Graz University of Technology

Institute for Computer Graphics and Vision

Master Thesis

Overview and Detail in Augmented Reality Browsers

Simon Kendler, BSc.

Graz, Austria, September 2014

Thesis supervisors

Dr. Denis Kalkofen

Dipl.-Ing. Markus Tatzgern

Prof. Dr. Dieter Schmalstieg

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

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 entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..
(Unterschrift)

Englische Fassung:

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 (signature)

The science of today is the technology of tomorrow.

Edward Teller

Abstracts

Abstract

In this thesis I present an approach for a new Augmented Reality Browser (AR browser). This AR browser allows to display the information of real world Points of Interest (POI) on mobile smart devices. The implementation of the AR browser handles a huge variety of tasks, beginning from the retrieval and filtering of the data for the POI to the final visualization on the screen of the device. These tasks include the clustering of the data to avoid clutter of the labels of the given data points. So, another topic discussed in this thesis is visualization and layout of the result of the cluster analysis. Additionally, colored glyphs are used to represent the quality of results filtered by category. Furthermore, the AR browser implements the concept of overview and detail in a single view. Therefore, an overview map is part of the content of the AR browser. In addition, the browser supports an AR location search to help a user at finding the location of nearby POI.

Keywords. Augmented reality, HCI, visualisation, overview+detail, clustering

Zusammenfassung

In dieser Masterarbeit stelle ich einen Ansatz für einen Augmented Reality Browser (AR Browser) vor. Dieser AR Browser erlaubt es sogenannte Points of Interest (POI oder Sehenswürdigkeiten) der realen Welt mit Hilfe eines mobilen Endgerätes (Smartphone) darzustellen. Die Umsetzung des AR Browsers kümmert sich um eine Vielzahl von unterschiedlichen Aufgaben. Diese Aufgaben beginnen bei der Umwandlung und Filterung der relevanten Daten der POI bis hin zur Visualisierung dieser am mobilen Endgerät. Eine weitere Aufgabe der Implementierung ist es die Daten zu Clustern zusammenzufassen, da der Platz für die Darstellung der Datenpunkte mit Hilfe von Labeln auf den Bildschirm des Gerätes sehr begrenzt ist. Weiters wird eine alternative Representation von Resultaten

einer Filterung nach Kategorien unterstützt. Dabei kommen verschiedenfarbige Glyphen zum Einsatz. Zusätzlich wird im AR Browsers die Idee von Overview+Detail (Übersicht und Detail) umgesetzt. Dabei wird eine Karte der Umgebung innerhalb der selben Ansicht wie Label der POI angezeigt. Die Implementierung einer einfachen AR unterstützen Ortssuche ist ein zusätzlicher Teil der Umsetzung des AR Browsers.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

Augmented Reality (AR) gives the user an augmented view of the world through the camera of a mobile device (e.g. 'smart phone'). Such an augmentation of the real world starts from the simple annotation of points of interest (POI) to the interaction between the user and her environment. The advantage of AR is that the information is displayed as an overlay of the camera input. Hence, the user's view is slightly disturbed, but she has the access to additional information of the POI in front of her. The applications of AR are versatile. Augmented reality is used as aid in complex tasks, i.e. in surgery, maintenance, or for navigation issues in civil and military vehicles.

The software used to display AR content is called an Augmented Reality browser (AR browser), which implementation runs either on a normal PC or a mobile smart device. In history, mobile AR systems had the drawback of bulky, expensive and heavy hardware for each part of system. But the arise of mobile smart devices available for consumers, so called *smart phones*, changed the possibilities for using AR systems. The modern mobile devices provide a large variety of sensors (e.g. GPS, gyroscope, magnetometer, accelerometer) to locate the position and orientation of the device and retrieve information from different networks. The development and availability of the cheap and powerful hardware in mobile devices has made AR browsers available for the common user on different mobile platforms. There are even devices on the market for the common user available , that are light and

small enough to be worn like glasses (*Google Glass**).

In this thesis I will present the implementation of an AR browser with the support for real world places. The thesis shows up problems occurring in the AR browser system and how to deal with them. Further, I describe the used methods and present an approach how to combine overview+detail in the AR browser.

1.1 Augmented Reality Browsers

There are a few established commercial AR browsers on the market for mobile applications (apps) for popular mobile operating systems (mobile OS), like Android, iOS, and Windows Phone (WP).

- *Wikitude* [57] (Android, iOS and WP)
- *Layar* [36] (Android, iOS)
- *Junaio* [42] (Android, iOS)
- *HERE City Lens* [46] (WP)

The popular AR browsers, that are available on the different mobile application markets, allow the user to retrieve additional multimedia content, i.e. text, images, and videos. Therefore, a continuous connection to the internet is necessary.

All of the commercial AR browsers allow the user to explore nearby locations. In some cases the user has the ability to select from different data sources. Commonly, the AR browser shows the information about the POI as an overlay of the recorded camera image on the screen. Additional information about the object is retrieved, if the user touches the annotated overlay. Furthermore, it is possible to search for locations or filter for specific categories of locations. Due to the small screen, clutter of data is often inevitable. Therefore cluster algorithms are used to aggregate nearby POI. Unfortunately, the method of aggregation is not mentioned by the creators of the different AR browsers.

A top-down-view of the surrounding area (*radar map*) allows the user to detect POI, which are not in the user's Field of View (FOV). In addition, a search can be performed to find easily any nearby location. Furthermore, a listing of nearby locations can be created in some cases.

The Layar and Junaio mobile applications have some additional features implemented. It is possible for the user to interact with her environment in some way. This AR browser

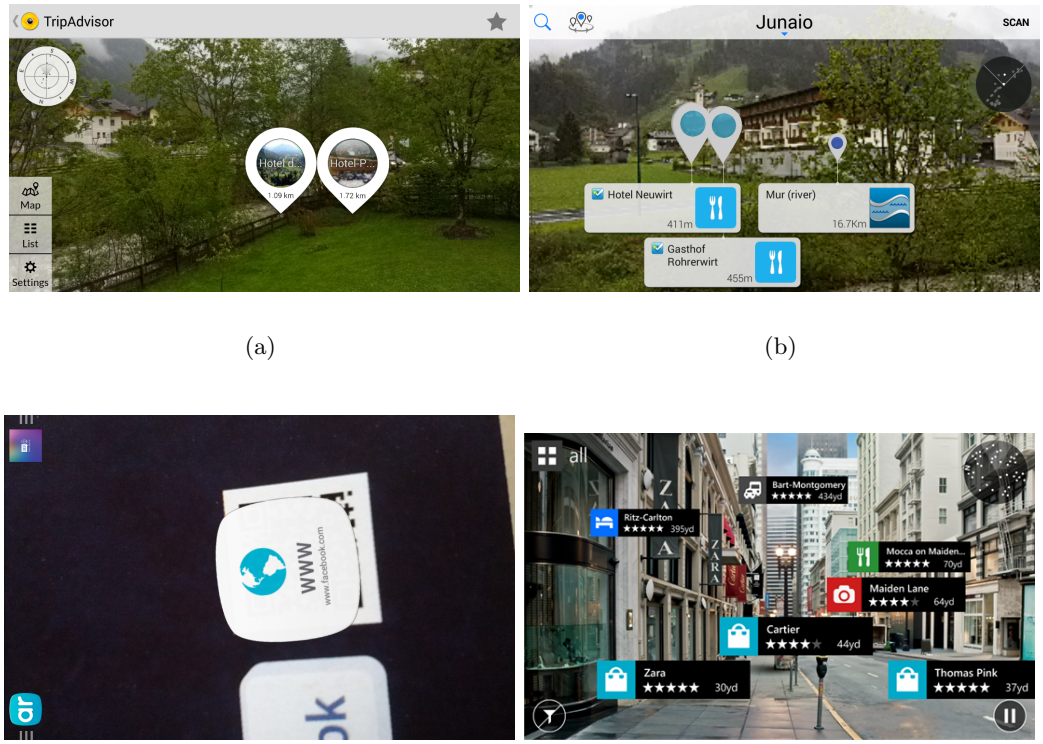
*http://www.google.com/glass/start/

type allows advertisers i.e. to present interested and possible future customers their products. Therefore, the user of the application has to scan a marker (i.e. *QR*-Tag[†]), which can be found in printed media, on posters or websites. After decoding the information of the tagged information, some sort of multi-media content will be shown on the screen of the mobile device. There is a huge variation of multi-media content types, beginning at a simple link to a website up to an interactive 3D model of an object. In the mentioned cases, this features of AR browsers have not the need to handle a large amount of data. Therefore, there is no need of a cluster analysis to find clusters within the data set and in a further step to deal with clutter, that occurs at the labelling of the POI on the screen of the device. More important for this type of AR browsers are image processing techniques like detection of the markers or image cues and the tracking of the detected markers.

In Fig. 1.1 impressions of the mentioned commercial AR browsers can be found. It can be seen that Wikitude (see Fig. 1.1(a)) and Junaio (see Fig. 1.1(b)) have a very similar user interface layout. Both AR browser use a radar map to show nearby POI and visualize the POI with markers. While Wikitude uses the markers as labels too, the marker and the according label for that marker are separated in Junaio. The connection is given by a leader line which connects the label and marker. The HERE City Lens application, which can be found in Fig. 1.1(d), uses labels only. In contrast, an image of Layar is given in Fig. 1.1(c), which shows that the main purpose of the application is to retrieve information from real world markers and the display of the correlating multi-media content. Therefore, the application supports no horizontal mode and tells the user to point onto a marker in his nearby environment.

In this thesis, I will address the first type of the previous mentioned AR browsers, which implementations support the presentation of real world places. Therefore, the data for the real world places is retrieved from external sources (Google Places API). The usage of external sources makes the connection to the internet mandatory, but the usage more flexible. Nevertheless, a caching mechanism using a database allows to reduce the amount of data which has to be demanded. The places are represented as labels on the screen of the mobile device. In the implementation, the representation of the labels is diverted into labels with detailed information and labels annotating clusters of points of interest. Each label with detailed information describes the POI by its name and additional information, i.e. the address or the distance from the current position.

[†]Quick Response Code, a 2D Code



(a) (b)

(c) (d)

Figure 1.1: Different commercial AR browsers available for different mobile OS platforms. (a) Wikitude AR browser (Android): *Tripadvisor*-mode to show nearby POI. (b) Junaio AR browser (Android): POI are labelled and annotated by markers. (c) Layar AR browser (Android): A *QR*-tag representing a hyperlink is found and overlaid by an icon for that type (Rotated 90° counter-clockwise). (d) HERE City Lens AR browser (WP): different POI are labelled and the categories are color coded. Image from [46].

In contrast to the introduced commercial AR browsers, my proposed implementation supports an overview map of the surrounding area within the main view of the AR browser. Additionally, a 'traditional' overview map with nearby POI can be shown in a separated viewer too. Common filtering and search tasks, e.g. filtering the data by category or searching for a specific location, can be performed in the implementation of the AR browser. In addition, an AR location search is supported by the browser, which helps the user to find a specific location in her surrounding area.

1.2 Problems

As mentioned above, one of the main problem of an AR browser is the fact that there is a huge amount of data to visualize. The large quantity of data in combination of a small screen of a mobile device makes it necessary to only show user relevant data. So, e.g. in large cities with a high density of different locations, a large data set representing different POI is possible. Therefore, the filtering of the data by some parameters is necessary. The filtering can be done in different ways. There are filtering methods, which use the distance of an object to the current position. These filtering approach is called spatial filtering. A common method of spatial filtering is to set a specific radius or region of interest. Everything within this range or area is shown to the user. The other type of filtering mechanisms is knowledge-based filtering. This type of filtering uses the properties of the given data points. An example of a knowledge-based filtering method is the filtering by one or more properties of an object, like the categories of POI, e.g. restaurants, hotels. These two filtering mechanisms can be combined to reduce the displayed amount of data. I will address this problem in detail in my approach.

Another problem is the visualization of the data points on the small screen of a mobile device. The representation should give as much as possible information about the POI, but should also allow the user to interact with the represented data. An easy way to enable the user interaction is to annotate each POI with an (fixed-size) label at the position on the screen representing its position in the real world. This simple labelling method introduces the problem of cluttering of the labels on the screen. Therefore, some sort of advanced clustering algorithm has to be performed on the given dataset. The problem of overlapping labels (clutter) is depicted in Fig. 1.2. Here, the standard k-means algorithm is used to determine clusters within a random generated point cloud (number of points $n = 450$) in an early implementation stage of the AR browser. Every label has the same width and height and its center is positioned at each cluster center.

After the clustering of the POI in a meaningful way, the next occurring problem is the way to display the POI within a cluster. The connection between the real world location and the representation on screen should be recognizable. This cluster labelling task causes the need of an additional clustering of the data points within the current cluster (sub-clustering). Whereas, the available space on screen of the mobile device should be exploit to allow the navigation within the cluster and the data points. I will present my approach to handle this problem in this thesis.

The field of view (FOV) of the camera is limited. Hence, only POI in front of the camera

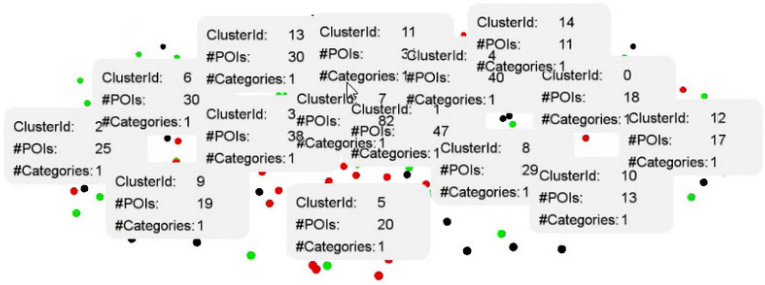


Figure 1.2: Example: clustering and clutter of labels. Using k-means to determine cluster and a simple cluster labelling. Random generated disc-shaped cloud of data points ($n = 450$) and $k = 15$ from an early stage of the AR browser.

respectively the user can be annotated. In case of an AR application for real world places the correlation between an annotation on the screen for a given POI and the position in real world is meaningful and useful. But, the visualization of this relation is not a trivial task on a small screen of a mobile device. Therefore, I will occupy myself with methods to deal with this problem in this thesis.

Additionally, a common issue is, that the user does not know where to find a specific location. Some AR browsers address this problem by providing a search option resulting in a list of matching nearby locations. But, an AR guidance, which helps to locate the looked-up POI in the real world, is not supported by existing consumer AR browser solutions. I will address this problem in my thesis in an approach of a AR location search. The AR location search will support the user by indicating her how to change the orientation of the device to find the desired location.

1.3 Purpose

The purpose of this thesis is to present a new AR browser. The AR browser uses a pipeline to retrieve, filter and visualize the data of nearby real world places. Furthermore, active user interactions are taken to account in the processing pipeline of the AR browser. These interactions may be position or orientation changes of the mobile device or the selection of a specific object on the screen of the mobile device. Therefore, the thesis will present methods to retrieve real world data by using online resources and how to convert these

resources for a further usage.

Further, the proposed implementation of the AR browser will list and discuss different methods to filter the data of the retrieved points of interest. Thus, the section of implementation of the filtering methods (see Section 4.3) will show up how to deal with the problem of the (possible) large amount of data points. Additionally, I deal with knowledge-based filtering using the example of filtering the multi-category data points by more than one category. The representation of the significance of the results of the filtered data set will be also a part of this thesis. Especially, an approach to show the quality of the results within a cluster with a single label without showing the the members of the cluster in detail.

Furthermore, the filtered data points have to be processed in such a way that a clutter free presentation of labels on the screen is achieved. This can be achieved by performing a cluster analysis on the given data points. I will compare some common used methods for cluster analysis and point out their advantages and disadvantages in Section 4.4.1. Furthermore, I discuss the proposed clustering algorithm, called '*Angular Clustering*' (see Section 4.4.1.3), which is implemented in the browser in detail. This thesis should illustrate how the used cluster algorithm superiors other cluster analysis methods at the usage in an AR browser.

In addition, I present a method to label the data of the POI within a cluster in a clear and easy to navigable way on the small screen of a mobile device. This labelling method (see Section 4.4.2) uses a fixed layout for the positions of each label on the screen. Furthermore, I will discuss a way to prevent intersections between the different labels and their leader lines to the annotated position of the real world place. Additionally, the labelling method supports a multiple assignment of a label to a cluster of places within a selected cluster ('sub-cluster clustering'). Therefore, I will present a hierarchical labelling structure, which allows to navigate within the cluster.

Furthermore, the problem of the user's limited FOV is addressed. As mentioned above, often a simple overview map ('radar' map) is used to show nearby locations. This approach allows the user to have a combined view of the world by showing an overview of the world by using a map and the detail (the user's FOV). The concept of overview+detail allows to illustrate the relationship between a point on the screen and in the real world in a single view. In my thesis, I show how to use overview+detail to represent this relationship for the user. The idea behind the proposed AR visualization of the map is based on a physical, 'traditional' map. In contrast a non-AR representation of an overview map is

implemented too. I will discuss my method to combine overview and detail in a single view in Section 4.6.

