

Paul Tolstoi

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

A Framework for location-based Augmented Reality Content on Mobile Devices

Ein Framework für standortbasierte Augmented Reality-Inhalte auf Mobilen Geräten

Author:	Paul Tolstoi
Supervisor:	Prof. Gudrun Klinker, Ph.D.
Advisor:	Dipl.-Inf. Univ. David A. Plecher, M.A.
Submission Date:	15.03.2019

I confirm that this Master's Thesis in Informatics: Games Engineering is my own work and I have documented all sources and material used.

Munich, 15.03.2019 Paul Tolstoi

Acknowledgments

First of all, I want to thank Prof. Gudrun Klinker, Ph.D. for giving me the opportunity to write my master's thesis at her chair.

Furthermore, I would like to express my gratitude to my advisor David Plecher for offering himself as my advisor, for supporting me and all the historic content he provided.

I appreciate all the participants in the user study for their feedback, support and the time they gave up for helping me.

From the bottom of my heart, I thank Timo Wagner for reading and annotating this thesis multiple times.

Abstract

There are many different approaches to use Augmented Reality for historical applications. However, these are mostly limited to small rooms or objects (museums, sculptures, etc.). Usually these applications use Augmented Reality to show only information in form of a text or virtual content. You rarely see e.g. missing parts of objects especially buildings superimposed onto the real world. This master's thesis aims to create a framework for presenting location dependant Augmented Reality content in outdoor environments. The ultimate goal is to create a generic Augmented Reality application, that is independent of environmental factors such as different light or weather conditions, and is not specifically tailored for particular building structures or traits.

Contents

1. Introduction

1.1. Motivation

There are many ways to teach a person about the history of human kind. In school every student has to learn many historical facts about different events of the past. Most of the time the only way for the students to acquire the knowledge is to read texts about what has happened. The text is accompanied with stories of a contemporary witnesses, sometimes there are some drawings or photos available; and for the most recent parts of the history there may be even video footage available.

Nevertheless, seeing a historic location with your own eyes might leave a lasting impression. Almost everyone has done some school trip to a historically relevant location in their life. More often than not those are some kind of ruins, where students learn about the past importance of the place and the period of time associated with it. The knowledge is usually conveyed either by the teacher or a guide; sometimes there are plaques with some historic photos or sketches. However, right now it is still not possible to dive into the past and see how the place looked like years or centuries ago.

Nowadays most EU citizen ([Eur19]) have a smartphone and schools are starting to have specific classes encouraging students to use a mobile device [Swe10]. So, it is only obvious to apply modern technologies to teach people historical information. Displaying virtual content in a historic location could motivate and engage people to explore and learn more about the days of the past.

1.2. Augmented reality

Augmented Reality (AR) is — as described by Azuma [Azu97] — a technology that superimposes virtual objects onto the real-world. This technology provides a contrast to Virtual Reality (VR), since VR does not allow the user to see any real-world content. He defines that a system should have three specific characteristics to allow AR to be free of specific technologies: Combination of real and virtual, interactivity in real time, and registration in three dimensions. As counterexample Azuma presents Computer-generated imagery (CGI), famously known from films, as not being interactive. Overlays, e.g. providing the score during a sports broadcast, are neither interactive nor "combined with the real-world in 3D" [Azu97]. Since the publication of Azuma's survey, the research and especially the industry managed to leap forward and created better hardware and published many different approaches and frameworks to tackle the issues presented by AR.

Nowadays most mobile devices like smartphones and tablets contain a built-in camera and inertial sensors. Additionally, they possess powerful processing units, and thus are capable of displaying AR content. An AR Software Development Kit (SDK) simplifies the implementation of AR applications by providing a simple Application Programming Interface (API) to consume. However, the latest iterations are (at least in some points) superior to the older ones. As described in my bachelor's thesis ([Tol14], [TD15]), tracking of an image target without additional stabilization fails quite easy, if the camera is moved too fast, the tracked object is obstructed or if the image is too shiny.

ARCore and ARKit AR SDKs like ARCore ([Goo19a]) and ARKit ([App19a]) are part of the latest iteration. They provide not only a simple API but more importantly hardware acceleration. These frameworks are deeply integrated into the operating system and allow development of mobile applications that are easy on the battery life while delivering a stable tracking performance.

The main principle behind both SDKs is the same. When an AR-capable application activates the tracking session, the frameworks begin to track the world around the device. The content positioned at a certain location inside the provided tracked space stays at the same location (relative to the real-world), even if the user turns or walks away. Both frameworks apply visual-inertial odometry for tracking of the environment ([App19b] [Goo19b]). This means they use both the image, provided by the camera, as well as the inertial sensor data of the device, to ensure a stable tracking even during periods of fast movement or short time occlusion (e.g. people walking by).

At the time of writing both AR SDKs require different platforms and thus are mutually exclusive:

- *ARCore* requires the devices to run Android 7.0 or later and is only supported by some specific smartphones and tablets [Goo19d]. On iOS only the Cloud Anchor feature is supported by the ARCore SDK [Goo19e].

- *ARKit* works only on devices with iOS 11.0 or later and requires an A9 chip or better [App19c]. Some features like face tracking only work with specific models (e.g. iPhone X/XR). They are, however, not needed for this master's thesis.

However, there is a way to target both platforms, without developing two different applications: The Unreal Engine ([Epi19]) as well as the Unity Engine ([Uni18]) provide official support for both AR SDKs. This allows to develop an AR application once that works on both iOS and Android devices.

Localization

To deliver location-based information an application must somehow know, where the user is located. On a smartphone there are different systems available for locating the device. The availability, precision and battery consumption vary drastically between different approaches. Systems only suitable for indoor navigation (e.g. Eddystone[1] and iBeacon[2]) do not provide a very good precision, but they are therefore very easy on the battery life. Since this master's thesis focuses on an outdoor application, these technologies are not further considered.

Global Positioning System (GPS) GPS is an outdoor location system that is available to many people. GPS uses "24 satellites, equally spaced in six orbital planes 20,200 kilometers above the Earth, that transmit [...] specially coded carrier signals" [DR01]. After receiving the signals, the device calculates the location with a precision up to 4.9 meters [DE15]. Originally it was developed and used by the US military. However, since 2000, high precision GPS is available and affordable for civilian use as well. As of today, almost every smartphone has a built-in GPS receiver and thus can be used for location-based applications e.g. every iPhone (except the iPhone 1) has a built-in receiver.

GPS, however, has some limitations:

1. It consumes a lot of power, e.g. it can reduce battery life down to 7.8 hours in contrast to 63 hours when using only GSM [Gao+08].
2. It can take up to several minutes to get the first location estimation [DR01].
3. The performance in urban environment is unstable, since it requires direct sight of at least four satellites, which cannot be assured due to e.g. interfering buildings.

[1]https://developers.google.com/beacons/
[2]https://developer.apple.com/ibeacon/

Additionally, tall buildings reflect the signals and therefore confuse the receiver [DR01] [DE15].

To reduce the latency that GPS-only localization can cause, there exists an improved iteration: Assisted GPS. A-GPS downloads the current satellite data using the data connection of a smartphone reducing the time until the first localization from minutes to seconds.

1.3. Goal of the thesis

The goal of this thesis is to create a framework that can be used for a range of location-based AR applications. Such an application should work stable in a large-scale outdoor environment and be as independent as possible of different environmental factors like various weather and light conditions. The user should be localized next to a building or monument, so the app can display AR content next to it, e.g. overlay the building with photos of how it looked like years ago. The detection and localization should not require markers, but nevertheless still work correctly with different kind of buildings, monuments or landmarks. Furthermore, the application should handle the augmentation well without requiring the objects to have certain traits like a facade, an arch or depend on availability of a 3D model.

2. Related Work

The goal of this thesis is inspired by preceding research as well as some applications discussed in this chapter. The idea is to utilize and improve some of them to create a generic framework.

2.1. Historic Methods

Historically there were some approaches that try to present the past appearance of buildings and cities to people, without being interactive or requiring the use of modern technologies (at least for the user).

2.1.1. Heidentor

The Heidentor in Carnuntum near Vienna in Austria is a triumphal arch from the 4th century. It is mostly dilapidated, however it was reconstructed multiple times throughout the history. The latest restoration was between 1998 and 2001 for which it was deeply analyzed. Today only small parts of it are standing, as only the western pillars were restored. On site, there is a glass plate installation (see Figure 2.1) that shows the outlines of how the original monument probably looked like [Jos02].

2.1.2. Cities now and then

Many photographers and historians often combine photos from different times to show people how cities around the world looked like years or centuries ago. New York (see Figure 2.2) underwent many changes, however, it was never destroyed by war. World War II (WW2) caused devastating destruction of Europe. Between September 1940 and May 1941 Nazi Germany bombed London and different other cities in the United Kingdom. The photographer Jim Dyson (from Getty Images [Eri16]) created a couple of images (see Figure 2.3) where he overlays pictures of London nowadays

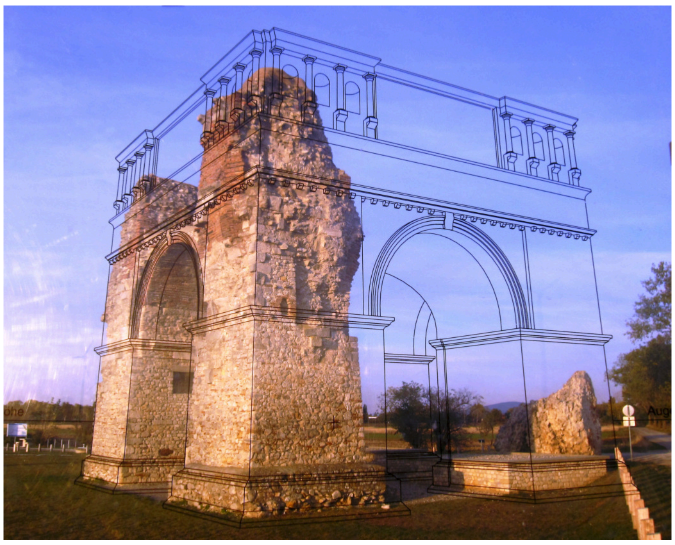


Figure 2.1.: The glass plate installation showing the outlines of the former Heidentor in Carnuntum [Wik16]

with the pictures taken during the time of the bombing. There are similar composed images from different cities, showing the destruction caused by the war: Nobody can imagine how it was to see big guns right in front of the Notre-Dame de Paris (see Figure 2.4). Just like many other cities in Germany, Munich was largely destroyed, which is nowadays hard to envision (see Figure 2.5).

2.2. Mobile Augmented Reality

Mobile AR applications use the built-in camera and sensors of a smartphone to augment the real world with virtual objects or information.

2.2.1. IKEA Place

One of the first AR apps created with Apple's new ARKit is IKEA Place. It was publicly announced during the Apple Worldwide Developers Conference (WWDC) in 2017, when ARKit was introduced for the first time [App17]. The app offers the user to browse through the catalog of IKEA and place life-size furniture in their home [IKE17]. Additionally, the user can share photos of their virtually furnished home (see Figure 2.6).

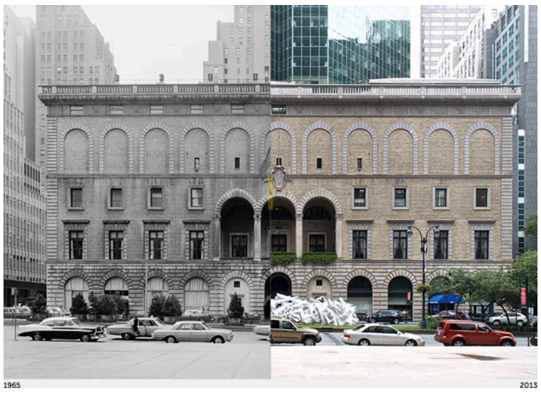


Figure 2.2.: Racquet and Tennis Club: left in 1965, right in 2013 [Zha13]

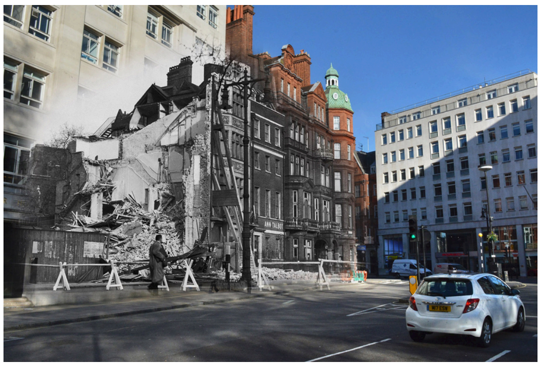


Figure 2.3.: London, destruction caused by the Nazi bomb attack; in the background London now with restored and new buildings [Eri16]

2.2.2. Cultural Heritage Presentation with Mixed Reality

This bachelor's thesis [Kön18] presents an approach, how museums can utilize AR to provide the visitors additional information on their exhibits. The information might be as simple as just additional text, but König presents several different kinds of interactions. One interaction visualizes how a statue most likely looked like in the past by augmenting its missing parts. The user can either select to display the whole statue, or to only add some of the missing parts (see Figure 2.7). For this to work, the application uses a Computer-aided design (CAD) model for tracking. Another interaction allows the user to dive into a complete small-scale (roughly 17-times smaller than the original) model of the Parthenon in Athens, Greece (see Figure 2.8) and examine the inside of the model, since it is not possible to enter and experience the interior due to its scale. Because there is no CAD model available, König uses Vuforia's Object Recognition. The user has to position the smartphone in front of the model and is then able to see through the building. Yet another way to interact, provided by the application, is the AR/VR-only presentation of the Acropolis, the hill where the Parthenon is located. The user can take a look at the ancient site, select buildings to get textual information and enter a VR mode to experience the actual proportions of the buildings.

