Graz University of Technology

Institute for Computer Graphics and Vision

Master's Thesis

Mobile Augmented Reality Campus Guide

Claus Degendorfer

0331357

November 2010, Graz, Austria

Thesis supervisors

Univ. Prof. DI Dr. Gerhard Reitmayr

Dipl. Mediensys.wiss. Tobias Langlotz

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....

Date Claus Degendorfer

Abstract

Smartphones are becoming increasingly interesting as a mobile Augmented Reality (AR) platform over the past few years because of improved hardware resources, including processing power, memory capabilities, built-in cameras and GPS sensors. With such devices it is possible to create mobile AR information systems which provide augmented reality anywhere, at anytime. Some limitations of current AR systems are a lack of appropriate AR content, the inaccuracy of current sensor-based annotation matching approaches and the poor matching rates of vision-based approaches under changing environment conditions which we want to address in this work.

We therefore developed a system to enable end-users to create textual AR annotations which provide information about the surrounding environment on a global scale. Furthermore, we investigated the possibilities of vision-based annotation matching and implemented three different improvements to annotation matching. Tests showed that a combination of these improvements can increase the annotation matching rate under difficult lighting situations by up to 50 %. This work was therefore one step in the evolution of mobile AR information systems.

Keywords: Mobile Augmented Reality, Smartphones, Outdoor AR Information Systems, Panoramas

Zusammenfassung

Smartphones sind in den letzten Jahren als Plattform für mobile Augmented Reality (AR) Anwendungen aufgrund verbesserter Hardwareressourcen wie Rechenleistung, Speicherkapazität, eingebauter Kameras und GPS-Sensoren, zunehmend interessant geworden. Mit Geräten dieser Art ist es möglich, mobile AR Informationssysteme zu entwickeln, die Augmented Reality jederzeit und überall ermöglichen. Diese Arbeit befasst sich mit der Verbesserung von Nachteilen derzeitiger AR Systeme, wie z.B. unzureichenden AR Inhalten, der Ungenauigkeit von sensorbasierten Annotationserkennungsverfahren und der Probleme bildbasierter Annotationskennung unter sich ändernden Beleuchtungssituationen.
Im Zuge dieser Arbeit wurde deshalb ein System entwickelt, das Endnutzern die weltweite Erstellung von textuellen AR Annotationen, die ihre Umgebung beschreiben, ermöglichen soll. Außerdem wurden die Möglichkeiten von bildbasierten Annotationserkennungsverfahren untersucht und drei unterschiedliche Verbesserungen implementiert, die die Erkennungsrate steigern sollen. Die Untersuchungen haben gezeigt, dass eine Kombination dieser Verbesserungen die Annotationserkennungsrate unter schwierigen Beleuchtungssituationen um bis zu 50 % steigern kann. Diese Arbeit ist daher ein weiterer Schritt in der Evolution mobiler AR Informationssysteme.

Keywords: Mobile Augmented Reality, Smartphones, Outdoor AR Information Systems, Panoramas

Danksagung

Ich möchte mich an dieser Stelle bei Tobias Langlotz und Gerhard Reitmayr für die ausgezeichnete Betreuung bedanken, ohne die meine Arbeit in dieser Form nicht möglich gewesen wäre. Ihr schnelles und konstruktives Feedback hat zweifelsohne zu einer enormen Qualitätssteigerung dieser Arbeit beigetragen.

Des weiteren danke ich dem gesamten ICG Team für seine Hilfsbereitschaft und die unbürokratische Unterstützung. Ein Dankeschön auch an Clemens Arth für den guten Serversupport an Wochenenden und Feiertagen.

Adam, dem freundlichen Australier, danke ich für das Korrekturlesen und die sprachliche Aufwertung meiner Arbeit.

Meinen Studienkollegen Stefan, Ferdinand und Robert danke ich für die zahlreichen Diskussionen und die gemeinsamen Jahre während des Studiums.

Weiters möchte ich mich ganz besonders bei meiner Familie bedanken, die mich während meines gesamten Studiums unterstützt hat.

Meiner Freundin Anita danke ich für die Toleranz bezüglich meiner Arbeitszeiten in der Abschlussphase dieser Masterarbeit.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In contrast to Virtual Reality (VR), where users experience completely artificial worlds which have to be created in advance, Augmented Reality (AR) uses the images of the real world coming from input sources e.g. webcams or smartphone cameras and enhances them by virtual content. Azuma [3] defines augmented reality as systems which have three characteristics:

1. Combines real and virtual
2. Interactive in real time
3. Registered in 3-D

Traditional AR systems often used a desktop computer with printed 2D markers for orientation tracking and were bound to their working environment. To move AR towards mobile applications different approaches were investigated. It is crucial for mobile AR systems to be lightweight, wearable and powerful enough to ensure interactive frame rates and to provide appropriate displays. Spohrer [25] distinguishes three different output systems which are suitable for mobile augmented reality applications:

1. Head Mounted Displays (see Figure 1.1)
2. Handheld devices
3. Projector displays

Figure 1.1: This Figure shows an example of a mobile AR system composed out of a backpack computer with Head Mounted Display by Wither et al. [37]

Although Head Mounted Displays (HMDs) and projector display systems are convenient for general mobile AR applications, we focus on the use of handheld devices in our work for several reasons. While head mounted displays are the best choice for many industrial applications, they are generally not available for non-expert users. Their advantage is, that HMDs can be worn continuously while performing a specific task where free hands are needed. A comprehensive list of different AR applications which use HMDs can be found in the work by Höllerer et al. [13]. Although HMDs also improved over the past few years, they are still bulky and quite expensive compared to current handheld devices. Therefore, many AR systems using HMDs remain scientific prototypes whereas current smartphones provide a promising field of application for end users. This is because they are cheap, commonly available

and offer all possibilities which are needed for mobile AR systems. As stated before, handheld devices fit best for our purpose at the moment and are used to build our AR Campus Guide prototype. Projector displays have some interesting fields of application where collaborative AR is desired, but again, for the same reasons which apply for HMDs they are unsuitable for our aim making AR available anywhere and anytime.

Spohrer [25] envisioned a global AR information system back in 1999, called WorldBoard, which should be able to associate information with their according places. He introduces three design goals of WorldBoard which are a prerequisite for success and also apply for our work in the long run:

1. The system should work on a planetary scale
2. As technology advances WorldBoard should be improved rapidly
3. WorldBoard should be simple and useful for people in their everyday life

As already described, the aim of augmented reality is to enrich the real world with further virtual content. This can be achieved by AR annotations which consist of text, 2D images, 3D models, sound or video information. The benefit of this approach is that virtual information can be displayed in context to real world objects. Physical objects, e.g. historical buildings in an urban environment, can be overlayed for example with a textual description of the construction date, name of the building or any other useful information as described by Höllerer et al. [13]. This has the advantage, that the required information is on the right place whenever needed and is therefore much more comfortable for users than looking it up in printed tourist guides or websites.

The idea of using augmented reality for navigation or information systems is not a new one. Feiner et al. [9] proposed a prototype of an AR campus information system, called the Touring Machine, already in 1997. They used a combination of a notebook computer with video see-through Head Mounted Display (HMD) and a 2D handheld display. Furthermore, they used a magnetometer for orientation tracking and Global Positioning System (GPS) for position tracking. Although they assembled the system with off-the-shelf lightweight components, it had an overall weight of approximately 20 kilograms (see Figure 1.2).

Figure 1.2: AR campus information system called Touring Machine developed by Feiner et al. [9] in 1997

Current smartphones provide similar functionality but have many advantages compared to HMDs with backpack computers. Although wearable computers and HMDs also miniaturized and made technical advances, smartphones are generally cheaper and their pervasiveness increases every year, as mentioned by Wagner et al. [33]. Therefore, potential users of mobile AR applications which already own a smartphone don't need to purchase any special hardware but only the AR software. This makes deployment of AR applications much easier and supports the vision of building AR systems on global scale.

1.1 Limitations of current AR systems

One issue of current AR systems which has to be addressed is a lack of content as described by Wither et al. [38]. They suggest to build systems where end-users are able to create and provide AR content similar to web 2.0 applications. For this reason, we want to develop a system which enables non-expert users to create, modify and delete AR annotations in a fast and easy way.

Another problem of mobile AR is to create a system which is interactive and works in real-time while also being available to a broad audience. As described above, backpack computers in combination with HMDs are unsuitable for this purpose as they are still too expensive and bulky for everyday use. We therefore want to establish mobile augmented reality on inexpensive off-the-shelf hardware. The challenge of using such hardware is to achieve interactive frame-rates despite the restricted resources provided by current smartphones.

Accurate tracking is crucial for augmented reality applications to avoid registration errors. Furthermore displaying AR content at its correct position in the real world is important to establish AR. This is a non-trivial task if these positions are calculated with a vision-based approach, especially in outdoor environments where lighting conditions are changing continuously. Using built-in smartphone sensors such as a digital compass leads to inaccuracies because of magnetic fields which disturb their proper operation. We therefore want to introduce a mobile outdoor AR system running on smartphones which is able to establish robust 3DOF tracking and to display virtual content with high accuracy, even in a changing environment.

We will explain how to address these issues in more detail in the next section.

1.2 Contribution

In this work we demonstrate a system with a similar field of application as the Touring Machine which provides users with additional virtual information about the buildings at our campus area. The difference is that our prototype works fully self-contained on smartphones and therefore scales very well. Furthermore, we use vision-based annotation matching whereas the touring machine relies on a sensor-based approach. We will discuss how our system works in detail in chapter 3.

The prototype developed in this Master's Thesis is based on the work of Wagner et al. [29] and Langlotz et al. [18]. They introduce a system which is able to create and track from panoramic maps in real-time (30Hz). With this technique pure rotational movement can be estimated. It is also possible to create textual annotations directly on the smartphone while exploring the environment. Afterwards the panoramic maps can be saved and uploaded together with the annotations to a content server for later use. At a later date, users can download these annotations if they are on the same location. Only the panoramas and the corresponding annotations are stored on a content server. This behavior supports the requirement of being operable on a global scale.

To support this requirement further, we introduce a content-pipeline in chapter 4 which organizes creation, modification and deletion of panoramas and annotations in a fast and easy way and also handles the communication between AR client and server. One part of the content-pipeline is a web-interface which we use as frontend for interaction with our content-server. This makes panorama and annotation organization simple and fast for non-expert users. Another feature of the web-interface is the ability to link annotations with their corresponding GPS address in the real world. This information allows system improvements to annotation matching in combination with compass data, which we explain in section 5.1.

As already described, the system of Langlotz et al. [18] uses a computer-vision approach to match downloaded annotations with new generated panorama maps. If

Figure 1.3: Snapshot of the mobile AR information system by Langlotz et al. [18] which is the basis of our work

a new panorama contains the same image patches as the annotations and matching is successful, they are displayed at their corresponding place. Vision-based annotation matching is very accurate but can fail if weather or lighting settings change significantly. We therefore present three major improvements to overcome these issues, namely

- Combination of vision-based annotation matching and compass data
- Extended dynamic range for image improvements
- Robust annotation matching model with the Random Sample Consensus (RANSAC) algorithm

We will describe the details of these approaches in chapter 5. In chapter 6, we evaluate the current system and the above described improvements separately and in different combinations. We give a conclusion of our results in chapter 7 and an outlook to future work in chapter 8. In the next chapter we discuss related work.

Chapter 2

Related work

2.1 Overview of mobile information systems

To give an overview of mobile information systems we divide them into two main categories:

1. Location based services
2. Mobile augmented reality systems

Furthermore, we can separate the latter into AR systems with high or low tracking accuracy.

2.1.1 Location based services

Location based services provide users with context sensitive information corresponding to their current location. However, they do not use augmented reality as an interface for providing additional information to the user. They also can display information, but in contrast to AR systems, this information is not registered in 3D within the real world but use 2D overview maps with marked Points of Interest (POIs) which may be relevant for users at the current position.

One common example of location based services are tourist guides. Schwinger et al. [24] compare and analyze nine different tourist guides in their work. The GUIDE project by Cheverst et al. [6] is a web-based handheld information system,

also designed to provide tourists with useful information. Facebook Places is another location based service [8] where users are informed about the location of their friends and are enabled to leave messages to nearby people. There are many more examples of location based services but we only want to convey an impression rather than listing many examples at this point.

Most location based services use sensors, such as GPS together with a digital compass to determine the current position and orientation of users. However, Bruns et al. [5] developed a vision-based system which is able to detect objects in the current camera image. This approach is somehow related to our goal of vision-based annotation matching although they do not determine any pose information of the detected objects and they only used it in smaller indoor environments.

2.1.2 Mobile AR systems

We can divide mobile AR systems according to their tracking accuracy into sensor-based and vision-based systems.

Low accuracy tracking using sensors: We presented one example of a sensor-based backpack system already in the introduction, the Touring Machine by Feiner et al. [9]. Another sensor-based HMD system was developed by Kooper et al. [17]. They built an indoor information system which is able to present 2D images containing text to users in an AR view after accessing a desired information node by gaze-selection. Reitmayr et al. [22] present a collaborative outdoor AR system for navigation and information browsing which uses a sensor-based orientation tracker and GPS for position tracking. These were only a few examples of mobile, sensor-based AR systems using HMDs.

Besides the research prototypes, there are also some commercial AR information systems on the market. Two of them are the Wikitude World Browser [20] and the Layar Reality Browser [19] which run on common smartphones. With these systems, users are able to get information about their surrounding environment, e.g. building names, bars, restaurants, etc. Although Wikitude and Layar have many users, their systems have a big disadvantage. They are using smartphone sensors for position and rotation estimation. The problem with these sensors is that they are sometimes

very inaccurate. Especially digital compasses are prone to errors if nearby magnetic fields are disturbing their operation, which is often the case in urban environments. This is a problem because errors in orientation tracking can lead to a comparative big registration error, especially if annotated objects are very far away, as stated by Azuma [4]. As a result of these errors, such a system often gives only a rough estimation where the objects of interest are located instead of an accurate direction.

High accuracy vision-based tracking: One example for vision-based tracking is provided by DiVerdi et al. [7] in their work about Envisor. Their system uses a combination of relative frame-to-frame tracking and absolute landmark tracking to achieve a better result. Frame-to-frame tracking introduces a small error which accumulates over long time periods. Therefore, landmark tracking is applied to correct these errors which leads to a better accuracy. Furthermore, they support online environment map creation similar to our system. The difference is that Envisor does not run on smartphones.

Takacs et al. [27] show an handheld outdoor AR system which takes a queue of query images with the integrated phone camera with a resolution of 640 x 480 pixels. Afterwards they extract significant keypoints of the image, called features, using SURF descriptors. These features of the query images are matched against features obtained by a feature database. If they find correspondences, they render 2D labels which describe Points of Interest (POIs) e.g. buildings, landmarks, etc. at their corresponding position. Although their system is quite similar to our work, they do not achieve interactive frame rates yet, but an average of about three seconds for one image matching procedure. Our prototype works in real-time, although one must state that annotation matching is done within a given time budget. Therefore it is possible that annotations are matched with a slight delay. We describe the annotation matching process in detail in section 3.2.

2.1.3 Comparison of the different categories of mobile information systems

As described above, location based services provide users with context sensitive information. The drawback of such systems is that they do not use AR as a natural

interface for viewing the real world overlayed by virtual information. Therefore, such systems cannot display e.g. detailed information about individual floors of buildings or parts of physical objects because they do not offer a possibility to present their information registered in 3D. For this reason, location based services without providing AR are not suitable for achieving our goals.

Many mobile sensor-based AR systems use backpack computers combined with HMDs to establish augmented reality. From our point of view, such systems are not suitable for a broad user base although they have some special applications where they are appropriate. The few AR systems available for smartphones cannot provide sufficient accuracy because of the error prone built-in sensors and are therefore unsuitable for our application as well.

Mobile AR systems using a vision-based approach are very accurate but similar systems either do not run on current smartphones at all or not with interactive frame rates. We therefore present a novel approach which achieves real-time performance on current smartphones developed by Wagner et al. [29] and Langlotz et al. [18] which we extend, improve and evaluate in this work. This system is able to find AR annotations with up to pixel-accuracy if the matching process succeeds. As already mentioned, one must state that the matching of annotations can fail in some cases. If the annotation is occluded by objects or weather settings change, it is possible that some annotations cannot be matched any longer. We will describe how matching works in detail in chapter 3 and also under which circumstances it can fail. In chapter 5, we explain how we improve the annotation matching process to minimize the number of unmatched annotations.

2.2 Evolution of mobile augmented reality

While augmented reality is a field of research since the late 1960's where Sutherland [26] developed the first HMD system, AR becomes increasingly interesting for handheld applications. Processing power and memory resources of smartphones have improved over the past few years and most current mobile phones also include a camera which provides sufficient image and video quality. Global Positioning System (GPS) sensors and improved networking capabilities complete the required compo-

nents which are needed for AR systems. Using affordable off-the-shelf hardware, which is provided by smartphones, as AR platform enables a wide range of mobile AR applications. This makes augmented reality available anywhere and anytime.

Besides capable AR hardware, special software is needed to establish an augmented reality environment. AR software was developed and mainly used on Personal Computers or notebooks in the past. This means that existing AR software was not optimized for handheld systems. Although smartphone performance is increasing continuously, there are still many limitations. Wagner et al. [32] notice in their work that processing power only doubled in five years and smartphones are still about 30 times slower than current desktop computers. They also show the evolution and miniaturization of mobile AR devices which evolved from carried notebooks with Head Mounted Display (HMD), over tablet PCs and Personal Digital Assistants (PDAs) to smartphones.

As mentioned before, many existing approaches and algorithms were not efficient enough to run on smartphones in real time, which is essential for interactive AR systems. Therefore, these approaches have been adapted and optimized. First steps to handheld AR were taken by Wagner et al. [31]. They ported the software library ARToolKit, which is able to track AR markers, to run on a PDA with attached camera (see Figure 2.1).

This system uses markers for pose tracking and therefore needs prepared scenes to establish augmented reality. This means that printed 2D markers have to be attached everywhere in the physical world, where virtual content should show up. Although marker tracking is very fast to compute, it has the big disadvantage that it only works in scenes prepared for it.

Another approach which also works in unprepared scenes is Natural Feature Tracking (NFT). This technique uses significant keypoints in images for pose estimation and tracking. The only prerequisite is that a feature database is known in advance, but the database creation can be done in situ on smartphones or with a client-server approach. Natural Feature Tracking is a computational intensive task which brings smartphones to their limits in its original version using SIFT or Ferns descriptors. For this reason NFT was adopted to run in real time on mobile phones by Wagner et al. [30].