Additionally, the thesis will address the problem of finding a specific real world location with the support of the AR browser. Therefore, I present a method to draw the attention to a point in the virtual representation within the AR. The approach of the 'guided' AR location search is presented and discussed in detail in this thesis in Section 4.5.

Chapter 2

Related Work

Contents

In this chapter, I will present the related work, which is associated with an AR browser. Due to the fact, that it is no single topic, I will deal separately with the different topics. The related work is separated into the following four topics:

- Augmented Reality Browsers
- Overview+Detail
- Filtering and Cluster Analysis in AR
- Representation and Labelling

2.1 Augmented Reality Browsers

The history of augmented reality reaches far back to the late 1960s, where Ivan Sutherland created the first augmented reality system [53]. This system showed via a head mounted system simple wire-frame graphics. In 1992 the term 'augmented reality' was referred by Caudell and Mizell [9] for the overlap of computer generated material over the real world with the help of a head-mounted display. Until the year 1997 augmented reality applications had been restricted for a use at a specific location, but this changed with

the presentation of the *Touring Machine* by Feiner et al. [17]. This system was the first Mobile Augmented Reality System (MARS), which used different ways to define the user's position and orientation. The Touring Machine introduced the concept of an AR browser too. Basically by showing landmarks with additional information on a campus of a university through a head-mounted display and the navigation via an additional stylus and a touchpad interface. An image of the prototype with the hardware of the Touring Machine can be seen in Fig. 2.1(a). The user wears a backpack with a head-worn display and is equipped with a hand-held display and stylus. The information shown through the head-worn display is depicted in 2.1(b), where some buildings of the campus are shown and labelled. In 1999 this concept was taken further by embedding multimedia content by Höllerer et al. [27].

(a) (b)

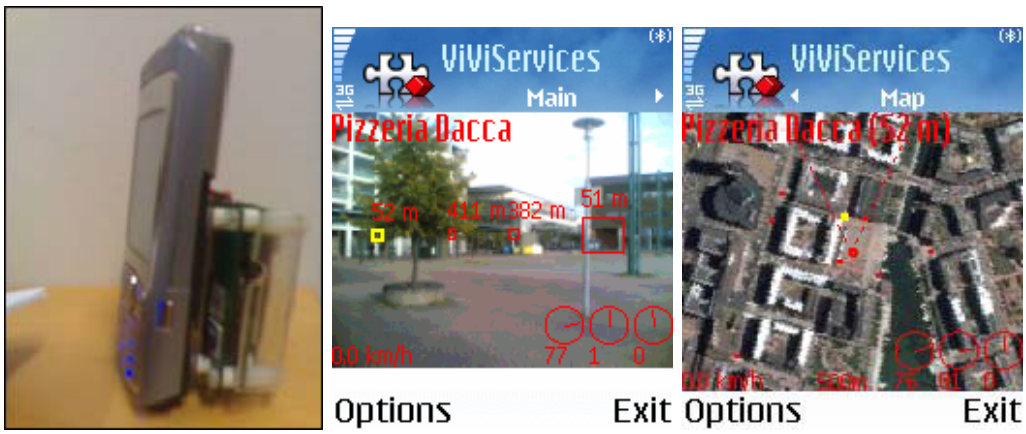
Figure 2.1: The Touring Machine by Feiner et al. from [17]. (a) Prototype of the hardware and accessories of a campus information system (the Touring Machine). (b) View-shot through the see-through head-worn display, showing campus buildings with overlaid labels.

Later, different platforms supporting and visualizing different types of media via augmented reality were presented. Like, the NEXUS architecture by [29], which offers the presentation of virtual information in the real world by supporting a spatial model and network requirements. Or, the Real-World Wide Web by [34] where the World Wide Web is presented with AR. In the previous two cases of augmented reality browsers, locations are presented as Uniform Resource Locators (URLs) and an embedded visualization of the corresponding web page. The Worldboard by Spohrer [52] takes a further step to present real world locations in combination with the current location retrieved by the Global

Positioning System (GPS) and shows the locations with additional information.

It should be mentioned that there are also non-visual concepts for augmented reality, like direct augmentation by geo-located post-its [48] or an audio based augmented reality approach by Bederson [5].

The trend of users for an active contribution in the Web and advanced Web technologies led to the Web 2.0. These developments had an impact on AR too. Therefore, Schmalstieg et al. [49] presented a concept of the Augmented Reality 2.0 by applying the idea of the Web 2.0 to the conventional augmented reality. In Augmented Reality 2.0 it should be possible for users to create and edit own content on place with interfaces provided by AR. In the recent years the evolution in the sector of mobile devices with a wide range of sensors ('smart phones') made it cheap and easy to do research in the field of AR browsers. Here, should be mentioned *MARA* (Mobile Augmented Reality Applications) from 2006 by Kähäri and Murphy [33], which was one of the first AR browsers on a mobile device. The hardware of MARA can be found in Fig. 2.2(a). It is a cell phone with camera support and additional equipment for positioning and orientation determination. The implementation of MARA supports a see-through-mode (see Fig. 2.2(b)) and a separated map-mode (see Fig. 2.2(c)). Furthermore, the concept of using layers to display different



(a) (b) (c)

Figure 2.2: Impressions of the MARA system from [33]. (a) The hardware of MARA: mobile imaging device (cell phone with camera support) with additional hardware. (b) The see-through-mode in MARA. (c) The map-mode in MARA.

types of information was presented by Lee et al. [37]. In their work they bother with the representation of data of an augmented web browser. Therefore, Lee et al. propose to

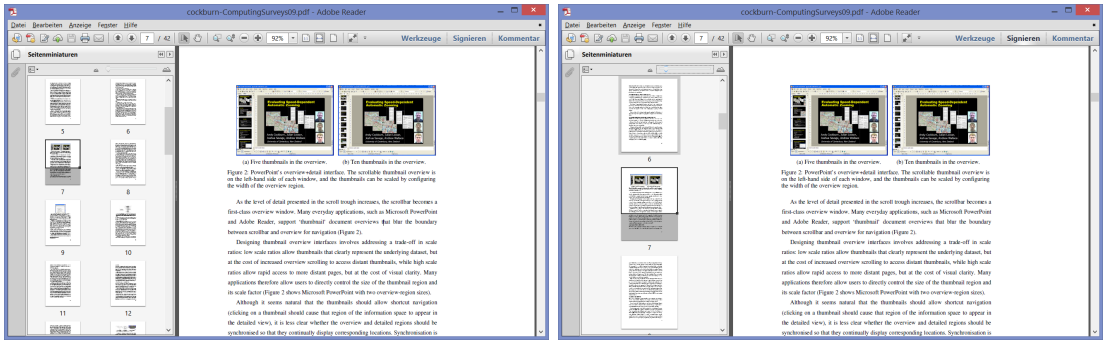
use layers to present different types of media in the augmented web browser in contrast to a single layer representation of the conventional HTML. In addition to research in AR browsers, data formats for editing and interchanging the augmented reality data are needed. One of such formats is KARML [32], which was first introduced in the ARGON browser [41] by MacIntyre et al..

Furthermore, there is a large variety of commercial and open-source AR browsers available on the market and can be found in different application stores. Here should be mentioned some popular AR browsers, like Wikitude by Wikitude GmbH [57], Layar[36], Junaio by metaio GmbH [42] and HERE City Lens by Nokia [46], which were introduced in Chapter 1.

2.2 Presentation of Context with Overview and Detail

The presentation of context is an important task of an AR browser. The user has only a small field of view represented on the screen of the device, thus only a small cover of the surrounding area is given. Therefore, she needs an overview of nearby POI. Furthermore, there should be a correlation between the depicted POI on the screen and their real position in the world. In [11] Cockburn et al. present a comprehensive survey over different context presentation methods like overview+detail, zooming and focus+context. Overview+detail is a presentation method to show both, the overview and detail of the data, within a single view simultaneously. The zooming presentation method uses as temporal separation of the views. Whereas, the idea behind focus+context is the elimination of the spatial and temporal separation by displaying the focus within the context in a single continuous view ('fisheye view'). The idea of overview+detail is often used in different document readers or presentation editors. An example can be found in Fig. 2.3. In the examples the implementation of overview+detail is shown in the case of the PDF-viewer software *Adobe Reader*. Another application of overview+detail is the usage of overview maps in online map services. Here, the current (zoomed) region of interest is annotated in a smaller map showing a larger region.

In the implementation of the AR browser I use two concepts presented by Cockburn et al.. On the one hand, the concept of overview+detail, which is the use of an overview map within the AR browser. On the other hand, the concept of zooming which is applied by using different levels of cluster data representation. AR browsers, i.e. Wikitude [57], use small maps, often called 'radar maps', with annotations of the nearby POI and the possibility to open a larger view of the map. The idea of the usage of a radar for this



(a) (b)

Figure 2.3: The overview+detail interface of Adobe Reader (PDF-Reader). Thumbnails of the previous, current and following pages are shown on the left-hand side. The size of the thumbnails can be changed and defines the number of shown thumbnails. Additionally the current shown region is annotated. (a) Small thumbnail size: 8 thumbnails are shown. (b) Large thumbnail size: only 3 thumbnails are shown.

purpose was proposed as *InfoRadar* by Rantanen et al. [47]. In [20] Fröhlich et al. show in their study that users rate such radar maps lower than other proposed alternatives. Therefore, I neglected the usage of a radar map in my suggested approach of the AR browser. My later presented approach uses another way of representation for nearby locations.

Another way to give the user an enhanced FOV (110 degrees) of the surrounding POI, but not a full overview, is the Context Compass as proposed by Lehikoinen and Suomela [38]. The Context Compass allows to show accessible objects within a linear drawn compass. In the one-dimensional compass the objects, regardless of their actual vertical position, are always shown on the same vertical position.

Mulloni et al. [43] present an approach by generating two overview types. First, the classical map view (exocentric view) is used, which is shown by pointing the mobile device on the floor. This gesture re-samples the behaviour with a physical map. The other overview method is an egocentric view mapping, which allows to show 360° in the user's limited field of view. An example of both zooming interfaces is depicted in Fig. 2.4. The example shows the case, where the user turns the camera of the device to the right side. The egocentric zoom is pictured in the upper part of the image and the exocentric zoom in the lower part of the image. In my approach, I adopted their idea of showing a map when pointing the camera of the mobile device on the floor.

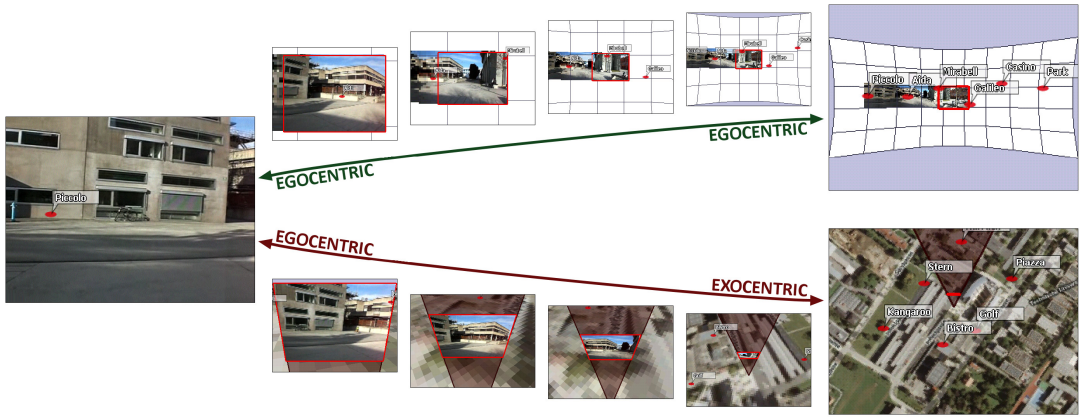


Figure 2.4: The proposed zooming interfaces from [43]. In the example the camera is turned to the right side. They propose an egocentric zoom that increases the field of view up to 360° (top) and an exocentric zoom that gives the user a top-down view onto the information (bottom).

Alongside with methods to present `POI` in general for the user, there are methods to draw the attention of the user directly to a specific point of interest. Baudisch and Rosenholtz [3] presented *Halo*, a technique to visualize off-screen locations in a map application. They use a visual cue, represented as a circle around each location, to draw attention to nearby locations. The size of the circle is proportional to the distance of the location outside the visual frame (see Fig. 2.5). Furthermore, they tested other representations for this problem, like the usage of arrows, which point in the direction of the corresponding location.

Figure 2.5: Animated example of the *Halo* technique from [3] (from left to right). The location (red arc icon with letter R) is moved left and disappears over time. A circle emerges with the center at the position of the location and the radius increases with the distance to the border of the frame.

Another, more advanced and newer method for supporting focus and context is *Around-*

plot by Jo et al. [30]. In their work, they propose a technique to project and represent objects which are not in the user's FOV by using cues for them. Therefore, the off-screen data points are mapped from the 3D environment to a 2D orthogonal fisheye. In Fig. 2.6 an usage example of Aroundplot is depicted. It shows an animation of the usage, beginning at the display of a cue for an object at the left margin up to the annotation of that object.

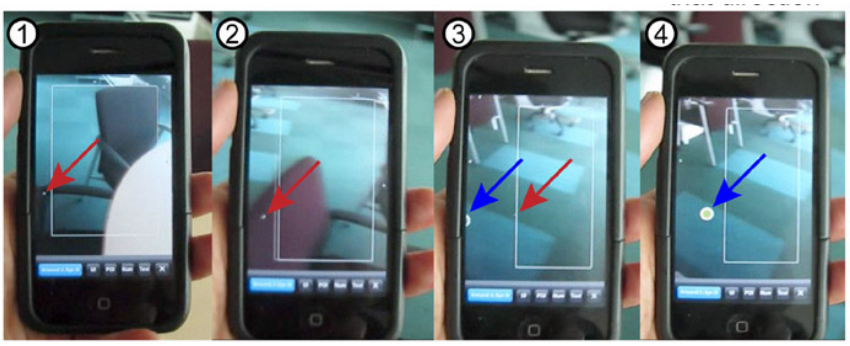
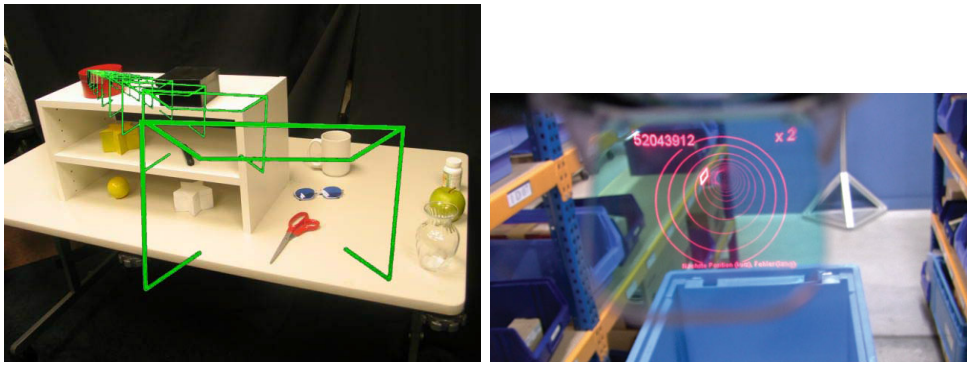


Figure 2.6: Animation sequence showing the usage of Aroundplot. (1) a cue for an object; (2) left margin is magnified; (3) the cue enters the boundary; (4) the object appears from that direction. Adapted from [30].

There are different approaches to draw attention in AR to a specific object in the real world. Biocca et al. [6] propose the 'attention funnel'. In their approach the user is guided by a funnel to a specific object by reacting to the user's position and (head-)orientation. An example of the attention funnel can be found in Fig. 2.7(a). In this case a virtual funnel shows the path to the desired object. Another approach of a guided object finding technique is *Pick-by-Vision* by Schwerdtfeger et al. [50]. An image of the implementation is depicted in Fig. 2.7(b). In the image the additional information to the attention drawn object is shown.

One of the main problem of such guiding methods is, that the user has to know what POI she is looking for. This may be useful in applications, i.e. logistic optimization in warehouses or at assembly and maintaining tasks in the auto-mobile industry. For the locating of a specific POI, I have implemented a slightly simple method for a supported search in my work. The method allows to be guided by AR to the POI by the usage of indicators pointing in the direction of a target point. The implementation of the AR location search will be discussed in detail in Section 4.5.



(a) (b)

Figure 2.7: Two examples of possibilities to draw attention to a specific object. (a) The attention funnel approach by Biocca et al. [6]. The attention is drawn to the red box on the shelf. (b) The *Pick-by-Vision* system from [50]. The user is guided by a bending circular tunnel to the object.