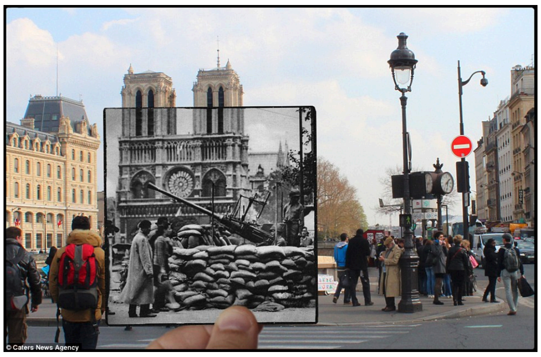


Figure 2.4.: A gun with a soldier in front of the Notre-Dame de Paris; in the background Notre-Dame today [Mic16]

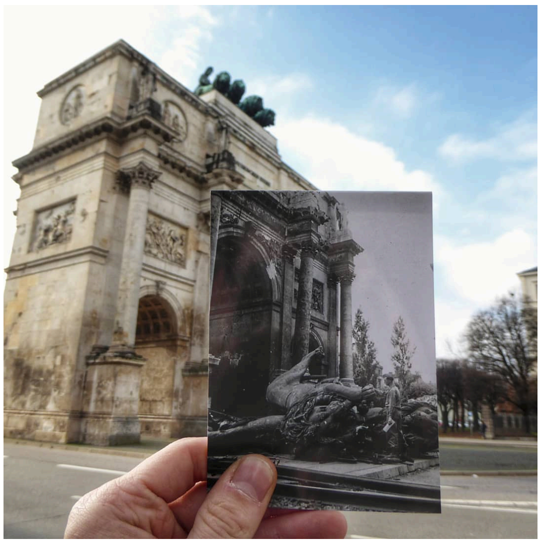


Figure 2.5.: The Siegestor in Munich, partly destructed during WW2; in background the same monument today [Sch18]

2.3. Location-based Applications

Location-based applications use the physical location of the user to present some kind of related information. The location of the user provides the context for the information.

2.3.1. Navigation Systems

Navigation Systems are probably one of the most commonly known location-based applications. They are available either as standalone devices (portable or built into e.g. a car) or as an application running on a smartphone. Independently of the device the common use case is to help the user navigate the best possible route from the user's current location to their desired destination. The system guides the user through the computed route and uses GPS to give helping directions to the driver (e.g. "in 100 meters turn left"), usually with the help of a computer generated voice. In addition, the interface presents information like estimated arrival time, upcoming traffic jams or interesting locations nearby, such as a gas station. Usually the device computes the route in advance, since way-finding on global scale on the fly is slow. Only if the

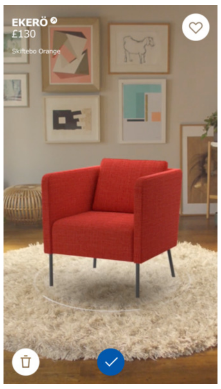
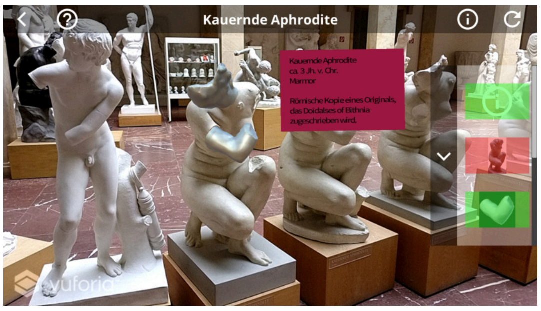


Figure 2.6.: Screenshot of IKEA Place: a red armchair is placed virtually in the living room using ARKit [IKE18]



Kauernde Aphrodite
ca. 3 Jh. v. Chr.
Marmor

Römische Kopie eines Originals, das Doidalses of Bithnia zugeschrieben wird.

Figure 2.7.: Crouching Aphrodite augmented with missing parts [Kön18]

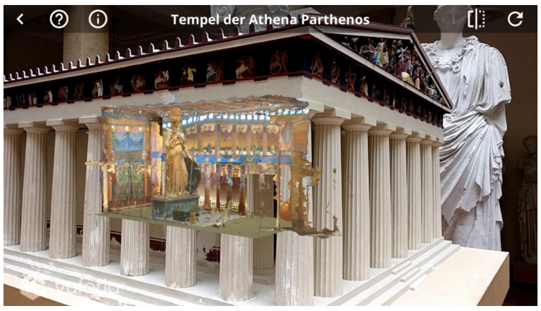


Figure 2.8.: An inside view of the Parthenon model [Kön18]

current course of the user greatly deviates from the calculated path, the system might recalculate and suggest a new route. Depending on the device or application the "best route" is computed with respect to different factors, usually defined by the user. The user might want to avoid traffic jams, tolls or specific kinds of transportation (e.g. only use subways) [Nag12].

Possible example applications are Google Maps (see Figure 2.10), available on iOS and Android, and TomTom, available as a standalone device (see Figure 2.9) or an app for iOS and Android.

Figure 2.9.: TomTom standalone navigation device [Tom19]

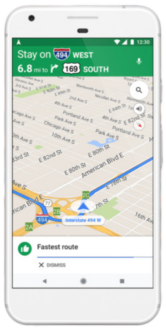


Figure 2.10.: Google Maps, a smartphone app for navigation [Goo19c]

2.3.2. Geocaching

One other location-based application is Geocaching. It is a game-like outdoor activity where people use a GPS-enabled device to find hidden caches. As described by [OHa08] a cache is usually "a small waterproof container which can be hidden anywhere in the world" (see Figure 2.11). The person hiding the cache publishes the GPS coordinates on a website for other "geocachers" to find. A cache usually contains a "log book" for the finder to write down when the cache was found. As noted by O'Hara, the main goal of the activity is not to find the cache. For the geocachers the experience is more important than the find itself. Additionally, it helps people to motivate themselves to start walking by sparking the flame of discovery and exploration of new and interesting places.

2.3.3. Ingress

A location-based game that combines geocaching and Capture the Flag (CTF) into one single game is Ingress [Dav16]. The game was developed by Niantic and has been installed more than 10 million times via the Google PlayStore [Nia19a]. The game screen shows a stylistic map surrounding the player (see Figure 2.12). The player takes on the role of an agent joining one of the two rebelling factions. The goal of the players is to capture portals and link them together to create control fields; the bigger the fields the more score the team gets. A portal in ingress is typically linked to a point of interest in the real world such as a statue, a monument or another publicly accessible place. Players have to move in the real world to come close to a portal. Then they can interact or claim a portal. Players can use a key (obtained from a claimed portal) to link two

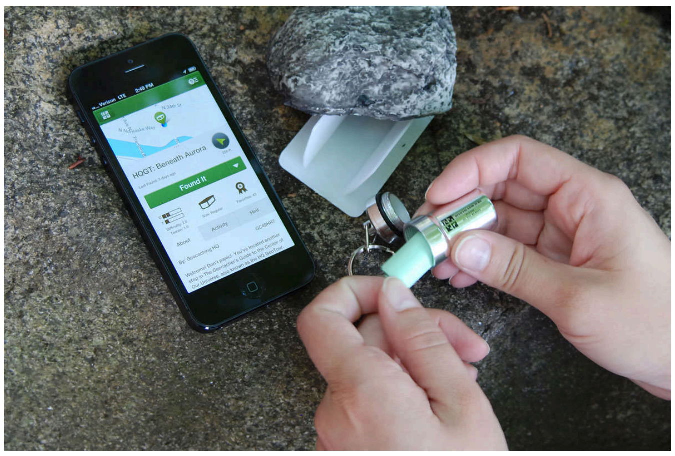


Figure 2.11.: A smartphone showing the location of the cache and person opening the cache [Geo19]

different portals. The links of portals can never cross already existing links. When three portals are linked together, they form a control field. The amount of people living within the control field is counted towards the team score.

2.4. Location-based Augmented Reality Applications

Location-based AR applications combine information based on the location provided by the device as well as display content in a way that combines the real and virtual world.

2.4.1. Wikitude World Browser

One of the first AR applications for a mobile consumer device is Wikitude World Browser [Hau18]. The app was released in 2008. The app allows the user to search for points of interest and display them using one of three different views: list, map or AR view. The AR view (see Figure 2.13) shows an overlay representing a point of interest and where it is relative to the smartphone's camera. The AR overlay does not provide any 3D object that combines the virtual and real world, so it is not strictly an AR app as defined by [Azu97]. In addition, a static overlay displays information about a selected place: the distance from the user to the location as well as some background information. Wikitude World Browser lets the user select out of a magnitude of "worlds" — data

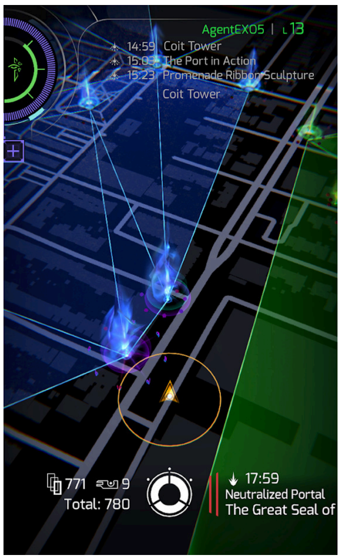
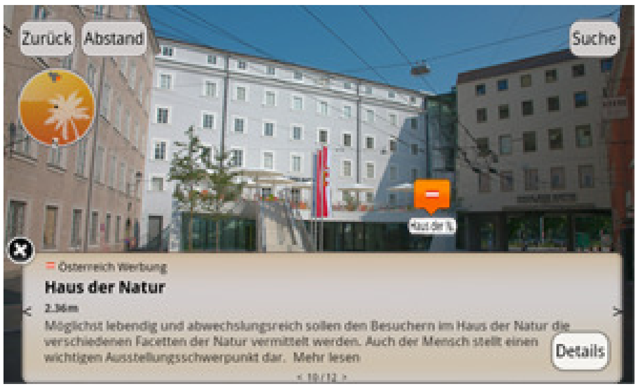


Figure 2.12.: Ingress: Main View of the game showing nearby linked portals [Nia19a]

provider with locations and information, that are either compiled by Wikitude or by third party sources.

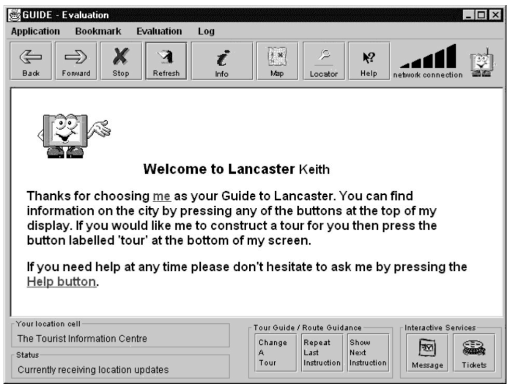


Österreich Werbung
Haus der Natur
2.36m
Möglichst lebendig und abwechslungsreich sollen den Besuchern im Haus der Natur die verschiedenen Facetten der Natur vermittelt werden. Auch der Mensch stellt einen wichtigen Ausstellungsschwerpunkt dar. Mehr lesen

Figure 2.13.: AR view of the Wikitude World Browser [Hau18]

2.4.2. City Guide on Mobile Devices

Cheverst, Davies, Mitchell, et al. present a location-aware city guide for a mobile device (Fujitsu TeamPad 7600) [Che+00]. This application is designed to support the visitors in multiple ways (see Figure 2.14). The app provides a feature enabling the user to create their own guided tour by selecting multiple attractions from one of the available categories like "Historic", "Recreational" or "Popular Attractions". When creating a tour, the guide automatically includes multiple factors like opening hours or when the location is not crowded. While walking around the city the user can ask the app about information for the nearby attractions. Additionally, a map provides the user an overview where they are, and what interesting locations are nearby. Furthermore using the guide app the user can book accommodation.



Welcome to Lancaster Keith

Thanks for choosing me as your Guide to Lancaster. You can find information on the city by pressing any of the buttons at the top of my display. If you would like me to construct a tour for you then press the button labelled 'tour' at the bottom of my screen.

If you need help at any time please don't hesitate to ask me by pressing the Help button.

Your location cell: The Tourist Information Centre

Status: Currently receiving location updates

Figure 2.14.: GUIDE: A context-aware electronic tourist guide [Che+00]

2.4.3. Pokémon GO

The probably most popular example for a location-based augmented reality application is the game Pokémon GO. Niantic developed the game and incorporated lessons learned from their game Ingress. 811 days after launch, the game generated more than $2 billion revenue ([Pau18]) and has more than 8 million monthly downloads ([Cru18]).