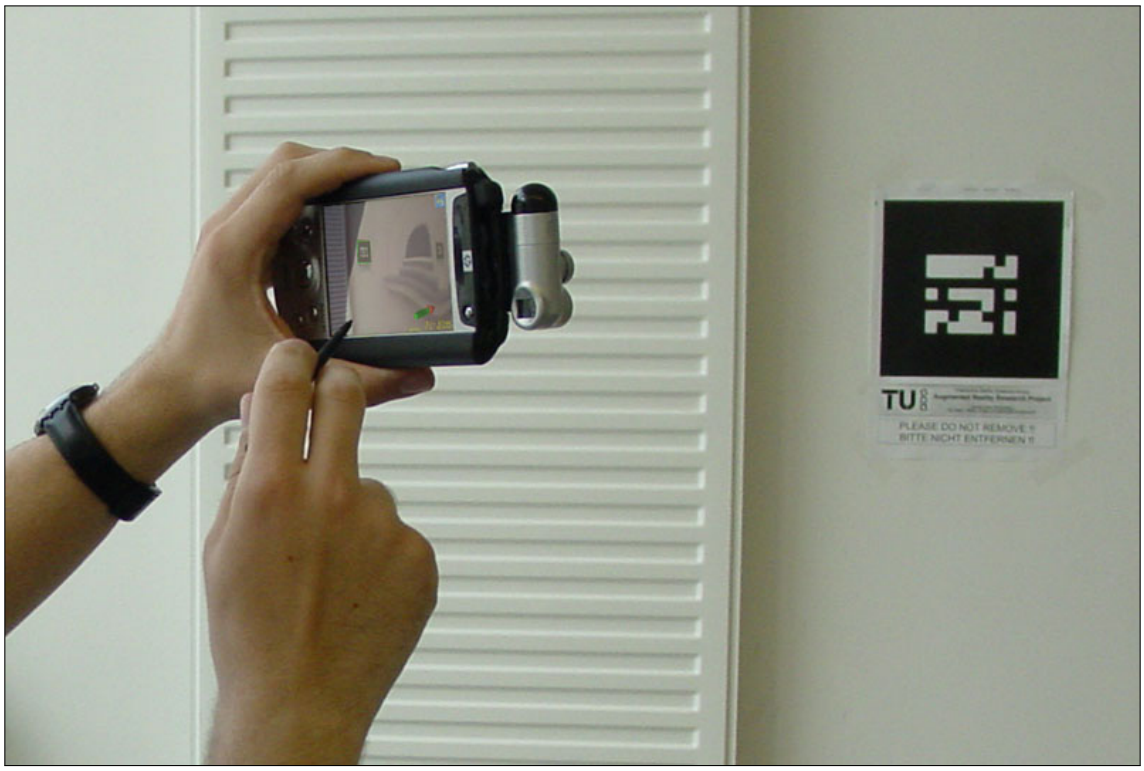

Figure 2.1: Marker-based AR system by Wagner et al. [31] running on a PDA

2.3 Augmented reality annotations in outdoor applications

After describing general mobile information systems and the evolution of AR in the previous sections, we want to focus on AR annotations and AR content creation in the following text because this issues play an important role for AR information systems and the AR campus guide which we present in this work. We use textual AR annotations which describe buildings or other points of interest and support users in orienting themselves. Determining the exact position of AR annotations in the real world is a common problem and an active field of research, as we describe in detail later on. Furthermore, a problem of current AR systems is a lack of content as mentioned by Wither et al. [38]. This is because many AR systems only use predefined content which is created in advance. To make an AR system successful on a global scale, it is crucial to enable every-day users to create AR content by themselves. They could leave for example geocoded messages at specific places

which are relevant to other users, e.g. descriptions or ratings of restaurants, hotels, etc. as suggested by Spohrer for WorldBoard [25].

Kooper et al. [17] envision a world wide system which they call the Real-World Wide Web (RWWW). Their idea is to link already existing web sites to locations whenever possible and useful to make them context sensitive. They also suggest the development of RWWW browsers to view the geo-tagged web sites. However, it is not possible with their RWWW browser prototype to create new content by users, although this approach would provide a lot of content which already exists in form of websites.

Schmalstieg et al. [23] introduce the term AR 2.0 by comparing the shift of AR content creation by some professionals to mainstream AR content creation with the development of web 2.0 technology. They present five key areas which have to be addressed to make AR 2.0 applications successful on a global scale:

1. A low-cost platform that combines AR display, tracking and processing
2. Mobility to realize AR in a global scale
3. Backend infrastructure for distribution of AR content and applications
4. Easy to use authoring tools for creating AR content
5. Large-scale AR tracking solutions which work in realtime

A mobile low-cost platform is provided by current smartphones, fulfilling point one and two. Content servers which use standard web technology can provide a basis for a backend infrastructure. We therefore use a server in our system which relies on such standard technologies. One key area we want to address in our work is the easy to use authoring of AR content. As we describe in chapter 4, we use a web interface which enables users to create and administrate AR annotations in a simple and fast way. Our large-scale AR tracking solution is provided by Wagner et al. [29] which we describe in more detail in the next chapter.

As we mentioned already in chapter 1, we use textual AR annotations to provide virtual information about our campus buildings to support users in orienting themselves. A big problem during outdoor AR annotation creation is the difficulty

of determining their exact position in the real world. Wither et al. [38] present four different approaches for this task:

- model based
- estimation based
- triangulation based
- measurement based

If a virtual model of the real world exists, the exact position of the annotation can be calculated. As described in detail by Reitmayr et al. [22], a ray can be cast into the scene and intersected with the virtual model. If the ray hits the model at a specific point, the corresponding physical location can be determined. Although this method is quite accurate, easy to use and fast, it has the big disadvantage that a model of the real world must be created in advance. Obviously, this approach does not scale very well and is not practical for global use. For this reason, we are not using the model based approach in our work.

Another technique described by Wither et al. [38] is position estimation done by the user while working with a mobile AR system. Some virtual markers, which visualize the distance between objects and users, are used for estimation assistance. But even with visual helpers and a top down view which shows the already placed annotations, the error made by users was about 10 % on average. This approach is very inaccurate and therefore also not suitable for our system.

One possibility for triangulation based annotation creation is to cast rays from different positions in the direction where the annotation should be created. Afterwards the exact position can be calculated by ray intersection tests. This method is very accurate but has the big disadvantage, that users must move to different locations for any annotation they want to create. This takes a lot of time and is not very convenient for everyday use.

Another method for annotation creation is described by Wither et al. in [37]. They use aerial photographs to support the annotation process. After casting a ray into the direction of the object which should be annotated, the view changes to the aerial photograph. Afterwards the annotation can be moved along the ray until

the correct depth is reached by rolling a trackball. With this approach, users need not change their position and can also create very accurate annotations. However, this system is composed out of a backpack computer with HMD and a mouse with trackball and is therefore unsuitable for applications using smartphones.

For the measurement based approach, Wither et al. [36] use a single-point laser range finder which is described in detail. They mounted the laser range finder parallel to the camera to cast rays in the same direction the user is looking. The laser range finder measures the exact distance between the user and the physical object in the real world, which is hit by the laser ray. With this information, the position of the annotation to be created can be calculated easily.

This system is very accurate (about one meter) and works up to 365 meters. It is also very convenient for users to create annotations because they only have to look at the point where they want to place an annotation and confirm the creation by a button click. This makes online annotation creation very fast and comfortable but has the disadvantage that special hardware is needed to measure the distance. The problem with this approach is that current smartphones are not equipped with laser range finders at the moment and it is questionable if they will in near future.

Chapter 3

Overview of the Studierstube ES AR framework

The work of Wagner et al. [29] and Langlotz et al. [18] is the basis of our improvements regarding annotation matching. We describe some parts of their system in detail in this chapter to understand the extensions we implemented in our work.

Studierstube ES is an AR framework developed by the handheld AR team [15] of Graz University of Technology. It evolved from the Studierstube 4 framework which was developed for the usage on PCs and was ported to run on mobile phones. However, most approaches and algorithms where not suitable for the restricted hardware resources provided by mobile phones. Therefore, this framework was rewritten completely and optimized, allowing real-time frame rates for AR applications on current devices, as described in detail by Wagner et al. [33, 34].

As one can see in Figure 3.1, Studierstube ES abstracts the different smartphone hardware, operating systems and special APIs from the rest of the system. This makes the exchange of different smartphones easy and allows fast development and testing with different hardware.

Studierstube ES consists of many different modules which provide special functionality to application programmers. Studierstube Math implements fixed-point math operations because most smartphones do not have an integrated Floating Point Unit (FPU). Therefore, significant speedups can be achieved, if calculations which would use double or float values normally, are replaced by fixed-point arith-

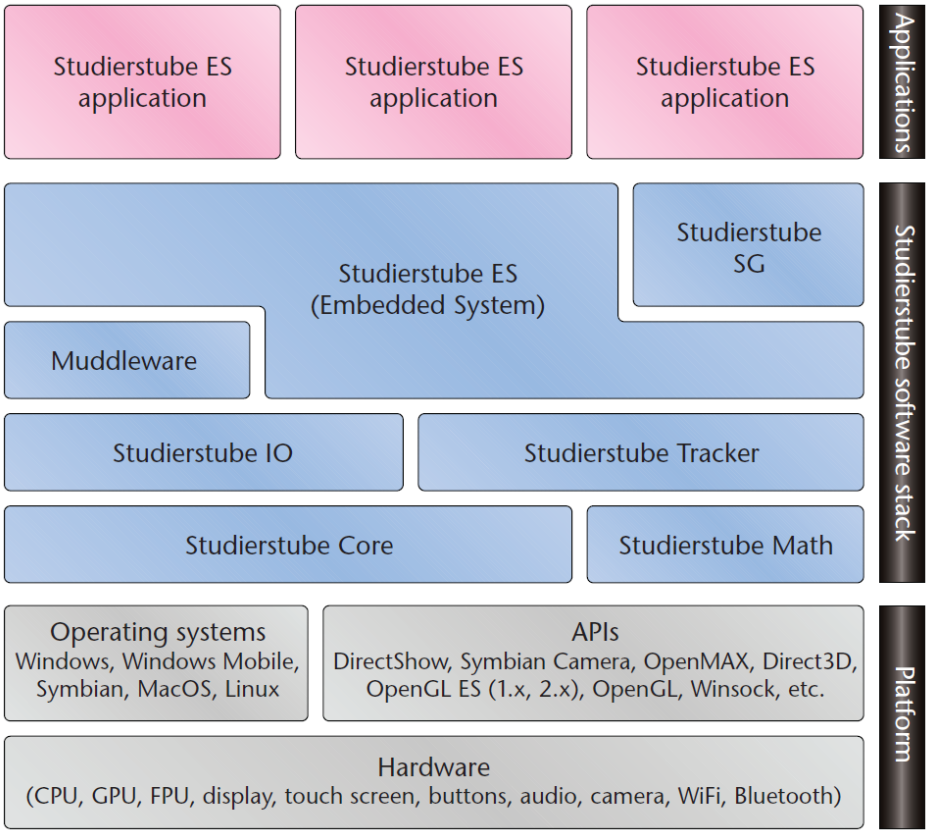

Figure 3.1: Overview of the Studierstube ES architecture developed by the handheld AR team [15]

metic. A detailed description about the differences of floating-point and fixed-point math is given in [34]. Studierstube ES can be compiled with either floating-point support for PC-usage or fixed-point math if the target platform is a mobile phone without FPU.

The Core module is responsible for managing threads, sockets, logging and handles different sensors, including GPS, digital compass and accelerometers. Furthermore it supports general data types, for example strings and vectors, to achieve platform independence. The Studierstube IO module implements functionality for XML and string parsing, handling HTTP requests and zipping/unzipping files. The Tracker module implements camera pose estimation and supports tracking of different markers and NFT targets. For efficient rendering, Studierstube ES uses a scenegraph which is implemented in Studierstube SG. This module handles all dif-

ferent scenegraph nodes, scenegraph traversal and rendering of the scenegraph.

The Studierstube ES module brings everything together and is the entry point for all Studierstube ES applications. Every Studierstube ES application has to inherit from the *Application* class which provides the initialization, update and render methods which can be overwritten and customized to fit the special needs of the application.

One big advantage of using the Studierstube ES framework is the possibility for fast prototyping. A lot of basic functionality which is needed in every AR application is already provided by the framework, e.g. video input, rendering, tracking, etc. This means, that developers can focus on their application specific needs and only need high level programming skills without concerning about the underlaying hardware or platform. Another advantage is that the framework is used in many different projects and therefore is well tested. This results in less debugging effort for the application developer.

3.1 Orientation tracking using panoramic maps

In this section, we explain some parts of the work of Wagner et al. [29] as we use the mechanisms of 3DOF orientation tracking in an outdoor environment in our work. In Figure 3.2 one can see a panorama map which was created online and in real-time by the orientation tracker by Wagner et al. [29]. This approach maps the first camera frame directly into the panorama. Every successive camera frame is compared to the map to determine if the visual information is already integrated into the panorama. If there are pixels in the camera frame which have not been mapped, they are also written into the panorama.

Besides 3DOF orientation tracking, the arising panorama map is also used for vision-based annotation matching which we explain in detail in the next section 3.2. To estimate the current orientation of the smartphone, the 2D panorama is mapped on a 3D cylinder as we show in Figure 3.3. When the camera is moved horizontally, the keypoints of new camera frames are compared to keypoints of the panorama map. If correspondences are found in both, the relative orientation can be estimated by projecting the according positions of the keypoints from map space

Figure 3.2: A sample panorama map where new pixels are added at runtime while the user explores his surroundings by Wagner et al. [29]

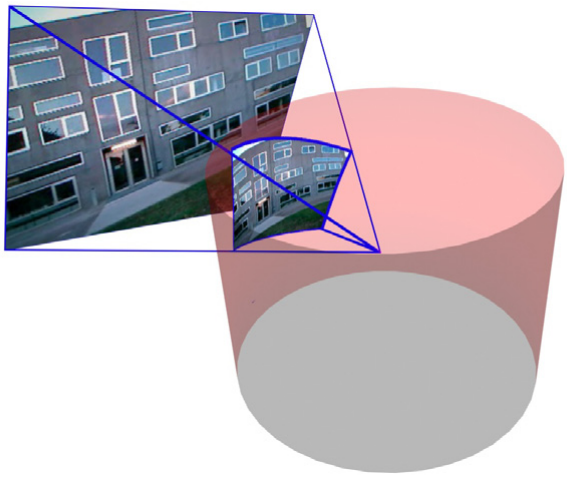

Figure 3.3: Schematic diagram of the cylindrical mapping process by Wagner et al. [29]

into the coordinate system of the cylinder. Afterwards the current orientation in the cylinder is calculated and used for mapping new pixels at their correct position in the 2D panorama map after forward projecting the camera frame into map space.

Mapping a panorama this way assumes pure rotational movement and the user is required to not change his position during the mapping process. This approach corresponds to a natural behavior of people exploring their environment as they stop and look around to investigate their surroundings.

In areas with very few keypoints it is likely that tracking fails because the orientation tracker needs a minimum amount of keypoints to compute and update the current orientation. This is often the case when pointing the smartphone into the sky or to the ground where similar texture leads to tracking difficulties. If tracking is lost, a built-in relocalizer tries to find a valid orientation again. This is

achieved by storing low-resolution keyframes and their corresponding orientations. The keypoints of these keyframes are then matched against the keypoints of the current camera image. If a valid correspondence is found, this orientation is used to re-initialize tracking after some refinement steps.

3.2 Vision-based annotation matching

The first approach of annotation matching by Langlotz et al. [18] was to store the created panorama map together with 2D image coordinates which described the annotation positions. The problem in this case are the high amounts of data which have to be submitted from the content servers to the mobile phone and also the unnecessary high memory requirements. Assuming only rotational motion, the camera frames were registered to the loaded panorama using a modified version of SIFT. This approach had problems if the user position was slightly different as in the panorama creation step. In such cases the registration was incorrect and the annotations where shown at wrong positions.

Langlotz et al. [18] demonstrated a new approach for vision-based annotation matching. For each annotation they extracted an image template consisting of a grayscale image with a size of 48x48 pixels. We show some example templates in Figure 3.4. Furthermore, each template is divided into nine image patches in a 3x3 configuration to improve the vision-based annotation matching performance (see left image of Figure 3.5).

After extracting image templates, they are stored together with the textual labels on the content server and are matched against the cells of new panorama maps. We show one example of such a map in Figure 3.6. If at least four out of nine template patches can be matched, the annotation label is displayed at the corresponding position. Matching the nine image templates separately is more robust than match-

Figure 3.4: This Figure shows four examples of the 48x48 grayscale image templates which are used for annotation matching.

Figure 3.5: Annotation template consisting of 9 image patches in a 3x3 configuration. This makes annotation matching more robust against small position changes. For further details see Langlotz et al. [18]

ing one big image and makes this approach able to tolerate slight position or scale errors which occur when users try to match the annotations at a different position as one can see on the right side of Figure 3.5.

If one panorama cell (Figure 3.6) is finished, the image templates (Figure 3.4) of all annotations are matched against this cell. This means, that matching is done on the panorama map and not directly on the current camera image which has the advantage that the annotation matching steps can be done in the background. All image templates of the annotations are queued and only processed if enough time is left for the calculations. Using this approach assures real-time framerates.

Langlotz et al. use Normalized Cross Correlation (NCC) as an efficient alterna-

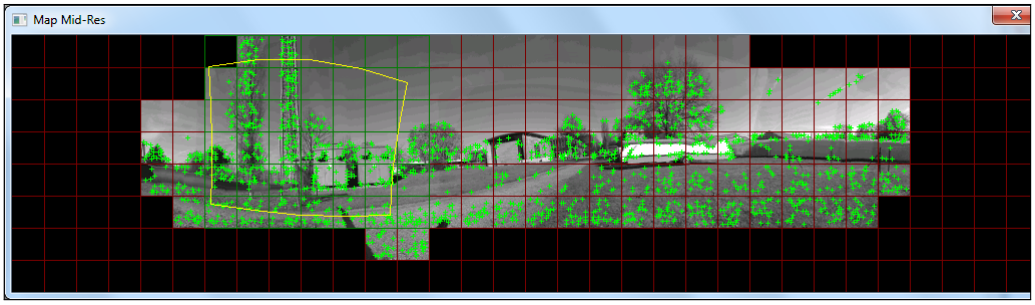

Figure 3.6: Grid of the panorama map consisting of 32x8 cells

tive to SIFT which would be too slow for real-time annotation matching. Furthermore, they use Walsh Transforms as an efficient pre-check before applying (NCC). For a detailed description about Normalized Cross-Correlation using Walsh Transforms see the work of Nillius et al. [21].

3.3 Limitations of the current system

If we want to operate our AR information system on a planetary scale, it must be interesting for many people to use it in their every day life. One critical factor of success is the quantity and quality of AR content provided by the system. At the moment, a limitation of our research prototype is a lack of appropriate AR content. We only use textual labels of building names around our university campus for development and testing purposes. Although Langlotz et al. [18] offer the possibility for users to create annotations directly on smartphones in real-time, there is no possibility to link this information with their corresponding position. Also the textual input with the aid of a software keyboard is slow if large amounts of text are to be entered or many annotations are made. We therefore present an annotation creation process via a web-based interface which is much faster and offers the possibility of linking annotations with their corresponding GPS address. Furthermore, we support using already existing data, in so called MashUps, in the annotation creation process which enables users to link geo-referenced content to newly created panorama images. We explain this approach in detail in section 4.3.2.

Another problem of the current system is that matching often fails if lighting conditions change. This happens throughout the day when the sun position changes and buildings are lit from a different angle. Also different weather settings and seasons influence the matching performance. Langlotz et al. [18] state in their work, that matching rates of only about 56 % can be achieved under these circumstances, although one must mention that they optimized their system to avoid false positives. A main goal of our work is to improve the matching performance under difficult lighting situations. We therefore present three different approaches in chapter 5 to address this issue and evaluate these improvements in chapter 6.

Chapter 4

Content pipeline for panorama and annotation creation

In this chapter we introduce a content pipeline to enable non-expert users to create, modify and delete panoramas and annotations. This supports the AR 2.0 approach where normal users act not only as content consumers but also as content providers. One main goal during the development of the content pipeline was to find ways which allow intuitive and fast interaction for users who are not familiar or specially trained to work with such an AR system. We therefore created a web-interface and support well known interaction technologies including drag & drop for linking the annotations with GPS positions. Also the client interface is very simple and easy to understand for casual and non-expert users of smartphones (see Figure 4.4). We give a rough overview of the content pipeline in Figure 4.1 and show the communication between the individual parts of the system. A detailed description about these parts follows in the next sections.