2.3 Filtering and Cluster Analysis

The data filtering can be solved in common by three different approaches: spatial filtering, knowledge-based filtering and a combination of both filtering methods. In [18] Feiner et al. present KARMA (Knowledgebased Augmented Reality for Maintenance Assistance), which uses a knowledge-based filtering algorithm. In KARMA the relevant information is selected by a rule-based approach and represented with `AR`. Their presented KARMA system prototype uses as example the application of a step-by-step instruction through a simple end-user laser printer maintenance.

On the other hand, a spatial filtering algorithm for augmented reality is proposed by Julier et al. [31]. They present their filtering algorithm for the application of a sniper avoidance system for urban areas. Therefore, their algorithm uses region-based filtering by using objective and subjective properties.

I will use spatial and knowledge-based filtering in my approach. So, the spatial filtering uses the quite simple approach of a threshold for the distance of an object. The knowledge-based filtering takes account of the properties (categories) of the `POI` beside the coordinates.

The filtering of the whole data reduces the number of data points, but there may be still to many of them to present the data. Therefore, a cluster analysis can be performed to find and cluster data points, which satisfy different criteria. The way to perform

cluster analysis (clustering) on a given n-dimensional data set can be generally segmented into four strategies. The first strategy is to perform a connectivity based clustering or hierarchical clustering. In this method, which is often used in data mining, the goal is to build hierarchies of cluster. The main idea is that nearby data points are more similar and belong to a cluster.

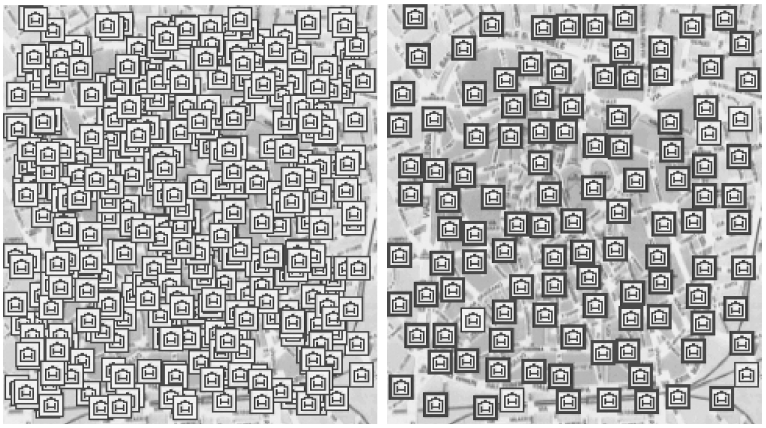
The second approach is to perform the cluster analysis centroid-based. Hence, its goal is to find cluster centers and data points which corresponds to this cluster center. The need that the cluster centers have to belong to the given data set is not necessary. One of the most popular algorithms is Lloyd's k-means algorithm [39], where the number of cluster centers is fixed.

Another way to find cluster within a data set is distribution-based clustering. This cluster analysis approach is related to statistics and based on distribution models. The goal is to find distributions that resembles the given data. The Expectation-Maximization algorithm by Dempster et al. [14] is one of the most popular algorithms to find a solution to this type of cluster analysis problem.

The density-based clustering approach tries to find areas of higher densities within the data set. Data points outside such areas of higher densities are seen as noise in the density-based cluster analysis. One of the popular clustering algorithms at density-based clustering is DBSCAN [16].

The above mentioned strategies and cluster analysis approaches are adopted to deal with the problem of cluttering of icons on map-based applications. Ellis and Dix [15] present in their work a taxonomy of different clutter reduction methods for the area of information visualization. They classified the different clutter reduction techniques, i.e. filtering, clustering, topological distortion, animation and several different other. Additionally, they tested the techniques for different criteria.

In [7], Burigat and Chittaro present some techniques to deal with clutter of icons occurring on mobile devices. Their approach tries to aggregate nearby icons to get rid of cluttering of the icons. An example of the results of the aggregation algorithm proposed by Burigat and Chittaro can be found in Fig. 2.8. de Matos et al. [12] compare the use of different generalization operators to handle the overlap of icons on maps. Furthermore, they performed a survey how these operators help to increase the usability. In [1] methods are presented by Andrienko et al. how cluster analysis can be used to find cluster in complex structured objects, like trajectories of moving objects on maps. They used in their work the generic-density based clustering algorithm OPTICS [2]. Hierarchical clustering is used



(a) (b)

Figure 2.8: Decluttering approach by the aggregation of icons on a map from [7]. (a) The original map with 400 icons without any aggregation. (b) The result of the proposed aggregation algorithm.

by Delort [13] to visualize large data sets of spatial data in web based mapping applications. Here, the clustering is performed for different scale levels of the shown map and it handles cluttering as well. Another paper by Guo et al. [23] presents different approaches of dealing with multi-variant spatial data by using (subspace) hierarchical clustering. Yang et al. [58] suggest a way to find POI in metropolitan areas from geo-tagged photos by using a self-tuning clustering algorithm. In their work they compare popular clustering techniques, like k-means and DBSCAN, with spectral clustering with self-tuning parameters.

The clustering algorithm presented in this thesis is a centroid-based approach, but uses not the often used k-means algorithm. I will show how the presented clustering algorithm works and discuss the advantages of my approach against other clustering algorithms.

2.4 Representation and Labelling

The representation of data on a screen (of a mobile device) is a recent topic in research since years. Especially in AR applications the representation of the given data and the possibility to interact with the data is important. Höllerer et al. [26, 28] present a pipeline (see Fig. 2.9) which describes how the data has to be processed. This pipeline starts at the information filtering, to the User Interface (UI) component design and the view management to the final representation on the AR display of a MARS.

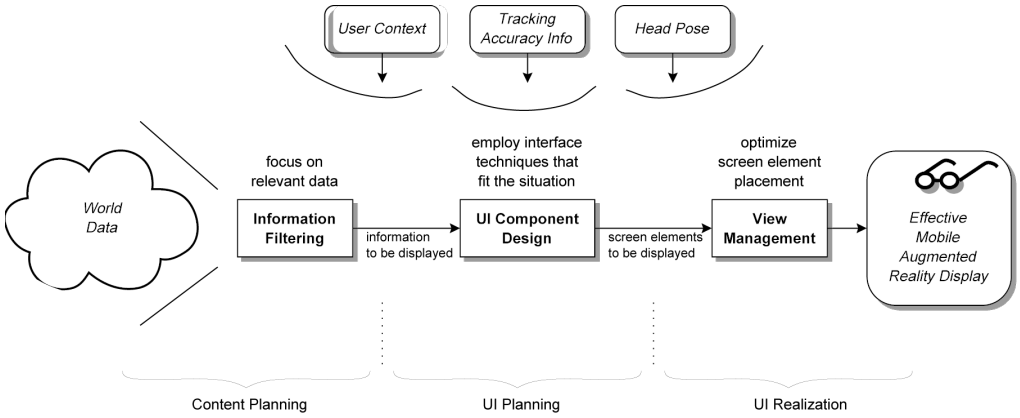


Figure 2.9: The MARS UI management model [26]: information filtering, UI component design, and view management.

The usage of text and icons is not the only way to represent data. There exist a large amount of glyphs, which can be used to visualize data in an easy to understandable way. Ward [56] give a taxonomy over different types of glyphs and how to place them for data-visualization in a meaningful way. In Fig. 2.10 some examples for glyphs from [56] are depicted.

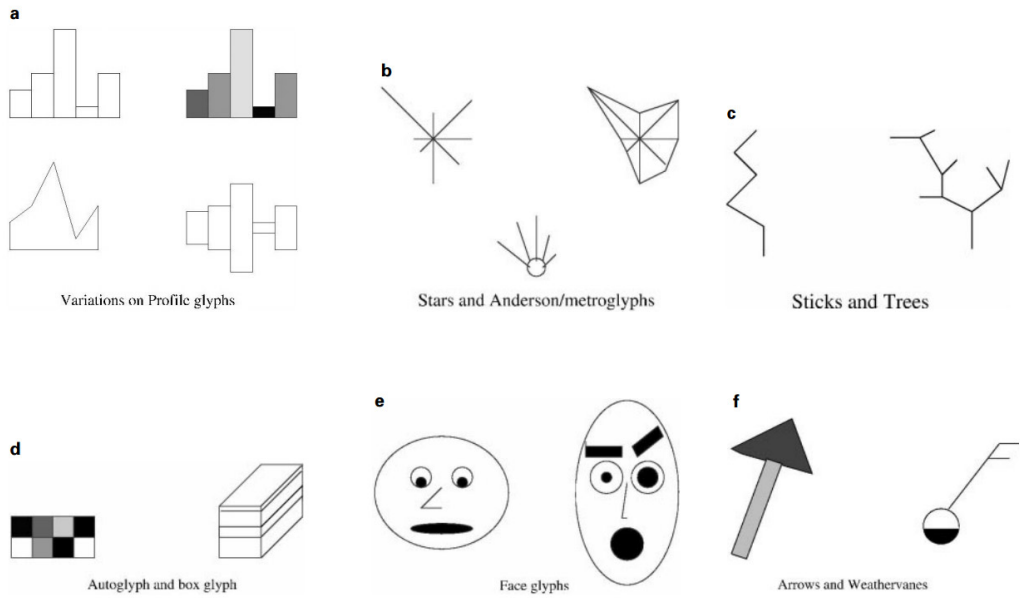


Figure 2.10: Examples of glyphs from [56]. Top: (a) variations on profile glyphs; (b) stars/metroglyphs. (c) stick figures and trees. Bottom: (d) autoglyphs and boxes; (e) faces; (f) arrows and weathervanes.

Chittaro [10] describes in his work the importance of the data representation on mobile devices. Therefore, possible solutions are described to deal with the problem of the representation of different data types on the screen.

The MoViSys (Mobile Visualization System) by Carmo et al. [8] proposes different types of visualizations for POI on mobile devices. Their idea is to avoid clutter on maps of mobile devices by using a single icon to present different types of POI, categories and even detail information. Therefore, cluttering icons of places on the map are aggregated to one single icon. This aggregation is based on a grid based clustering of the elements within a cell of the grid. Furthermore, the user's possible interest for every POI is calculated and used for filtering issues. The usefulness of a grid based aggregation for places in an AR browser is discussed later in Chapter 4. In Fig. 2.11 examples of the proposed visualizations for points of interests are depicted.

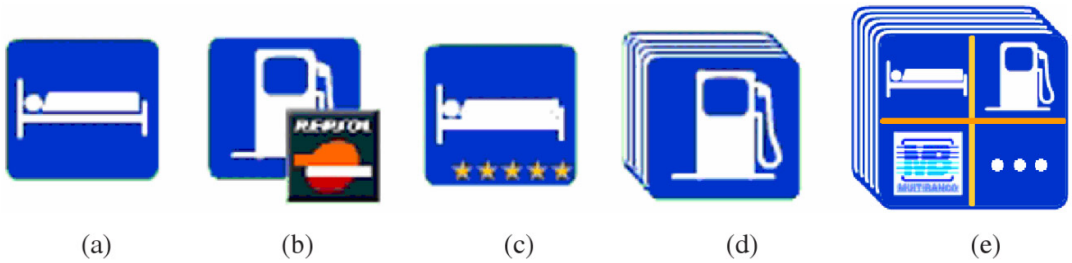
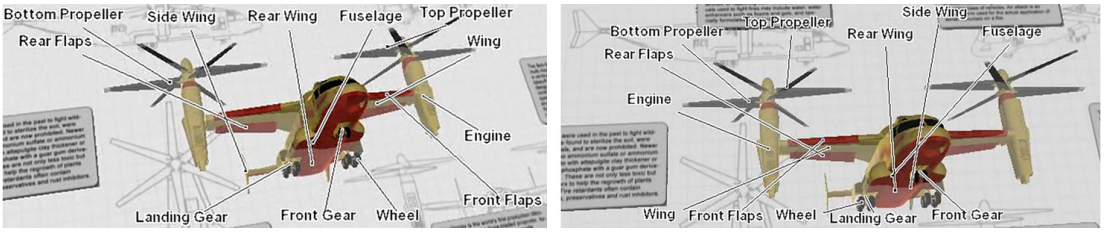


Figure 2.11: Visualizations of POI from [8]: (a) individual symbol; (b) and (c) individual symbol with detail; (d) aggregation; (e) multiple category aggregation.

In [55], Tatzgern et al. describe the problem of information overload occurring with the usage of labels and annotations in AR applications. The problem is discussed on the example of 3D model exploration and labelling in augmented reality. In their presented method for annotation layouts, they adopt the force-based approach of Hartmann et al. [25]. Furthermore, they proceed on a combined filter and layout approach which was developed for explosion diagrams [54]. They try to deal with the problem of distractions between the labels and the leader lines connecting to the corresponding part of the 3D model. This distractions may be intersecting leader lines or flickering labels and annotations. The approach of using a fixed order for the labels and position of the anchor points will cause intersections of the leader lines (see Fig. 2.12(a)). So, they try to solve this problem by maximizing the distribution of the annotations of the layout. Thus, this method reduces the appearance of distractions, but they cannot be fully eliminated. The result of the method can be found in Fig. 2.12(b).

In [4], Baudisch et al. present *Starburst* for labelling data points on a map. It creates



(a) (b)

Figure 2.12: Methods for minimizing potential distractions from [54]. (a) Fixed order of the labels and position of the anchor points causes intersections. (b) Maximizing the distribution of the annotations of the layout: reduced number of intersections between lead lines.

tiles which belong to an individual data point on the map. So, their method aims to improve the selection of an individual data point on a touch sensitive screen. Baudisch et al. have run a user survey to show the advantage of their presented *Starburst*-method against a Voronoi-cell labelling on different cluster types (uniform-, loose- and tight-cluster). The results of their survey show, that the proposed method will lead to smaller error rates and shorter targeting times for data points on a touch sensitive screen. In Fig. 2.13 the results of the *Starburst* and a Voronoi tessellation of different cluster sizes are depicted. The *Starburst* method for selectable tile creating could be useful at the selection on mobile devices. Anyway, the method does not handle the positioning of labels filled with text and/or icons.

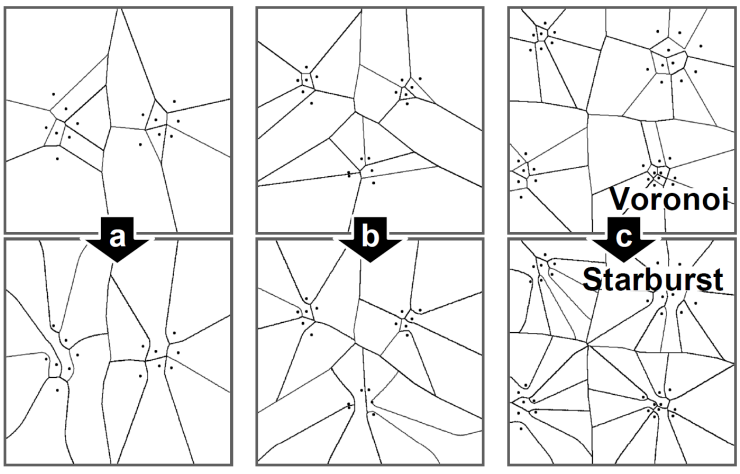


Figure 2.13: Starburst tiles algorithm by [4]. Results of Voronoi (top) and *Starburst* (bottom) tessellations for layouts with (a) 2, (b) 3, and (c) 4 clusters.

Maass and Döllner [40] use a 2.5 dimensional approach for labelling planar maps and 3D landscapes. Therefore, they compute the depth value of each annotation point and sort these points in ascending (near-to-far) order. Afterwards the position on the view plane is determined for each point. Their proposed view management for placing labels uses interval-slots which use intervals to indicate space occupied by labels. An example of their approach of placing labels is depicted in Fig. 2.14.

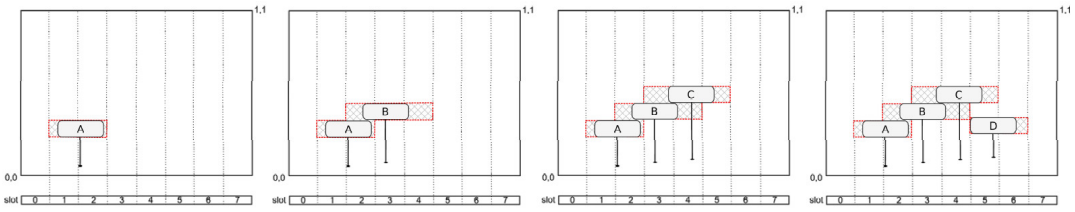
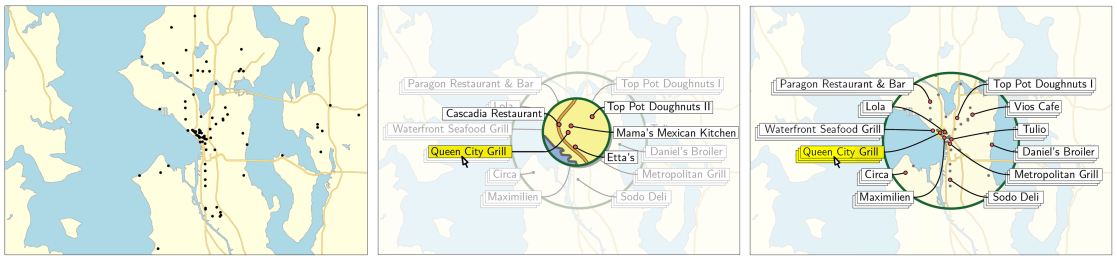


Figure 2.14: Placing labels (A-D) using the interval-slot view management proposed by Maass and Döllner [40].