The player becomes a Pokémon trainer and can walk around the world, catch Pokémon, battle at Gyms or gather items at PokéStops. Gyms and PokéStops are placed at points of interest in the real world (see Figure 2.15), just like the portals in Ingress. When walking around the player encounters wild Pokémon which they can catch by throwing Poké Balls. While catching, the player can use an AR view (see Figure 2.16) to augment the Pokémon and even take photos and share them.

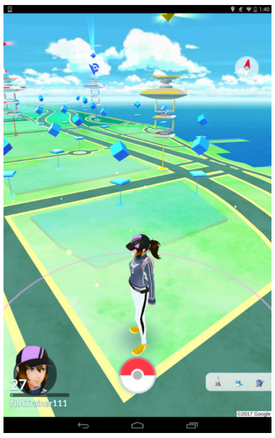


Figure 2.15.: Pokémon GO: A stylistic map view showing nearby Arenas and PokéStops [Nia19b]

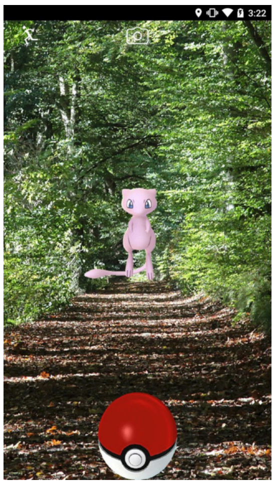


Figure 2.16.: Pokémon GO: AR view when catching a virtual Pokémon [Nia19b]

2.4.4. Augmented Reality using Building Facade

In his master's thesis [McC13], McClean developed an AR system that uses the planar facade of buildings for displaying content. To locate the user and display appropriate content the app tracks the geographic location using GPS and sends it together with the current image of the camera to the backend service. The backend processes the image and returns the corresponding content and camera pose. The mobile application uses the returned data for displaying content (see Figure 2.17).

The processing on the backend happens in multiple steps. First the facade is extracted using line segmentation, tilt rectification and homography estimation. From the extracted facade the backend then extracts features and matches them against a database. Once matched the backend can then retrieve the authored AR content and send it back to the client.

2.4.5. Timetraveler: Time of the Berlin Wall in AR

Timetraveler is an app developed by Timetraveler Augmented and Metaio for experiencing "the history of the Berlin Wall [...] like never before" [Met15]. By using GPS, the app allows the user to participate in a historical tour and stop by at one of the eleven major locations. Thanks to optical tracking, at each of the location of historic

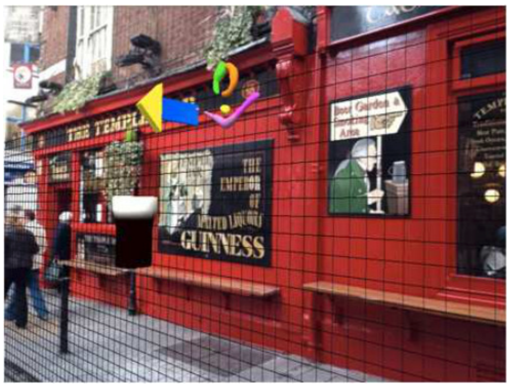


Figure 2.17.: Building Facade augmented with a grid and other content [McC13]

importance Timetraveler can take the user back in time. The user can look at historic videos (see Figure 2.18) and images and relive the time of the Berlin Wall.

After Apple bought Metaio in 2015 ([Ron15]), the app and the webpage were discontinued.

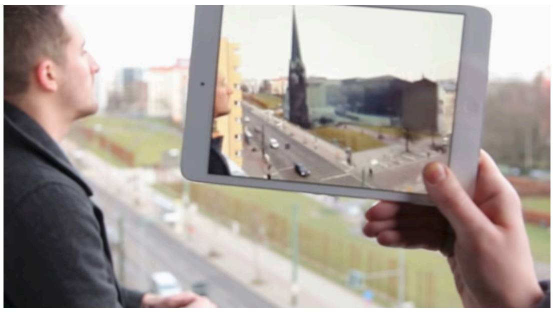


Figure 2.18.: Demolition of the Church of Reconciliation in Berlin overlaid over the former location [Met15]

2.4.6. MauAR: Experience Berlin Wall in AR

MauAR [Str18] is an app that was developed by a team of four during the Coding Da Vinci hackathon in 2017 [Opea]. The idea behind the Coding Da Vinci [Opeb] hackathon is to bring software developers and cultural institutions together to develop a prototype during a six-week phase. The prototype should use the data, information and/or objects provided by an institution in a novel engaging way.

MauAR uses a combination of ARKit and CoreLocation (framework for obtaining geographic location and orientation) to show the user where the Berlin Wall stood and how it looked like. In addition, the app shows visual content such as videos or images at their historic location. The content was provided by "Stiftung Berliner Mauer".

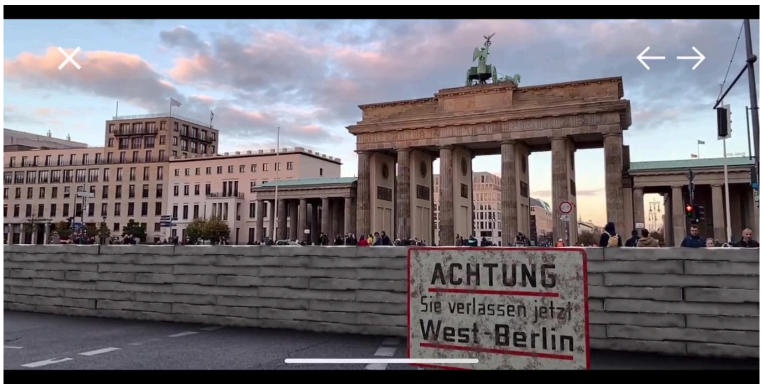


Figure 2.19.: MauAR: Berlin Wall in front of the Brandenburg Gate [Str18]

2.4.7. HistoricARGuide

In his master's thesis [Sch16], Schwarzmann uses a markerless approach to augment historic content on top of buildings. The camera image is processed in multiple steps. First it is converted to grayscale and normalized using histogram equalization to make it "illumination invariant". The grayscale image is blurred and edge detection is applied. Because feature detection would be too expensive to perform on a smartphone, Schwarzmann uses contour detection since most of the historic buildings have some sort of arcs, that can be detected and tracked. After detecting and matching the arcs a corresponding image was overlaid. In addition to an image overlay, the app can display more information about the building.

Schwarzmann points out in the "Future Work" chapter multiple improvements for the app. It is only possible to add plain images as an augmentation, neither 3D objects nor distorted images depending on the user position can be added.

2.4.8. Mobile Outdoors AR Cultural Heritage

Another approach is presented by Panou, Ragia, Dimelli, and Mania. The application is a "complete mobile tourist guide for cultural heritage sites located in the old town of

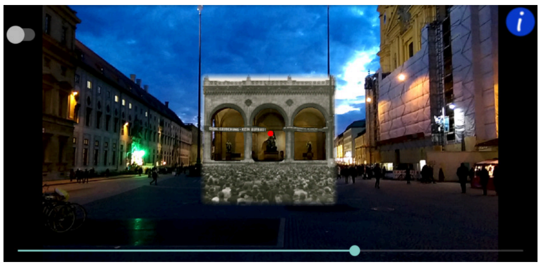


Figure 2.20.: Feldherrnhalle augmented with historic photo [Sch16]

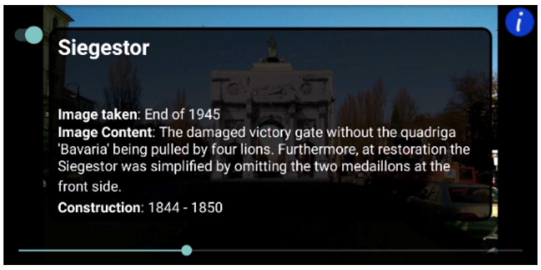

Siegestor

Image taken: End of 1945
Image Content: The damaged victory gate without the quadriga 'Bavaria' being pulled by four lions. Furthermore, at restoration the Siegestor was simplified by omitting the two medaillons at the front side.
Construction: 1844 - 1850

Figure 2.21.: Siegestor augmented with historic photo [Sch16]

Chania, Crete, Greece" [Pan+18]. The app has two main modes: map navigation and camera navigation. The map navigation shows a map of Chania with corresponding points of interest which the user can select to get more information. The camera view provides a dot for each point of interest with a small thumbnail (similar to 2.4.1). When near an AR capable location, the user can use the smartphone to take a look at the 3D model. The user can change between augmenting the whole building or only the missing parts.

For the AR view the application uses the Mobile AR Framework by HITLabNZ[1] as well as the Wikitude JavaScript API[2].

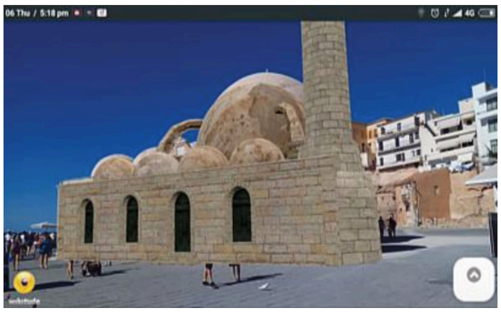


Figure 2.22.: An Architecture for Mobile Outdoors Augmented Reality for Cultural Heritage [Pan+18]

[1]https://www.hitlabnz.org/index.php/products/mobile-ar-framework/
[2]https://www.wikitude.com/products/wikitude-sdk/

2.5. Localization and Camera Pose Estimation Approaches

For displaying the AR content correctly, first the devices must be located. There are many different approaches to do that.

2.5.1. Localization using Facade Detection

There are multiple approaches to utilize facades for localizing the user. The fundamental assumption for all of them is that the man-made world is to some extend a Manhattan World ([CY99]). From this assumption follows: each image has some prominent edges that can be detected, in addition the extension of those usually meet in two or three specific points (vanishing points).

Chu, Gallagher, and Chen propose a GPS position refinement with additional camera orientation estimation [CGC14]. For that they use a single photo provided by a camera and 2D map of the nearby region. The system detects the vertical edges of buildings and matches them against outlines of nearby buildings on the 2D map.

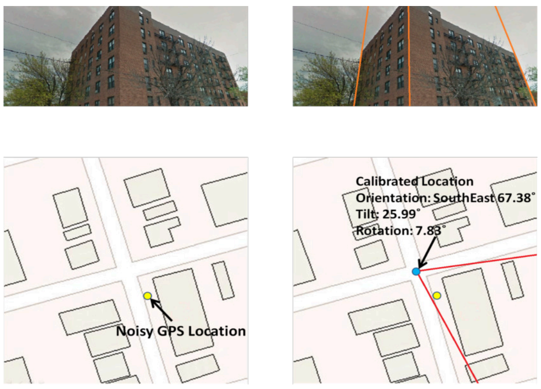


Figure 2.23.: GPS refinement and camera orientation estimation from a single image and a 2D building outline map [CGC14]

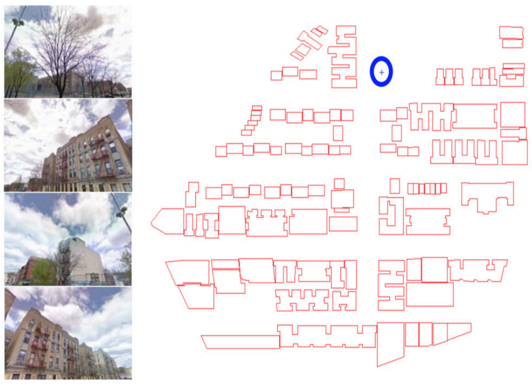


Figure 2.24.: Estimating Camera Pose from a Single Urban Ground-View Omnidirectional Image and a 2D Building Outline Map [Cha+10]

Cham, Ciptadi, Tan, et al. [Cha+10] propose a similar approach but use instead of a simple camera photo an omnidirectional image. This approach extracts the vertical corner edges of buildings as well, additionally it also extracts the normals of the walls. After the extraction the edges and corresponding walls are grouped, so that they can be matched against the 2D map of buildings. From the matches the camera position is extracted (see Figure 2.24).

Fond, Berger, and Simon do not use a 2D map for the detection and localization of the camera. Instead Fond, Berger, and Simon propose to use a Convolutional Neural Network to extract and identify a facade of a building. If the reference database is extended with geometric information, "a complete 6DOF camera pose can be estimated for each recognized facade".

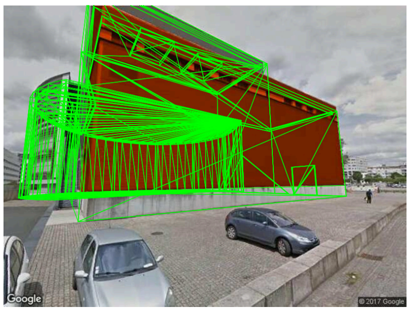


Figure 2.25.: Facade Proposals for Urban Augmented Reality: facade detection and virtual object augmentation on an image from Google StreetView [FBS17]

2.5.2. Robust Model-based Tracking for Outdoor AR

Reitmayr and Drummond present a hybrid approach for tracking and displaying of AR content in large-scale outdoor environments by incorporating multiple systems for robustness [RD06]. To guarantee a robust tracking experience the system relies on an edge-tracker that utilizes a 3D model of the buildings. In contrast to other approaches, the used 3D model is simple (just simple box-like objects) but it is textured, which provides better results, even though it has less polygonal details. The "missing" information is provided by the texture, which additionally allows for automatic filtering of edges based on the estimated pose of the camera. To minimize the instabilities created by fast movement, the system incorporates additionally the inertial and magnetometer sensors. The outdoor environment provides further obstacles such as pedestrians or cars passing by. To compensate the tracking failures, the approach uses a "point-based image matching component" to help recover from such failures.