As described already in chapter 3, users can create panoramas directly on smartphones while exploring their environment (Client on the left side in Figure 4.1). We then save the panorama locally and upload it together with the current GPS address of the user who is creating the panorama. Afterwards this panorama is stored and indexed via the GPS address of the location where it was created on our content server. Although it is still possible to annotate objects in place using the the system of Langlotz et al. [18], we introduce a web-interface which is responsible for

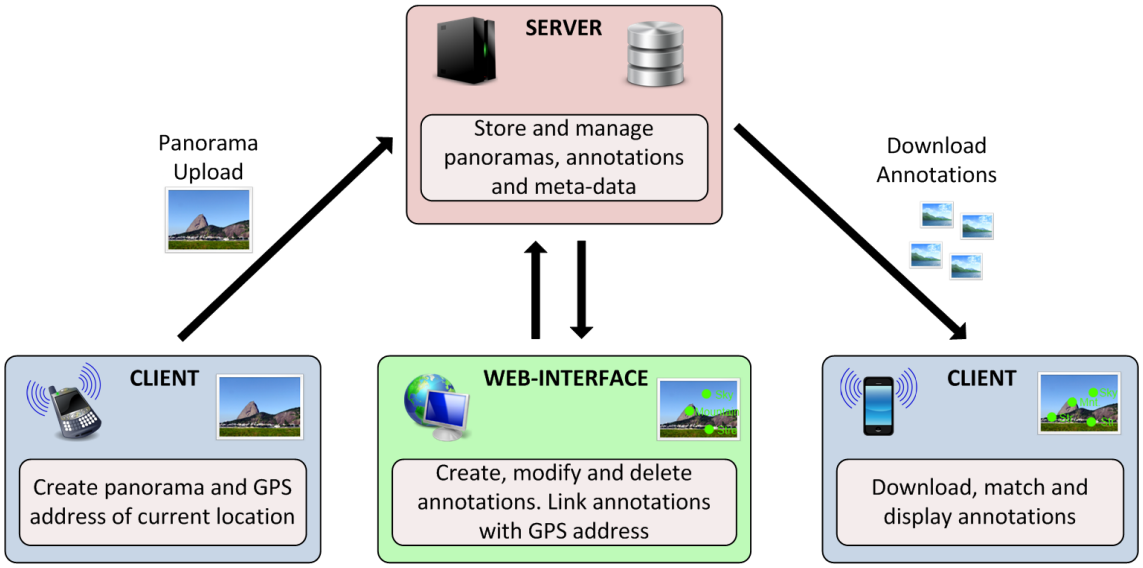

Figure 4.1: Overview of the content pipeline which we use for panorama and annotation management

annotation creation. This is because we need the corresponding GPS addresses, not only for the panorama itself, but also for all annotations for our system improvements. This step is not possible at the moment directly on the smartphone but is supported by our web-interface. Furthermore we support the modification and deletion of annotations which was not possible in the previous prototype. The web-interface directly interacts with the content server, which means that panoramas are downloaded and provided to the user for annotation management. After creating, modifying or deleting annotations, the panorama, annotations and corresponding meta-data (GPS position, label texts, etc.) are uploaded again to the content server.

After making the above described steps, users can explore their environment and download annotations created by all contributing users. The annotations are only downloaded if the current GPS position is within a certain range of the previous created annotations. This is to ensure that the difference between positions are not large enough to influence the matching process negatively. Although the system can tolerate a difference of a few meters, matching works better generally if the positions are nearly the same.

4.1 File and data structure of the panorama and annotation content

In Figure 4.2 we present our file and information structure which we use to store panoramas and annotations on the content server. To encapsulate our content into logical units we use ZIP files. The outer zip file is illustrated by the blue area which contains the panorama image in PNG file format and another ZIP file with the annotations. This is because our content server is able to unzip files automatically and can therefore provide only the inner ZIP file containing all annotations.

We decided to design the file structure this way, because we need the panorama for manipulation via the web-interface (see section 4.3). Furthermore we want to group the panorama images with their according annotations to logical units so that it is clear which annotation is belonging to the current panorama image. Another reason for modeling this type of information structure is because our AR client application is able to download just the annotation ZIP container. This has the big advantage, that only a few kilobytes instead of some megabytes must be transfered from the content server to the AR client running on the smartphone using a 3G network. This makes the loading process much faster which takes about 2-3 seconds on average.

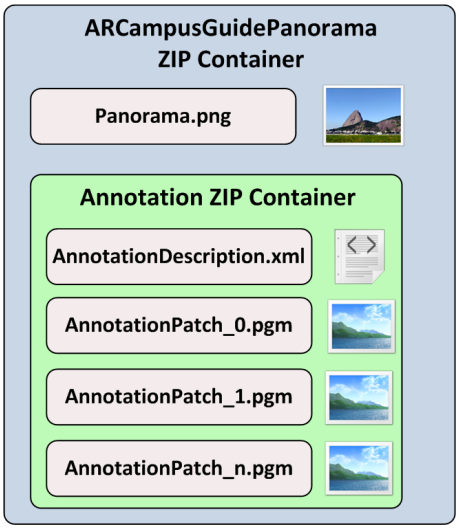

Figure 4.2: Schematic diagram of the ARCampusGuidePanorama containers which are stored together with their GPS address on our content server

As one can see in the green area of Figure 4.2, the inner ZIP file contains the individual annotation patches, which are stored as PGM files, and a XML file which contains meta information about the annotations. The PGM files are grayscale image templates which are used for annotation matching as explained in section 3.2. Figure 4.3 illustrates a minimal example of an AnnotationDescription.xml file which gives detailed information about one annotation.

```
<Annotations>
  <Annotation>
    <Id>0</Id>
    <Filename>AnnotationPatch_0.pgm</Filename>
    <Text>Institute for Computer Graphics and Vision</Text>
    <ScreenPosition x="256" y="350"/>
    <GPSPosition lat="47.05949465296363" lng="15.458569034393314"/>
    <CompassAngle>12.377539905009556</CompassAngle>
  </Annotation>
</Annotations>
```

Figure 4.3: Minimal example of a XML meta data file

This XML file contains meta information about the individual annotation including their ID, filename of the corresponding PGM image template or the label text. We also save the screen position of the annotations which is used by the web-interface for manipulation and display tasks. We do not use the screen position to display the annotation in the AR client annotation because this can lead to errors if the newly created panorama does not exactly match the old one. We determine the GPS position of each annotation with help of our web-interface and store the corresponding latitude and longitude values in our XML file for later use. Details about the linking process of annotations with their GPS address are presented in section 4.3.2. Knowing the GPS address of the panorama and the annotation, we can calculate a compass angle which we use for our system improvements presented in chapter 5.

4.2 Client-side handling of panorama and annotation content

Figure 4.4 shows the client interface of our AR information system. Users are able to create and save panoramas or load annotations from the content-server via the load/save buttons

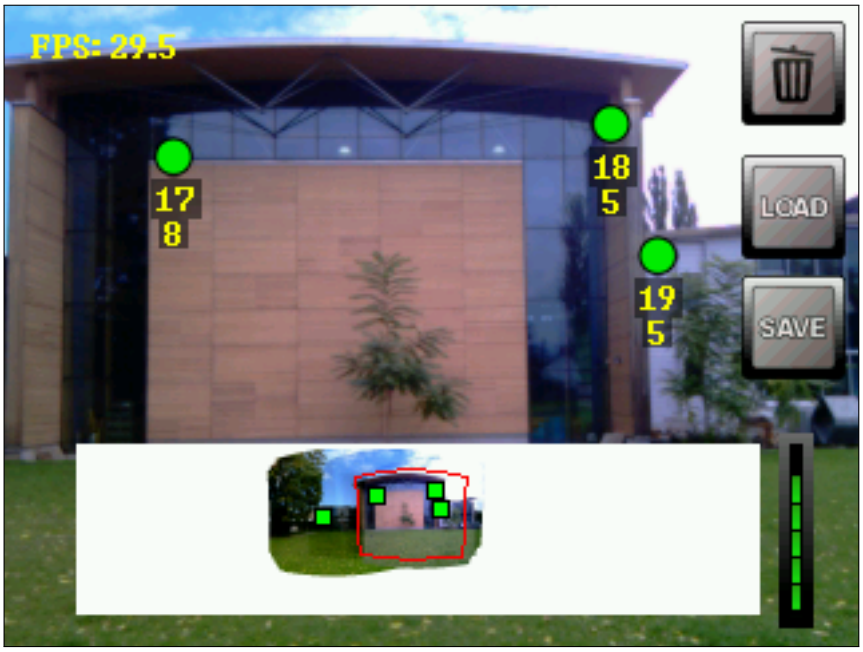

Figure 4.4: Snapshot of the client application running on current smartphones in real-time

which can be triggered by the touchscreen of current smartphones. This corresponds to the communication illustrated by the blue client parts of the schematic diagram shown in Figure 4.1. The transfer of big images over a 3G network used by mobile phones can be slow. In our case, uploading a panorama with one megabyte takes approximately 20-30 seconds. For this reason, we execute the up- and download in a separate thread. This has the advantage that users are able to continue working with the system while images are up- or downloaded in background.

In the lower area of Figure 4.4 one can see a preview of the panorama image which is created on the fly by rotating the smartphone. This gives the user feedback about unexplored regions and support to fill the panorama completely. If the "trash" button is pressed, the panorama image and the preview are cleared to allow users to start a new panorama creation process.

Next to the preview area we display a panel which shows the accuracy of the GPS signal. The higher the GPS indicator the better the signal received by the smartphone is. If the signal quality drops under a certain limit, the GPS indicator gets lower and the green indicator elements change to red, signaling to the user bad GPS accuracy. This is especially important when uploading panoramas or downloading annotations because if

the GPS signal is inaccurate it can cause either wrong annotations to be downloaded or panorama images to be uploaded with the wrong location data.

However, a wrong position of the panorama image can be corrected easily by our web-interface, as we describe in section 4.3.1. If wrong annotations are downloaded because of an inaccurate GPS signal, the download can be repeated when the signal gets better. Furthermore, it is possible to download annotations from several panoramas within a certain area instead of downloading only one panorama container. The number of panorama containers which should be downloaded at once can be set via a configuration file at the AR client.

4.2.1 Panorama upload

As described in detail in chapter 3, we create a panoramic map with a resolution of 2048x512 pixels. We save this map as a PNG image and send it to the content server. This image is used later on by the web-interface to allow users to create annotations. To keep the entries of the content server consistent, we create a ZIP file with the structure described in 4.1. This means that we also create and include an Annotations.zip file which contains an empty AnnotationDescription.xml file if users have not made any annotations directly on the smartphone.

If there are annotations, they are included as PGM image patches and corresponding meta information in the XML file but without GPS positions corresponding to the annotations. Without this GPS information, the original system works as described in chapter 3 but the improvements using the compass data of the smartphone is not available. The missing GPS information can be added easily later on by the web-interface as we describe in section 4.3.2.

4.2.2 Annotation download

To download annotations, we query our content server by the AR client running on the smartphone. The following steps are needed to download and use annotations on the smartphone client:

1. Send HTTP request containing the current GPS position
2. Query panorama IDs sorted by their distance to the current GPS position
3. Determine the file IDs of the Annotations.zip files of the nearest panoramas

4. Download and save the nearest Annotations.zip files on the smartphone

5. Unzip all Annotations.zip files and integrate them into the AR client system

As our content server supports unzipping files, we use this mechanism to download only the inner Annotations.zip file as one can see in Figure 4.2. This can be done by determining the concrete file ID for the Annotations.zip file as we describe in point three. More precisely we parse the answer of the content server after querying the panoramas related to their distance of the current position getting the needed file ID as result.

This approach allows querying the annotations only and avoids downloading the complete ZIP container which also includes the Panorama.png file which can have a size up to several megabytes. For this reason, the download size decreases to e few kilobytes, making the transfer over a smartphone's 3G network much faster. We need 2-3 seconds on average for downloading 5-7 annotations. As already mentioned, we execute the downloading process in an individual thread running in background, allowing the users to create a panorama map which is used for annotation matching in the meantime.

4.3 Web-interface for panorama and annotation management

One of our long term goals is to provide our system to a broad user base in an AR 2.0 approach where people are able to participate in the content creation process. We therefore designed and developed a prototype web-interface which enables users to create and manipulate AR annotations. The web-interface offers the possibility to interact in a simple and straight forward way with our content server without or with little previous knowledge. One reason to use a web-interface is because it is accessible with standard web technologies from all over the world without the requirement of downloading or installing any additional software. This way, we can provide our content to users who do not have access to current smartphones. Users are also able to view the uploaded panoramas via a web browser and create or manipulate annotations. Despite users are not able to view annotations in place with this method they are still able to contribute content.

Another reason is that it is inconvenient to to make all annotation creation and manipulation tasks directly on the smartphone because of the small displays and low resolution of current smartphone screens. Nevertheless, this would be an alternative possibility if users don't have access to desktop computers or notebooks with internet connection at

the moment. The only feature which is not supported currently on the AR client is the possibility to connect an annotation with its corresponding GPS position because we think this task can be done simpler and faster with a web-interface approach.

Using a web-interface for manipulating data stored on the content server also has the advantage that it can be presented to the user in a much more convenient way. As the separate data entries are stored in lists they can get very unclear if content starts to grow. Instead of browsing all entries in lists, we present them by visual markers which we insert at their corresponding GPS position in a map. We use the GoogleMaps API [12] for Flash which provides a lot of functionality regarding the visual presentation of digital maps, navigation and manipulation tasks and many more. We decided to take advantage of this API because it enables rapid prototyping as standard functionality is already provided. Furthermore, GoogleMaps and the interaction with it is well known among internet users which has the advantage that they need not acquire additional knowledge about using digital maps.

Before making our decision on which programming language we should choose for developing the web-interface, we evaluated the requirements of our system and the possibilities provided by different languages. One requirement was to use our already existing content server infrastructure. For this reason, our alternatives were quite restricted and server-side scripting languages did not come into consideration. The advantage of using an existing content server is to avoid maintenance problems as it is also used in other projects. We therefore decided to leave the server-side code completely unaffected.

The restriction of using client-side scripting, in the end, only led to making the decision between JavaScript and ActionScript. Adobe ActionScript 3.0 provides special functionality concerning image manipulation tasks which we need during annotation creation and we decided to use ActionScript for this reason. The code of our web-interface is therefore based on examples from the Adobe ActionScript API documentation [2] and tutorials of an Actionscript book [35]. Although the same functionality is provided with the HTML5 standard in conjunction with JavaScript, this standard was not implemented in current web browsers at the time of the web-interface development and was therefore in contrast to the wide spread of Adobe's Flash Player browser plugin.

In the following sections, we describe the particular functionality of the web-interface and how it supports the tasks of content creation.

4.3.1 Selection of panorama images

As one can see in Figure 4.5 the web-interface is divided into three main areas. On top, the menu bar which offers different functionality including annotation upload or panorama queries which we explain in more detail in the corresponding sections. Below the menu bar, we display the panorama image which is created on the AR client using a current smartphone with integrated camera. At the bottom we insert a GoogleMaps window which is responsible e.g. for selecting different panoramas and annotations.

Panorama selection: All panoramas within the GoogleMaps window are queried by a button click from the content server and displayed as blue markers with a "P" letter. This avoids querying redundant data and visual cluttering. By clicking one panorama

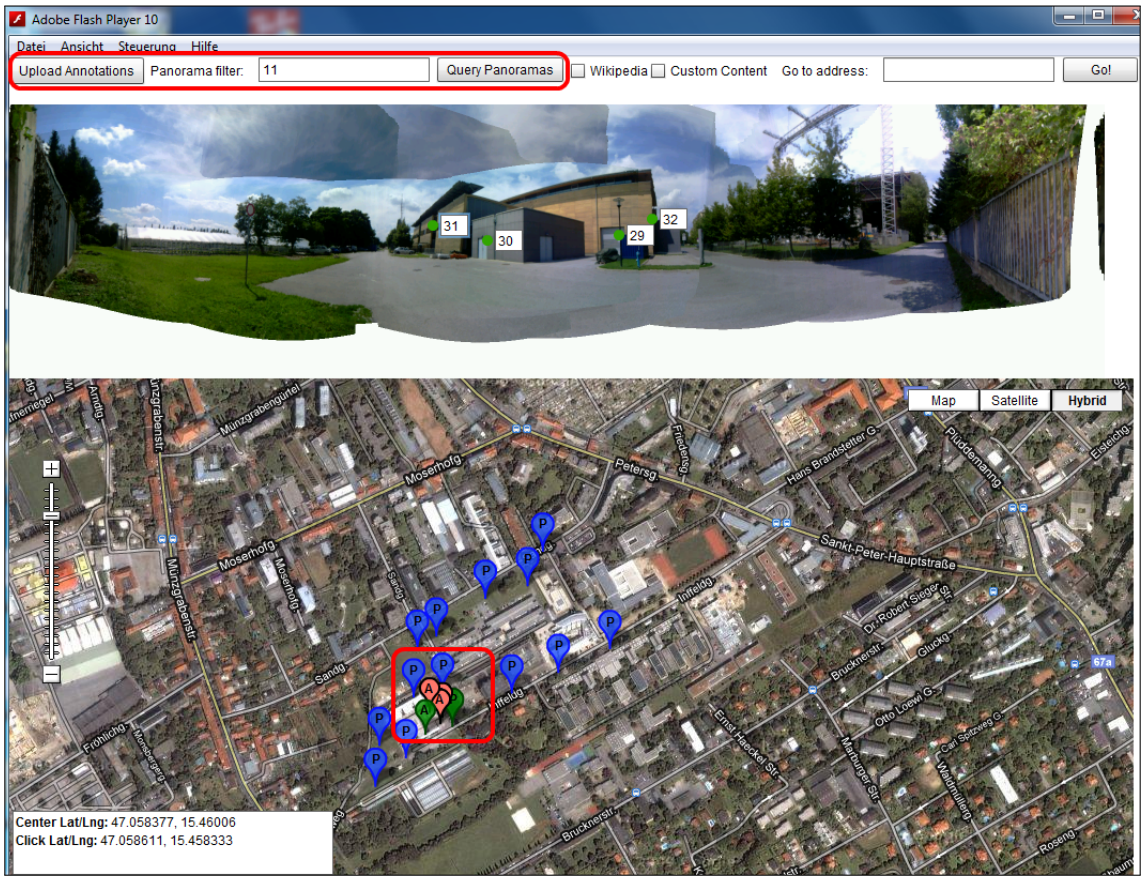

Figure 4.5: Web-interface which we use to interact with the content server. Panorama images can be selected, downloaded and displayed in a simple way using markers of the GoogleMaps window.

marker, the according panorama image is requested from the content server and displayed on top of the GoogleMaps window. If one panorama is selected, the color of the marker changes to green to visualize the current selected image, as one can see in the red marked area of Figure 4.5. Therefore it is clear at every moment which panorama is modified.

Changing panorama positions: We allow users to adjust the panorama position via our web-interface as GPS can be inaccurate and some meters off the real position where the panorama image is taken. Instead of looking up the correct entry on the content server and changing the GPS position manually, one can simply drag and drop the panorama marker to it's correct position. The position change is updated immediately in the database of the content server.

Annotation selection: Besides the panorama markers, we use red annotation markers which are labeled by an "A" letter. If such an annotation marker is clicked, it also changes its color to green showing the user which annotation is active. Furthermore we give the focus to the corresponding text field which we use for labeling annotations. A detailed description about the annotation creation and modification process follows in the next section.

Selection of different panorama sets: Spohrer envisions in his work [25] about WorldBoard different WorldBoard channels to provide users with contextual information. For example, if one person is looking for a restaurant in his current proximity, the system could display only restaurant guides with menus or ratings of other restaurant customers. This avoids that too much information is displayed at a moment which could confuse users or draw their attention to unimportant information. For this reason, we demonstrate a similar system where textual IDs can be used for information filtering. As one can see in the upper red marked area in Figure 4.5, users can enter a panorama filter before hitting the "Query Panorama" button. Afterwards all panoramas with a matching ID are displayed by panorama markers within the GoolgeMaps window.

We used this approach in our work mainly for debugging purposes to test for example different lighting settings. The ID can be set on the AR client via a configuration file which is applied to all created panoramas. After uploading these panoramas, they are stored together with the ID entered on the client side. We created panorama sets with different IDs at the same location but at different times of a day and on different days during half a year. Afterwards we tested these panorama sets with our system and the

improvements to determine the performance gain of our solutions.