Another labelling approach is presented by Fink et al. [19] for labelling focused regions on maps or diagrams. They present their idea to determine label-leaders and generate label-stacks to avoid overlap of labels. Furthermore, Fink et al. show methods to avoid intersections between the connections of label and annotation. Additionally, they discuss different ways to draw the leader lines.

In my work I adopted and combined the ideas of the interval-slot view management [40] and the labelling of focus regions [19] to deal with sub-cluster labelling. For the leader lines, I use a more simple approach than the proposed usage of Bézier-Curves of [19].



(a) (b) (c)

Figure 2.15: Labelling of focus regions as presented by Fink et al. [19]. (a) Map of a city area (Seattle) with different locations (black dots). (b) The labelling after selecting a focus area on the map. Each label is connected to location or a cluster of locations on the map. (c) The detail labelling after clicking on a label representing a cluster.

Chapter 3

Concept

Contents

In this chapter the concept behind the implemented AR browser is described and discussed. Therefore, the following sections will explain the aspects of the chosen approaches. Furthermore, the layout of the user interface and the possible tasks which can be performed with the AR browser are described in detail. The implementation of the AR browser will be discussed and described later in detail in Chapter 4.

The AR browser is set up hierarchically in different visualisation types for the given data set. These visualizations will be discussed in the following sections. An organization chart of different visualization ways is depicted in Fig. 3.1. The implementation of the AR browser is separated in two viewers for the data: the *Map Viewer* (see Section 3.6) and the *AR Overview+Detail Viewer* or simply the *AR Viewer* (see Section 3.1). The map viewer holds a only map for an overview of the nearby POI. In the AR viewer the data of the points of interest is visualized by using a combination of overview+detail techniques. Whereas, the map viewer only gives an overview of the given data. Due to the nature of the viewers, only one of both can be shown on the screen of the mobile device.

The augmented reality viewer uses clustering to generate cluster for an overview of the data. The result of the cluster analysis task is the *Cluster Data Visualization*, which is discussed in Section 3.2. This visualization uses different labels depending on the cluster size. The members of the clusters are displayed in a different way as their representing cluster. This visualization level is called the *Detail Visualization* of the clustered data (Section 3.3). The main task of the data detail visualization is to solve the problem occurring with the small spatial distance between the (projected) data points. Therefore, in this visualization a fixed layout for the labels is used. For the filtered data, there are alternative visualizations of the cluster and detail data. These alternative representations are described in Section 3.4. Furthermore, the AR viewer holds an integrated overview map of the POI. The features and idea behind this map are discussed in Section 3.5.

In contrast to the visualizations of the whole data set, the AR overview+detail viewer allows to execute an *AR Location Search* (Section 3.7), which means a guided search for a single POI. The AR location search will turn off the other visualizations except the integrated overview map.

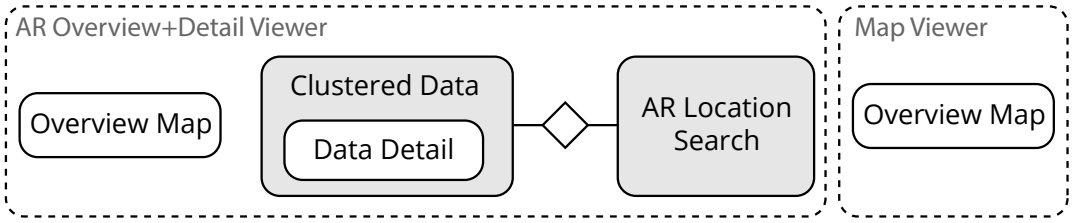


Figure 3.1: Organizational chart of of the AR browser: the AR overview+detail viewer uses either different visualizations for clusters and their members or holds an AR location search. Additionally, this viewer has an integrated overview map. The map viewer holds an overview map only.

3.1 AR Overview+Detail Viewer

The screen-space of a mobile device is limited. Therefore, the user interface is designed to use the maximum of the screen-space to show the user what is in front of her. An image of the user interface of the AR overview+detail viewer can be found in Fig. 3.2. Only a small fraction of the screen space is occupied by a semi-transparent, so called ActionBar (AB) (see Fig. 3.2 - (1)). The AB allows the user perform several tasks, like the search possibility of a POI or updating the location manually. This so called **actions** are annotated in Fig. 3.2 at (1.a). Furthermore, a context-menu allows the access to less often

needed actions, i.e. changing the zoomlevel of the map or set the location manually. The context-menu is opened by touching on the icon at the annotated area in Fig. 3.2 - (1.b).

The implementation of the later discussed clustering method will lead to the clustered data visualization. The cluster analysis results in two cluster types. A single data point cluster which represents only one place of interest. Such a `POI` has enough distance to other `POI` on the screen and will be labelled in detail (see Fig. 3.2 - (2.a)). Whereas a cluster of more than one `POI` has a different and more simple representation on the screen. The visualization of a multi-item cluster can be found at annotation (2.b) in Fig. 3.2.

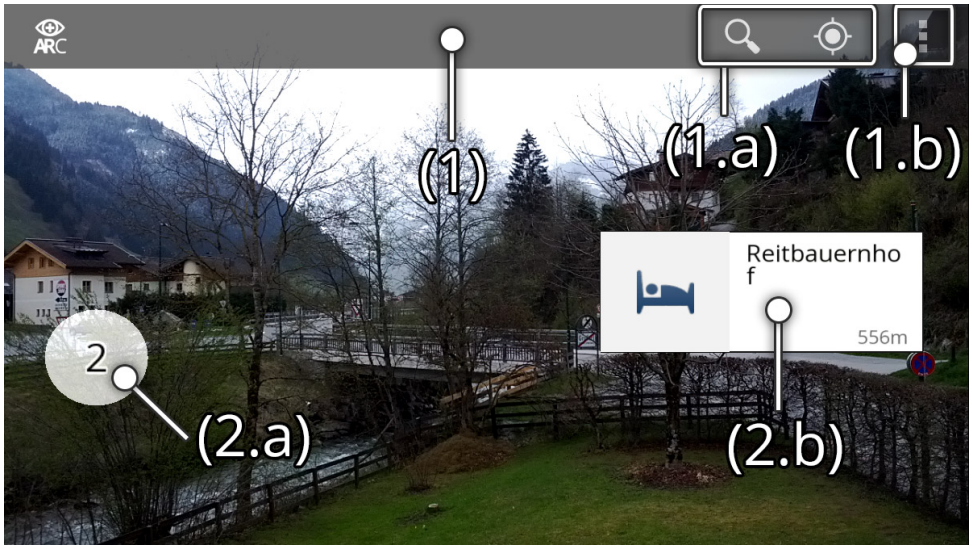


Figure 3.2: `AR` browser: `AR` overview+detail viewer. (1) is the `AB` with icons to perform actions (1.a) and the icon to open the context-menu (1.b). The different cluster visualizations: (2.a) label for a cluster and (2.b) detail label.

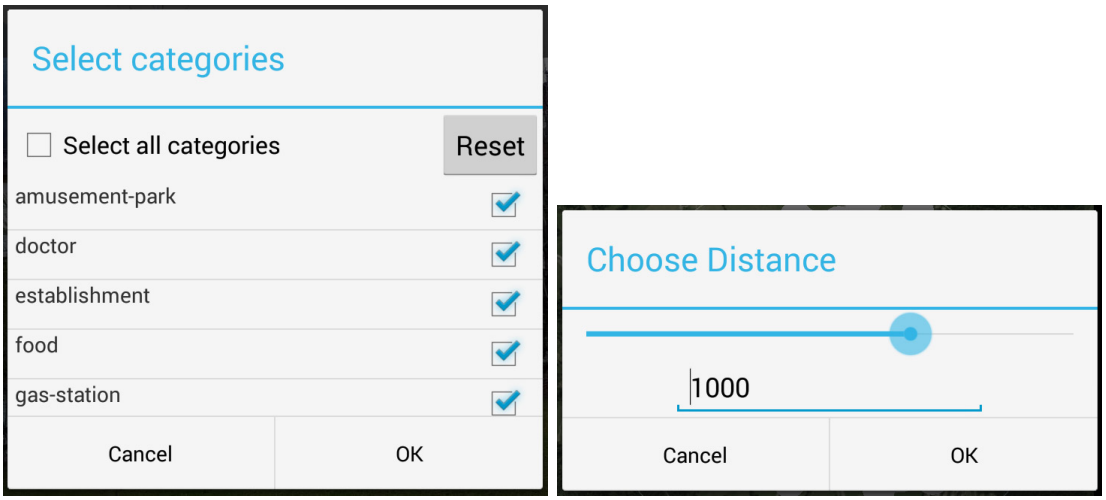
The `AB` context-menu (see Fig. 3.1) offers different actions as mentioned before. The first menu entry 'Set categories' opens a dialog (see Fig. 3.4(a)), where the user can select from different categories for the shown places. After the selection of some categories, the matching `POI` will be selected and be ranked into three groups: bad results (one match), good results (some matches) and perfect results (all categories match). The next entry 'Set distance' allows to change the radius for the distance based filter for the points of interest (see Fig. 3.4(b)). Therefore, the user can use the slider to change the distance smoothly or set the exact value in a input box. It should be mentioned that the distance has a predefined maximum value.

The 'Mapsettings' entry opens a dialog to change different settings for the shown overview map, like changing the visibility of the integrated map, setting the type, and

Figure 3.3: AR browser: the opened context-menu in the AR viewer (1).

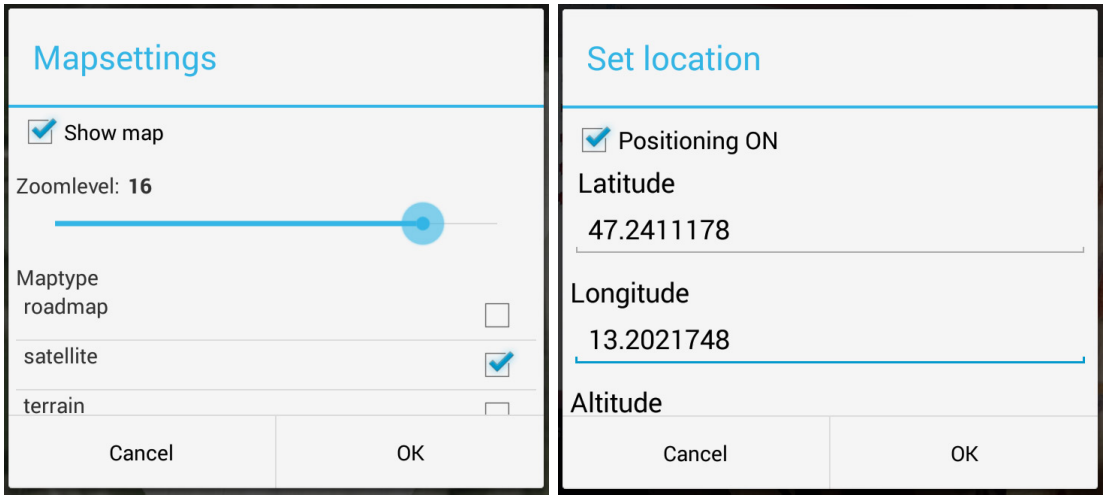
changing the zoomlevel of the map. This dialog is shown in Fig. 3.5(a). After the change of the settings a new map will be requested from the Google Static Maps API.

Another interesting context-menu entry is 'Set location'. The 'Set location'-entry allows to change the position manually in an appearing dialog and to set the state of the automatic location updates. An image from this dialog can be found in Fig. 3.5(b).



(a) (b)

Figure 3.4: AR browser: different dialogs of the AR viewer context-menu opened (I). (a) The 'Set categories' dialog and (b) 'Set distance' dialog.



(a) (b)

Figure 3.5: AR browser: different dialogs of the AR viewer context-menu opened (II). (a) 'Mapsettings' dialog and (n) 'Set location' dialog.

3.2 Clustered Data Visualization

There are two different result types after the clustering process as mentioned before. The first result type is a cluster which consists only of one element. Thus, it is possible to label this element in detail with the name of the location, additional information and a possible icon. The used clustering analysis method ensures that a label is not occluded by other labels. I will discuss the implemented clustering algorithm for the clustered data visualization in detail in Section 4.4.1. An impression of the clustered data visualization can be found in Fig. 3.6 - (2). A label for a single item cluster consists of an icon, which indicates the category of the place, the name of the place and the linear distance. The additional information is shown in a dialog by tapping on the label (see Fig. 3.7). In Fig. 3.7 - (1) the details of the POI are shown the opened dialog. The direct connection between the POI annotated on the screen and on the overview map can be shown. Therefore, the 'Show place on map'-button has to be clicked (see Fig. 3.7 - (2)). Additionally, the categories are shown in a list at (3) in Fig. 3.7.

The second type of cluster representations (see Fig. 3.6 - (1)) on the screen will be shown at the position of multi-item clusters. Such a cluster is visualized by a circle and a number, which represents the number of points within this cluster. By tapping on the

Figure 3.6: AR browser - clustered data visualization: labels for cluster (1) and a label with details (2) for a single-item cluster POI.

circle, a new visualization of the cluster members is shown: the detail visualization for the clustered data. This visualization will be discussed in detail in Section 3.3.

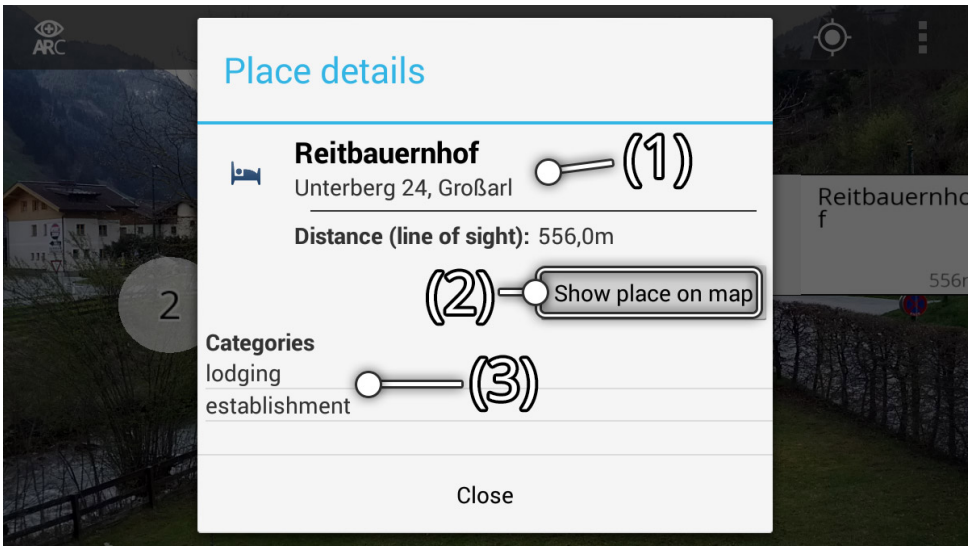
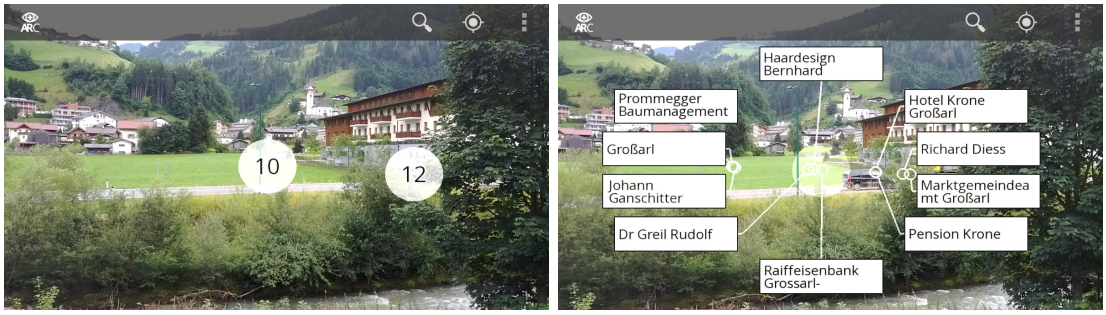


Figure 3.7: AR browser: the detail dialog of a selected POI . (1) Details about the POI. (2) shows connection between point on screen and on map. (3) the categories of the location.