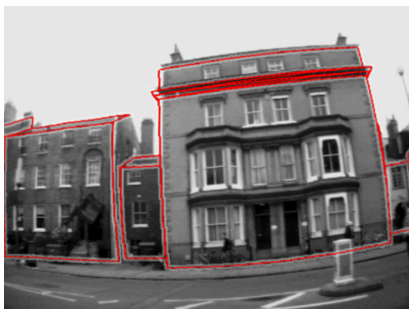


Figure 2.26.: Model-based estimated pose with overlaid building outline [RD06]

2.5.3. PoseNet: neural network for pose estimation

Kendall, Grimes, and Cipolla from the University of Cambridge use a convolutional neural network "PoseNet" to generate a "6-DoF pose relative to a scene", using only a "single 224x224 RGB image" [KGC15]. The base idea is training a neural network that processes a single image and outputs the pose of the camera. However, to train the network it is required to first obtain a set of images with the corresponding pose. To gather this data Kendall, Grimes, and Cipolla propose to first record a number of videos and then simply extract some frames as images. These images can be fed into a sparse reconstruction application (such as [SF16]). Finally, the neural network can be trained with the results of the reconstruction as the ground truth data (namely the position and rotation associated with each image in the reconstruction). The architecture of the used neural network is a modified version of the GoogLeNet [Sze+15], a convolutional network trained to classify an image into one of the 1000 categories. The PoseNet network is trained using transfer learning, that was "pretrained as a classifier, on immense image recognition datasets". PoseNet was pretrained with the Places dataset [Zho+14] that contains different indoor and outdoor scenes with a label. Kendall, Grimes, and Cipolla state the training takes about an hour on the hardware that they had available. The results have a median error of "approximately 2m and 3 degrees" for outdoor environments.

2.5.4. ColMap: Structure from Motion

ColMap is an application by Schönberger and Frahm for an Incremental Structure from Motion (SfM) pipeline ([SF16]). The application provides additional features e.g. Multi-View Stereo that are not discussed in this thesis. The result of the Incremental SfM pipeline is a sparse reconstruction of the scene with corresponding pose of each

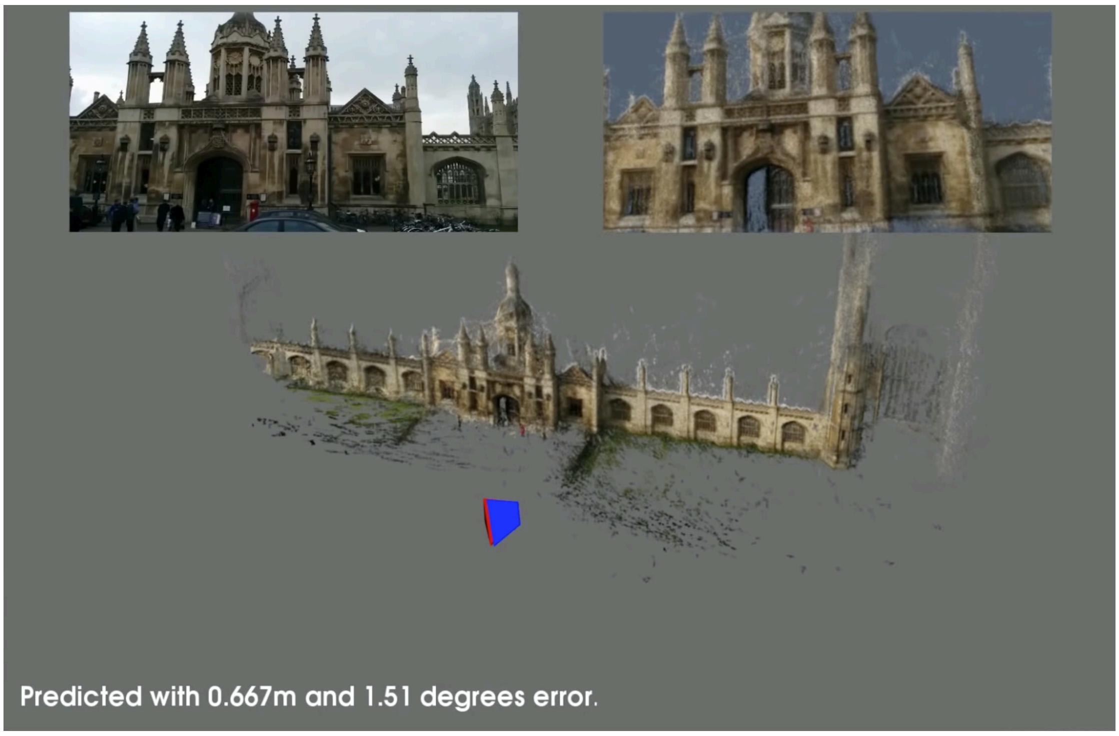


Figure 2.27.: PoseNet: The estimated pose from the input image (top left) predicted with 0.667 m and 1.51 degree error.
Top Right: rendering of the scene from the predicted position [KGC15]

camera. One of the main assumptions of SfM is that the world does not change between the images. This means that either all the images should be taken at the same time, or that there must be some compensation for objects that move between the frames (e.g. a moving car should not influence the pose estimation). Figure 2.28 shows a reconstruction of the Siegestor in Munich with only 36 images and 4341 points.

The Incremental SfM pipeline consists of two main steps.

Correspondence Search Before the system can begin the reconstruction, some correspondences between the images of the dataset have to be established. To do that, first ColMap extracts features from each of the images. For the feature detection and extraction Scale-invariant feature transform (SIFT) is used, since it was implemented working fast on the GPU. After the extraction, ColMap provides multiple strategies for matching (exhaustive, sequential, VocabTree and other). Underneath however, it uses a GPU-based SIFT matching. After the matching, each match is verified geometrically.

Incremental reconstruction Now that there is some correspondence between images, the application can begin the reconstruction. After an initial pair is matched, one by one an image is selected and registered with the reconstruction. The positions of the points are triangulated and afterwards the poses of all registered images are refined using Bundle Adjustment. To increase the stability after each Bundle Adjustment the outliers are filtered out. The reconstruction finishes when all images of the dataset are either registered or cannot be incorporated.

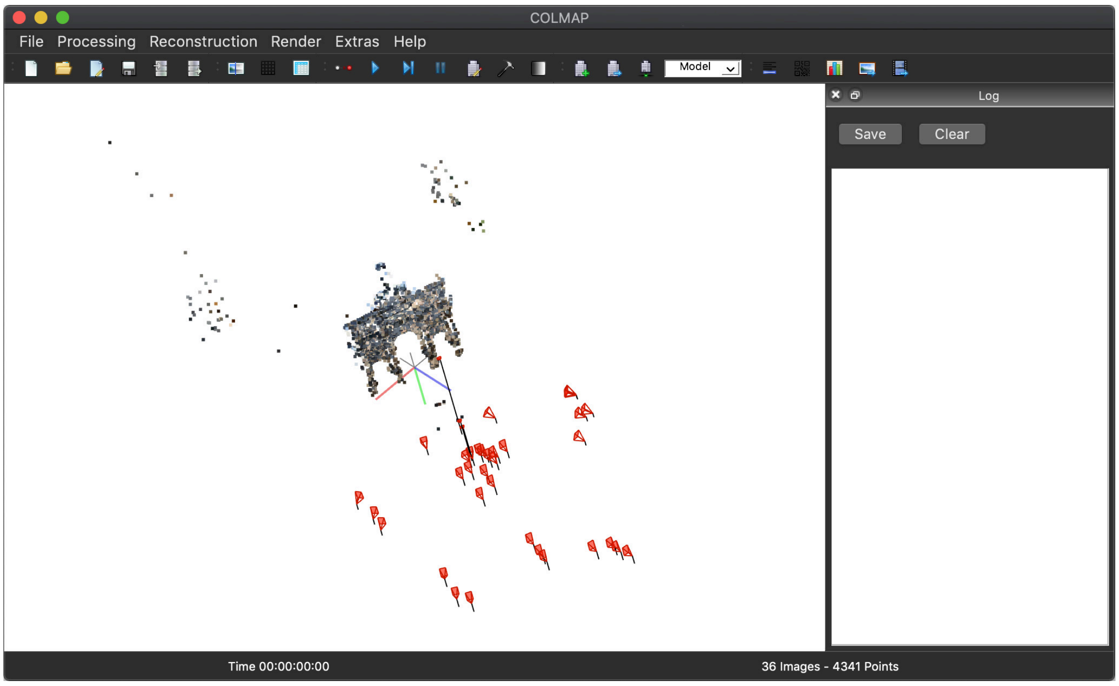


Figure 2.28.: ColMap [SF16]: Reconstruction of the Siegestor using SfM

3. Implementation

As already stated, the purpose of this master's thesis is to create a generic framework for displaying Augmented Reality (AR) content on mobile devices, that is location specific and works reliably in different lighting and weather conditions. Furthermore, it should not depend on specific traits of buildings (prominent facade, arcs, etc.) or require models of the buildings or monuments.

The master's thesis by Schwarzmann [Sch16] requires that the buildings must have at least one arc. The arc assumption is in breach of one of the main principles for this work. Same is true for the works of McClean [McC13], Chu, Gallagher, and Chen [CGC14] and Fond, Berger, and Simon [FBS17], since they need a facade for the localization and augmentation. König [Kön18] uses a CAD model for tracking and augmenting, which is simply not possible to obtain for large-scale outdoor buildings or monuments. Similarly, the work by Reitmayr and Drummond [RD06] needs a model as well, however not a detailed but a textured one. The approach by Cham, Ciptadi, Tan, et al. [Cha+10] depends upon an omnidirectional camera, which is lacking in every smartphone nowadays.

So, there are multiple problems need that need to be solved:

- Tracking of outdoor scenery proves to be difficult due to varying lighting during different times of day or weather conditions as well as due to occlusion or unrelated movement of pedestrians or cars.
- It is not possible to place markers to ease the task of tracking.
- Tracked objects should not require specific traits such as an arc or a facade since they might not always be present — a statue for example has neither.
- Obtaining or creating a model of buildings or monuments is either time-consuming or simply infeasible.

Content

As outlined in subsection 2.1.1 and subsection 2.1.2, it would be nice to be able to glance at history that happened at specific places. Overlaying old photos on top of historically relevant locations and enriching them with information is the implementational focus of this master's thesis.

There are many locations in Munich of historic significance. Many of them date back to the foundation of the city, others were erected after historically important events. One of the later is the Siegestor (Triumphal Arch). The construction was initiated in 1840 by the king of Bavaria Ludwig I after the victory over Napoleon and dedicated to the Bavarian army. During WW2 the monument was partly destroyed and not fully restored afterwards. Nowadays it serves as a reminder to peace, as the inscription on the back side says: "Dedicated to victory, destroyed by war, urging peace" (Dem Sieg geweiht, vom Krieg zerstört, zum Frieden mahnend).

For this master's thesis I focused on the Siegestor and there are many reasons for that. Neil MacGregor mentions one of them. As described in his book "Germany - Memories of a nation": history of Germany is different than the history of other countries [Mac14]. As a lead-in he uses the Siegestor as an example: In contrast to the triumphal arches in London or Paris, the one in Munich does not only celebrate the victory of a country, but additionally reminds that dedications to victories can be destroyed by wars. In conclusion he writes, that this is a good example for the role history plays in Germany, it shows a past that urges to learn from it. This is the reason why the Siegestor is such an important monument, which led me to choose the Siegestor as the focus for this master's thesis. As an important prominent monument one can easily find many historic image footages, that can be augmented onto it. Its location is another good reason, since it stands quite isolated and thus is rarely occluded by pedestrians or cars. This allows for more uninterrupted testing without disturbing people or the traffic.

3.1. General Idea

The high-level idea for the use case is quite simple: The user starts the app, points the tablet or the smartphone at a building or a monument and then takes a look at some AR content, that is positioned with respect to the object e.g. overlaying an image showing how the building looked years ago.

The core idea for the implementation is to utilize the stable tracking provided by

ARKit/ARCore. As outlined in *section 1.2 ARCore and ARKit* to be able to create a cross-platform mobile application and use the respective SDK, the easiest way would be to use either the Unreal or the Unity engine. Due to the bigger market adaptation of Unity in the mobile sector— 50% of mobile games use Unity as their engine [Uni19] — and due to preexisting experience with the engine, I selected Unity for the mobile application.

Independent of the SDK, when the world is being tracked, the application will place the AR content at the correct position with respect to the real world (e.g. replace the facade of a building with a historic photo). For that the app must know *what content* to show at which *"correct position"*.