In the long run the ID system can be used similar to Spohrer's vision of presenting users only relevant information. It could also be used to create or retrieve content only for those individuals or user groups who are subscribed to a channel or ID, similar to current social networks. This channels could be password protected to ensure privacy or traceability of content sources.

4.3.2 Creation and management of annotations

To keep the burden of annotation creation low, we focus on a fast and simple way to create textual annotations describing objects in the real world. This is possible with our web-interface after selecting a panorama as already described in the previous section. After the selection step, the panorama which should be annotated shows up below the menu bar as one can see in Figure 4.6.

Annotation creation: If a user wants to create an annotation, they simply have to click at the object in the panorama image. Afterwards we create an editable text field and set the focus to this field automatically so that a description can be entered directly. After this step, the annotation itself is created but not yet linked to its according GPS position. To finish the creation process by linking the annotation to the corresponding GPS address, the user has only to drag and drop the annotation into the GoogleMaps window as we show in the read marked area of Figure 4.6. After releasing the mouse button, we create a red annotation marker and read the longitude and latitude values at the corresponding mouse position. These values are then assigned and stored together with the annotation.

Although we find that it is much easier to link the GPS position to the annotations right after creating them, it is also possible to make this step at any time. It is therefore possible to create the annotations directly on the smartphone with the integrated software keyboard doing the GPS position linking step later via the web-interface. If a GPS position was made on the wrong position, it is also possible to change it by another drag and drop step later on. We always overwrite the previous position with the new one.

To avoid visual cluttering in case of many annotations are created in one panorama image, we adjust the size of the text field dynamically to the number of letters entered. Furthermore we restrict the maximum horizontal size of a text field allowing users to scroll vertically if the description text of an annotation is longer than the maximum text field size.

Annotation deletion: To delete annotations, they must be selected either by clicking an annotation marker in the GoolgeMaps window or into the corresponding text field in the panorama image. Pressing Ctrl-Del deletes the annotation and their corresponding text field and annotation marker. We use this shortcut to avoid accidental deletion of annotations. The missing possibility of changing the annotation position in the panorama image can be by-passed by deleting and recreating the annotation to achieve the same effect. Enabling users to change annotation positions in the panorama images directly can be done in future work to enhance the usability of our web-interface further.

Uploading annotations to the content server: After making all creation, modification and deletion steps, the annotations can be uploaded for later use by clicking the "Upload Annotations" button highlighted in the upper area in Figure 4.5. Before we upload the annotations, we write the meta information containing label texts, GPS positions,

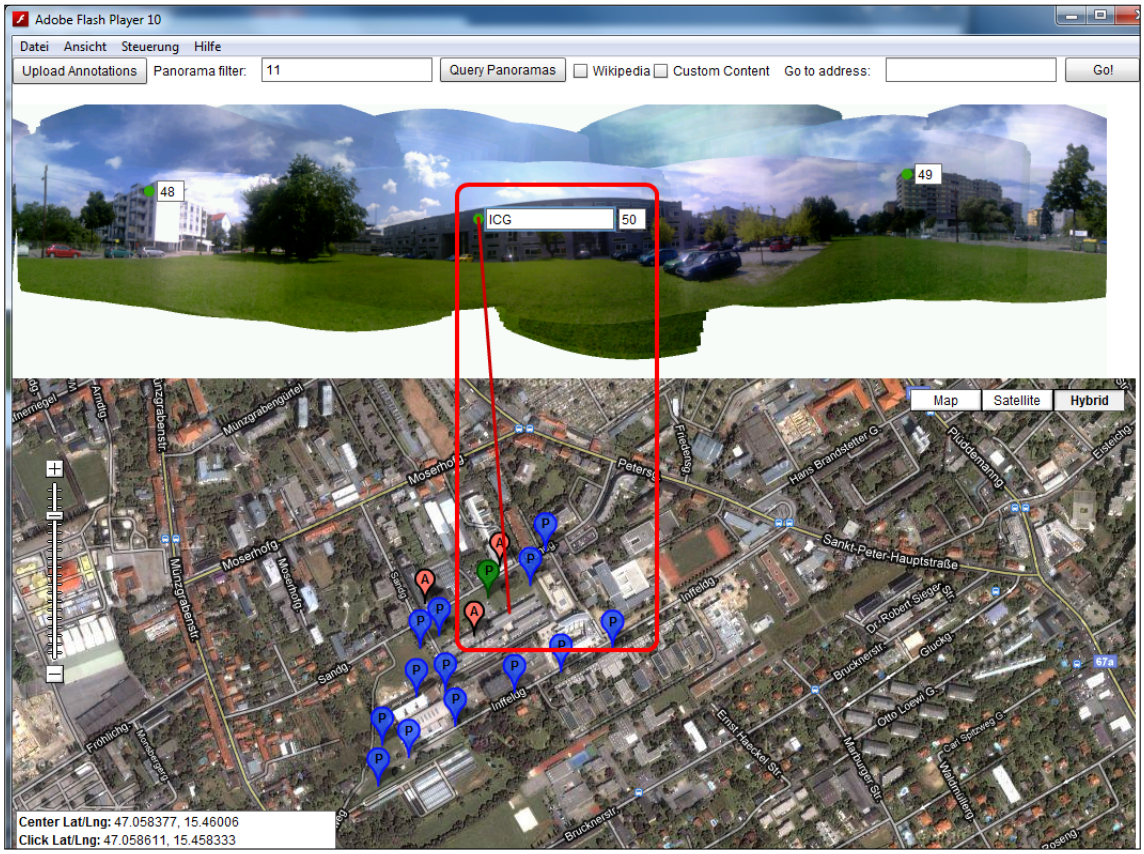

Figure 4.6: Screenshot of the web-interface which allows content creation and manipulation

etc. in the XML file as described in section 4.1. Furthermore, we extract image patches for every annotation with a size of 48x48 pixels which is used on the smartphone client for annotation matching. This image patch is centered at the green point of an annotation in the panorama image as one can see e.g. in Figure 4.6. After copying this image patch, we convert it into a grayscale image and save it in the PGM file format together with the XML as shown in Figure 4.2.

With the approach of creating annotations via a web-interface we do not create them completely online any longer. This has the limitation that a computer with internet access must be available to finish the annotation creation process. However, we find that this procedure is sufficient in many cases and is much more comfortable and faster than creating annotations directly on the smartphone. Nevertheless, there might some circumstances where online annotation creation is desired. We therefore could enhance the AR client to support the described functionality in future work.

4.3.3 Web-interface navigation

It is important to offer effective navigation possibilities to support users finding the correct position in the real world which is needed for linking annotations to their according GPS positions. We therefore use the well known navigation features of GoolgeMaps and integrate them into our web-interface.

GoolgeMaps geocoding: As we show in the upper right red area of Figure 4.7, we use the geocoding feature of GoogleMaps which is explained in detail in the GoogleMaps API [12]. The geocoding service translates a given address to its corresponding longitude and latitude values which we use for navigation. If a user enters an address in the "Go to address:" text field, we translate the first hit into longitude and latitude values and center the map view automatically to this position after the "Go!" button is hit. This functionality provides a very fast way to "jump" to any location in the map.

Map zooming: Another important feature is the zooming mechanism which is controlled by the panel on the left side of Figure 4.7. We also support zooming by mouse wheel because it is often faster and more comfortable than clicking the zoom bar several times. While low zooming levels give a good overview of the existing panoramas in a certain area, high zooming levels are essential for the annotation creation process. From this it follows that the accuracy of the GPS positions linked to annotations is directly

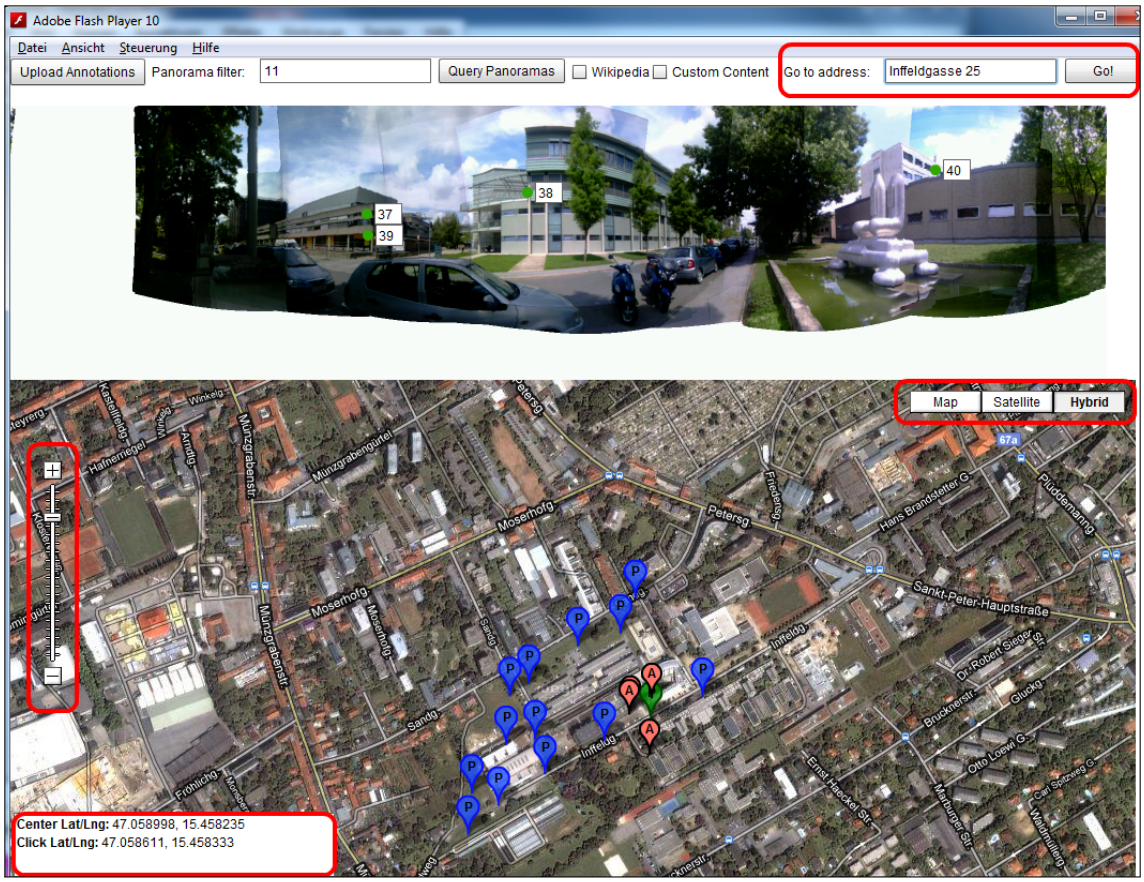

Figure 4.7: We use navigation techniques which are well-known by GoogleMaps users to keep the user burden for working with our system low

related to the highest possible zoom level.

Map views: We also offer different map views to support users in navigating to the locations they are looking for. The "Map" view is suitable for rough orientation tasks on a low zoom level because in this view, only streets, parks, etc. and their according names are displayed, but it is difficult to find individual buildings or small objects. This can be achieved by the "Satellite" view, although we find it is easier to use the "Hybrid View" which combines the satellite images with street names and other labels.

GPS position output window: In the lower left corner of the GoogleMaps window (see Figure 4.7) we integrate an output window which displays the longitude and latitude values of the map center and the current mouse position if the map is clicked. This feature is very helpful for debugging purposes where accurate GPS positions should be determined.

In future releases this feature can be discarded because it adds no further value for end users.

While developing the web-interface, we focused on usability for casual users with little or no previous knowledge about AR annotations. We therefore try to keep the interaction mechanisms as simple as possible. Especially for the navigation tasks, we use features provided by GoogleMaps which are well known by many users already, including geocoding, zooming and selecting different map views. This approach should lower the user burden for using our system and we therefore hope to make one step towards the AR2.0 vision.

4.3.4 Custom content and Wikipedia annotations

One promising approach in content generation for AR applications is using so called Mashups. In our context we mean combining already existing content sources with our application. One example is shown in Figure 4.8. The white markers in the GoogleMaps window are content units provided by a free geographical database called GeoNames [11] which offers geo-coded Wikipedia entries among other things.

Wikipedia annotations: If the Wikipedia checkbox in the menubar is activated, we query all existing entries from the GeoNames webservice which are within the map area. Afterwards these entries can be used to create annotations automatically. The only thing one must do is selecting a panorama by clicking a blue "P" marker in the map and then drag and drop a "W" marker into the panorama image. This is exactly the opposite process as we described in section 4.3.2. After releasing the mouse button within the panorama image, an annotation is created and labeled automatically with the title of the Wikipedia entry. We also could add the complete description to the annotation but we found that such big amounts of text are confusing if used together with our small text labels. We could enhance the textfields in future releases to better fit large amounts of data or develop an alternative way of presenting texts, e.g. by tooltips which show up on mouse over.

This approach of using already existing content for annotation creation has the advantage of avoiding the manual input of textual descriptions of objects which should be annotated and therefore is a fast and simple way to create AR content. Another advantage of using this kind of content creation is that the GPS position is already known. This means that we can directly transfer it into our system for later use.

Custom content annotations: However, using geo-tagged Wikipedia entries is only one possibility of content creation for AR systems. Another one is using user defined content which already exists in some way, for example in Keyhole Markup Language (KML) files. KML uses a similar structure as XML to describe objects but with the extension of coordinates which refer to a place in the real world. Schmalstieg et al. [23] suggest using a more open format, as KML is restricted to GoogleMaps, which they call Augmented Reality Markup Language (ARML) which is based on KML and also describes geo-tagged objects.

The above described approaches are first steps in the direction of AR2.0 and offer a great possibility for users to create AR content in a simple and fast way. This is because already existing content which is provided by webservices, databases or special files, e.g. KML or ARML can be re-used with little effort. Although we have not implemented

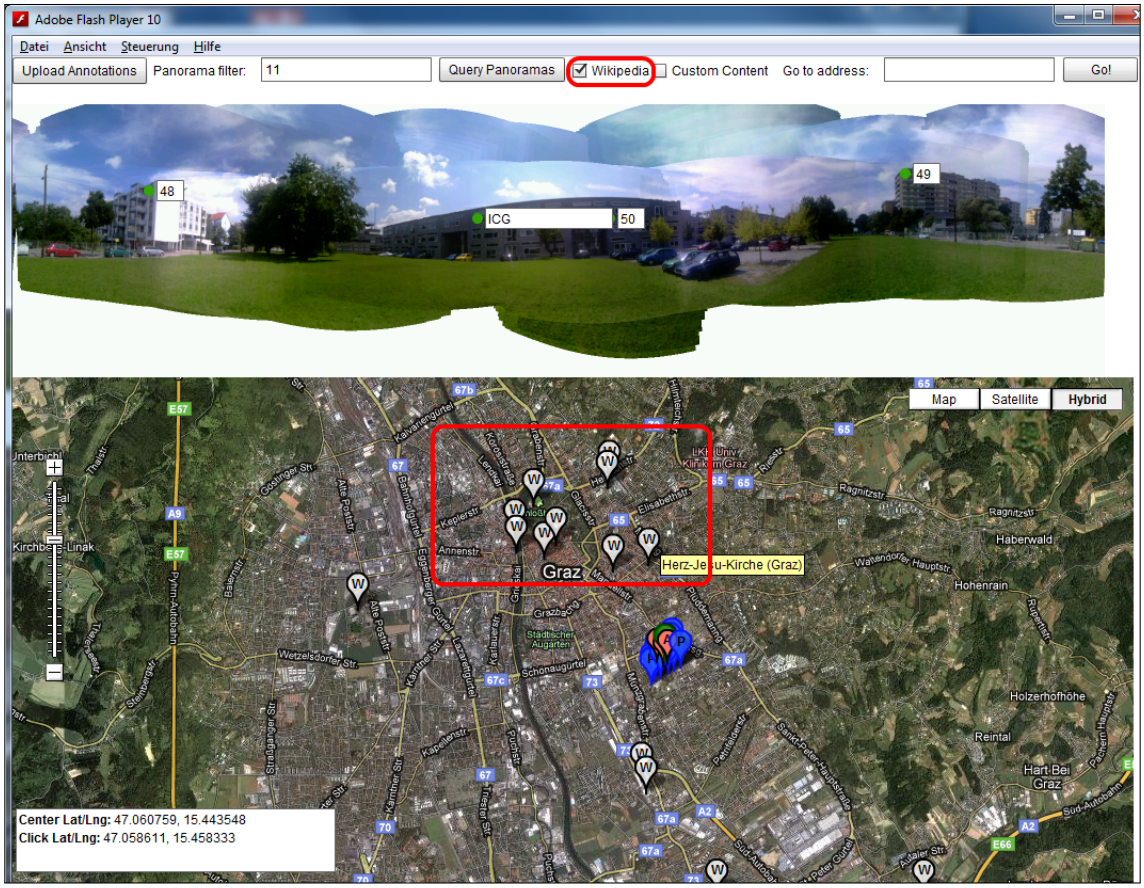

Figure 4.8: Example of a MashUp using geo-tagged Wikipedia entries which are provided by an already existing webservice [11]

KML or ARML support in the current version, it would be easy to extend our system by incorporating this feature. The only thing which has to be done is parsing the KML or ARML files and integrate the gained information in our existing system and file structure.

Chapter 5

Improvements to annotation matching

The system of Langlotz et al. [18] has some limitations concerning the annotation matching performance. The matching rates can drop drastically, if the lighting settings in the outdoor environment change throughout the day or if weather changes between annotation creation and annotation matching. This is because they use a vision-based approach for annotation matching which compares the stored image templates of the annotations with the cells from the currently created panorama map.

If a panorama and its annotations are created e.g. in the morning and the matching process is done in the afternoon, the lighting of buildings etc. can be very different. In such cases it is possible that the transition between light and dark areas is completely oppositional between the panoramas which leads to matching difficulties. Furthermore, shadows on buildings are cast from different directions depending on the position of the sun. A more detailed description about the situations in which annotation matching is likely to fail can be found in the work of Langlotz et al. [18]. To cope with these difficulties we investigate and present three different techniques in the following sections which should provide solutions to the matching problem under difficult lighting settings.

5.1 Combination of vision-based annotation matching and compass data

As we described in section 4.3.2, we create annotations via a web-interface and link them to their according GPS position with help of GoolgeMaps. In this section we explain how we take advantage of this information for improving the annotation matching quality.

Langlotz et al. [18] match every annotation against every map cell of the panorama. For this reason it is likely that some similar areas in the panorama image occur which can lead to wrong matching results if an area of the panorama gets a higher matching score than the correct one. Therefore they set the NCC threshold which is responsible for accepting or declining a correct match very high to avoid false positives.

In our work, we experiment with this matching threshold to obtain better results. As we mentioned already in section 3.2, we only accept a match as correct, if four out of nine image templates can be matched. Therefore, it is possible that a suggestion for an annotation match would be correct, but only two or three templates can be matched. This results in rejecting the annotation match because the threshold which is responsible for accepting a template is too high. We therefore suggest an approach which finds more correct annotation matches while keeping the number of false positives low.

One possibility for improving the matching result is to lower the threshold, which is set to 100 in the current system. By reducing this threshold it is possible to match more annotations correctly but if the threshold is reduced globally for the whole panorama, also false positives occur. We therefore only reduce it in the area where the annotation is likely to be located. We achieve this by using a built in digital compass which is available in some current smartphones, including the HTC HD2 which we use for development and testing purposes. Together with the GPS position of the stored panorama and the annotation positions, we use this compass data to estimate the position of the annotations in the newly created panorama.

In Figure 5.1 we give a visual explanation of how to calculate the angle between a panorama and an annotation in degrees where 0° and 360° means north, 90° east, 180° south and 270° west. For a better understanding, we overlay the image by a red circle which is centered at the panorama position. The circumference of the circle intersects with the annotation from which we want to calculate the angle.