3.3 Detail Visualization of Clustered Data

After tapping on the label of a multi-item cluster, the display changes to the cluster data detail visualization. In this visualization the leading n-elements of the cluster are shown. The number n of elements is a fixed number and it is determined by the available spatial screen space and pixel density. Due to the fact that the orientation of the device changes permanent in the world reference frame, the possible labels for the `POI` have a fixed position in relation to the center of the cluster. Such a layout makes it easier for the user to select a label to retrieve additional information about the `POI`. Fig. 3.8 gives an insight into the idea of the layout of the cluster detail visualization. The proposed approach for labelling the cluster detail is as mentioned before in the related work an adaptation and combination of the approaches by [19, 40].

Each annotation for a `POI` is represented as a circle at the projected location of the data point on the screen. This annotation is connected via a leader line to the corresponding label. It could be possible that there is only a label shown. This will be the case, if the label is in front of the location.

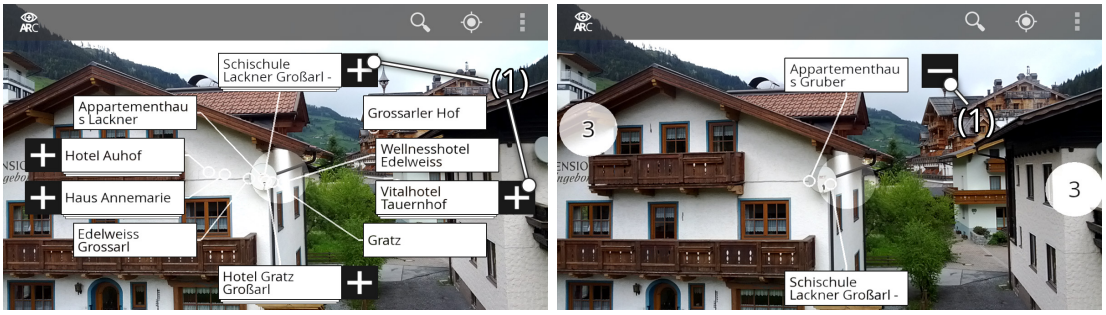


(a) (b)

Figure 3.8: `AR` browser: (a) visualization of clustered data (two cluster); (b) cluster detail visualization for a cluster.

In case if there are more points within the cluster than possible labels, a hierarchical menu structure is generated and shown. The user can spot such a hierarchy within the cluster by an alternative label layout. In the alternative label layout the label symbolizes a stack of 'cards' and a plus-sign-icon is attached to the label (see Fig. 3.9(a) - (1)). A touch on the plus-sign-icon leads the user to the next level of the hierarchy of the current selected label. If the user wants back to the previous level of the hierarchy, she can navigate to the previous level by tapping onto the appearing minus-sign-icon (Fig. 3.9(b)). The cluster

detail visualization is closed, when the user touches the cluster center (semi-transparent white area) or points the device away from the cluster center. Through the different levels of the hierarchy the representation is always centered at the primary center position of the whole cluster. This means, that the center is not computed as the mean value of the points for each level of the hierarchy.



(a) (b)

Figure 3.9: AR browser: the detail visualization of the clustered data for a selected cluster with hierarchical menu structure. (a) The first level of the hierarchy: at (1) labels with additional levels can be found. (b) The last level of the hierarchy: a touch on (1) allows to get back to the lower level of the hierarchy.

3.4 Filtered Data Visualization

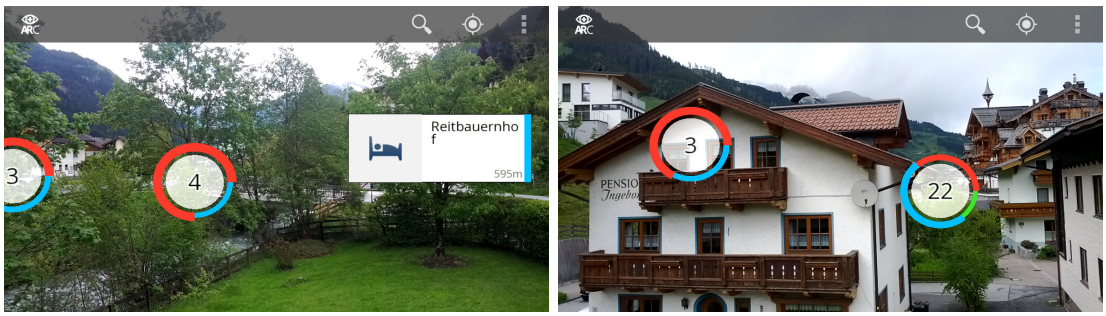
The AR browser allows to filter the data set by selecting from different categories for the given POI. Therefore, the quality of the results of the category filter have to visualized in an understandable way. Therefore, there is an alternative visualization for the clustered data and the details of the clustered data. So, the labels of the cluster are used to indicate the quality of the results of the category filter. The quality can be represented in different ways, i.e. in percent, as rank or by choosing an appropriate symbol. But, these representations will need extra space on the labels or be more confusing than helpful. Therefore, in my approach the quality of a search results is indicated by different colored glyphs. The distinct colors and the glyph design will allow to spot at first sight the difference between the results. There are four possible states and according colors:

- no match
- match in one category: bad result (color red)

- match in one or more categories: good result (color blue)
- match in all selected categories: perfect result (color green)

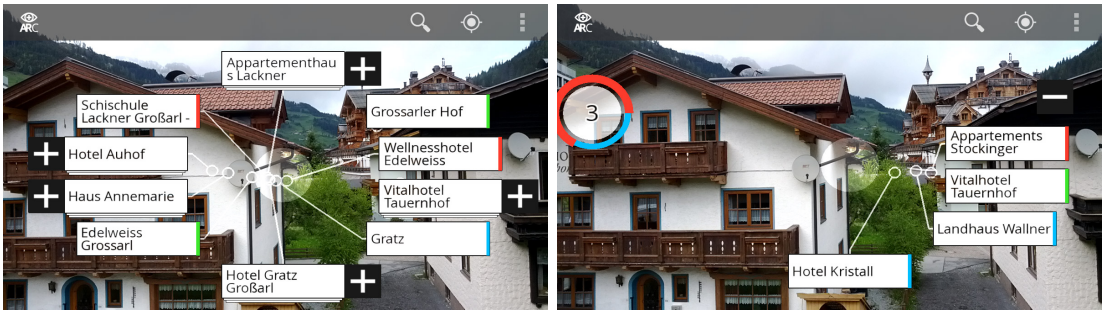
The results of the first result state - no match - are excluded from the data set and not shown in the clustered data visualization. The remaining subset of data points is used to create a new clustering. After the clustering, quality of a single-item cluster label is indicated on the right hand side of the label.

The different qualities of the results for the matching `POI`, which are in the same cluster, are visualized outside the detail visualization of a cluster too. This visualization uses glyphs to express the quality of the filter results. Therefore, the cluster label is outlined by a (multi-)colored circle. This outline can be separated in sectors, where the size of the sectors is proportional to the number of data points which share the same quality within the cluster. Additionally, the thickness of the segment depends on the number of results of the same kind too. I have chosen this layout because it allows to give as much as information about the quality of the cluster content in an easy to understandable and space-saving way. This visualization can be found in Fig. 3.10. In the detail visualization for the clustered data the labels show the quality of the result on the right hand side, like the detail label of a single `POI`. If there is a label, which is the parent of another cluster, no indicator is shown. Both cases of labelling in the cluster detail visualization are depicted in Fig. 3.11.



(a) (b)

Figure 3.10: `AR` browser - clustered data visualization: representation after category-filtering. (a) indicators at multi-entry cluster labels (right) and a detailed label (left). Color and size coded indicator outline on the cluster label. (b) two cluster representations including one large cluster (22 data points) with all sort of qualities for the results. Color coding: bad results (red), good results (blue) and perfect results (green).



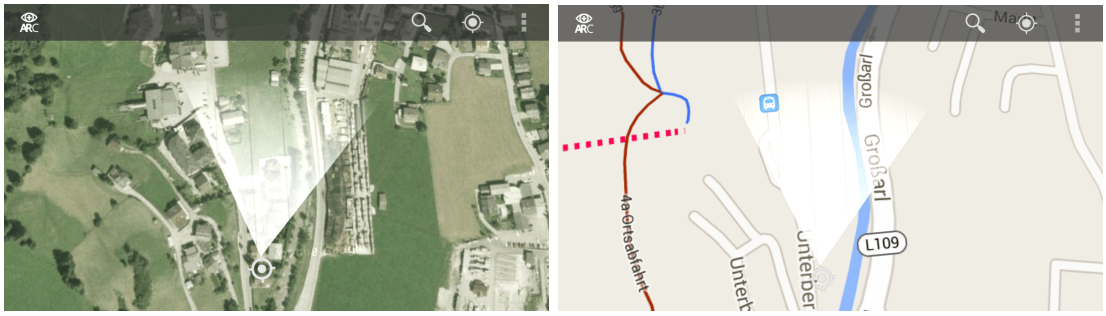
(a) (b)

Figure 3.11: AR browser - data visualization of clustered data: representation after applying the category filter. Color coded indicator in the cluster detail visualization. (a) the first level of the detail visualization. (b) the second level of a label in the hierarchy with color coding. Color coding: bad results (red), good results (blue) and perfect result (green).

3.5 Integrated Overview Map

An overview map is integrated in the AR overview+detail viewer, when the user points the camera of the device on the floor. This integrated map can be used by the user like a traditional map to find her current location. Furthermore, it is possible to select the type of the map (satellite, roadmap, terrain or hybrid) and change the zoom level of the map to get e.g. a better overview of the area from a dialog accessible over the context-menu. It is possible to zoom the map in and out. To do so, the user can use the pinch-to-zoom* gesture. It will move the map to the user or away. So, the map can be explored in detail by changing the orientation of the device. Some impressions of the map types can be found in Fig. 3.12 On the other hand, it has some additional features. The possibility to show the relationship between a POI in the real world and the corresponding location on the map is given. For this purpose the user's view from the POI in 3D space to the point on the map is guided by a line and annotated by a marker on the map. Furthermore, the results of the category filter are represented on the map by colored markers.

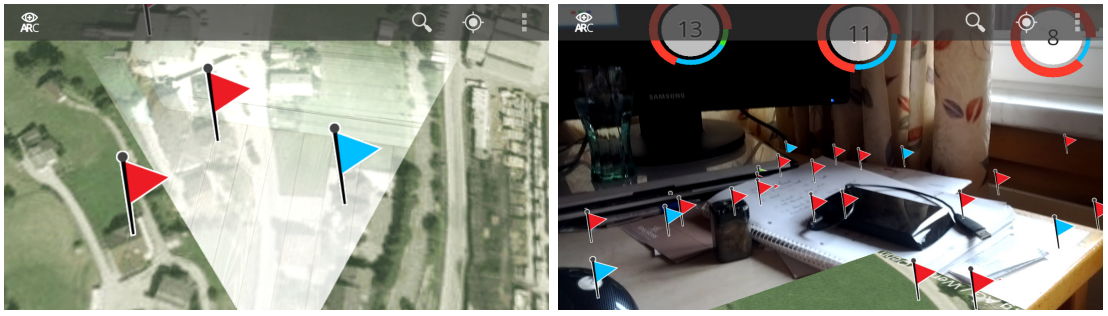
*pinch-to-zoom: change the distance between two fingers; proportion of the start- and end-distance define the scale



(a) (b)

(c)

Figure 3.12: AR browser - integrated overview map: showing different map types. (a) 'satellite'-map, (b) 'roadmap', (c) 'hybrid'-map.



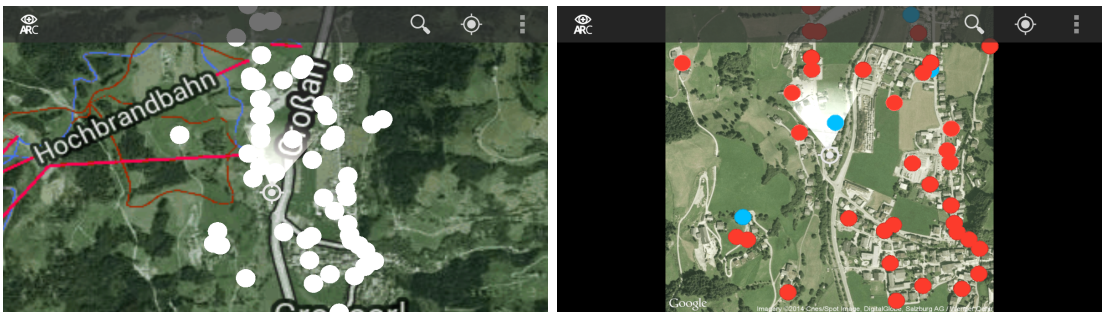
(a) (b)

Figure 3.13: AR browser - integrated overview map: marker representation of the filter results. In (a) the marker of the category filter can be found. (b) depicts the possibility of markers outside of the map. Color coding: bad results (red), good results (blue) and perfect result (green).

3.6 Map Viewer

If the user wants an overview map only, she can change to the map viewer. The viewer holds only an overview map of the current position and all nearby POI are indicated by markers. An example of the default map state is depicted in Fig. 3.14(a). The map in the map viewer can be zoomed in the same way as the map in the AR viewer. Additionally, it is possible to move the map. If the distance chosen by the corresponding dialog is within the map boundaries, a circle of the distance range will be shown as map overlay. So, the user is able to estimate the distance of the annotated POI.

If categories are selected, the markers will indicate the quality of the results in the same way as in the integrated overview map of the AR overview+detail viewer by using different colors (see Fig. 3.14(b)).



(a) (b)

Figure 3.14: AR browser - map viewer: (a) shows the default overview map with white markers for POI (zoomed in). In (b) the overview map with different marker colors indicating the quality of category filtering is shown. Color coding: neutral (white), bad results (red), good results (blue) and perfect results (green).

Both visualization methods for the overview map in the AR browser, the perspective visualization in the AR overview+detail viewer and the orthogonal visualization in the map viewer, have their advantages and disadvantages.

The perspective representation in the AR viewer allows primary to show overview and detail in one reference frame (AR overiew+detail viewer). Furthermore, this visualization approach allows to display the correlation between points on the map and in the real world. Major drawbacks are the limited possibilities for interaction and the display of additional information on the overview map. This will lead to an information overload within the limited screen area of the mobile device. Furthermore, the distortion of content

of the map due to the perspective rendering may be a small drawback.

In contrast, the orthogonal visualization of the map allows a better interaction of the user with the displayed map. Additionally, there is the possibility of the usage of existing, more dynamic, map displaying techniques, like Google Maps, to exploit the available screen space. But, this will lead to different approaches to handle labelling and the possible occurring clutter. One of the main disadvantages of an own map viewer is the separation of the overview and detail data, especially at the usage in an AR browser. It is more convenient for the user to stay in the same viewer to show the augmented world and the overview map. This disadvantage could be solved by an smooth transition between the different viewers without touch-gestures as proposed in [43] by Mulloni et al..

3.7 AR Location Search

The AR location search in the AR viewer allows the user to search for POI by name or address, and further to be guided to the specific nearby POI, which location she does not know. Therefore, she invokes the search by touching on the search symbol (a magnifying glass). The user can enter some keywords (see Fig. 3.15(a)). This starts a query for possible candidates of the desired POI. These candidates are selected and ordered by descending distance to the current location. The representation of the search results is shown in Fig. 3.15(b). By tapping on the left symbol (Fig. 3.15(b) - (1)) the user can start the AR location search after inspecting the list of results. The set of places from the result list can be show in the AR overview+detail viewer by selecting the 'Show results'-option in the dialog window (see Fig. 3.15(b) - (2)).

3.7.1 AR Guided Search

The AR guided search is shown in Fig. 3.16. All other POI disappear from the screen and only the selected POI is shown. To remind the user of the location, what she is looking for, the name of the selected POI is shown in the ActionBar (Fig. 3.16 - (1)). In the center of the screen some sort of 'crosshairs' can be found. In this case it is represented as a point at the center of the screen and a larger surrounding circle (see Fig. 3.16 - (3)). The radius of the outer circle around the center point represents the distance where the projected data point counts as hit and ends the AR location search.

Due to the fact, that the user normally does not know where the location is, she needs to be guided to the location. Therefore the AR location search provides two visual

indicators. First, there is an arrow on the edge, where she has to turn (Fig. 3.16 - (3.a)). The size of this arrow changes with the distance to the target. The size decreases, when the user comes closer to the POI and the size increases, if she turns away. Second, there is a circle, which indicates the position of the POI (see Fig. 3.16 - (3.b)). In addition, the size of the circle is animated and it is used to attract the user's attention. Furthermore, the radius of circle becomes smaller, if the center of the screen comes closer to the POI . The AR location search is closed either by touching on the 'Exit search mode'-button (Fig. 3.16 - (4)) or pointing on the target data point. The latter is achieved by 'capturing' the target POI with the outer circle of the crosshairs. If the target is reached, the AR viewer replaces the content of the AR location search with the clustered data visualization. If the cluster of the looked up POI consists of more than one data point, the detail visualization is displayed and the hierarchy traversed up to the right level with the label of the point of interest. An image of the result with an opened hierarchy can be found in Fig. 3.17(a). Furthermore, the location of the place, that the user is looking for, is marked on the map (see Fig. 3.17(b) - (1)). So, the POI can be found, even if there is no clear field of view, e.g. occlusion by other buildings or trees.