What content to show The content to be displayed is location-dependent. A historic photo of a building, that was standing 5 kilometers away, might not make sense to be displayed. To determine the correct content the real-world location of the user must be used. The position estimation can be retrieved using GPS to prune among the different sites. Depending on the current accuracy of the GPS, it might not be clear, which of the sites is the nearest one. However, for this master's thesis, the different sites, that were tested during implementation, were sufficiently far apart from each other so the content can be unambiguously selected via GPS localization (Siegestor and Feldherrnhalle). Furthermore, during the user study only a single site was used, so the users didn't have to spend a large amount of time of the study on traveling from one location to a different one.

Where to show the content After the decision what content to show is made, the app has to determine where to position the content inside the local AR space that is provided and tracked by the SDK. The basic idea is to locate the user once, then set an anchor in the AR space and take advantage of the tracking. To compensate for some drifting, the application should try to relocate the user from time to time.

For this master's thesis I tried to utilize two different approaches: First I tried to integrate the method provided by [KGC15] (see subsection 2.5.3 PoseNet: neural network for pose estimation). And after the method proved not suitable, I extended the approach by [SF16] (see subsection 2.5.4 ColMap: Structure from Motion) for usage in AR.

3.2. Neural Network for Pose Estimation

As already described in *subsection 2.5.3*, PoseNet provides a possible solution to the problem of the estimation of the camera pose inside a certain space. For the evaluation of the method I used a Keras based implementation of Posenet by Kent Sommer[1]. I chose a Keras implementation, because Apple provides a simple tool[2] for conversion from Keras to CoreML[3], which can be executed directly on an iOS device.

I attempted the training multiple times, since slight changes in lighting and weather conditions impacted the localization negatively. The images used as the dataset were extracted from a number of videos of the northern side of the Siegestor, Munich. They were taken with a smartphone, filming from different angles. The training dataset was incrementally enlarged:

1. *500* images taken during late afternoon (Figure 3.1 image 1).
2. *206* images taken during midday (Figure 3.1 image 2).
3. *176* images taken during late evening (Figure 3.1 image 3).
4. *367* images taken during night (Figure 3.1 image 4).
5. *149* images taken during rain at midday (Figure 3.1 image 5).

Hence, the final dataset contains a total of *1398* images. I filmed the videos from different angles, pointing towards the monument (see Figure 3.2). The datasets were split 80%/20% for training and validation respectively.

After covering almost all weather and lighting conditions, I tried to integrate the PoseNet results into an AR app. However, during the integration I found out, that the network produces different pose depending on the zoom of the camera (the dataset images were taken from videos with a slight zoom for stabilization; the AR view uses not zoomed camera output). If a camera with a different zoom (e.g. a different smartphone or tablet) is used the estimation gets really unstable (see Figure 3.3).

At this point I decided to abandon this approach and try a different way to localize the user.

[1]https://github.com/kentsommer/keras-posenet

[2]https://developer.apple.com/documentation/coreml/converting_trained_models_to_core_ml/

[3]Hardware-accelerated framework for executing neural networks: https://developer.apple.com/machine-learning/

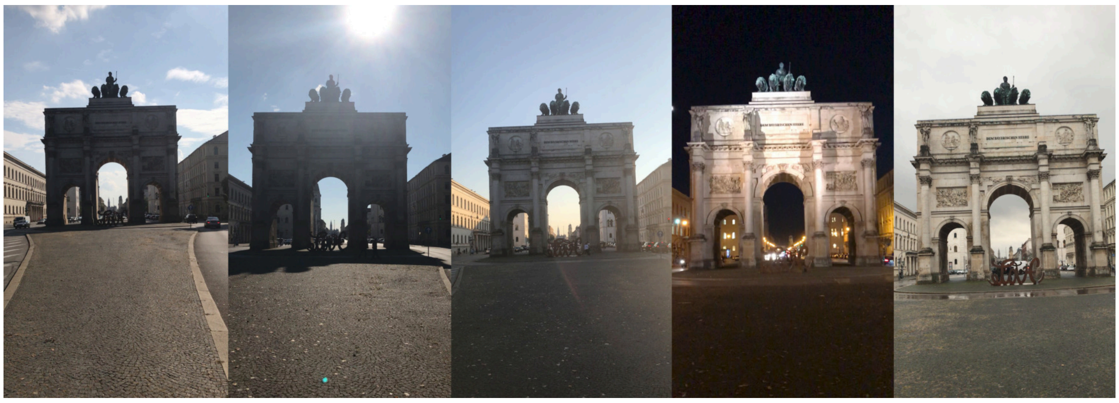


Figure 3.1.: Examples for the different parts of the dataset
different images of the Siegestor, Munich; from left to right: late afternoon, midday, late evening, at night, midday during rain

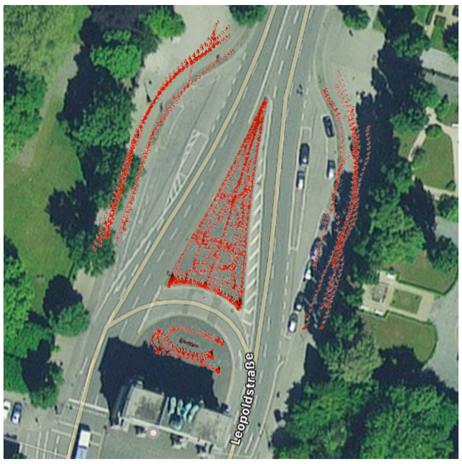


Figure 3.2.: Location of the dataset overlaid on top of a map (Satellite image taken from Apple Maps)

Conclusion

The model of PoseNet is about 50 megabyte in size, which should be small enough to be able to deploy it on a mobile device. During tests using CoreML on an iPhone 7 Plus, the device was able to execute the neural network in less than 1 second. As stated by Kendall, Grimes, and Cipolla, right now it is not known what the upper limit for the neural network is (with respect to the maximal "physical area that it can learn to localize within" [KGC15]). Hence, a single PoseNet model might be able to contain

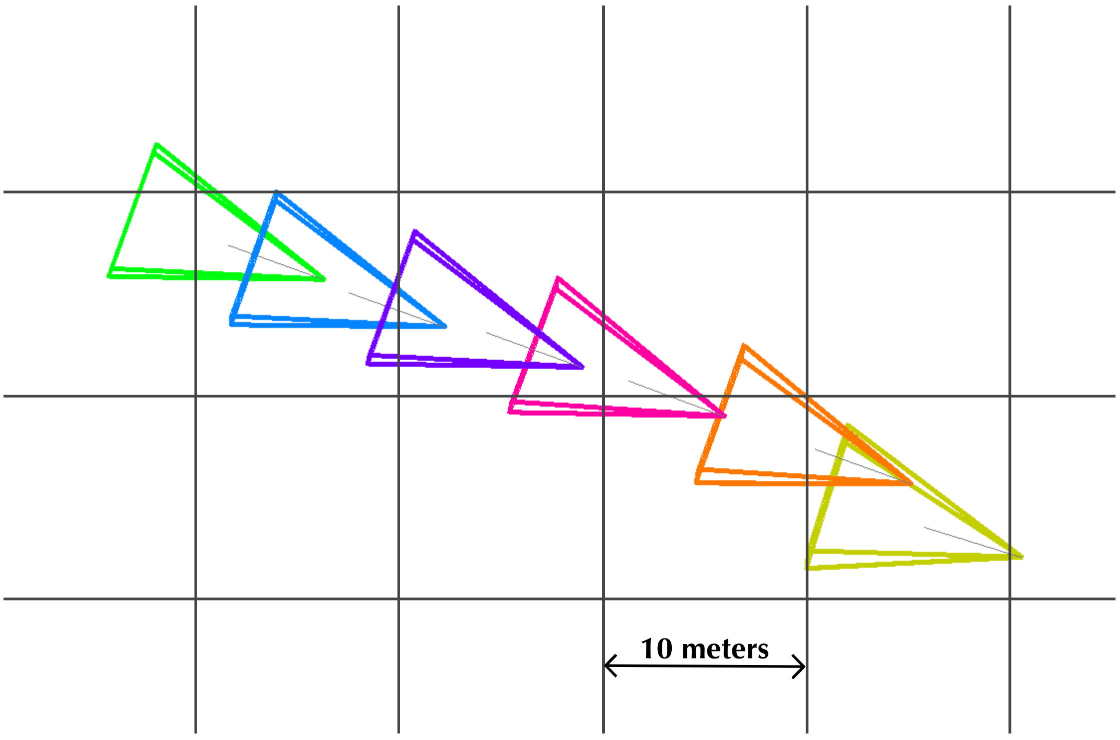


Figure 3.3.: PoseNet: Predictions are unstable depending on camera zoom
from left to right: original (green), 90% zoom, 80% zoom, 70% zoom, 60% zoom, 50% zoom
grid unit is 10 meters

multiple locations, which should affect the storage and data usage on a mobile device in a positive way. However, there are a couple of issues with this approach, that led me to stop pursuing it.

Prone to changing outdoor conditions First and foremost the pose estimation is not reliable enough for an AR application. It is not camera independent since a simple change in focal length (zoom) results in a different pose. Furthermore, it is very susceptible to different lighting conditions such as night vs midday or rainy weather vs late evening.

Time-consuming preparation Secondly, the preparation steps required before Pose-Net can be used for estimation are too time-consuming. You have to create multiple videos (or image sets) during different times of day and during different weather conditions. The more images you have, the more time the sparse reconstruction requires (the feature detection, matching and the reconstruction on my computer takes about 4 hours for 1398 images). And finally, the training of the neural network needs additional time (on my computer the training finishes in about 8 hours; see section A.1).

Outputs a pose for nonsense input Thirdly, the neural network requires some additional tweaking. PoseNet is trained only on images that have a valid pose and show a meaningful image (e.g. of the monument or its surroundings). Feeding images of the Eiffel Tower to a model that was trained on images of the Siegestor, still results in a pose output, which simply does not make any sense. This can be prevented e.g. by only executing the neural network after first localizing the user via GPS. An image of only the sky or the ground near the Siegestor, however, still outputs a pose.

3.3. Pose Estimation As A Service

The main idea behind this approach is to use an SfM application (such as ColMap [SF16]) for estimation of the camera pose. As described in subsection 2.5.4 ColMap: Structure from Motion, the application must first detect features, then match them against other images and finally compute the pose of the images. Since an SfM pipeline is quite computationally expensive, it makes no sense to run it on a mobile device. The result would be high battery consumption and it would require too much time to run. Additionally, parts of the pipeline that are used for stabilization, like Bundle Adjustment, may take minutes to run, even on a powerful computer. However, if one strips away those expensive steps, at the cost of stability and precision, it should be possible to run the application on a powerful computer which offers a remote service to circumvent the shortcomings of a mobile device. The mobile application has to only send an image to the backend and use the returned pose within the AR application.

I decided to extend ColMap with a webservice, since it is open source and supports GPU acceleration for feature detection and matching.

The webservice combines multiple projects in a single workspace. A workspace assigns each project a name, a GPS location and a reconstruction. This compound is called *site*. A project is provided already by ColMap and contains the images, their features and properties. The sparse reconstruction consists of image poses, a point cloud and camera parameters for each image. The project as well as the reconstruction were provided by ColMap and did not need any extensions.

Process

For pose estimation the application exposes an HTTP endpoint which expects an image as the payload as well as either a name or a GPS coordinate (latitude and longitude) as parameters. The processing happens in multiple steps:

1. **Site Finding** If a name of a site is passed, the application searches for the site by name. Otherwise it selects the nearest site to the GPS location passed via the parameters.
2. **Pose Estimation** When the application knows which site is the closest, it starts the pose estimation. Just like during the Incremental SfM (see subsection 2.5.4 ColMap: Structure from Motion), first the backend extracts features from the image. Afterwards the image is matched "smartly" against the images that were used for the reconstruction. Finally, the pose is estimated and refined from the matched images. The refined pose is then sent back.

"Smart" Matching During the startup of the webservice the application creates a matching queue for each of the sites. First, all the images are sorted by their third most match. This favors images that have high matches with at least three images. One by one the images are added to the matching queue. 10% of the images that match with the just inserted image (sorted by amount of matches) are remembered as neighbor images and added to a to-be-skipped list. This insertion is applied to all remaining images in the sorted list and that are not in the to-be-skipped list. In the end all skipped images are added to the end of the matching queue.

When a new image should be matched, the matching queue is taken as basis. The images are taken out of the queue, one by one, and matched against the image from the mobile application. If there are enough matches, all the remembered neighbor images are prepended to the queue. This ensures that the application favors images that have many matches. The matching only happens once for each image, and is canceled after a certain timeout, so the application does not take too long if there are no or not enough matches.

Creating a site

An important question is how to create a site. This task has some different constraints when we use the sparse reconstruction for pose estimation instead of recovering some geometry. Usually the more images are used the better the reconstruction gets. However, since the application exploits SfM for pose estimation, having a high number of images increases the computation time. But, a small amount of images results in worse pose estimation. So, you need enough images to cover the location/monument from different perspectives, but not too many since it would simply take too long to match. I found out that a good balance is to have more images near the object to capture more details, and only a couple of images from far away (see Figure 2.28). Including images from different times of day is recommended. Additionally, trying to not include objects in images, that are not part of the scene (e.g. parking cars or people walking by), since they might hide parts of the monument. After ColMap completes the sparse reconstruction it is really important to scale it, so the reconstruction has the same size as in the real world, otherwise the AR content will be positioned wrongly and appear somewhere floating around. For easier content creation the reconstruction should be rotated to match the vertical axis.