With the two GPS positions lon_1 and lat_1 of the panorama and lon_2, lat_2 of the current annotation we can calculate the exact angle between north and the annotation

by the formulas of Vincenty [28]. Equation (5.1) calculates the difference λ between the longitude positions of the panorama and the annotation.

$$\lambda = lon_1 - lon_2 \tag{5.1}$$

We can use λ in equation (5.2) to determine α,

$$\alpha = -\arctan 2(\sin(\lambda) * \cos(lat_2), \cos(lat_1) * \sin(lat_2) - \sin(lat_1) * \cos(lat_2) * \cos(\lambda)) \tag{5.2}$$

the angle between north direction and the current annotation.

After placing an annotation marker in the GoogleMaps window, we calculate the angle as described above. We then save it to the AnnotationDescription.xml meta-data file to use it on the AR client. If annotations are downloaded to the client, we parse the meta-data XML file and use the compass angle to add the individual annotations to the panorama preview.

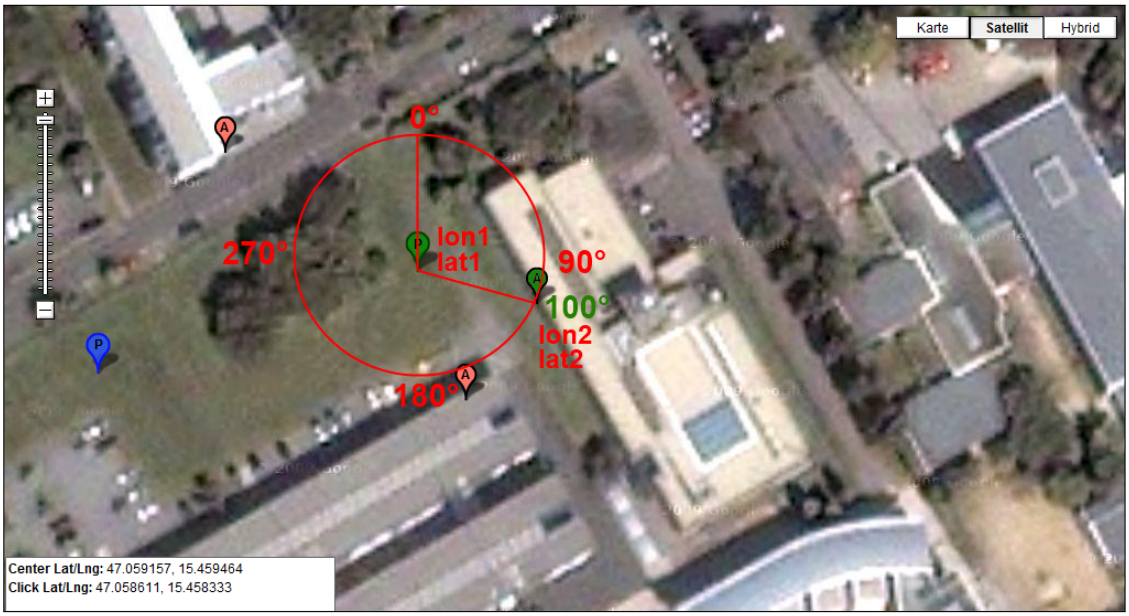

Figure 5.1: This Figure shows a schematic description of how we calculate the angle between a panorama an its annotations. The panorama can be seen as center of a circle, whereas the circle border intersects the annotation. With the formula by Vincenty [28] we calculate the angle between north (0°) and the annotation (100° in this example).

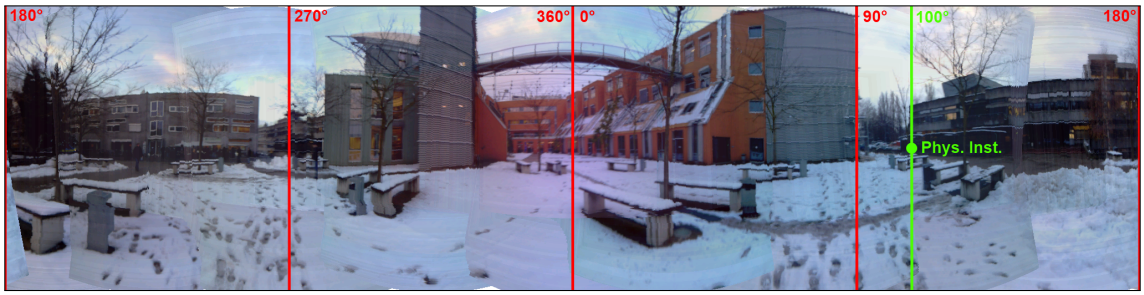

Figure 5.2: This Figure shows a schematic visualization of where we display annotations in our panorama preview. If the user is looking exactly into north direction it is sufficient to display the annotation at the position in the 2D map which corresponds to the angle between panorama and annotation position.

Before displaying the annotations in the preview, we align the angles to the current orientation of the smartphone, which is again determined by the digital compass. One can think about this process as we outline in Figure 5.2. In this Figure we assume north is in exactly the middle of the panorama map for an easier understanding. If this would be the case, we could display the annotations directly in the panorama preview using the compass angles which we calculated during annotation creation. In our example, it would be sufficient to display our annotation 100° right of the panorama center.

It is not often the case that a user looks directly north, but is orientated randomly, looking towards interesting objects or building of which he wants to get virtual information. To predict the annotations at the right position in the panorama preview, we have to shift the alignment according to the current compass data. This means that if we are looking in south-east direction our panorama center is located at about 130°. With this information, we can calculate an offset which can be added to the panorama center. In the example of Figure 5.3 we have an annotation at 100°. Next we subtract the current orientation of the user by the angle of the annotation. The offset is therefore 100° - 130° which results in -30° which we add to the center of the panorama map at 130°. To display the annotation at the correct position, we convert the degrees into pixels. As already mentioned, our panorama map has a width of 2048 pixels. We can calculate the degrees per pixel by dividing 2048px/360° which means that one pixel corresponds to 0.176°.

If we want to place our annotation in our panorama preview, we assume the map center at the pixel 1024 and add an offset of (-30°/360°)*2048px which is about -171 pixels. The position where we place our annotation in the preview step is calculated therefore by 1024-171 and results at pixel 853. We then set the position in the 2D panorama map at x=853

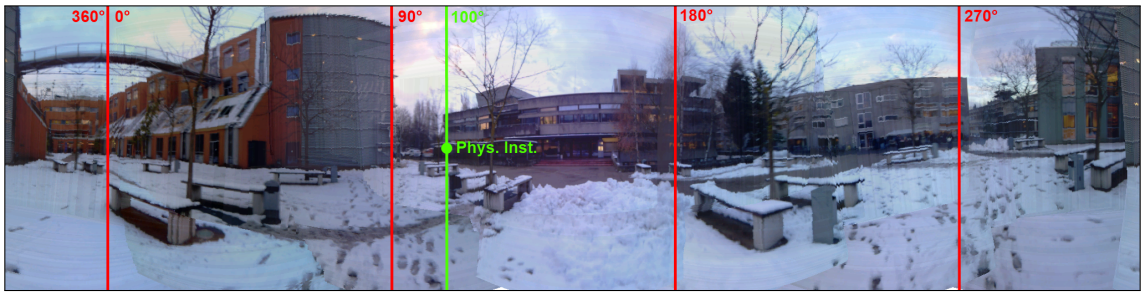

Figure 5.3: This illustration is comparable to Figure 5.2 with the difference that the user was looking in another direction, e.g. south-east which corresponds to 130°, when he started the panorama creation process. The user direction of 130° is measured by the built in digital compass and is used to calculate an offset between the current direction and the annotation angle which is used to align the annotations at their according positions in the preview.

and y=256. We use the center in y-direction because we have no information about the height of the annotation in the image.

This position is used as a first guess to improve the annotation matching process and to support the user in finding annotations. Furthermore, this approach has the advantage that a rough estimation of the annotation position is displayed as a fallback solution. This works similar to Wikitude [20] or Layar [19] which use a compass to display the annotations. Although we also use the compass information as a fallback if matching fails completely, the primary application is to localize the area of the panorama map where a specific annotation is likely to be located. We then set the matching threshold in the determined cell and in its neighbor cells to a less restrictive value. This way, we avoid a lower threshold in all other areas of the panorama and therefore minimize the risk of gaining a false positive which could occur if there are many similar areas in the panorama.

During the development of this improvement we tested several different thresholds, starting with 50 (half of the original one) and increased the threshold by steps of 10. We met our goal of finding a good trade off between finding more correct annotation matches and avoiding more false positives at a threshold of 70. Lower thresholds tend to cause many false positives which decreases the matching performance of the original implementation, whereas higher ones do not find more correct matches.

To support the user in finding annotations, we overlay the cells with the lower threshold in the preview image by a transparent green image patch as one can see in Figure 5.4. The middle and right annotations could not be matched yet. We therefore display these unmatched annotations as red dots in the preview image surrounded by transparent

Figure 5.4: The green patches of the panorama map preview mark the areas where the annotations are likely to be located. We therefore set the annotation matching threshold to a lower value in these regions.

green areas. We only display the green areas if an map cell is filled completely. Therefore they are added successively while the user explores the environment around an annotation. This also supports users with the matching process, because they immediately see which cells are still unexplored and where the annotation should be located. If the annotation can be matched, we change the color in the preview image from red to green and do not display the compass area around the annotation any longer as one can see at the left annotation of Figure 5.4.

We discuss the impact of the compass improvement on the overall annotation matching performance and compare it to the other improvements in chapter 6.

5.2 Extended dynamic range for image improvements

The image quality of both, the annotation templates and the panorama is a crucial factor for the matching performance since we use a vision-based approach for matching the annotation templates against the newly created panoramic map. One problem for the

Figure 5.5: This image shows the artifacts which occur because of automatic exposure adjustment of current mobile phone cameras.

task of template matching is the automatic exposure adjustment of current smartphone cameras which often occurs in very bright scenes. We show an example in Figure 5.5. In this image one can see the color changes especially in areas surrounding the sun.

This effect occurs because we write the first camera frame directly into the panorama map when we start to create a new panorama. Afterwards we add new pixels which are in the camera frame but not yet in the map. This results in significant color changes between consecutive frames caused by the different exposure settings.

If annotations are placed directly on color transition lines this can cause problems in the matching process because these artifacts would not show up under different lighting settings. This leads to matching difficulties if either color artifacts occur in the panorama used for annotation creation or in the panorama used for annotation matching.

One solution to solve this problem would be to disable the automatic exposure adjustment of the camera and use a fixed one instead, but to the best of our knowledge there is no such possibility on current smartphones except the Nokia N900 which is used in the Frankencamera project [1]. We therefore decided to investigate the impact of image improvements by implementing Extended Dynamic Range (EDR) for the panorama map and the annotation templates. We show one example of our results in image 6.3 and explain our implementation in more detail below.

At the beginning of the panorama creation process, we write the first camera frame into the panorama map. Afterwards we use the color information of these pixels for further image improvement steps. The main idea of our EDR implementation is to compensate too drastic color changes which occur if the camera is pointed alternately towards bright and dark areas during panorama creation. We therefore use the keypoints of the current camera image and the panorama map which are already calculated for orientation tracking. This approach does not cause any additional effort for this reason.

Figure 5.6: The same panorama image as we show in Figure 5.5 but with EDR applied. Although there are still artifacts left, they are not as significant as without activated EDR.

After finding corresponding keypoints in both, the camera frame and the map, we calculate the average color difference per color channel. We add this difference to all pixels of each color channel of the camera frame before mapping them in the panorama. This approach leads to color values below 0 and above 255 which is the range of our 8-bit color buffers per channel. To avoid buffer under- or overflows we use 16-bit buffers per color channel instead and determine the minimum and maximum color value for each channel. With this information we can apply tone mapping before saving the panorama which converts the extended color range again to values within the 0 to 255 bounds. We use this approach when creating panoramas on the AR client which is only one part of our EDR implementation. We also apply EDR to all cells of the panorama which are tried to be matched against the annotation templates. Here, we also determine the minimum and maximum values of the grayscale image everytime a cell is filled completely. Afterwards we again apply tone mapping before creating the grayscale image patch of the cell. The following listings give an overview of the EDR approach.

To apply EDR on the panorama map we take the following steps:

1. Write first camera frame into the panorama map
2. Find corresponding keypoints between the camera frame and the panorama map
3. Calculate the average lighting difference between the keypoints
4. Add this lighting difference to the camera pixel values before mapping
5. Calculate the minimum and maximum color for every color channel (RGB)
6. Map pixels in a 16-bit buffer per color channel

7. Apply tone mapping before saving the panorama

Applying EDR for annotation patches:

1. Determine the minimum and maximum grayscale values for the current panorama cell

2. Apply tone mapping before creating the image patch

To explore the possibilities of EDR we evaluated our system with and without activated EDR and in combination with the other two improvements. We present and interpret the results in the chapter 6 about the system evaluation.

5.3 Robust annotation matching model

We use a vision-based approach to match every annotation against every finished cell of the currently created panorama map. For this reason it is likely that more than one position is found in the map which is a possible candidate for the correct match. The system therefore is over-determined which allows us to use the Random Sample Consensus (RANSAC) Algorithm to exploit this fact.

RANSAC is used where more samples are available than needed for e.g. in image analysis tasks where one goal is to find inliers or to remove outliers which would affect the result in a negative way. A detailed description about the algorithm can be found in the work of Fischler et al. [10]. In our case we use RANSAC to build a global model of our annotations and their relations between them.

We determine where annotations are located in a panorama saved on our content server and bring them in relation to the annotations which are downloaded and matched with the currently created panorama on the AR client. In Figure 5.7 one can see a schematic representation of these relations. At the bottom of the Figure, we show a panorama map which is annotated and stored on our content server. The green dots represent the annotation position with their according label which was created either directly on the smartphone or via our web-interface in a previous annotation creation step.

The image on top of Figure 5.7 shows a panorama map which is created on the AR client and used for annotation matching. As one can see, the two images have a slight rotational offset because the users looked in different directions when they started the panorama creation task. As already discussed, we map the first camera frame directly

Figure 5.7: This Figure shows a panorama with according annotations stored on our content server (bottom) and a new created panorama where the annotations are tried to be matched (top). For every annotation we store a maximum of three best matches. The green dots in the upper image have the best matchings scores and are therefore used for label placement. The red ones are the second and third best matches of an annotation which makes them suitable for a possible correct match.

into the panorama map and then add new pixels when they appear in the camera frame. For this reason we have to align the rotations of both panorama cylinders as one can see in Figure 5.8.

As one can see in the upper image of Figure 5.7 there are several possible annotation positions. This is because we store a maximum of three positions which have the highest matching scores and most matched templates with each annotation as indicated by the green and red dots. A green dot in the upper image means, that this point is the best match and therefore shows up as labeled annotation at the corresponding position at the display of the AR client. The red dots are the second or third best match but are not displayed as annotation on the smartphone at the moment. However, they are good candidates for building a RANSAC model which we describe in detail later. As one can see in Figure 5.7, the leftmost annotation of the upper image is correct, the other two annotations show an incorrect match, because the green dot is not on the same position as in the lower panorama map of the content server. We therefore apply a RANSAC calculation which should verify the annotation positions, or if a better model is found, correct the positions.

If many annotations are checked, the RANSAC calculation can be time consuming.

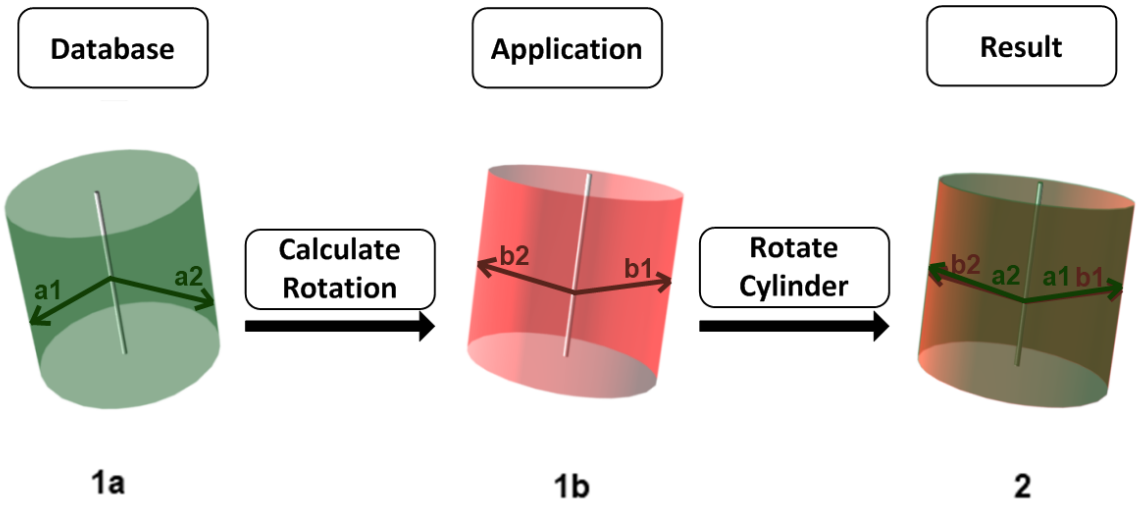

Figure 5.8: The left cylinder (1a) of this Figure corresponds to a panorama map stored on our content server. The two vectors of 1a and 1b are two annotation positions in the cylindrical panorama map. The middle cylinder (1b) belongs to a panorama which is currently created on the smartphone client. After rotating one cylinder into the other in order to align both vectors of each cylinder with a minimal error, we get a rotation as result which can be used in a RANSAC calculation to determine a model with a sufficient small error.

Therefore, we only execute it if there is enough time left in one frame to ensure real-time performance on current smartphones. One prerequisite of RANSAC is that the samples are chosen randomly. We therefore create an index list which contains combinations of all database annotations stored on the content server and the best matches of every annotation determined on the AR client. After building this index list, we choose the entries randomly for every RANSAC cycle.

Each index entry contains a quadruple of positions which belong to two different annotations which we choose randomly. We give one example in Figure 5.9. Here, the points *a1* and *b1* belong to the same annotation, once coming from the database of our content server (*a1*) and from the currently created panorama *b1* which we use for annotation matching. The same applies to *a2* and *b2*.

After selecting those four positions randomly we convert them into 3D vectors corresponding to a point in the cylindrical panorama map. The two vector pairs *a1, a2* and *b1, b2* are in two different cylinders and therefore in different coordinate systems which we align as we explain in Figure 5.8. We give a detailed description about the algorithm we use for the orientation calculation in appendix A.

Figure 5.9: We use the four positions *a1, b1, a2, b2* as input for the RANSAC algorithm. The points *a1* and *a2* are the position in the stored database panorama. The point *b1* belongs to one of the three best matches of the annotation *a1*, the same is true for *b2* and *a2*. These four points are converted to 3D vectors in the cylindrical map and are used for the rotation calculation as we show in Figure 5.8.

The result of this calculation is a rotation which aligns the two different cylinders with minimal error. We use this rotation as hypothesis for the RANSAC algorithm. In the next step we loop through all annotations of the database converting them from 2D map coordinates into 3D cylinder vectors. Afterwards we apply the hypothesis rotation to the 3D vectors and convert them back into 2D map coordinates. We then calculate the distance between these map positions and the current positions of the corresponding annotations in the panorama map of the AR client. If the distance is small enough, we count this annotation as inlier because it supports the current hypothesis and add the error to an error sum which is calculated while looping through all annotations giving an overall error of the currently investigated hypothesis.