(a) (b)

Figure 3.15: AR browser - AR location search: Invocation of the location search and showing the result list. (a) Initialization of the AR location search by tapping on the magnify glass icon and entering a search term. (b) List of search results ordered by distance. A click on the icon (1) starts the guided search. The 'Show results'-button shows only the POI from the results in the AR viewer.

3.7.2 Search Results

The search results are shown in the AR overview+detail viewer, if the user selects the 'Show results'-button (see Fig. 3.15(b) - (2)). This shows only the points of interest of the result

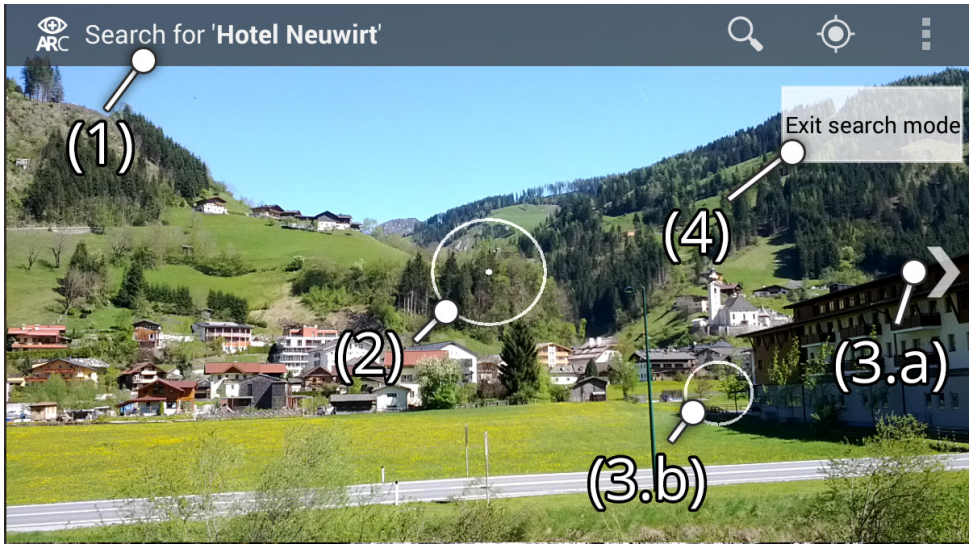
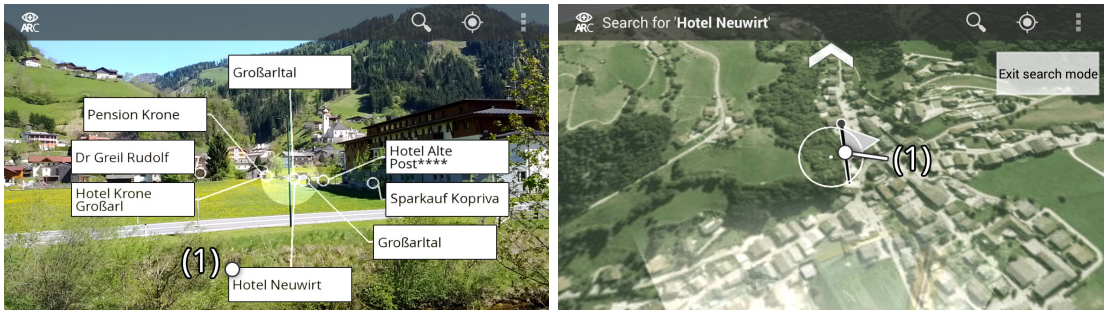


Figure 3.16: AR browser - AR location search: guidance for the user to the looked-up location. (1): Name of the selected POI to be guided to. A circle with an inner point ('crosshairs') is used as a support to find the POI. In (3.a) an arrow of the direction to turn and (3.b) an animated circle to draw attention to the desired POI. (4) exits the location search and returns to the clustered data visualization.



(a) (b)

Figure 3.17: AR browser - AR location search: (a) The looked-up POI is annotated by a marker (1) in the integrated overview map. (b) Search result within cluster. The detail visualization is traversed to the right level of the hierarchy. The result POI can be found at the position of annotation (1).

list in the AR viewer. An image of such a result can be found in Fig. 3.18(a). The return to the normal non-filtered data set is possible by clicking on the 'Show POIs'-button. Additionally, each result is represented as a marker on the overview map in the AR viewer (see Fig. 3.18(b)) and in the map viewer. Further, in Fig. 3.18 the comparison between search results and unfiltered dataset within the map viewer is shown. The filtered data set can be found in Fig. 3.18(c), in contrast to the non-filtered data in Fig. 3.18(d).

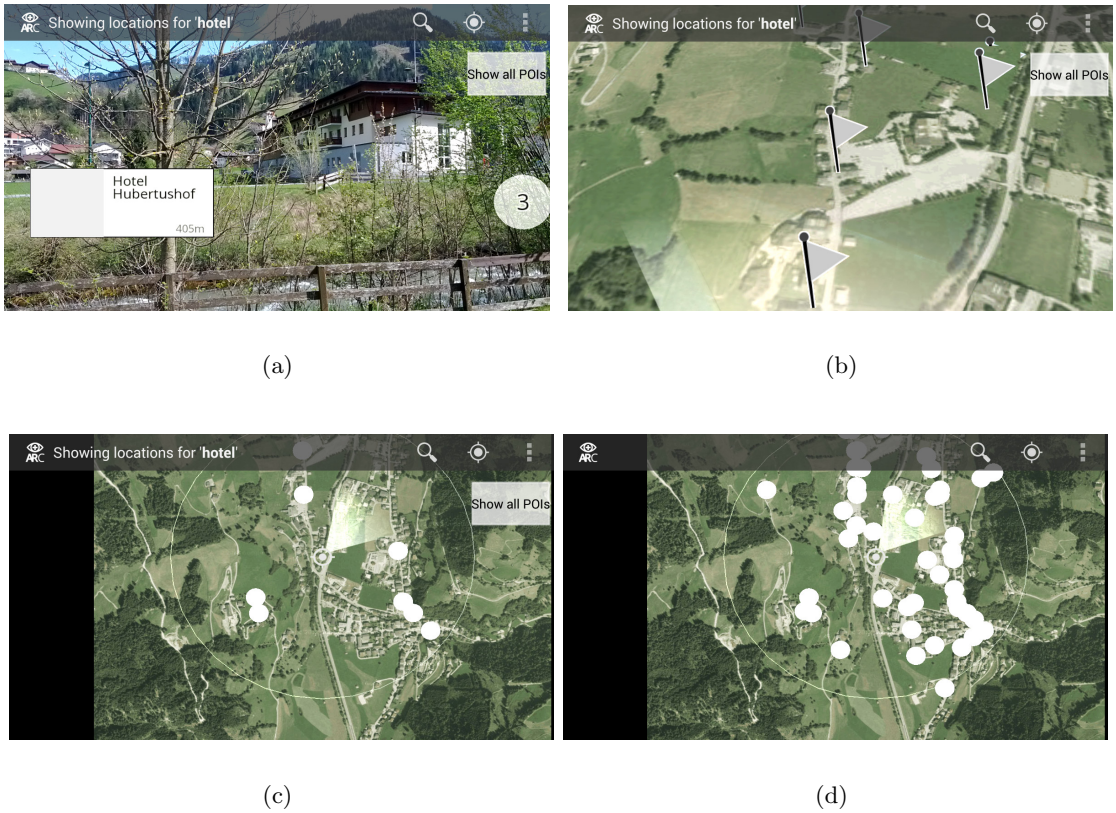


Figure 3.18: AR browser - AR location search: showing multiple search results. (a) All results from the search result list are shown in the AR viewer. (b) All results from the search result list are shown on the overview map by markers; comparison between filtered and non-filtered data set. (c) Only search results are shown on the map in the map viewer by markers. (d) Return to the non-filtered results after clicking the 'Show all POI '-button.

Chapter 4

Implementation

Contents

In this chapter I present the implementation of the in Chapter 3 presented `AR` browser for Android. The framework of the `AR` browser handles different tasks:

- handling of sensor data (see Section 4.1): the raw sensor data has to be processed to handle changes in position and orientation
- data retrieval: usage of location and network based services (see Fig. 4.1), and converting the retrieved data for further usage (see Section 4.2)
- data filtering (see Section 4.3): the whole dataset may not be needed and/or useful, so, filtering the data set is necessary
- clustering of the data points (see Section 4.4): to avoid clutter of the labels by aggregation of nearby data points

- preparation and representation of the data on the screen (see Section 4.4): the user interface should represent the given data in a clear and understandable way; furthermore it should be possible to navigate through the data via the generated labels

- handling of user input: the user should be able to modify the dataset and get additional information about a specific POI or cluster

Further, the implementation of the AR browser system can be separated into two programming-, respectively, coding-parts. The first part is written in native C++ and handles the steps for the representation: clustering, labelling and rendering of the POI and generated cluster. The second part is Java code for Android. This part deals with the data of the sensors of the device (orientation, camera, GPS, etc.) and prepares the retrieved data for further use in the native code. Furthermore, it is used to present detailed information of a POI and is used as interface for different methods written in native code. The user-browser-interaction is done by touch events and gestures.

4.1 Sensor Data and Data Sources

The handling of sensor data is mostly done by the OS. The implementation of the AR browser only accesses the required data and converts it to the needed data formats for the native code. Furthermore, a simple logic handles the retrieved position data updates if the location is changed. The AR browser uses different Application Program Interfaces (APIs) as data sources which are provided by Google. Mainly, the *Google Places API*[*] is used to get location based information. The retrieved data consists of the geographic coordinates (latitude and longitude), the name and address, the categories, a rating and a URL to an icon for the place. The data can be fetched by a nearby search of the current geographic location of the device and a given radius in meters. The maximum number of results is limited to 60 per request by the used API.

Due to the limited number of results, all results are cached in a SQLite database on the Android device. The caching of the data reduces the data traffic and allows to show data points immediately, if the location was visited earlier.

Another Google API used in the implementation is the *Elevation API*. The Elevation API returns the interpolated altitude of a geographic location based on the contour lines from the map data. A returned result of a Google Places API request misses the altitude

[*] https://developers.google.com/places/

information. Therefore, the Elevation `API` is needed to get the the altitude of the retrieved nearby places. Further, it is used to get the altitude of the user, if the location sensor has wrong or no data about the altitude.

The shown map data is retrieved as a static image (PNG file format) from the *Static Maps* `API`[†]. It allows to request a map for a location which is given by latitude and longitude, or an address. Furthermore, the type of map (satellite, roadmap, etc.) can be chosen and it is possible to set a zoomlevel (level of detail) for the requested map. The center of the image is the requested geographic location.

4.2 Data Conversion

The retrieved data has to be processed for the usage in the framework. So, there are different conversions of the location based data (geographic coordinates) to Cartesian coordinates for the usage in the OpenSceneGraph graphics toolkit (see Section 4.7). Therefore, an appropriate projection of the geographic coordinates is needed.

4.2.1 Geographic Coordinates to Cartesian Coordinates

The retrieved locations are represented as geographic coordinates by latitude φ ($\varphi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$) and longitude λ ($\lambda \in [-\pi, \pi]$). Geographic coordinates are a special case of spherical coordinates. In spherical coordinates a point on the sphere is given by the radius r ($r \in [0, \infty]$), the angle θ ($\theta \in [0, \pi]$) and ϕ ($\phi \in [0, 2\pi]$). For further use in the framework, a conversion from the geographic coordinates to Cartesian coordinates (3D space, 3 coordinates: point $\mathbf{x} = (x, y, z)$) is needed. This conversation can not by simply done by transforming the spherical coordinates to Cartesian coordinates (see Equation (4.1)).

$$
\begin{aligned}
x &= r \sin\theta \cos\phi \\
y &= r \sin\theta \sin\phi \\
z &= r \cos\phi
\end{aligned}
$$

The geographic coordinates must be projected onto a plane by using a map projection. A common projection method for maps is the cylindrical map projection by Mercator (Mercator projection) [51]. This type of map projection is used by Google Maps too as mentioned

[†]`https://developers.google.com/maps/documentation/staticmaps/`

in [21]. The determination of the position values x and y is given in Equation (4.2). As radius R we use the semi-major axis of the WGS84[‡] [45] ellipsoid reference. The longitude λ_0 respectively the y-coordinate of the current position has to be subtracted, if the current position is the origin of the coordinate system. Additionally, the z-coordinate can be treated as zero or the altitude of the position z_{alt} can be taken, whereas $z_{alt,0}$ is the altitude of the viewers position.

$$
\begin{aligned}
x &= R\,(\lambda - \lambda_0) \\
y &= R\left(\ln\left[\tan\left(\frac{\pi}{4} + \frac{\varphi}{2}\right)\right]\right) - y_0 \\
&= R\left(\frac{1}{2}\ln\left(\frac{1+\sin\varphi}{1-\sin\varphi}\right)\right) - y_0 \\
z &= z_{alt} - z_{alt,0}
\end{aligned}
$$

4.2.2 Position on the retrieved Google Map Image

As mentioned before, the Google Maps `API` allows to define a zoomlevel for the retrieved image of the map. Furthermore, the center of the image is the requested location given by latitude and longitude. For further use, i.e. putting markers on the map, it is necessary to know the conversion of a given geographic location for a `POI` to the position on the map. The computation is the same as in Equation (4.2), but with small modifications. A zoomfactor f_{zoom} is defined by the given zoomlevel l_{zoom}. This zoomfactor f_{zoom} and the width w_{map} of the quadratic map units is used to define the units per degree (u_{degree}) and units per rad (u_{rad}). The final equations for the conversion are found in Equation (4.4).

$$
\begin{aligned}
f_{zoom} &= 2^{l_{zoom}-1} \\
u_{degree} &= \frac{w_{map} f_{zoom}}{360^\circ} \\
u_{rad} &= \frac{w_{map} f_{zoom}}{2\pi}
\end{aligned}
$$

[‡]World Geodetic System 1984

$$x = u_{degree}\,(\lambda - \lambda_0)$$
$$y = u_{rad}\left(\frac{1}{2}\ln\left(\frac{1+\sin\varphi}{1-\sin\varphi}\right)\right) - y_0$$

4.3 Data Filtering

The dataset of `POI` has to be filtered in a useful way to present the user only the desired information. Therefore, I use three methods for filtering: distance based filtering, filtering by term accordance (search) and category based filtering. In my implementation a combination of one or more of this filtering methods is used.

4.3.1 Distance based Filtering

Distance based filtering is an easy way to reduce the dataset before the clustering step. Each data point is represented with coordinates in the world space after the previous described conversion from geographic coordinates to Cartesian coordinates. The distance is computed as the Euclidean distance $d_n(\mathbf{p}, \mathbf{q})$ (see Equation (4.5), n is the number of dimensions of the given Euclidean n-space) between the current position $\mathbf{x}_0$ and the position of the `POI` $\mathbf{x}_{\text{POI}}$. Only the subset of the whole data set is taken into account for the following steps (see Section 4.4).

$$d_n(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{4.5}$$

The distance based filtering uses at least one distance $d_{\max}$ as parameter which can be changed with the `UI`. The distance $d_{\max}$ defines the total distance range $r_{d,\text{total}} \in [0, d_{\max}]$ for the filtering. Furthermore, the native part of the implementation supports m sub-distances $d_{\text{sub},i} \forall i \in [0, m-1]$ where $0 < d_{\text{sub},i} < d_{\max}$. The sub-distances result in $m+1$ sub-ranges $r_{d,i}$ for $r_{d,\text{total}}$. The sub-ranges are taken into account at the later presented Angular Clustering method (see Section 4.4.1.3). But it should be mentioned that the sub-rages are not supported to be set or changed by the `UI`.

4.3.2 Filtering by Search Term

In an AR browser the user wants to search for a specific point of interest. Therefore, a matching by search term(s) is useful. Due to the usage of caching the data of the locations with a SQLite data base, it is possible to search within nearby places offline. The search results are represented as a list of places and ranked by distance to the current position. This allows the user to perform a AR location search or to only show the search results in the AR overview+detail viewer (see Section 3.7). The filtering by search term uses the given space separated search terms as parameters. Each term must (partially) match with either the name or the address of the place. The order of the search terms is not crucial because the search terms are conjuncted.

4.3.3 Category based Filtering

The Google Places API provides for every returned place at least one of 90 categories [22] (Google calls them types). Therefore, a filtering by these categories is very useful. The implementation allows to select multiple categories from a category list. After selecting one or more categories and applying the filter by categories on the full category set $\mathbf{C}_{all}$ with size N, the filtered result is the category subset $\mathbf{C}_{\text{sub}}$ with size n. Each POI p_i has its subset of categories $\mathbf{C}_{p,i}$ with size $n_{p,i} \in [1, N]$. Categories, which are not in the selected subset of categories, are neglected at the matching step. This means, that non-selected categories do not have action on the quality of the filter result.