Conclusion

Depending on the amount of images and the size of the images, saving a site requires 300-500 megabyte disk space. Since the reconstructions used in this thesis consists of a small number of images (~100), ColMap's SfM requires less than 2 hours on my computer (see section A.1). The time between beginning of the pose estimation and the incorporation of the pose into the AR application took usually between two and three seconds.

For the mobile app the main issue is the time between the start of the application and the first pose estimation. The app must somehow distract the user from the waiting time, which can vary depending on multiple factors:

Precision As already stated, the pose estimation alone requires a big amount of time. Successful feature matching increases the precision of the pose, at the cost of computation time. Additionally, it is not feasible to run Bundle Adjustment for improvement of the pose estimation, since it would simple take too long to compute.

Network latency In addition to the computation on the backend, one should not neglect the network latency. Depending on the bandwidth and latency available during the upload, round-trip time might increase by a couple of seconds. Depending on the location and time of day, the mobile data network might operate at (almost) full capacity, because many people use their mobile devices at the same time, so it may take longer until the image is sent to the backend.

3.4. Mobile Application

The mobile application is an Unity[4] application for mobile devices. It uses AR Foundation[5], a framework that simplifies the usage of ARKit and ARCore within Unity. For pose estimation the application sends a single frame to the backend while simultaneously remembering the pose. When the backend returns the camera rotation and position, the application instantiates the corresponding Prefab relative to the original location of the camera. This ensures that the user does not have to hold the camera perfectly still during the estimation. Right now, the user gets a fake progress bar (see Figure 3.6) that completes after 5 seconds (usually the estimation is done within 3-4 seconds), which completes fast once the pose estimation is received.

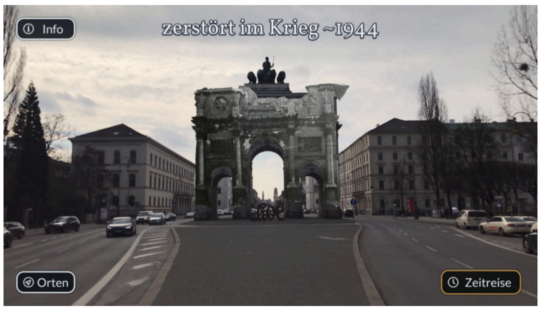


Figure 3.4.: AR view of the destroyed Siegestor (Front)

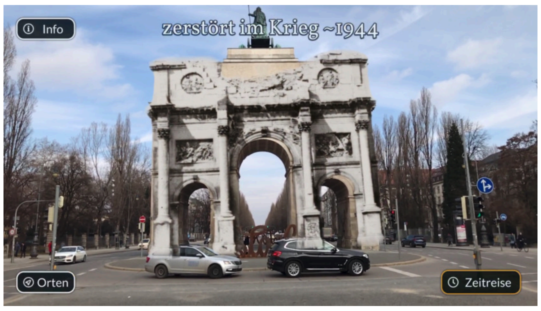


Figure 3.5.: AR view of the destroyed Siegestor (Back)

The application provides a "time travel" button allowing users to switch between different augmentations, e.g. images of the Siegestor before and after the WW2. The application was tested on multiple devices (see section A.1).

[4]https://unity3d.com/
[5]https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@1.0/manual/index.html

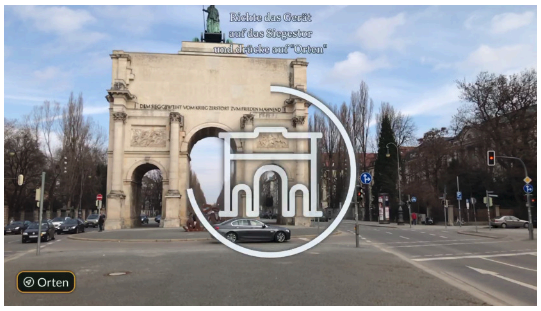


Figure 3.6.: Fake Progress Bar distracting the user from the pose estimation

Content Authoring

The AR content is authored within Unity and resides only inside the mobile application. For the content authoring, I created a helper tool in Unity, that connects to the backend and pulls the sites that are contained within a workspace. The backend exposes additionally the point cloud of the reconstruction for each site. The Unity helper tool shows the point cloud (see Figure 3.7) during the editing of the Prefab, which contains all the content for a certain site, so the user can position and scale the content appropriately. When the content author is done with editing the Prefabs, the application has to be deployed to the mobile device.

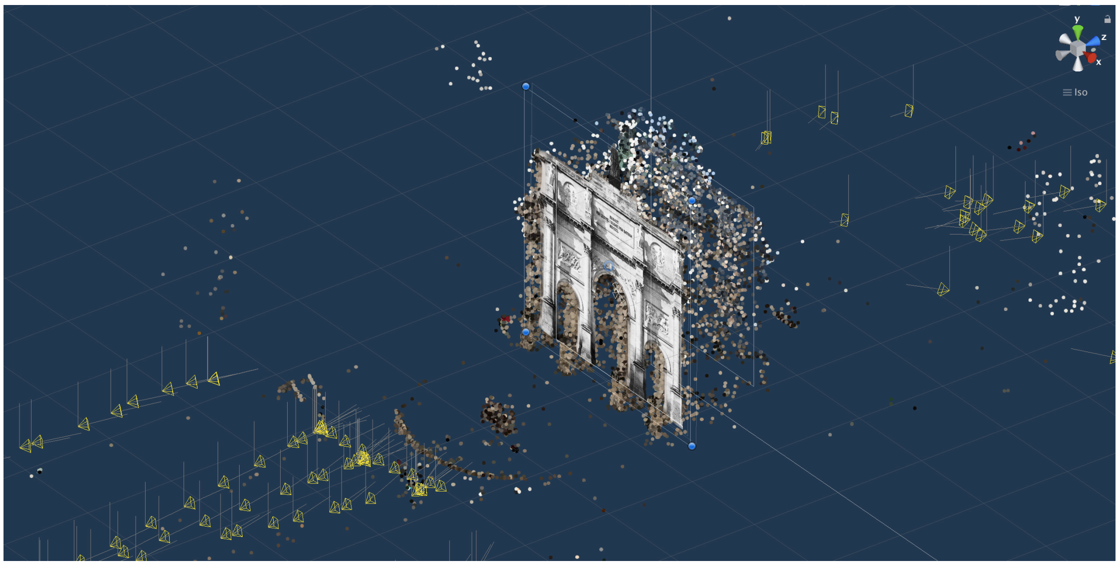


Figure 3.7.: Content Authoring: Point Cloud for better orientation

Conclusion

Technically the mobile application using ARCore/ARKit in combination with the remote backend for pose estimation works quite well. The upload and estimation take usually 3-4 seconds and return a good enough estimation for the AR content to feel at the right place. Sometimes, however, the scene drifts a little bit after correct pose estimation (see Figure 3.8). Presumably ARKit/ARCore starts to compensate for moving cars/pedestrians. It happens more frequently if the camera is pointed to the street while many cars pass by (e.g. during a green traffic light). This could partially be fixed by requesting a new estimation from time to time.

Since the AR content is only overlaid, the historic images occlude cars or walking by pedestrians. However, the overall performance and stability of the mobile app is quite promising.

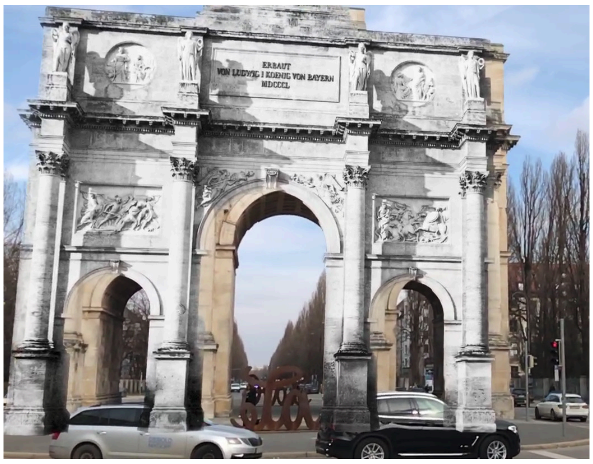


Figure 3.8.: Siegestor overlay: Tracking starts to drift likely due to cars driving by

4. Evaluation

4.1. Overview

Procedure

Before the practical part of the study takes place, the users are asked to answer a questionnaire with some general questions (see B.1 Pre-Study Questionnaire). The practical part of the user study starts at the northern part of the Siegestor. First the users are asked to start the app and follow the instructions on screen. When handing over the smartphone, the supervisor asks the users to think aloud, and records their comments. After the start, the users need to initiate the tracking session by moving the device sidewise. When the session is initialized, the test person is prompted to point the device at the monument and then to press the "Locate" button. After a successful localization the supervisor asks to try out the application by walking around and to interact with the interface. After the user uses the timetravel function, looks at the information text and interacts with the medallions the study is concluded.

Due to a wrong matching during the localization in the study of the first two users, the backend located the user not on the northern side but rather on the southern side. For this reason, I modified the app to ensure the same conditions for all participants. Thus, the study was conducted at the north side only, and the application was restricted to locate and display only the AR content intended for the southern part of the monument.

After the study, the users are asked to answer a System Usability Scale (SUS) survey (see B.2 Post-Study Questionnaire, [Bro+96]). The eighth question was changed according to [Fin06] so non-native English speaker do not distort the survey.

Concluding there is a short interview with the following questions:

- How did you like the AR interaction?
- What parts did you dislike or did confuse you?

- For the future: Are there any monuments/buildings where you could image this app could be used?
- Do you have any other comments?

4.2. Results and User Feedback

In total 7 persons participated in the user study, 2 identified as female and 5 as male. They stated an age of 26 to 29 except for one user who stated being between 30 and 39. The highest degree or level of school of the participants was Master/Diploma or similar; except for 1 person with a bachelor's degree. The main discipline of 4 interviewees was Computer Science/Informatics; 2 Biology/Chemistry and 1 Printing and Media (Druck und Medien). All participants used a smartphone on a daily basis. One person used AR frequently, the remaining stated they used it a couple times.

The user studies were conducted during different times of day: one at midday (12:30), three in the afternoon (15:00), one in the evening (17:30) and two in the night (22:00).

During the study all users had to use the locate functionality to relocalize the monument, so the AR content appeared back at the correct location. Only two people had to be hinted that they can tap the medallions, all other discovered the function themselves. Most users tried to zoom using the pinch gesture so they could read the inscriptions or see the medallions better.

Feedback

The results of the SUS survey can be seen in Figure 4.1 and Figure 4.2. The score ranges between 77.5 and 95. The total median score of the survey is 82.5 (the average is 85.4), which corresponds to a grade B as proposed by Bangor, Kortum, and Miller [BKM09].

During the interviews most users expressed that they like the AR content and that it is easy to use. They liked the interaction and the amount of text. However, most of the time they criticized the drifting content. Following statements were said by respectively one proband:

- It could be a nice app for tourists.
- It is an AR app that makes sense.

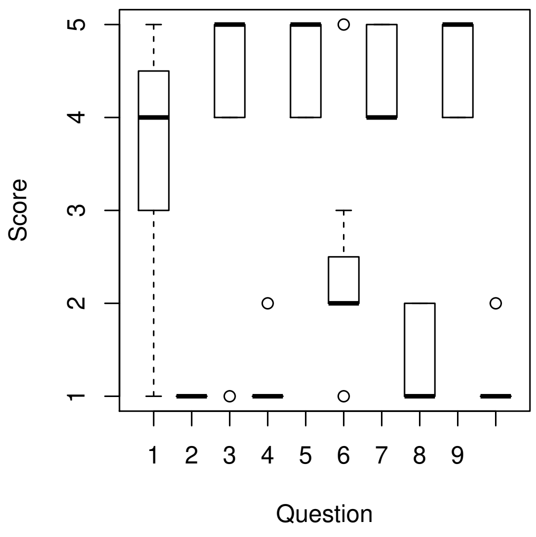


Figure 4.1.: Results of the SUS survey, itemized by question section B.2

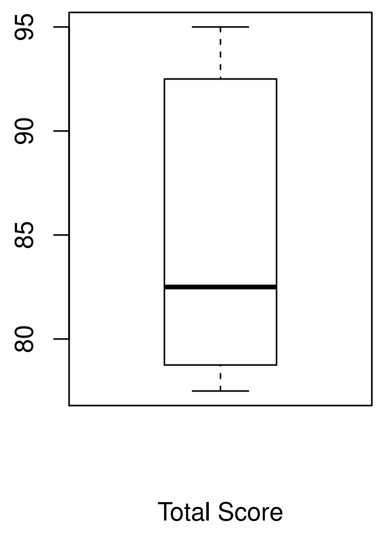


Figure 4.2.: Total score of the SUS survey

- A photo feature could probably engage young people to use the app.
- It would be nice to see the text in AR, floating next to the monument.
- More content was expected.
- 3D content was expected.
- You still see the remaining parts of the Siegestor, when you travel back in time. It would be nice to hide current parts (e.g. the quadriga).
- A slider or list for selecting the different points in time instead of just a single button would be nice.