After finishing the loop where we calculate the distance error sum, we check if the number of inliers is above 50 %. If the number of inliers is less than 50 %, we reject this hypothesis because it is too weak to support the assumption of a valid model. We count a point as inlier, if the distance error is below a certain threshold, which can be changed via a configuration file on the AR client. In our empirical tests we found that a maximum error of 10 pixels in x or y direction in the 2D map is a good value because this allows

that RANSAC is applied in many cases while the overall model error is relatively low. If the current hypothesis fulfills the inlier prerequisite, we divide the error sum through the number of inliers and check if this error is smaller than the best error determined in previous RANSAC steps. If it is smaller, we save this error together with the hypothesis rotation.

After each RANSAC loop cycle we take another index containing an annotation position quadruple as described above which was not used before. We use this data to calculate a new hypothesis rotation. Afterwards we check again, how many annotations support the hypothesis and save it in case of a smaller error. We continue with these steps until every index was checked or all RANSAC iterations are done.

If we find a hypothesis which has an error below our threshold, we apply this hypothesis rotation to all annotation positions of the database annotations. Afterwards we replace the positions in the AR client panorama by these calculated positions. As a result, we are able to display all annotations of the panorama map at their corresponding positions, even if some annotations could not be matched before. Although this approach is quite accurate one must mention that an error of a few pixels can occur because the hypotheses are approximations and not completely correct.

The following enumeration gives a compact overview regarding the individual steps we apply to build a RANSAC model which we then use for annotation position verification or correction:

1. If enough time is left, build RANSAC model
2. Create RANSAC index list out of the three best matches of every annotation on the client and the annotation positions of the stored panorama
3. Take RANSAC indices randomly until every index was used or all RANSAC iterations are done
4. Convert the four panorama map positions (two points of the saved annotations (DB) and two of the currently created panorama (APP)) determined by the indices to 3D vectors in the cylinder coordinate system
5. Compute the orientation of the two vector pairs as shown in Figure 5.8 (Rotates one cylinder into the other one so that the corresponding points are almost on the same position)
6. Use this rotation as RANSAC hypothesis

7. Loop through all annotations and convert their database map position into 3D vectors of the cylinder coordinate system
8. Apply the hypothesis rotation to every previously calculated 3D vector
9. Convert the rotated position into 2D map coordinates
10. Compare the distance between the calculated hypothesis position with the corresponding annotation position in the newly created panorama map
11. If the distance is below 10 pixels in x- and y- direction in the 2D map, the difference is added to the error sum
12. If the inlier ratio is greater than 50%, the error sum is divided through the number of inliers
13. Check if this error is smaller than the best one from previous hypotheses and store the new error together with the hypothesis rotation if better
14. Repeat steps 3 to 13 until all indices were used or all RANSAC iterations are done
15. RANSAC loop finished
16. If the best error is below 10 pixels in each, x- and y-direction in the 2D panorama map coordinate system, convert all annotation map positions to 3D vectors of the cylinder coordinate system and multiply them with the hypothesis rotation. Afterwards convert the 3D vectors to 2D map positions again, store the new positions and replace the old annotation positions in the panorama map created on the AR client.

As one can see, this system works if at least 50 % of inliers are found, and their error is below 10 pixels in 2D map coordinates. During our research and evaluation phase, we tested different error thresholds. We found that a pixel error of 10 is a good compromise between minimizing the overall error and the likelyhood of the applicability of the RANSAC approach. This error is introduced when we apply RANSAC to all annotations and the hypothesis model is not fully correct but only an approximation of the correct solution. The reason for this slight inaccuracy is the position offset between the two different panoramas used for annotation creation and annotation matching. DiVerdi et al. [7] give a detailed explanation about this behavior in their error analysis of their work about Envisor.

However, reducing the error threshold e.g. to 5 pixels leads to a smaller overall error, but the precondition for the RANSAC model also gets stricter. This leads to a smaller chance of applying RANSAC in cases where not many good matches are found. On the other hand, if we increase the error threshold e.g. to 20 pixels, it is possible that a false annotation relation model is applied and all annotations are displayed at incorrect positions. To avoid such a behavior we use an error threshold of 10 pixels which did not lead to incorrect models during our evaluation.

RANSAC works best in cases where many annotations are already found and does not provide any performance gain in cases where no or very few annotations can be matched. We also noticed this behavior in our evaluation where the matching rates did not increase drastically when RANSAC was applied in scenarios with generally poor matching rates. We achieved an annotation matching performance gain of approximately 18 % when we applied RANSAC to the current system which reaches an overall matching rate of about 40 % without RANSAC. In contrast, when we applied RANSAC to the system combined with our compass improvement which achieved a matching rate of 55 %, RANSAC was able to increase this matching performance by nearly 31 %. We present the detailed results and an explanation of our evaluation in the next chapter.

Chapter 6

System evaluation

To control whether our improvements led to the expected results, we decided to evaluate our system under different environment conditions. Langlotz et al. [18] achieve annotation matching rates of up to 90 % if the annotations are created directly before they are beeing matched. Although this is a good matching rate, this scenario does not occur very often in practical use, because in most cases users want to view annotations which are created days, weeks or month before by other users. This leads to a more difficult requirement for the system because different lighting and environmental conditions have an influence on the matching performance.

These changing conditions are problematic, because we use a vision-based approach for annotation matching. Langlotz et al. [18] explain in their work under which circumstances the matching is likely to fail. For example, if shadows on buildings change throughout the day which leads to a change in the grayscale annotation templates which we use for matching. Under such circumstances the matching rate drops to 56 % which is unsatisfactory for AR2.0 scenary in which non-experts use the system in their every day life.

We therefore formulated our goal of improving the matching rate under difficult and changing lighting situations up to 80 %. In the following text we explain how we arranged our system evaluation. The evaluation demonstrates the impact of the individual improvements and how different combinations affect the overall matching performance.

6.1 Evaluation setup

Although we made several field tests directly with a smartphone, we decided to create a test scenario in which we could compare individual improvements more objectively. In a field test with a smartphone we would have different environmental conditions in every evaluation cycle. Because lighting situations have a great effect on matching results and thus effect the results of the individual improvements, we determined our field tests were unsuitable as an objective test baseline.

We therefore decided to record videos at the panorama spots with a notebook computer and a web cam. Furthermore we use an InterSense InertiaCube3^{TM}[16] sensor as compass and write the compass data in a text file every frame. Afterwards we use these videos and the compass data as input for our AR client application on a desktop computer. With this approach we have the same conditions for every evaluation cycle and can therefore compare the individual improved components to each other in a more objective way.

In the first evaluation step, we created 12 panoramas at different positions around our campus area to ensure a diverse set of images and environment conditions. After creating the panoramas we uploaded them to our content server and annotated them via our web-interface. For each panorama we created 4-6 annotations, which led to 58 annotations in sum. We did this both for panorama images created with activated EDR and without EDR. Afterwards we used these annotations for our evaluation and tried to match them with the new created panoramas coming from the recorded video streams.

To test the matching performance under difficult lighting settings, we created the first panorama test set on a sunny day one hour before sunset and the second one on a different day about noon. This led to situations in which we had completely different shadows on parts of buildings which we wanted to match against annotation templates without these shadows. Furthermore we got lighting artifacts including lens flares and white blobs which where mapped directly into the panorama image making it very difficult to match annotations in such areas. We noticed that the matching performance strongly depends on the quality of the annotation templates for several reasons. As mentioned above, vision-based matching becomes impossible in incorrectly matched areas because of lighting artifacts. Furthermore an annotation template must contain a minimum number of good keypoints for the matching process. We therefore placed our annotations in areas which provide a good template quality e.g. near doors, windows or roofs of buildings.

Below we present the results of our evaluation and explain the impact and conclusions

of the gained outcome. Although we took the matching rate of 56 % from Langlotz et al. [18] as reference one cannot directly compare both evaluations because they gained their results from an early user study with eight users at only one panorama spot. In contrast, we evaluated 12 different panorama spots but with only one experienced user. For this reason, the same system only achieves a matching rate of about 40 % without any improvement instead of 56 %. We present the detailed matching results in Figure 6.1.

6.2 Evaluation results

What we can see in the following Figures 6.1 to 6.8 are the results of our evaluation. We show one diagram for every individual improvement, all possible combinations of improvements and one diagram without any improvements. The diagrams demonstrate a clear comparison between possible annotation matching performance gains. Every diagram contains the results of 12 different panoramas which are described by their panorama ID. The blue bars show the number of correctly matched annotations per panorama. The yellow bars mark the total number of annotations per panorama which is the maximum of possible matches. The difference between the yellow and blue bars are the annotations which could not be matched correctly at all. To maintain clarity, we do not show the number of unmatched annotations in the charts. Finally we present an overview of the individual evaluation results in Figure 6.9 in ascending order.

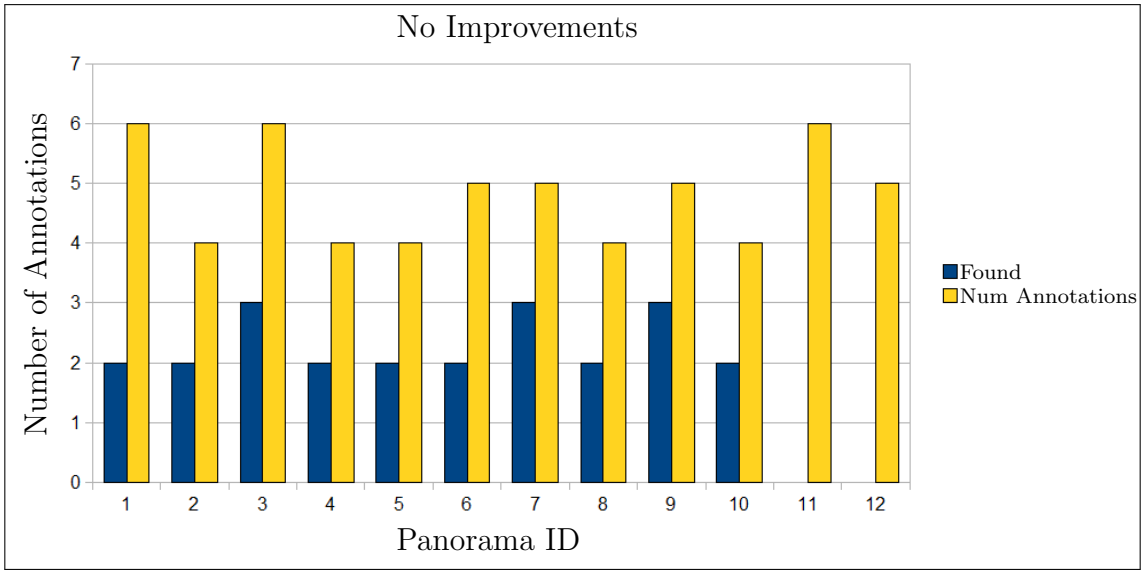

Figure 6.1: We present the matching results for every single panorama spot without any improvements in this Figure.

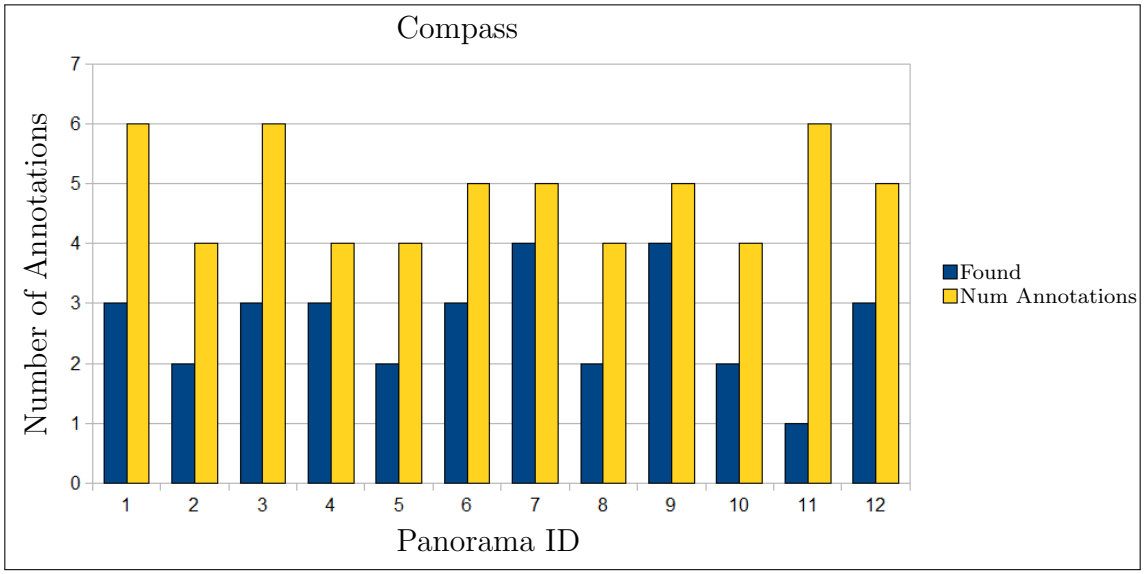

Figure 6.2: In this Figure we show the annotation matching performance gain achieved by a digital compass used for preselecting the annotation area.

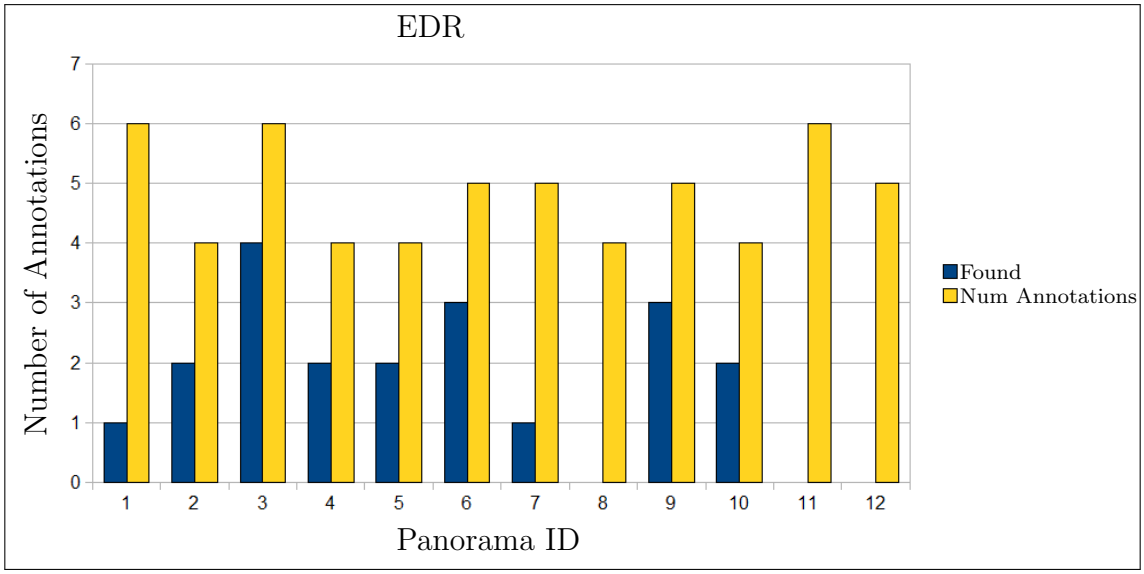

Figure 6.3: As one can see in this Figure, the matching rate with activated EDR is relatively low with 34,48 % correct matches.

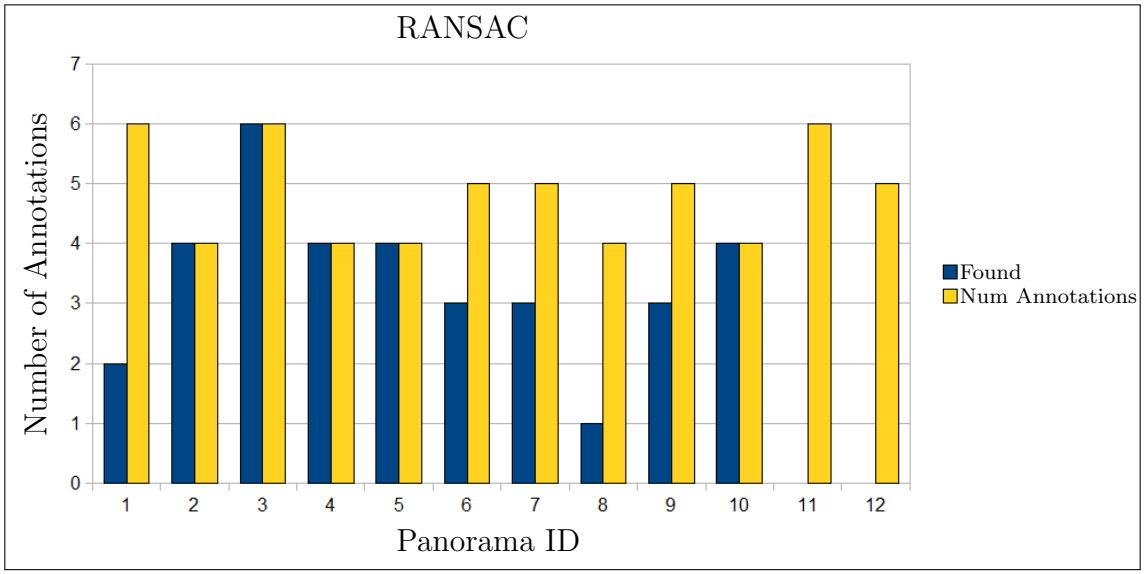

Figure 6.4: This Figure shows the performance of our RANSAC approach which achieves a matching rate of 58,62 %.

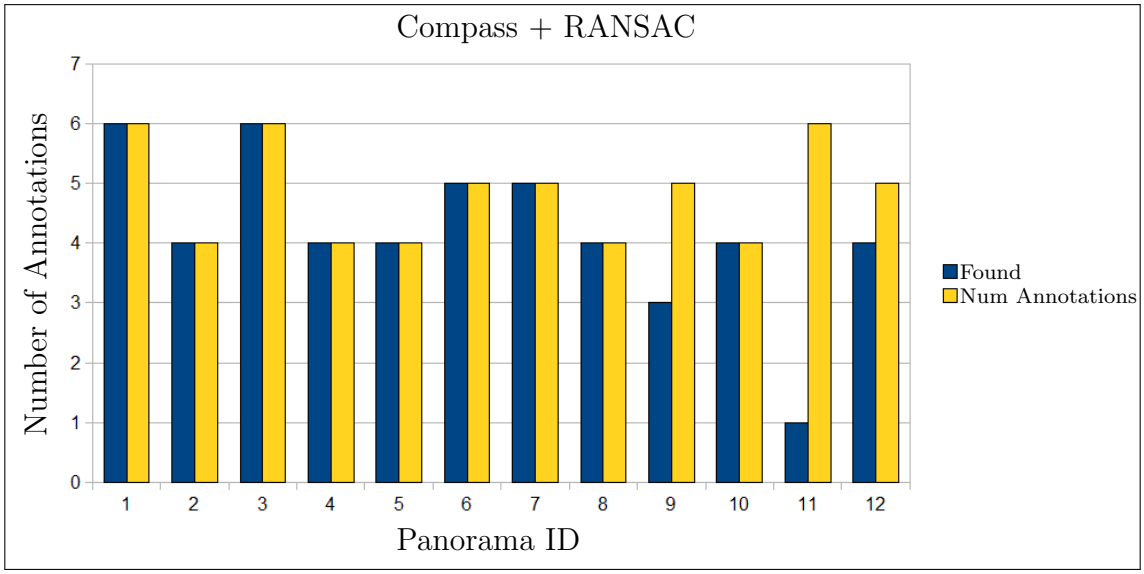

Figure 6.5: Using a digital compass and the RANSAC approach leads to a strong combination which takes advantage of both improvements.