The quality of a filter result rq_i ($rq_i \in [0, 1]$) is defined in Equation (4.6), where $m_{p,i}$ is the number of matching categories of place p_i from the selected set of categories $\mathbf{C}_{\text{sub}}$ (see Equation (4.7).

$$rq_i = \frac{m_{p,i}}{n} \tag{4.6}$$

$$m_{p,i} = \mathbf{C}_{p,i} \wedge \mathbf{C}_{\text{sub}} \tag{4.7}$$

The category filtering is combined with the distance based filtering - only POI within the selected range are taken into account by category based filtering of the data. This means, that data points are not taken into account of the category based filtering, which are outside of the desired distance. At the final representation in the AR viewer, the quality of a search result is indicated by different colors and non-matching points of interest are not shown. The color based approach has the advantage that the quality can be indicated even outside of a (large) cluster (see Section 3.4).

4.4 Data Clustering and Data Representation

The given data points are defined by coordinates in the world coordinate space and some sort of additional data like a unique ID and/or a name. A single data point has no real meaning for the user, therefore some sort of labelling is needed to get the context of a data point and its other properties. Thus, a simple label for each data point could be shown at its projection position on the screen, but this causes cluttering of the labels, due to the lack of space on the screen of the device. Hence, some sort of cluster analysis is needed to represent the data in a reasonable and understandable way.

The data representation can be separated into two different parts. First, there is the representation of the whole data set of points in the surrounding area (world data). By using a suitable clustering algorithm, the result is a number of clusters containing one or more data points after applying the clustering. I call this clustering the primary cluster analysis (see Section 4.4.1). The data set from this clustering is part of the clustered data visualization, which was discussed in the concept in Section 3.2. This rises the question how to handle and display the data within these generated clusters as proposed in Section 3.3. The implementation of the clustering and visualization of the cluster detail data is presented in Section 4.4.2 and called sub-cluster clustering. I will present in the following sections the clustering and the handling of the labelling of the data points in the `AR` browser framework.

4.4.1 Primary Cluster Analysis

As mentioned before, the goal of the primary cluster analysis is to find data point cluster. Such a cluster can be labelled immediately, if the size of the cluster equals one. Otherwise, we have to find another way to get a reasonable labelling. I will deal with this problem in Section 4.4.2. The next sections will show some common ways to perform a cluster analysis and the finally used clustering algorithm.

4.4.1.1 Grid based Clustering

A grid based clustering approach is one of the easiest way to find clusters of data points. Given an area A or volume V, the available space is divided into N cells with same area respectively volume. The clustering is performed by checking, if a data point is within such a cell. After the cluster analysis, the results are empty and non-empty cells and a cell mapping for each data point.

The grid based approach has the advantage, that it is very fast and very little computational power is needed. The major disadvantage is, that is it not very useful in an AR browser because the position of a data point on the screen is not static due to movements with the device. Hence, the assignment of a data point to a cell can change easy, when the data point lies near the border of two cells or the cell size is to small. This behaviour is not desired on a mobile device and will lead to a very bad user experience. Nevertheless, the grid based clustering approach is used by Carmo et al. in their proposed method to aggregate icons on a 2D map [8] as mentioned in Chapter 2.

4.4.1.2 k-Means Clustering

k-means is a very common centroid-based clustering method. The optimization problem is referred as NP-hard, but there are approximation algorithms, which results after a few iterations in a local minimum. The name of the clustering method is given by the number of needed k-cluster centers. A well known approximate method is Lloyd's algorithm [39]. But there exists many other variations of the k-means algorithms which distinguish i.e. in the method of computing the center of a cluster or how the initial cluster centers are selected.

The algorithm minimizes the within-cluster sum of squares (WCSS) as described in Equation (4.8). Where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is a d-dimensional observation (data point) vector with length n and μ_i is the mean of the current cluster. This observations are separated into k sets $\mathbf{S}_1, \ldots, \mathbf{S}_k$, where $k \leq n$.

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in \mathbf{S}_i} \|\mathbf{x}_j - \mu_i\|^2 \tag{4.8}$$

Lloyd's Algorithm - Standard Algorithm. Lloyd's algorithm is a common k-means clustering technique and can be seen as the standard algorithm for k-means clustering. The algorithm uses an iterative refinement technique and performs two main steps after the initialization of the cluster and observations. The whole refinement starts with an initial set of k means $\mathbf{M}^{(1)} = \{m_1^{(1)}, m_2^{(1)}, \ldots, m_k\}$.

Initialization. There are two commonly used methods for the initialization of the algorithm: the Forgy and Random Partition [24] method. Both initialization methods lead to different final results. The Forgy method chooses randomly k observations and

sets them as initial means for the cluster. While, the Random Partition method assigns a cluster to each observation and skips the assignment step in the first iteration.

Assignment Step. Each observation x_p is assigned to the cluster, where the mean yields to the WCSS (see Equation (4.9)). Each observation x_p is only assigned to one $\mathbf{S}^{(t)}$.

$$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 \leq \|x_p - m_j^{(t)}\|^2 \forall j, 1 \leq j \leq k\} \tag{4.9}$$

Update Step. The centroids of each cluster are generated by calculating the new means of the cluster by using the arithmetic mean (see Equation (4.10)). This step minimizes the WCSS objective too, since the arithmetic mean is a least-squares estimator.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{4.10}$$

Lloyd's algorithm converges, when the assignments of the observations to a specific cluster no longer change. In this case the algorithm has reached a local optimum, which is not guaranteed to be the global optimum. The original algorithm tries to minimize the within-cluster sum of squares (least sum of squares), but the assignment to a cluster can be done by Euclidean distance as well.

Practical Use This clustering method was tested for the usage in the `AR` browser, but was neglected after a short testing time. There are some major drawbacks of k-means clustering for the usage in an `AR` browser.

- selection of the coordinate space of the data set
- fixed cluster size (parameter k)
- clustering every time when data set changes
- clustering does not take care of the representation of the data

The above mentioned drawbacks cannot be seen individual problem. More or less the drawbacks are connected. There is the problem of selecting the coordinate frame of data set. We have two possibilities, which data coordinate space is used. On the one hand, there is the data in the world coordinate frame ('world data', 3 dimensions). The position of the data points is static, but it is possible that new data points are added to the data

set. Therefore, only at the changes on the data set, a new clustering for the data has to be performed.

On the other hand, we have the projection of the data points on the camera respectively the screen coordinates (2 dimensions). Here, we have to deal with only a subset of the data points of the 'world data'. But the position of a data point can change every frame and the number of visible data points may change very often. This results in a clustering in (nearly) every frame. Depending on the size of the data set, the number of clusters k and the number of iterations needed to find the local optimum, a lot of computation resources will be spend on the clustering algorithm.

However, there is the question how to determine the parameter k for the number of cluster. We perform the clustering to avoid cluttering of labels for the data points. Unfortunately, k-means clustering takes only account of the position of the each observation in the d-dimensional space, but not of the representation of the data point (area) on the 2-dimensional screen space. Thus, there is a need of additionally steps to merge overlapping cluster representations.

4.4.1.3 Angular Clustering

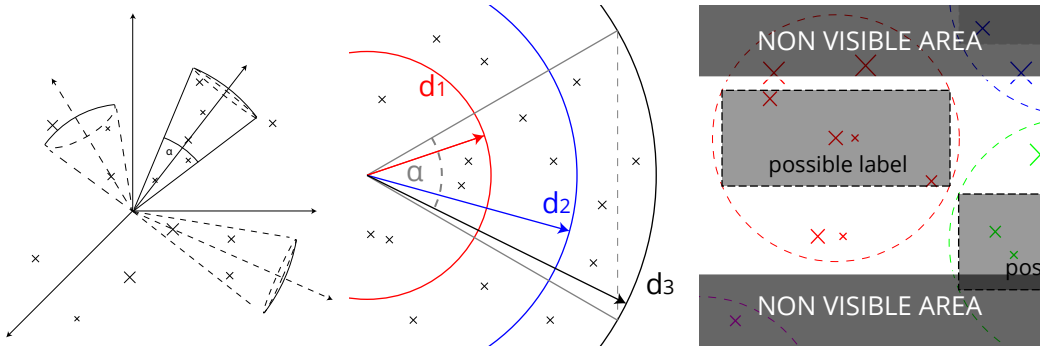
As mentioned in the occurring problems of the previous cluster analysis methods, I present in this thesis a method to prevent cluttering of labels within a `AR` browser. The proposed approach is called *Angular Clustering*-method. This sort of cluster analysis uses the angle between vectors as decision condition. Additionally, the distance between a data point and the devices position in the world reference frame can be taken into account too. The idea behind the Angular Clustering is, that the clustering is performed before the projection from world coordinates to view-port coordinates takes place. This makes the clustering only necessary, if the position of the camera is changed or the data set is changed. Therefore, the clustering will not be needed every frame.

The clustering will result in a cone shaped cluster in the 3-dimensional Euclidean space with circular base with the apex at the position of the viewer, which is normally the origin of the Euclidean space. The axis of the cone is normal on the center of the circular base. Another way to define the cone is to see it as an isosceles triangle which is revolved around its height. The angle α between the two sides with the same length is for the Angular Clustering an important parameter. Some examples of the result of the Angular Clustering can be found in Fig. 4.1(a). Let the angle α be $\alpha \in [0, \alpha_{\max}]$, where the lower bound is given for a cluster with only one data point and everything else can

be at a maximum $\alpha_{\max}$. Furthermore, the angle between two cone axis has to be at least $2\alpha_{\max}$. The computation of $\alpha_{\max}$ is described later.

The resulting cluster cone can be separated in distance layers without a new clustering run is needed (see Fig. 4.1(b)). Thus, it is possible to turn specific layers (distance ranges) on or off. It should be mentioned that this feature of the algorithm is not accessible in the implementation of the `AR` browser from the `UI`

In Fig. 4.1(c) the visualization of the cluster including a possible label is shown. The projected data points and the surrounding circle of the cluster are colored differently for a better understanding. An overlap of labels can not occur because new data points are always added to a cluster with the smallest angle between the vector of the cluster center and the vector of the data point.



(a) (b) (c)

Figure 4.1: The idea behind Angular Clustering. (a) Some examples for results of the Angular Clustering. Each cluster is cone shaped and has a maximum angle $\alpha_{\max}$ between two vectors within the cluster. (b) A cone shaped cluster. The cluster can be separated into volumes with different distances. (c) The model for a possible representation on the screen of a device. Each cluster is colored different and an overlap of labels is per definition not possible.

In an initial step the conversion between the needed pixels on the screen and angle for the cone volume has to be found. The angle of view (`AV`) have to be known for this conversion. This value, which can be given as horizontal, vertical or diagonal `AV`, describes the angular extend of the scene which is taken into account by the camera and is given in degrees (or rad). Furthermore, the height and width in pixels of the screen or the viewport has to be known. By using these parameters, it is possible to compute the pixel per degree (PpD) (see Equation (4.11)) or the degrees per pixel (DpP) (see Equation (4.12)).

This will result either in one value for both dimensions (quadratic pixel) or in two separate values (non-quadratic pixel).

$$PpD_x = \frac{\text{width}}{AV_x} \qquad \text{or} \qquad PpD_y = \frac{\text{height}}{AV_y} \tag{4.11}$$

$$DpP_x = \frac{AV_x}{\text{width}} \qquad \text{or} \qquad DpP_y = \frac{AV_y}{\text{height}} \tag{4.12}$$

The next step in the initialization is to define a size for the label in pixels $(l_{\text{width}}, l_{\text{height}})$. Finally, the angle in degrees for the cone α_{cone} is $DpP = \max(l_{\text{width}}, l_{\text{height}})$. Additionally, the half angle α_{half} is computed, where $\alpha_{\text{half}} = \frac{\alpha_{\text{cone}}}{2}$.

The following steps will be performed at each run of the Angular Clustering algorithm:

Algorithm 4.1 Angular Clustering.

```
filter data set
random select data point as center seed
while untested data set has data points do
  select untested data point from data set
  for all centers do
    if data point addable to center then
      add to center
    else
      create new center
    end if
  end for
end while
```

In the first step, the data points are pre-selected by a filter. This can be either a distance based filtering or a combination of the previous mentioned filtering methods (see Section 4.3). This step is necessary because only a clustering of relevant data points is needed. Next, a data point from the sub-selection is chosen. This, so called 'seed', will be the first cluster center. Therefore, either the first data point from the list or a random point is picked. The position of the data point and the current position of the camera center will be the axis vector of the cone.

In the next step, every other data point is checked, if it is possible to add the point to an existing cluster or it is necessary to create a new cluster from this data point. This is done by computing the angle θ between the vector pointing from the camera center to the data point and the axis vector. The angle θ is given by the dot-product of the two vectors

a and **b** (see Equation (4.13)) and results in $\theta \in [0, \pi]$ respectively in degrees $\theta \in [0°, 180°]$.

$$\theta = \arccos \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \tag{4.13}$$

There are two conditions, which have to be fulfilled for a data point dp to be added to an existing cluster center cc_i. First, the angle θ has to be smaller than the value of α_{half}. This condition results only in cluster centers, which are in close range of the data point dp. This sub-set of cluster centers $\mathbf{S}_{cc}$ will be taken only into account for the second condition. The second condition is, that the cluster center cc_i has to have minimum angular distance to the data point dp (see Equation (4.14)).

$$\arg\min_{cc_i} \theta_{cc_i,dp} \tag{4.14}$$

If both conditions are satisfied, the data point is assigned to the cluster. Otherwise, a new cluster is created from the current data point with cone axis through the position of the data point and added to the set of clusters.

4.4.2 Sub-Cluster Clustering

The primary clustering with Angular Clustering results in clusters with either one or multiple items. A single-item cluster can be represented easily with a label, but the representation of the members of a multi-item cluster is not quite simple to achieve. Therefore, a sub-cluster clustering takes account of the layout and representation of the content of a cluster from the primary clustering.

The sub-cluster clustering uses a defined number of positions around the cluster center, so called slots. This layout allows an easier assignment of the data points to a cluster center and a stable behaviour of the clustering process. A model of the slot arrangement can be found in Fig. 4.2. In this case, we have ten available slots, but more than 10 data points. Therefore, there is the need to assign each data point to a slot. Furthermore, we want to connect the position of the data point to a label by connecting them via a leader line. This step is needed because the position of each label is static (seen from the cluster center). Additionally, intersections between the leader lines of the data points have to be avoided.

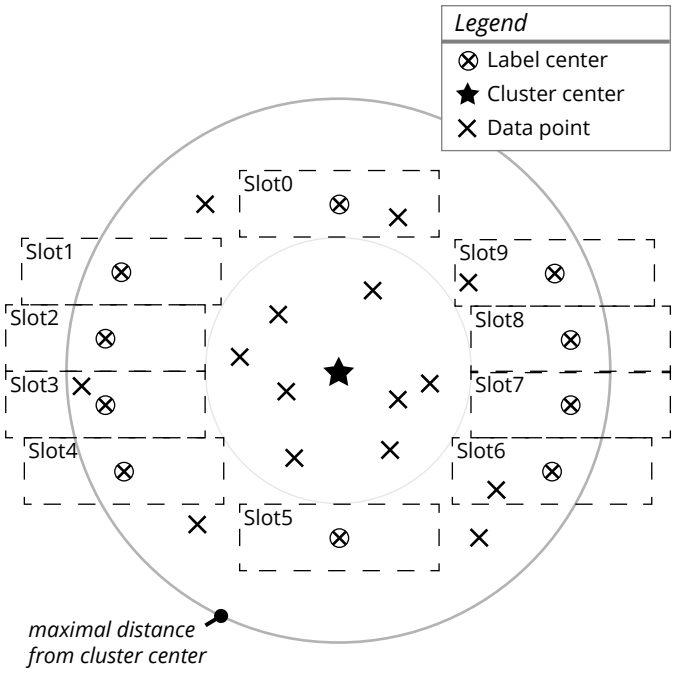


Figure 4.2: Model of the layout for the sub-cluster clustering and labelling.

4.4.2.1 Slot Layout

As mentioned before, a number N of slots is used, N depends on the available screenspace and the height of a cluster label for the sub-cluster clustering. The positions of these labels are fixed and the labels according to the slots are only shown, if they are not empty. The proposed layout can be found in Fig. 4.2. I have chosen a trade-off between needed screen space for a label and visibility of the matching to the corresponding data point. The label size is taking account of different pixel densities on Android devices. On the screen, the labels are arranged horizontally in a circle around the cluster center. A relative large area is left blank to allow a clear annotation of a data point. If the number of data points is larger than the available slots, it is necessary to select a subset of `POI`. This selection is done by sorting the data points by distance to the current position and taking the first N data points from the cluster. Another way of pre-selection can be done by sorting by relevance of a `POI`, but this pre-selection technique is not accessible by the `UI`.