The users suggested different use cases for the future extension:

- Other (historical) buildings and locations in Munich: Feldherrnhalle, the Town Hall with its Clock chimes, Viktualienmarkt.
- Augmentation of a whole street not only a single building.
- Buildings in different cities: Dresden Frauenkirche, World Trade Center.

- On the Autobahn, there are sometimes brown signs announcing a culturally important monument or location nearby.
- Augmentations from different vista points showing the location and (former) buildings or monument.
- A location that changed much throughout the time: first the forest/grassland with animals; then some settlements of native people; later the urbanization and industrialization.

4.3. Interpretation

The median and average SUS is quite high, indicating that the app is intuitive to use. There are two reason to explain this fact. First the app is quite simple: It only offers three different buttons for the interaction as well as only one more complex interaction: taping the medallions for more information. Additionally, since all participants use a smartphone on a daily basis and already have at least some experience with AR, the high acceptance was not a surprise. The wish for more as well as 3D content supports the fact that the app is rather simple, the users, however, liked it and would like to see more of it.

The users criticized the sometimes failing tracking which resulted in a slow drifting of the AR content. Because both framework (ARCore and ARKit) are designed with indoor environments in mind, passing by cars and pedestrians and the more difficult lighting (low light in the night, glare caused by direct sun light) often induce instability and thus cause drifting.

5. Future Work

As outlined in chapter 3, I developed a complete end-to-end scenario for this thesis, starting with pose estimation, adding content and concluding with a usable mobile application. However, there are still many issues that can be improved and developed even further.

5.1. Backend Service

For the backend arise multiple issues that were not dealt with during the development of this master's thesis. Most of them were simply out of scope for this work, others were ignored to allow creation of a working prototype.

- A big issue, which needs addressing before the app can be offered publicly or even commercially, is of a legal origin: The SIFT feature detection and matching, that is used by the backend, is patented by David Lowe and the University of British Columbia[Low04]. One possible work around would be to exchange SIFT with a different free feature detector. Since the publication of SIFT a big amount of different detectors was developed by the community, some of them might even be faster.
- Another major issue is currently the blocking of the OpenGL context, which means, only one feature detection and matching can happen at a given time, preventing multiple users from simultaneously performing a pose estimation. To be able to offer the app to the public, it should be able to scale appropriately.
- Because we exploit SfM for pose estimation only, it might make sense to use a different approach (or at least a different feature detection) that accounts for things like pedestrians or cars passing by.
- The detection and matching could be optimized e.g. by improving the matching or by using different technologies like CUDA support on the backend.

- As already mentioned in section 3.1, the app uses GPS for selection of the nearest site. If there are several sites nearby, the app might select the wrong site due to GPS inaccuracy and thus the backend won't be able to estimate any pose.
- The images, that are uploaded to the backend, are discarded after the pose estimation. Meaningfully incorporating them could increase the stability e.g. if the environment changes slowly over time.
- Even though I stopped pursuing the machine learning approach, it still promises a solution that might not require a backend at all, should one fix the stability issues that were discussed before.

5.2. Mobile Application

Not only the backend but the mobile application has many points as well that might be addressed during future work.

- Battery life is really important for mobile devices. If the app drains the battery too much, nobody will be willing to use it. At the moment the GPS localization is used at maximal accuracy available.
- An additional aspect to consider is the data usage. Currently the upload is limited to 1MB, but during the testing the images were only around 350KB.
- The users criticized the drifting that occured sometimes during the user study. Automatically relocating could create a feasable workaround, or other tracking technologies could improve the stability.
- Right now, the user has no way of knowing, where a site is located and if there is any AR content available. The app could offer the user for example a map, which marks the special locations where content is available. Location-based notifications could be used to notify the user, that a site to visit is nearby.
- Users might be interested in a guided tour around the city or around an excavation. The app could guide the user through an exhibition and show the different points of interest.
- The application was not developed with indoor environment in mind. For a possible usage e.g. within a museum, the app must address the issue of missing or imprecise GPS localization. The pose estimation on the backend does not require a GPS position per se, each site can be addressed by a name as well.

- Many users wanted to zoom in to take a better look at the medallions or to read the inscriptions. For that they tried to use a pinch gesture instead of taping on the medallions.

5.3. Content Creation

Even though the content lives only on the mobile device, it is worth dedicating its own section. Currently the content creation is quite tedious. It requires numerous steps, long waiting times and multiple applications. And in the end, the app needs a redeployment so the content can be used on a mobile device. There are many different ways to improve the pipeline:

- A single application, which handles everything, would make it easier, since one has to master only one tool. For example, by exposing an appropriate interface at the backend, all the work could be done inside Unity only.
- Another approach would be to create or extend the mobile app, to enable the user to simultaneously capture the environment and meanwhile create the reconstruction on the backend or utilize the pose that is provided by ARKit/ARCore to create the reconstruction faster.
- A workflow inspired by the Microsoft HoloLens[1], where the user can walk around and place AR content, would allow even a non-technical person to create content.
- When adding or updating sites, the app must be redeployed (e.g. manually by connecting a mobile device to the computer or via an update over the AppStore or Google Play). A possible solution would be to deliver the content dynamically and directly to the app (e.g. by using Unity's Addressable Asset System[2]).
- For the user study of this thesis only one location was used. It could be extended with additional locations and points of interest. During the interview, the users suggested several interesting as well as challenging locations.

[1]https://www.microsoft.com/en-us/hololens
[2]https://docs.unity3d.com/Packages/com.unity.addressables@0.4/manual/index.html

6. Conclusion

Nowadays most people in the western world have a smartphone capable of Augmented Reality (AR). To take advantage of this fact, the goal of this thesis was to create a framework that allows to show location-based AR content in an outdoor environment. The tracking and augmentation should withstand the difficult lighting und weather conditions caused by the outdoor setting.

I investigated two different approaches to create a prototype. The first attempt was to utilize a convolutional deep neural network that accepts a single image and outputs a camera pose estimation. However, this method was not reliable enough, since it was susceptible to different weather or lighting conditions as well as to different cameras. The second attempt was to use Structure from Motion (SfM) for camera pose estimation. Given that it involves very heavy computation, it could not be done on the mobile device alone. For this reason, I extended ColMap — an application that uses SfM — with a webservice, that the mobile app can employ to localize the user. This approach was then tested in a small user study.

As the testing during implementation and the user study showed, the overall approach seems quite promising. The probands liked the application and thought that it was easy to use. Furthermore, some of them wanted to see even more content. However, there are still some teething problems to be fixed. The localization provided by ColMap is relatively stable, yet the tracking provided by ARCore/ARKit sometimes becomes unstable resulting in displaced AR content. Additionally, the backend is limited only to a single concurrent pose estimation preventing the application from proper simultaneous multi-user support.

An open question remains, if such an AR application can teach and motivate people to learn more about history.

Appendices

A. Tools and Device Specifications

A.1. Device Specifications

Desktop Computer (Backend Server)

- **CPU** Intel© Core™ i7-3770
- **GPU** GeForce GTX 670
- **SSD** Crucial MX300 (512GB)
- **RAM** 24 GB DDR3-1600
- **OS** Windows 10 64 bit 1803 `17134.590`

Mobile Devices

- iPhone 7 Plus: iOS 12.1
- iPhone 8: iOS 12.1
- Samsung Galaxy Tab S4: Android 8.1.0 / ARCore `1.7.190128066`

A.2. Used Tools and Libraries

- Unity3D 2018.3.0f2
  - AR Foundation (`com.unity.xr.arfoundation 1.0.0-preview.20`)

    `https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@1.0/manual/index.html`

  - ARCore XR Plugin (`com.unity.xr.arcore 1.0.0-preview.23`)

    `https://docs.unity3d.com/Packages/com.unity.xr.arcore@1.0/manual/index.html`

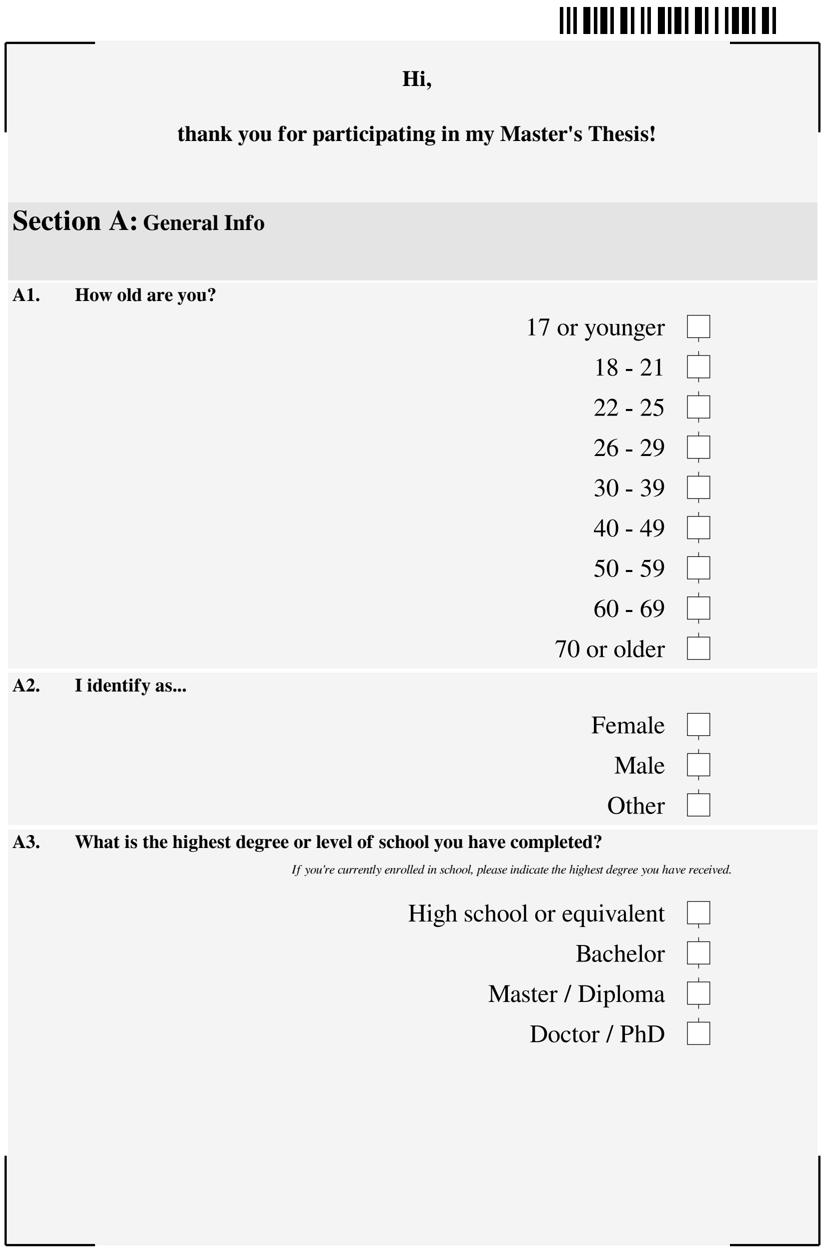
  - ARKit XR Plugin (`com.unity.xr.arkit 1.0.0-preview.17`)

    `https://docs.unity3d.com/Packages/com.unity.xr.arkit@1.0/manual/index.html`

- Visual Studio 2017
- Visual Studio Code
- Xcode 10.1
- GIT

B. Questionnaire

B.1. Pre-Study Questionnaire



Hi,

thank you for participating in my Master's Thesis!

Section A: General Info

A1. How old are you?

17 or younger	☐
18 - 21	☐
22 - 25	☐
26 - 29	☐
30 - 39	☐
40 - 49	☐
50 - 59	☐
60 - 69	☐
70 or older	☐

A2. I identify as...

Female	☐
Male	☐
Other	☐

A3. What is the highest degree or level of school you have completed?

If you're currently enrolled in school, please indicate the highest degree you have received.

High school or equivalent	☐
Bachelor	☐
Master / Diploma	☐
Doctor / PhD	☐

A4. **What is your main discipline?**

If you are a student, please select your current subject. If you are employed, please select your working area.

Informatics ☐
History ☐
Touristic ☐
Other ☐

Other

A5. **How often do you use a smartphone**

Every day ☐
Weekly ☐
Monthly ☐
Never ☐

A6. **How familiar are you with Augmented Reality (AR)?**

Never heard of ☐
Heard of, but never used ☐
Used a couple times ☐
Use frequently ☐
Use almost daily ☐
I work with AR ☐

Thank you very much! You really saved me and my Master's Thesis!

B.2. Post-Study Questionnaire

SUS Survey

For each of the following questions:

- Keep in mind the system you just used.
- Reflect your immediate response to each statement.
- Don't think too long on each one.
- Make sure you respond to each statement.

1. I think that I would like to use this system frequently.	O Strongly Disagree	O	O	O	O Strongly Agree
2. I found the system unnecessarily complex.	O Strongly Disagree	O	O	O	O Strongly Agree
3. I thought the system was easy to use.	O Strongly Disagree	O	O	O	O Strongly Agree
4. I think that I would need the support of a technical person to be able to use this system.	O Strongly Disagree	O	O	O	O Strongly Agree
5. I found the various functions in this system were well integrated.	O Strongly Disagree	O	O	O	O Strongly Agree
6. I thought there was too much inconsistency in this system.	O Strongly Disagree	O	O	O	O Strongly Agree
7. I would imagine that most people would learn to use this system very quickly.	O Strongly Disagree	O	O	O	O Strongly Agree
8. I found the system very cumbersome/ awkward to use.	O Strongly Disagree	O	O	O	O Strongly Agree
9. I felt very confident using the system.	O Strongly Disagree	O	O	O	O Strongly Agree
10. I needed to learn a lot of things before I could get going with this system.	O Strongly Disagree	O	O	O	O Strongly Agree

Acronyms

A-GPS Assisted Global Positioning System.