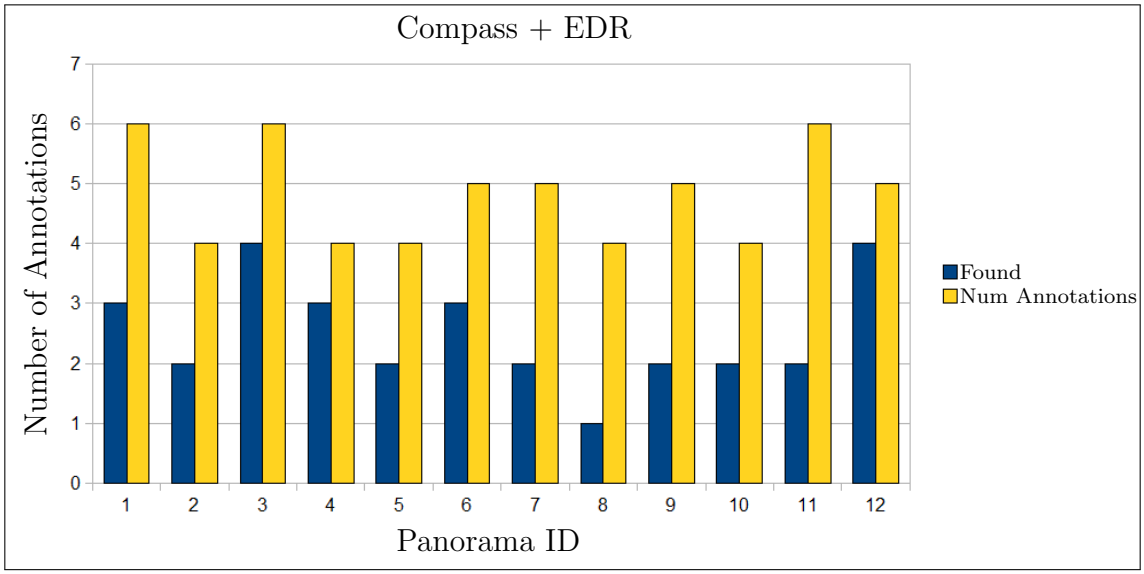

Figure 6.6: A combination of compass and EDR leads to a matching rate of 51,72 %.

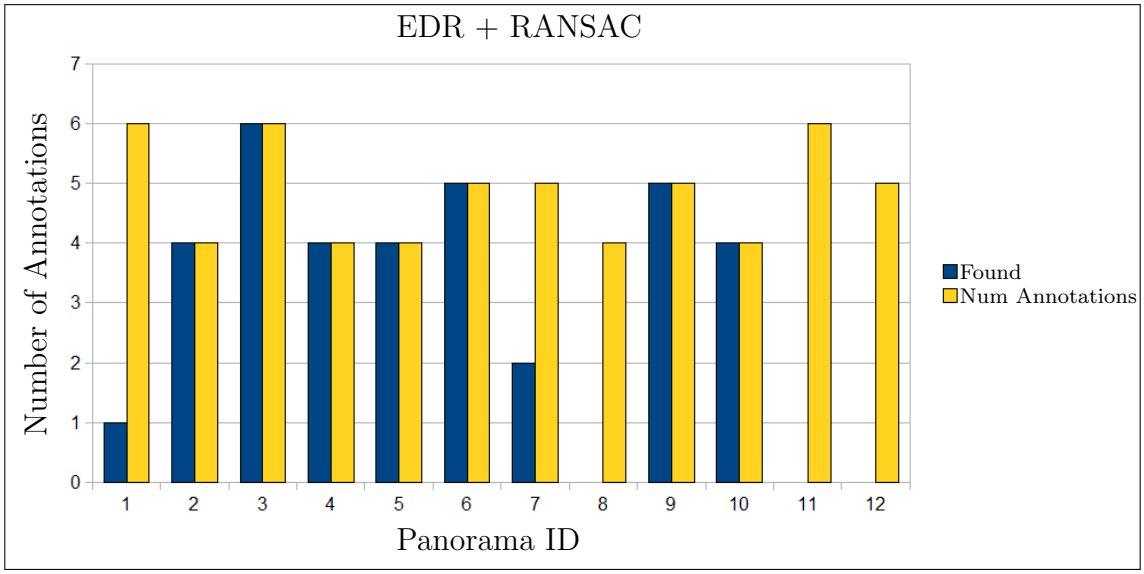

Figure 6.7: This Figure shows the combination of EDR with RANSAC and achieves a matching rate of 60,34 %.

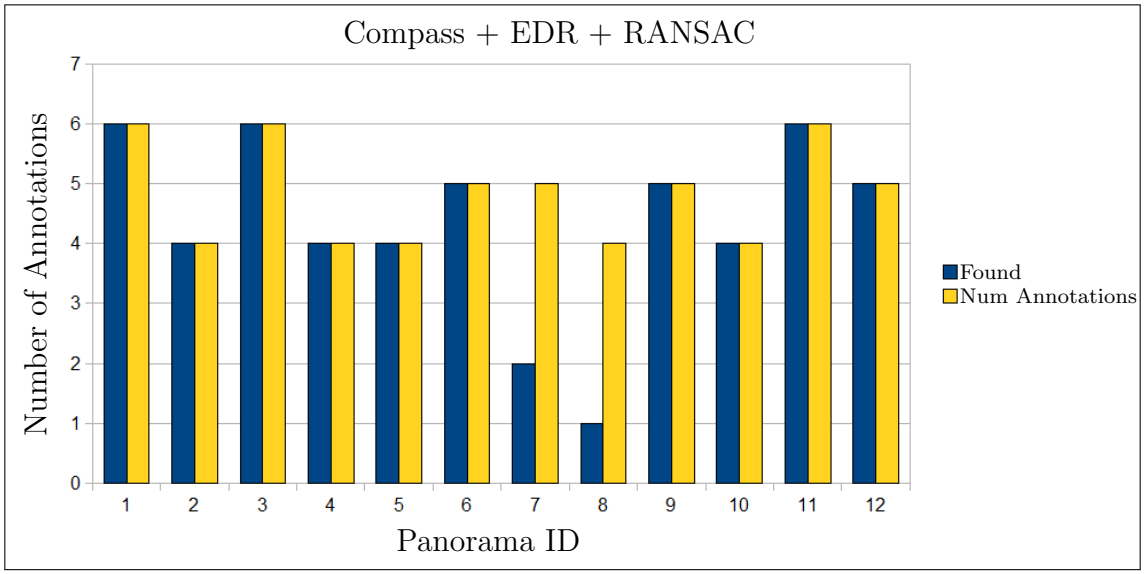

Figure 6.8: Combining compass, EDR and RANSAC results in a matching rate rate of 89,66 %.

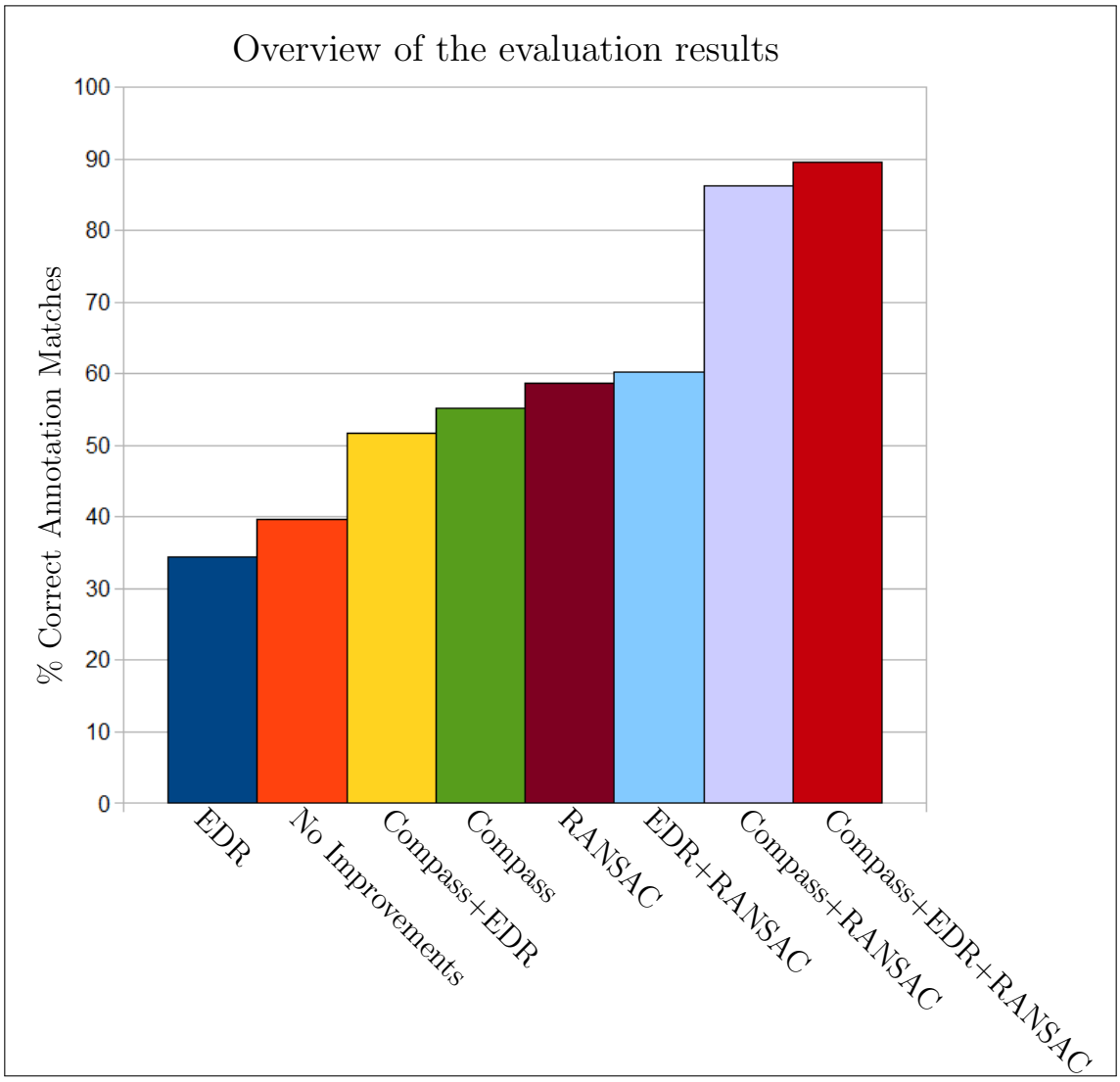

Figure 6.9: This Figure shows the annotation matching results in percent of the different improvements and the combinations of them.

6.3 Description of the evaluation results

Without any improvements we achieve an annotation matching rate of approximately 40 %. As we present in Figure 6.1 in this setting we can match nearly half of all annotations except in panorama 11 and 12 where not even one annotation can be matched correctly.

Description of the individual improvements: The next three Figures 6.2 to 6.4 represent the results of the individual improvements to annotation matching which we

implemented in this work. Figure 6.2 shows the matching performance of the activated digital compass. This improvement achieves a matching rate of more than 55 % and is able to match at least half of all annotations in every panorama except panorama 11 where only one annotation is found. Figure 6.3 illustrates the annotation matching rate with EDR applied during annotation creation and annotation matching. Here, only one annotation could be matched for panorama 1 and 7 and no single annotation for panorama 8, 11, and 12. This leads to an overall matching rate of 34,48 %. As one can see, the performance of this approach is slightly below the results without any activated improvements. RANSAC achieves an overall matching result of 58,62 %. We present the details for every panorama in Figure 6.4. As one can see, the performance of panorama 1, 8, 11 and 12 is also poor similar to EDR but in contrast all annotations of panorama 2, 3, 4, 5 and 10 could be matched correctly.

Description of the different improvement combinations: The next four Figures 6.5 to 6.8 show different combinations of the individual improvements to investigate whether synergy effects occur as a result of combining them. Figure 6.5 illustrates the effect of combining a digital compass with RANSAC. In this setting we achieve an annotation matching rate of 86,21 %. The only panoramas where not all annotations can be matched are number 9, 11 and 12. All other annotations can be matched completely. Combining compass and EDR leads to a slightly lower result as with compass only. Similar to applying EDR only which reduced the overall annotation matching performance of the system. Although there is no panorama without any correct matched annotation, as one can see in Figure 6.6, only 51,72 % annotations could be matched. Figure 6.7 shows that all annotations of seven panoramas could be matched correctly after applying EDR in combination with the RANSAC approach. On the other hand, the matching rates of the remaining five panoramas are poor. Nevertheless, combining EDR and RANSAC leads to an overall matching rate of 60,34 %. In Figure 6.8 we present the results of all three improvements in combination. In this setting we are able to match all annotations except of panorama 7 and 8 and therefore achieve the best matching rate of 89,66 %.

Overview of the results: To give an overview of the different improvements, we show them in figure 6.9 in ascending order. The improvement with the lowest matching score is EDR with 34,48 %, followed by 39,66 without any improvements applied. Combining compass with EDR leads to an annotation matching rate of 51,72 % whereas compass only achieves 55,17 %. The RANSAC approach leads to 58,62 % and in combination

with EDR to 60,34 %. Combinations of the compass and RANSAC lead to a considerable performance increase of 86,21 %. Finally the three methods combined score the best performance with 89,66 %.

6.4 Interpretation of the evaluation results

In this section we draw conclusions and analyze the annotation matching results of our evaluation.

Using a digital compass for annotation matching leads to a plus of 15 % compared to the previous results with no improvements. As one can see, using the compass improvement increases the matching rate moderately. This is because we reduce the NCC matching threshold in areas where annotations are predicted by the compass data as we explain in section 5.1. Generally this approach leads to better matching results. This method can however increase false positives if many similar looking image patches are found in the area with the reduced matching threshold, e.g. windows of a building. Because of magnetic fields in the environment our compass occasionally provided incorrect data. If this occurs we use the same matching threshold used in other areas outside the predicted area which leads to a similar matching performance as without any improvement.

The results of applying EDR are not satisfying in this test run because they are even lower than without any improvements. We could only match 20 out of 58 annotations with the EDR approach which are three annotations less than without EDR. As already mentioned, we integrated EDR to improve the overall image quality of the annotation templates as well as the quality of the panorama cells which should lead to a better matching performance.

As we stated at the beginning of this chapter, the evaluation results are strongly dependent on the annotation quality. If annotations are placed directly on color shift lines caused by the auto exposure adjustment of current smartphone cameras, EDR would increase the matching quality. However, in our case we did not have such cases and therefore had no improvements in matching quality. Furthermore, with activated EDR the larger range of 16-bit values is transformed into 8-bit values during tone mapping process which means that we loose some contrast in the final image. We assume that this is the reason of the slightly weaker matching performance of our EDR implementation. However, we have to investigate this behavior further in future work and also improve the quality of our EDR approach to avoid image artifacts which disturb the matching process.

The third improvement we implemented and evaluated is the RANSAC approach which exploits the overdetermined information about the relations between the individual annotations. RANSAC uses this information and treats annotations as a complete annotation set instead of single, unrelated annotations. As one can see in Figure 6.4 this approach outperforms well with a matching rate of 58,62 % which is a plus of nearly 19 %. This result is compareable to the compass results and is a good step forward to meet our goal of an improved matching performance. If we compare the figures 6.2 and 5.7 one can see that with the RANSAC approach all annotations of one panorama are found in may cases whereas with compass support there are no panoramas where not even one annotation is found. This is because the compass improvement increases the chance of finding individual annotations in contrast to RANSAC where a minimum set of annotations must be found to find a good hypothesis which can be used to match and display all annotations of one panorama at their correct place. Nevertheless, these two improvements do not increase the performance drastically if considered separately. The full strength of the improvements show up in their combination which we present in the following.

A very powerful combination is the RANSAC approach with the possibilities of a digital compass which we present in Figure 6.5. In this combination we achieve an annotation matching rate of 86,21 %. This is because RANSAC needs a minimum amount of best matching predictions to build a good hypothesis model of all annotation positions. This prestep of finding enough annotations can be achieved by the compass as we show in Figure 6.2. In combination with RANSAC we are able to meet our goal of finding 80 % or above of all annotations.

One could also note that it would be possible to lower the matching threshold for the annotations generally so that the chance of finding annotations increases even without using a digital compass. However, with this approach also the number of false positives would increase which can lead to a false RANSAC hypothesis which is supported by many wrong matches. Here, all annotations would be displayed at the wrong positions, which was never the case in the combination of RANSAC with our compass improvement.

Similar to the results of EDR only, the incorporation of EDR with the compass approach does not lead to any performance gain and is approximately +/- 5% as with compass only. Figure 6.7 shows we achieve a similar result for the combination of EDR with the RANSAC approach.

Figure 6.8 shows the combination of all our improvements. In this setting using the compass improvement, EDR and RANSAC we are able to increase the annotation match-

ing rate to nearly 90 % which is a plus of almost 50 % compared to the version without any activated improvements. If we compare Figure 6.8 with Figure 6.5 we notice that the only big difference can be found in the panorama with ID eleven where all annotations could be matched in the case where all improvements where activated and only one correct match could be found in the combination of RANSAC with the compass improvement. Although the two other panoramas where not every annotation could be matched are different in the two versions, the discrepancy's between the two combination sets are not very crucial.

In Figure 6.9 and table 6.1 we compare the individual improvements, as well as all different combinations with the matching performance of the system provided by Langlotz et al. [18]. Here in this diagram, the original system without improvements achieves a matching rate of 40 % on average. The next step of improvements are provided by the RANSAC and the compass approach with and without EDR activated which adds a performance gain of approximately 15-20 % on average and leads to an overall matching rate of about 55 % to 60 %.

The next big performance step is provided if RANSAC is combined with our compass improvement which is presented by the last two bars of Figure 6.9. The version with activated EDR found only 2 out of 58 annotations more than the RANSAC - compass combination and is therefore very similar to the version without EDR. We want to emphasize at this point that these results were generated in one evaluation process and can vary slightly in other evaluations. This is because lighting and environment conditions change during annotation creation and annotation matching which leads to different images which have to be matched against each other. Because of the environmental factor, we tried to evaluate in a different lighting setting to test our system in a worst case scenario.

Although the annotation matching results can vary over evaluations in different test sets, the general trend and the relations between the individual improvements and their combinations are very similar. The matching rates without any improvements are generally below 50 % whereas compass and RANSAC individually improve these results by 15 % to 20 %. We investigated that RANSAC outperformeces better if the overall matching rate is generally higher which is explainable by the minimum amount of best matches which are a prerequisite for this algorithm. Furthermore, if RANSAC is combined with the digital compass, either with our without activated EDR, matching rates increase noticeable up to 80 % - 90 % which is a good outcome for our improvements.

As already mentioned, we evaluated the system using a notebook, webcam and a digital compass. To investigate if the results are transferable to current smartphone systems

Improvement	% Correct Matches
EDR	34,48
No Improvement	39,66
Compass + EDR	51,72
Compass	55,17
RANSAC	58,62
EDR + RANSAC	60,34
Compass + RANSAC	86,21
Compass + EDR + RANSAC	89,66

Table 6.1: Evaluation Results

we tested a combination of compass, EDR and RANSAC on a HTC HD2 smartphone with a digital compass and a 1 GHz processor running on Windows Mobile. The result of this evaluation achieved 90 % correct matches which supports the outcome of the evaluation presented in this chapter and met our expectations regarding the combination of our improvements.

However, one must mention that these results can be achieved only, if the annotation templates contain a minimum amount of matchable features. If annotations are placed in areas with few or no features e.g. completely white walls, matching will fail with or without any improvement. Nevertheless, in such cases we can use the data provided by the digital compass as a fallback solution to display the annotation at least nearby its correct location. Although this approach is very useful we did not take it into account in our evaluation.

We suggest to improve the annotation quality by a preliminary step during annotation creation since the compass data can be influenced by magnetic fields and not every smartphone is equipped with a digital compass. The annotation template which we cut out from the panorama with our web-interface could be analyzed regarding the quality and amount of features which can be found in this image patch. If it is unlikely, that this annotation template can be matched on the smartphone client, we could refuse the annotation creation asking the user to create the annotation in an area of better image quality. However, this is a task which we did not implement yet and is therefore is left for future work.