API Application Programming Interface.

AR Augmented Reality.

CAD Computer-aided design.

CGI Computer-generated imagery.

CTF Capture the Flag.

DoF Degree of Freedom.

GPS Global Positioning System.

SDK Software Development Kit.

SfM Structure from Motion.

SIFT Scale-invariant feature transform.

SLAM Simultaneous Localization and Mapping.

SUS System Usability Scale.

VR Virtual Reality.

WW2 World War II.

WWDC Apple Worldwide Developers Conference.

List of Figures

Bibliography

[App17] Apple. *Introducing ARKit: Augmented Reality for iOS - WWDC 2017 - Videos - Apple Developer*. 2017. URL: https://developer.apple.com/videos/play/wwdc2017/602/ (visited on 01/27/2019).

[App19a] Apple. *ARKit - Apple Developer*. 2019. URL: https://developer.apple.com/arkit/ (visited on 01/25/2019).

[App19b] Apple. *Understanding World Tracking in ARKit | Apple Developer Documentation*. 2019. URL: https://developer.apple.com/documentation/arkit/understanding_world_tracking_in_arkit (visited on 02/04/2019).

[App19c] Apple. *Verifying Device Support and User Permission | Apple Developer Documentation*. 2019. URL: https://developer.apple.com/documentation/arkit/verifying_device_support_and_user_permission (visited on 02/01/2019).

[Azu97] R. T. Azuma. "A survey of augmented reality." In: *Presence: Teleoperators & Virtual Environments* 6.4 (1997), pp. 355–385.

[BKM09] A. Bangor, P. Kortum, and J. Miller. "Determining what individual SUS scores mean: Adding an adjective rating scale." In: *Journal of usability studies* 4.3 (2009), pp. 114–123.

[Bro+96] J. Brooke et al. "SUS-A quick and dirty usability scale." In: *Usability evaluation in industry* 189.194 (1996), pp. 4–7.

[CGC14] H. Chu, A. Gallagher, and T. Chen. "GPS refinement and camera orientation estimation from a single image and a 2D map." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014, pp. 171–178.

[Cha+10] T.-J. Cham, A. Ciptadi, W.-C. Tan, M.-T. Pham, and L.-T. Chia. "Estimating camera pose from a single urban ground-view omnidirectional image and a 2D building outline map." In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE. 2010, pp. 366–373.

[Che+00] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. "Developing a context-aware electronic tourist guide: some issues and experiences." In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM. 2000, pp. 17–24.

[Cru18] Crunchbase. *Pokémon GO | Crunchbase*. 2018. URL: `https://www.crunchbase.com/apptopia_app/275a8937-7af4-450e-8ebe-d7b2fc2806ff` (visited on 01/29/2019).

[CY99] J. M. Coughlan and A. L. Yuille. "Manhattan world: Compass direction from a single image by bayesian inference." In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Vol. 2. IEEE. 1999, pp. 941–947.

[Dav16] M. Davis. "Ingress in Geography: Portals to Academic Success?" In: *Journal of Geography* 116 (Oct. 2016), pp. 1–9. DOI: `10.1080/00221341.2016.1227356`.

[DE15] F. van Diggelen and P. Enge. "The worlds first gps mooc and worldwide laboratory using smartphones." In: *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2015)*. 2015, pp. 361–369.

[DR01] G. M. Djuknic and R. E. Richton. "Geolocation and assisted GPS." In: *Computer* 2 (2001), pp. 123–125.

[Epi19] Epic Games. *ARKit Prerequisites*. 2019. URL: `https://docs.unrealengine.com/en-us/Platforms/AR/HandheldAR/AROverview` (visited on 02/01/2019).

[Eri16] Eric Harlow / Keystone / Getty – Jim Dyson / Getty. 2016. URL: `https://www.theatlantic.com/photo/2016/05/london-during-the-blitz-then-and-now-photographs/481851/#img07` (visited on 01/27/2019).

[Eur19] Eurostat, the statistical office of the European Union. *Personen, die für den Zugang zum Internet unterwegs mobile Geräte genutzt haben - Eurostat*. 2019. URL: `https://ec.europa.eu/eurostat/web/products-datasets/-/tin00083` (visited on 03/05/2019).

[FBS17] A. Fond, M.-O. Berger, and G. Simon. "Facade proposals for urban augmented reality." In: *Mixed and Augmented Reality (ISMAR), 2017 IEEE International Symposium on*. IEEE. 2017, pp. 32–41.

[Fin06] K. Finstad. "The system usability scale and non-native english speakers." In: *Journal of usability studies* 1.4 (2006), pp. 185–188.

[Gao+08] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt. "Micro-blog: sharing and querying content through mobile phones and social participation." In: *Proceedings of the 6th international conference on Mobile systems, applications, and services*. ACM. 2008, pp. 174–186.

[Geo19] Geocaching. *Geocaching*. 2019. URL: `https://www.geocaching.com/play` (visited on 01/29/2019).

[Goo19a] Google. *ARCore - Google Developer | ARCore | Google Developers*. 2019. URL: `https://developers.google.com/ar/` (visited on 01/25/2019).

[Goo19b] Google. *Fundamental Concepts | ARCore | Google Developers*. 2019. URL: `https://developers.google.com/ar/discover/concepts` (visited on 02/04/2019).

[Goo19c] Google. *Maps - Navigate & Explore - Apps on Google Play*. 2019. URL: `https://play.google.com/store/apps/details?id=com.google.android.apps.maps` (visited on 01/29/2019).

[Goo19d] Google. *Supported Devices | ARCore | Google Developers*. 2019. URL: `https://developers.google.com/ar/discover/supported-devices` (visited on 02/01/2019).

[Goo19e] Google. *Welcome to the ARCore SDK for iOS | ARCore | Google Developers*. 2019. URL: `https://developers.google.com/ar/develop/ios/overview` (visited on 02/01/2019).

[Hau18] A. Hauser. *Wikitude World Browser - Wikitude*. 2018. URL: `https://www.wikitude.com/wikitude-world-browser-augmented-reality/` (visited on 01/24/2019).

[IKE17] IKEA. *IKEA Launches IKEA Place, a New App that Allows People to Virtually Place Furniture in Their Home - IKEA*. 2017. URL: `https://www.ikea.com/us/en/about_ikea/newsitem/091217_IKEA_Launches_IKEA_Place` (visited on 01/27/2019).

[IKE18] IKEA - Apple. *IKEA Place on the App Store*. 2018. URL: `https://itunes.apple.com/us/app/ikea-place/id1279244498` (visited on 01/27/2019).

[Jos02] W. Jost. *Das Heidentor von Petronell-Carnuntum*. Verlag der Österreichischen Akademie der Wissenschaften, 2002.

[KGC15] A. Kendall, M. Grimes, and R. Cipolla. "Posenet: A convolutional network for real-time 6-dof camera relocalization." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2938–2946.

[Kön18] C. König. "Cultural Heritage Presentation with Mixed Reality." Bachelor's thesis. Technical University of Munich, 2018.

[Low04] D. G. Lowe. *Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image*. US Patent 6,711,293. Mar. 2004.

[Mac14] N. MacGregor. *Germany: Memories of a nation*. Penguin UK, 2014.

[McC13] E. McClean. "An Augmented Reality System for Urban Environments using a Planar Building Facade Model." Master's thesis. National University of Ireland Maynooth, 2013.

[Met15] Metaio. *metaio | Augmented Reality Provides Window into the Past with Berlin Wall Time Traveler App | Augmented Reality Products & Solutions*. 2015. URL: `https://web.archive.org/web/20150508165344/http://www-origin.metaio.com/about/press/press-release/2014/augmented-reality-provides-window-into-the-past-with-berlin-wall-time-traveler-app/` (visited on 01/24/2019).

[Mic16] Michael Gadd for MailOnline. *Then and now: Incredible composite images compare iconic Paris attractions during Nazi occupation with today's tourist traps | Daily Mail Online*. 2016. URL: `https://www.dailymail.co.uk/travel/travel_news/article-3067154/Then-Incredible-composite-images-compare-iconic-Paris-attractions-Nazi-occupation-today-s-tourist-traps.html` (visited on 01/25/2019).

[Nag12] K. Nagaki. "Evolution of In-Car Navigation Systems." In: *Handbook of Intelligent Vehicles*. Ed. by A. Eskandarian. London: Springer London, 2012, pp. 463–487. ISBN: 978-0-85729-085-4. DOI: `10.1007/978-0-85729-085-4_18`.

[Nia19a] Niantic - Google. *Ingress Prime - Apps on Google Play*. 2019. URL: `https://play.google.com/store/apps/details?id=com.nianticproject.ingress` (visited on 01/29/2019).

[Nia19b] Niantic - Google. *Pokémon GO - Apps on Google Play*. 2019. URL: `https://play.google.com/store/apps/details?id=com.nianticlabs.pokemongo` (visited on 01/29/2019).

[OHa08] K. O'Hara. "Understanding geocaching practices and motivations." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2008, pp. 1177–1186.

[Opea] Open Knowledge Foundation Deutschland (OKF De). *Coding Da Vinci*. URL: `https://codingdavinci.de/events/berlin/` (visited on 01/24/2019).

[Opeb] Open Knowledge Foundation Deutschland (OKF De). *Coding Da Vinci*. URL: `https://codingdavinci.de/` (visited on 01/24/2019).

[Pan+18] C. Panou, L. Ragia, D. Dimelli, and K. Mania. “An Architecture for Mobile Outdoors Augmented Reality for Cultural Heritage.” In: vol. 7. 12. Multidisciplinary Digital Publishing Institute, 2018, p. 463.

[Pau18] Paul Tassi - Forbes. *'Pokémon GO' Has Just Crossed $2 Billion In Revenue Since Launch, Almost An Industry Best*. 2018. URL: `https://www.forbes.com/sites/insertcoin/2018/09/25/pokemon-go-has-just-crossed-2-billion-in-revenue-since-launch-almost-an-industry-best/` (visited on 01/29/2019).

[RD06] G. Reitmayr and T. Drummond. “Going out: robust model-based tracking for outdoor augmented reality.” In: *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society. 2006, pp. 109–118.

[Ron15] Ron Miller, Josh Constine - TechCrunch. *Apple Acquires Augmented Reality Company Metaio | TechCrunch*. 2015. URL: `https://techcrunch.com/2015/05/28/apple-metaio/` (visited on 01/31/2019).

[Sch16] M. T. Schwarzmann. “HistoricARGuide - A markerless approach for displaying historic location-based Augmented Reality content.” Master's thesis. Technical University of Munich, 2016.

[Sch18] S. Schneider. *Instagram Post by therealsebis - Deskgram*. 2018. URL: `https://deskgram.net/p/1789418435315586405_7808930372` (visited on 01/25/2019).

[SF16] J. L. Schönberger and J.-M. Frahm. “Structure-from-Motion Revisited.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[Str18] S. Strauss. *About the app | MauAR*. 2018. URL: `https://mauar.berlin/en/` (visited on 01/24/2019).

[Swe10] C. Swertz. *Smartphones im Klassenzimmer - Medienimpulse*. 2010. URL: `http://www.medienimpulse.at/articles/view/251` (visited on 03/10/2019).

[Sze+15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going deeper with convolutions.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[TD15] P. Tolstoi and A. Dippon. “Towering defense: An augmented reality multi-device game.” In: *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM. 2015, pp. 89–92.

[Tol14] P. Tolstoi. “Towering Defense: An Augmented Reality Multi-Device Game.” Bachelor's thesis. Technical University of Munich, 2014.

[Tom19] TomTom. *TomTom GO 620*. 2019. URL: `https://www.tomtom.com/en_us/drive/car/products/go-620/` (visited on 01/29/2019).

[Uni18] Unity. *Unity's Handheld AR Ecosystem: AR Foundation, ARCore and ARKit – Unity Blog*. 2018. URL: `https://blogs.unity3d.com/2018/12/18/unitys-handheld-ar-ecosystem-ar-foundation-arcore-and-arkit/` (visited on 02/04/2019).

[Uni19] Unity. *Unity Public Relations*. 2019. URL: `https://unity3d.com/public-relations` (visited on 02/04/2019).

[Wik16] Wikimedia Commons. *File:Heidentor Carnuntum 4487.jpg — Wikimedia Commons, the free media repository*. 2016. URL: `https://commons.wikimedia.org/w/index.php?title=File:Heidentor_Carnuntum_4487.jpg&oldid=224108524` (visited on 01/27/2019).

[Zha13] M. Zhang. *Then-and-Now Photos of New York City*. 2013. URL: `https://petapixel.com/2013/08/17/then-and-now-photos-of-new-york-city/` (visited on 01/30/2019).

[Zho+14] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. "Learning deep features for scene recognition using places database." In: *Advances in neural information processing systems*. 2014, pp. 487–495.