Table 6.1 shows a complete list of all annotation matching results sorted by their performance. One annotation accounts for approximately 1,7 % of the overall result which means that a difference of three annotations which could not be matched leads to a result

5,18 % lower as one can see when comparing the EDR results with the result of no activated improvements. Also the next four improvements, compass, compass with EDR, RANSAC and RANSAC with EDR differ only by a few annotations if compared to each other. The next big step is achieved by combining compass with RANSAC or all three improvements with each other which brings the matching performance from 86 % to nearly 90 %. Although this matching performance already meets our expectations, this are only first results. We think that the overall matching rate can be enhanced further by improving the EDR approach or by some more experimenting with the annotation matching threshold used in the compass improvement. Another optimization could be possible by investigating the maximum pixel error which we allow for the RANSAC algorithm or the number of inliers needed to activate RANSAC. However, we leave these optimization steps for future work.

Chapter 7

Conclusion

In this work, we addressed some limitations of current mobile AR information systems which is a lack of content and the unsatisfying performance of annotation matching under changing environment conditions. To generate AR content in a fast and inexpensive way on a global scale, user participation is necessary. We therefore developed a prototype of a web-based interface which allows end-users with no or little previous knowledge about AR information systems to create, modify or delete textual AR annotations.

One problem of AR annotation creation is to determine the exact position in the real world. We mentioned several approaches that had some advantages but also disadvantages making them unsuitable for mobile AR systems running on current smartphones. We therefore decided to integrate linking textual AR annotations to their according GPS position using GoogleMaps which is commonly known and used by a broad user base. Although we developed this interface with usability considerations in mind, it is only a prototype and should be extended in future work.

After creating annotations and linking them to their real-world position, it is possible to calculate the angle between the panorama and its corresponding annotations. We use this information for both, a fallback solution if vision-based annotation matching fails and to improve the annotation matching quality. Furthermore, we are able to provide a preview of the annotations to support the user in finding and matching them to their correct position in the AR view.

This approach works similar to commercial sensor-based AR information systems which use a compass to compute the position of AR annotations. However, a drawback of such systems is that they are generally not very accurate; magnetic fields surrounding the mobile devices disturb the correct operation of digital compasses. Another option for

determining the position of annotations is the use of vision-based algorithms instead of sensor-based approaches. We also used such a system as basis for our work. However, we discovered that vision-based annotation matching often fails if lighting and environmental conditions change between annotation creation and annotation matching.

We therefore investigated three different approaches to improve the overall annotation matching performance. One improvement was to extend the range of the values for each color channel of the panorama image dynamically and use this extended range to compensate the color artifacts which are introduced because of automatic exposure adjustment of current smartphone cameras. As we discovered in a first evaluation step, this improvement has no significant impact on the annotation matching performance. Furthermore, in some cases EDR even decreases the matching rates as shown in our evaluation. We therefore suggest to investigate and improve this approach further in future work.

Although vision-based annotation matching can provide pixel-accuracy in contrast to sensor-based approaches, we use a built-in digital compass to improve the annotation matching quality and as a fallback if matching fails completely. We use the absolute orientation provided by the compass together with the calculated angle between panorama and annotations to display them in a preview and to localize areas where annotations are likely to be. Furthermore, we lower the vision-based matching criteria in those areas to find annotations easier while avoiding more false positives because we leave the matching criteria high in all remaining areas. Our evaluation showed that this approach is able to increase the overall matching rates by approximately 15 %.

Another promising improvement which we implemented and evaluated is the RANSAC algorithm to build hypotheses of annotation position models. The main idea of this approach is to exploit the information about the relations of the textual annotations among each other. We therefore treat annotations not as separate units any longer, but as a linked set of annotations where the relations among them are known.

We use this information of the saved annotations of our content-server to predict the annotation position in the newly created panorama during the annotation matching process. Depending on the orientation of the user when he starts to create the panorama, the cylinder on which we map our panorama is oriented arbitrarily. This means that we get different cylinder orientations of the panorama which was used to create the annotations and the panorama which is created during the annotation matching process. We therefore have to align both cylinders so that they have the same orientation if we want to make a prediction of annotation positions by transferring the positions of the saved panorama

into the new one. To achieve this, we try to find a rotation which is able to transform all annotations of the saved panorama in a way that they are at the same, or close to the position in the panorama used for matching.

We use this rotation as a hypothesis and try to find enough best matches in the new panorama which supports this model. If the model fulfills the criteria of heaving at least 50 % inliers with a maximum offset of 10 pixels in x- and y-direction regarding the position difference of the database annotations and the client's annotation positions, we accept it as accurate enough. Afterwards we transform and display all annotations at their corresponding position in the panorama map on the AR client. Although this method introduces a small error because of the pixel offset tolerance, it is still very accurate. If one can tolerate this inaccuracy, this approach is able to improve the annotation matching performance significantly because if enough best matches are found to support a correct hypthesis, all annotations can be displayed at their correct position.

To investigate the impact of our improvements under changing environment conditions, we created a test scenario under difficult lighting settings. To make the individual improvements comparable to each other as well as the system without any improvements, we created test videos and panoramas at 12 different locations around our university campus. With this data we evaluated the system without improvements, with activated compass, EDR and RANSAC and with all possible combinations of them which led to eight different alternatives.

The outcome of this first evaluation was that the system could only match about 40 % of all annotations correctly under changing lighting conditions. However, with activated compass we were able to improve this result by nearly 15 % and the RANSAC approach led to a performance gain of approximately 19 %. In contrast to our expectations, EDR did not lead to significantly better matching results. Furthermore, in some cases it even reduced the annotation matching performance. However, we found some interesting outcomes by combining the RANSAC approach with the compass improvement which is able to find 80 % - 90 % of all annotations. This is because our compass improvement increases the matching rate over 50 % and is able to find possible candidates for best matches which is a prerequisite for building a good RANSAC model. If this step succeeds, RANSAC is able to display all annotations at their correct position.

Although this work showed how to improve annotation matching and made some suggestions for developing a system to create and manage AR annotations on a global scale, this is only a small part of the evolutionary field of mobile AR research.

Chapter 8

Future work

As we discovered in our evaluation, the quality of the annotation templates is crucial for the overall matching performance. For this reason, it would be preferable to check the image quality of annotations before creating them with the web-based interface. If a certain number of good features are found in the image patch the user should be allowed to create the annotation at the desired position in the panorama map. However, a good threshold which decides if an annotation is accepted or not has to be determined.

An interesting question is what to do if users want to place annotations in areas where few or no appropriate features are found, e.g. on completely white walls of buildings. In such cases annotation matching will fail on the AR client and the overall annotation matching performance will decrease. One way to manage this situation could be to implement an automatic interest point detector which finds areas of good image quality with a high amount of features. These areas could be suggested to the user to support him in finding appropriate positions in the panorama which are suitable for annotation creation.

Another idea is to create annotations without any label automatically at positions in the panorama which are retrieved by an interest point detector. These annotations would be of high feature quality supporting the RANSAC approach in being successfully applied. This would enable users to place their annotations everywhere in the panorama, even in areas with no or very few features, while ensuring the correct display of all annotations because of the robust annotation matching model using RANSAC.

Since our implementation of the EDR approach did not meet our expectations, we should investigate the concrete circumstances under which annotation matching is influenced in a negative way by applying EDR. Furthermore, we should improve the EDR implementation and also address vignetting effects which are introduced by the smart-

phone cameras and lead to visual artifacts in the resulting panorama image. The reason for not investigating vignetting in this work is that this effect is compensated partially in the built-in camera modules already.

Although the compass and RANSAC improvements delivered good results, we could still optimize them further by experimenting with the thresholds of both approaches. In the RANSAC approach we could adjust the pixel error tolerance which is responsible for accepting a hypothesis as good model. The goal is to find a value which allows no false models at all while increasing the number of successful applications of the RANSAC algorithm. The threshold of the compass improvement is responsible for deciding if a part of an annotation template is accepted or declined as a correct match. The challenge of finding an ideal value for this threshold which maximizes the number of correct matches while keeping false positives low remains.

A further step for improving the presented mobile AR system is to support more different content-types besides the textual AR annotations. Some examples are images, audio or video content which could be used to augment the real world and would enhance the users perception. However, the web-based interface and the AR client have to be enabled to support such content formats in the future.

It would be interesting to evaluate our system on a larger scale with several hundred annotations created over a longer time period than a year. This would provide more test cases under changing environment conditions and different seasons and could give a better understanding on the several factors which influence the annotation matching performance. Furthermore, we could gain better average matching results as our evaluation was only an initial test.

Another important step on the way to AR 2.0 is to present our system to non-experienced end users in a user study to examine their acceptance and get feedback concerning usability issues. Furthermore, it would be interesting to investigate which annotation matching rates could be achieved if non-experts use our system in contrast to our evaluation results where only one expert user performed the evaluation.

Appendix A

Orientation calculation

We present a detailed description of how to calculate the orientation between two different coordinate systems so that two vector pairs a_1, a_2 and b_1, b_2 can be aligned to each other with a minimal error between a_1 and b_1 and a_2 and b_2. The approach presented in this chapter is a special solution for only two vector pairs. For a more general approach see the work of Horn [14].
Equation (A.1) calculates the rotation $\otimes$ of the coordinate system of the two vectors a and b.

$$\otimes = \Big(\; a, \quad (a \times b) \times a, \quad a \times b \; \Big) \tag{A.1}$$

With the rotation calculation (A.2) we can determine the orientation r_1 of the coordinate system of the vectors a_1 and a_2 which corresponds to the orientation of the first cylinder 1a of Figure A.1.

$$r_1 = a_1 \otimes a_2 \tag{A.2}$$

Equation (A.3) determines the orientation r_2 as illustrated in our example of cylinder 1b of Figure A.1.

$$r_2 = b_1 \otimes b_2 \tag{A.3}$$

We use the two rotations r_1 and r_2 to calculate rAB_1 which aligns cylinder 1a to 1b so that both cylinders have the same up vector and a_1 and b_1 are equal (see equation (A.4)).

$$rAB_1 = r_2 * r_1^{-1} \tag{A.4}$$

The next three equations (A.5) to (A.7) are similar to (A.2) to (A.4) with the difference that now the two coordinate systems of cylinder 1a and 1b are aligned in a way that a_2 and b_2 are equal, again with the same up vector.

$$r_3 = a_2 \otimes a_1 \tag{A.5}$$

$$r_4 = b_2 \otimes b_1 \tag{A.6}$$

$$rAB_2 = r_4 * r_3^{-1} \tag{A.7}$$

We can use the orientations of rAB_1 and rAB_2 to calculate the difference Δ between them in equation (A.8).

$$\Delta = rAB_2 * rAB_1^{-1} \tag{A.8}$$

The $\log(R)$ of a rotation R is a skew-symmetric 3x3 matrix r representing rotation angle and axis. Scaling the matrix by 0.5 creates a rotation around the same axis with half the angle. $\exp(r)$ is then again the corresponding rotation matrix, now representing half the original rotation.

$$R = \exp(\log(\Delta) * 0.5) * rAB_1 \tag{A.9}$$

The result R of equation (A.9) is a rotation which can be used to align the two vector pairs a_1, a_2 and b_1, b_2 so that a_1 and b_1 and a_2 and b_2 point in the same direction with an equally distributed error. This effect is illustrated in cylinder 2 of Figure A.1.

A small error remains if the angle between the vector pairs a_1, a_2 and b_1, b_2 are different. These different angles occur if the two vector pairs originate from different positions and initial rotations during panorama creation. This effect of introducing a translation and rotation error is explained in more detail by DiVerdi et al. [7] in their work about Envisor, an application for online environment map creation.

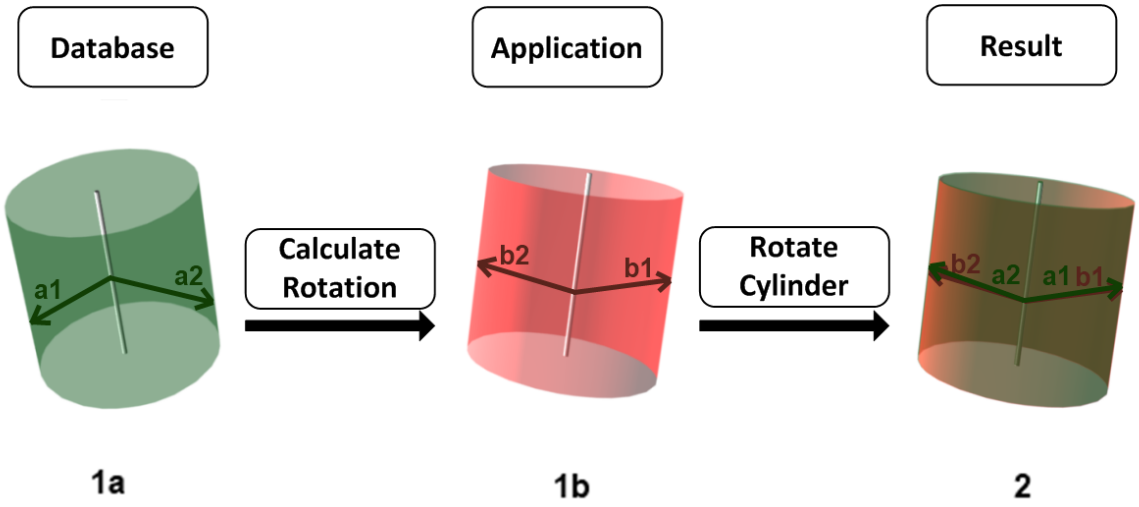

Figure A.1: Visual representation of an orientation calculation which aligns one coordinate system to another one.

Bibliography

[1] Adams, A., Talvala, E.-V., Park, S. H., Jacobs, D. E., Ajdin, B., Gelfand, N., Dolson, J., Vaquero, D., Baek, J., Tico, M., Lensch, H. P. A., Matusik, W., Pulli, K., Horowitz, M., and Levoy, M. (2010). The frankencamera: an experimental platform for computational photography. *ACM Trans. Graph.*, 29:29:1–29:12.

[2] Adobe (2010). Adobe actionscript api. http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/. last visited on October 3, 2010.

[3] Azuma, R. (1997). A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385.

[4] Azuma, R. (1999). The challenge of making augmented reality work outdoors. *Mixed reality: Merging real and virtual worlds*, pages 379–390.

[5] Bruns, E. and Bimber, O. (2009). Adaptive training of video sets for image recognition on mobile phones. *Personal Ubiquitous Comput.*, 13:165–178.

[6] Cheverst, K., Davies, N., Mitchell, K., and Friday, A. (2000). Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 20–31. ACM.

[7] DiVerdi, S., Wither, J., and Höllerer, T. (2008). Envisor: Online environment map construction for mixed reality. *Proc. IEEE VR 2008 (10th International Conference on Virtual Reality)*, pages 19–26.

[8] Facebook (2010). Facebook places. http://www.facebook.com/places/. last visited on November 30, 2010.

[9] Feiner, S., MacIntyre, B., Höllerer, T., and Webster, A. (1997). A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. *Personal Technologies*, 1(4):208–217.

[10] Fischler, M. and Bolles, R. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6).

[11] GeoNames (2010). Geonames geographical database. http://www.geonames.org/maps/wikipedia.html. last visited on October 29, 2010.

[12] Google (2010). Google maps api. http://code.google.com/intl/en-EN/apis/maps/documentation/flash/. last visited on October 1, 2010.

[13] Höllerer, T. and Feiner, S. (2004). Mobile Augmented Reality. *Telegeoinformatics: Location-based Computing and Services*, pages 1–39.

[14] Horn, B. K. (1987). Closed form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4:629–642.

[15] Institute for Computer Graphics and Vision (2009). Handheld augmented reality. http://studierstube.icg.tu-graz.ac.at/handheld_ar/index.php. last visited on September 23, 2010.

[16] InterSense Inc. (2010). Intersense. http://www.intersense.com/InertiaCube_Sensors.aspx. last visited on November 20, 2010.

[17] Kooper, R. and MacIntyre, B. (2003). Browsing the Real-World Wide Web: Maintaining awareness of virtual information in an AR information space. *International Journal of Human-Computer Interaction*, 16(3):425–446.

[18] Langlotz, T., Wagner, D., Mulloni, A., and Schmalstieg, D. (2010). Online creation of panoramic augmented reality annotations on mobile phones. *IEEE Pervasive Computing*, 99(PrePrints).

[19] Layar (2010). Augmented reality - layar reality browser. http://www.layar.com/. last visited on October 21, 2010.

[20] Mobilizy Mobile Software (2010). Wikitude. http://www.wikitude.org/. last visited on October 21, 2010.

[21] Nillius, P. and Eklundh, J. (2002). Fast block matching with normalized cross-correlation using walsh transforms. *Computational Vision and Active Perception Laboratory, Stockholm, Sweden, Tech. Rep. TRITA-NA-P02/11*.

[22] Reitmayr, G. and Schmalstieg, D. (2004). Collaborative Augmented Reality for Outdoor Navigation and Information Browsing. *Location Based Services and TeleCartography*, pages 31–41.

[23] Schmalstieg, D., Langlotz, T., and Billinghurst, M. (2010). *Augmented Reality 2.0*. Springer-Verlag.

[24] Schwinger, W., Grün, C., Pröll, B., Retschitzegger, W., and Schauerhuber, A. (2009). Context-awareness in Mobile Tourism Guides. *Handbook of Research on Mobile Multimedia, 2nd edition, published by Information Science Reference*.

[25] Spohrer, J. C. (1999). Information in places. *IBM SYSTEMS JOURNAL*, 38(4):602–628.

[26] Sutherland, I. E. (1968). A head-mounted three dimensional display. *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)*, 1866(16):757.

[27] Takacs, G., Chandrasekhar, V., Gelfand, N., Xiong, Y., Chen, W.-C., Bismpigiannis, T., Grzeszczuk, R., Pulli, K., and Girod, B. (2008). Outdoors augmented reality on mobile phone using loxel-based visual feature organization. *IEEE Trans. Pattern Analysis and Machine Intelligence*.

[28] Vincenty, T. (1975). Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review*, 12(176):88–93.

[29] Wagner, D., Mulloni, A., Langlotz, T., and Schmalstieg, D. (2010). Real-time panoramic mapping and tracking on mobile phones. *2010 IEEE Virtual Reality Conference (VR)*, pages 211–218.

[30] Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., and Schmalstieg, D. (2008). Pose tracking from natural features on mobile phones. *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134.

[31] Wagner, D. and Schmalstieg, D. (2003). First steps towards handheld augmented reality. *Seventh IEEE International Symposium on Wearable Computers, 2003. Proceedings.*, pages 127–135.

[32] Wagner, D. and Schmalstieg, D. (2009a). History and future of tracking for mobile phone augmented reality. In *ISUVR '09: Proceedings of the 2009 International Symposium on Ubiquitous Virtual Reality*, pages 7–10, Washington, DC, USA. IEEE Computer Society.

[33] Wagner, D. and Schmalstieg, D. (2009b). Making augmented reality practical on mobile phones, part 1. *IEEE Comput. Graph. Appl.*, 29(3):12–15.

[34] Wagner, D. and Schmalstieg, D. (2009c). Making augmented reality practical on mobile phones, part 2. *Computer Graphics and Applications, IEEE*, 29(4):6–9.

[35] Weschkalnies, N. (2009). *Adobe Flash CS4 - Das umfassende Handbuch*. Galileo Press, Rheinwerkallee 4, 53227 Bonn, Germany, first edition.

[36] Wither, J., Coffin, C., Ventura, J., and Höllerer, T. (2008). Fast annotation and modeling with a single-point laser range finder. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 65–68. IEEE.

[37] Wither, J., DiVerdi, S., and Höllerer, T. (2006). Using aerial photographs for improved mobile AR annotation. In *IEEE/ACM International Symposium on Mixed and Augmented Reality, 2006. ISMAR 2006*, pages 159–162.

[38] Wither, J., DiVerdi, S., and Höllerer, T. (2009). Annotation in outdoor augmented reality. *Computers & Graphics*, 33(6):679–689.