The layout is determined as follows. Every slot is defined as a rectangle with the coordinates for the top-left and bottom-right corner. The width w_{label} and the height h_{label} of the label is for all slots the same. Additionally, the center of the rectangle is computed. The cluster center is seen as the origin for the 2-dimensional coordinates.

	coordinate	
quadrant	x	y
top-right	$+$	$+$
top-left	$-$	$+$
bottom-left	$-$	$-$
bottom-right	$+$	$-$

Table 4.1: Conversion table for coordinates of a the quadrants.

So, the coordinates for each rectangle are relative to this coordinate frame. The whole layout is symmetric on both axes. Therefore, only the coordinates of the slots for one quadrant have to be determined. The other coordinates can be defined by mirroring the coordinates along the x- and y-axis. The total number of cluster labels is determined by the radius of the circle r_{circle}, which encloses a detailed label from the clustered date visualization. The reason for that arrangement is that so there is enough space left to show the annotations of the `POI` within the cluster. To compute the number N of slots the formula in Equation (4.15) can be used, where $N_{\text{slots per side}}$ is the number of slots of each side without the top and bottom slots.

$$N_{\text{slots per side}} = \lfloor \frac{2r_{\text{circle}}}{h_{\text{label}}} \rfloor$$
$$N = 2 + 2N_{\text{slots per side}}$$

The center of the of the top and bottom slot is centered at the vertical axis (y-axis) and the distance of the center to the inner circle is half the height of the slot label. For the remaining slots a vertical step size s_y is determined, where $s_y = \frac{2r_{\text{circle}}}{N_{\text{slots per side}}}$. The x-coordinates $x_{\text{left},i}$ for the left edged of the top-left quadrant slots can be determined by Equation (4.17), where the index i is the row number ($i \in [1, \lfloor N_{\text{slots per side}}/2 \rfloor]$) and $y_{\text{top},i}$ is the y-coordinate of the top edge of the slot (see Equation (4.16)). So, the top-left, bottom-right and center coordinates are computed for each slot of the quadrant. The coordinates for the other quadrants can be determined by using the conversion table Table 4.1. If $N_{\text{slots per side}}$ is odd, two slots must be added centered at the horizontal axis.

$$y_{\text{top},i} = r_{\text{circle}} - s_y i \tag{4.16}$$

$$x_{\text{left},i} = \sqrt{r_{\text{circle}}^2 - y_{\text{top},i}^2} \tag{4.17}$$

4.4.2.2 Label Layout and Leader Line Layout

The static slot layout makes it necessary to use connections between a data point and the correlating labels, so called 'leader lines'. This leader lines can be either presented as single straight line or a line with two segments. Both representations can be found in Fig. 4.3.

The decision, which representation of the leader line is draw, is made as follows. For every slot a *connection point* is defined. The connection point is in the center of the border of the slot with the minimum distance to the center of the cluster. This is the left border of the slots on the right side respectively the right border of the slots on the left side. The labels within the slot use a margin to achieve a visual separation to the border of the other labels. Additionally, it allows to draw the leader lines for data points, which are near a slot-slot border.

The defined connection point is used as decision boundary. Therefore, the x-coordinate of the projection of the data point on the screen is tested, if it is within or outside the slot. This is a larger value for the x-coordinate of the data point than the x-coordinate of the connection point for the slots on the left side. In the case of the slots on the right side, the x-coordinate value has to be smaller. For the top and bottom slots the y-coordinate is taken into account at the decision.

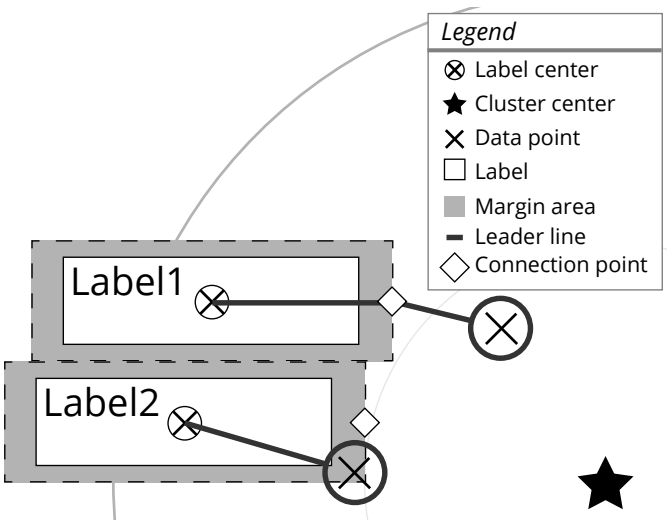
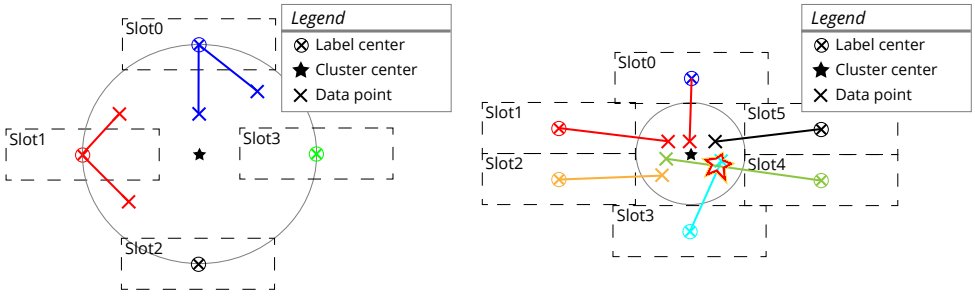


Figure 4.3: The leader line layout: a label can be connected to the annotation via a single straight leader line (*Label2*) or a leader line with two segments (*Label1*). The connection type depends on the position of the corresponding data point.

4.4.2.3 Label Assignment for the Detail Visualization

After selecting the subset of data points the data points ave to be assigned to the labels. The first idea is to simple assign each data point to the nearest label (center). This will lead to a problem: a label could be assigned to more than one of the previous selected data points. A possible result of such a labelling method is shown in Fig. 4.4(a). But, the usage of all available slots is desired to have an optimal use of them.

This will lead to another simple try to solve the problem. In this approach every data point get its own slot by removing not available slots from the list of possible label candidates. This is done by finding the nearest slot for the first point, removing the found slot from the candidates list and moving on to the next data point. This possible simple solution will lead to possible intersections of the leader lines between label and data point. An illustration of the result of the labelling method can be found in Fig. 4.4(b).



(a) (b)

Figure 4.4: Examples of two simple labelling methods. (a) Assigning label by minimal distance. (b) Assigning label by distance, improved method: better usage, but intersecting lines.

Therefore, a more advanced method of assigning a data point to a label is needed. The problem can be seen as a weighted bipartite graph, where the labels and the data points can be seen as vertices of a graph G. Further, the labels and data points represent two disjoint sets (U, V). The vertices of set U can be connected to one or more vertices in set V via an edge E. So, it is possible that a vertex of U is connected to more than one vertex in V. In the first view, every vertex of U is connected to all vertices in B in the bipartite graph G. The edges E are weighted for this labelling problem. The weight of each edge is given by the distance between the corresponding data point of the vertex in U and the corresponding label center of the vertex in V. The weights can be seen as costs. So, the

	JobA	**JobB**	**JobC**
Worker1	$cost_{1,A}$	$cost_{1,B}$	$cost_{1,C}$
Worker2	$cost_{2,A}$	$cost_{2,B}$	$cost_{2,C}$
Worker3	$cost_{3,A}$	$cost_{3,B}$	$cost_{3,C}$

Table 4.2: Example of a $n \times n$ cost matrix where $n = 3$.

aim is to optimize the cost of the graph by finding the optimal combination of one-to-one (bijective) connections between the two sets.

This combinatorial optimization problem can be solved by the *Hungarian algorithm* (Kuhn-Munkres algorithm) [35, 44].
Before the Hungarian algorithm can be applied a $n \times n$ cost matrix has to be set up. The cost matrix deals with a set of workers and a number of possible jobs. For every worker (row) the cost for a job (column) is known and can be found in the i-th row in the j-th column. If there are more workers than jobs or vice-versa, the matrix has to be filled up with additionally columns respectively rows. The aim of the algorithm is to find an assignment for each job to a worker and get a total cost minimum. If the problem is seen as maximization problem, the problem can be solved by finding the maximum cost $c_{\max}$ and subtracting each cost $c_{i,j}$ of $c_{\max}$. The cost matrix can be seen as complete bipartite graph $G = \{U, V, E\}$ as mentioned before. Let us define a function $y : (U \cup V) \mapsto \mathbb{R}$, which is a potential, if $y(i) + y(j) \leq c(i,j) \forall i \in U, j \in V$. The value of y is given by $\sum_{v \in U \cup V} y(v)$, where v is a vertex of G. The Hungarian algorithm finds a perfect matching and a potential with equal cost/value which proves the optimality of both. This perfect matching is found for tight edges. An edge of the bipartite graph is called tight when $y(i) + y(j) = c(i,j)$. The sub-graph of tight edges is denoted as G_y.

While the algorithm is running, a potential y and an orientation $\overrightarrow{G_y}$ of G_y are maintained. The property of $\overrightarrow{G_y}$ is that the edges are orientated from V to U and form a matching M, where all edges in M are tight edges. The algorithm is initialized by setting the potential everywhere to zero and the edges orientated from U to V (empty matching M). In each step the potential y is modified to increase its value or the orientation is modified to obtain a matching with more edges.

In the first step, we define the vertices, which are not covered by M, as $R_U \subseteq U$ and $R_V \subseteq V$. So, R_U consists of vertices with no incoming edges in U and R_V has only vertices of V with no outgoing edges. Let Z the set of vertices in $\overrightarrow{G_y}$ be, which are reachable from R_U in a direct path only following tight edges.

There can be two states for for the set $R_V \cap Z$: a non-empty or empty set. If the case

of a non-empty set occurs, the orientation of the directed path $\overrightarrow{G_y}$ is reversed from R_U to R_V. This will increase the size of the corresponding matching by one.

In contrast, if $R_V \cap Z$ is empty, let us define $\Delta := \min\{c(i,j) - y(i) - y(j) : i \in Z \cap U, j \in V \setminus Z\}$. Δ will be positive because there are no tight edges between $Z \cap U$ and $V \setminus Z$. The potential y is increased by Δ on the vertices of $Z \cap U$ and the potential is decreased on vertices of $Z \cap V$. This will lead to changing graph $\overrightarrow{G_y}$, but the graph still contains M. The new edges have to be orientated from U to V. The number of vertices in Z reachable from R_U increases by the definition of Δ, but it will not necessarily increase the number of tight edges. The steps are repeated until M is a perfect matching and therefore a minimum cost assignment.

4.4.2.4 Sub-Cluster Hierarchy

In the previous section the assignment of a data point to a label without intersections of the leader lines was discussed. The presented algorithm handles only the data points within the cluster, which will be assigned to a label, but not the remaining non-visualized data points. These data points will lead to a hierarchical structure of the cluster. Therefore, the remaining data points have to be assigned to a root or parent slot in the hierarchy. To do so, only the children of a next level for a slot are determined. The remaining data points are assigned to the slot with the minimum Euclidean distance (distance: slot center to data point). The label assignment of the data points in the next level of a slot is not computed until the level is opened by the user as proposed in the concept.

4.5 AR Guided Search

The `AR` guided search is implemented as follows. First, the position $\mathbf{x}_{\text{loc}}$ in the world coordinate frame of the `POI` is determined. Additionally, the current position $\mathbf{x}_0$ is known. The target vector $\mathbf{x}_{target}$ is defined by $\mathbf{x}_{target} = \mathbf{x}_{\text{loc}} - \mathbf{x}_0$. Furthermore, we need the direction, where the device is pointed at. This vector is called the focal vector $\mathbf{c}_{\text{focal}}$. The vector $\mathbf{y}_{\text{focal}}$ is retrieved from the current center of the camera $\mathbf{c}_{\text{center}}$, where $\mathbf{c}_{\text{center}} = \mathbf{x}_0$, and the eye vector of the camera $\mathbf{c}_{\text{eye}}$ ($\mathbf{c}_{\text{focal}} = \mathbf{c}_{\text{eye}} - \mathbf{c}_{\text{center}}$). By taking these two vectors, $\mathbf{x}_{target}$ and $\mathbf{c}_{focal}$, the angle α between the vectors can be computed. Although, the angle α gives information about the angular-'distance' from the desired target point, it is not possible to use only α to guide the user to the target. Therefore, the camera, in this case the mobile device, has to be changed in orientation. There are three possible rotations

around the axes of the camera (see Fig. 4.5).

- roll φ: rotation around the focal axis of the camera
- pitch ψ: rotation around the horizontal axis of the camera
- yaw γ: rotation around the vertical axis of the camera

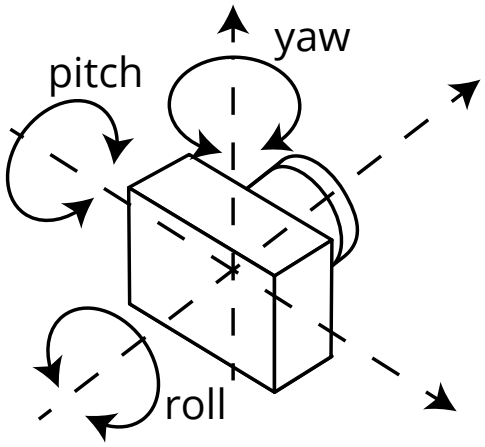


Figure 4.5: The three possible rotations around the axis of the camera.

Thus, three different angles are needed for each of the three axes. The roll angle φ could be neglected because the device is held in the horizontal landscape mode and therefore this has not to be corrected. So, there are only two rotations to be done: pitching and yawing. The pitch angle ψ is determined as follows. The angles between the z-axis vector $\mathbf{v}_z$, where $\mathbf{v}_z = (0, 0, 1)$, and the target vector $\mathbf{x}_{target}$ respectively the $\mathbf{c}_{focal}$ vector are calculated from their dot products (see Equation (4.13)). The resulting angles ψ_{target} and ψ_{focal} are used to retrieve the final pitch angle ψ in Equation (4.18).

$$\psi = \psi_{\text{target}} - \psi_{\text{focal}} \tag{4.18}$$

The yaw angle γ is computed in a different way. First, the target and focal vector are projected onto the XY-Plane. The projection of a point $\mathbf{x}$ onto a plane is defined by its normal vector $\mathbf{n}$ and a point on the plane $\mathbf{n}_0$ and can be found defined in Equation (4.19). In the case of the XY-plane the normal is the z-Axis ($\mathbf{n} = (0, 0, 1)$ and $\mathbf{n}_0 = (0, 0, 0)$). In this special case of the z-axis vector, the projection can be skipped or simplified by only

taking the first two dimensions (x and y) of the vector which has to be projected.

$$\mathbf{x}_{proj} = \mathbf{x} - ((\mathbf{x} - \mathbf{n}_0) \cdot \mathbf{n})\mathbf{n} \tag{4.19}$$

In the next step the projected vectors are normalized. Due the normalization the length of each vector equals one. So the angle θ in the unit circle can be computed according to Equation (4.20). Where r is the length of the vector (in this case $r = 1$) and x and y are the first and second component of the projected vectors.

$$\theta = \begin{cases} \arccos \frac{x}{y} & \text{if } y \geq 0 \\ 2\pi - \arccos \frac{x}{y} & \text{for } y < 0 \end{cases} \tag{4.20}$$

Finally, both angles γ_{target} and γ_{focal} are subtracted to get the yaw-angle γ:

$$\gamma = \gamma_{\text{target}} - \gamma_{\text{focal}}$$

This may result in a non optimal value for γ. This means absolute a value for γ which is over π respectively 180°. Therefore, the final value for γ is determined as in Equation (4.21).

$$\gamma = \begin{cases} \gamma + 2\pi & \text{if } \gamma < \pi \\ \gamma - 2\pi & \text{if } \gamma > \pi \\ \gamma & \text{else} \end{cases} \tag{4.21}$$

In the approach a simple `AR` guiding system is used. It uses only four possible direction annotations on the screen to guide to the desired target for simplicity. The annotations are arrows, which point in the direction of the rotation and only one annotation is shown on the screen simultaneously. This results in the following directions: left, up, right, or down. The user can rotate easily around her own center axis (normal to the ground) by turning left or right. But, the moveability around the horizontal axis of the head is restricted. Therefore, a user's turn to the left or to the right is more preferable than an up- or down-movement. So, at the beginning the guidance should bring the user to a horizontal starting point. This is achieved by computing the pitch angle between the camera and the world XY-plane. The pitch angle $\psi_{\text{focal,XY-plane}}$ between the focal vector $\mathbf{c}_{\text{focal}}$ and the XY-plane is determined by the vector and the normal vector $\mathbf{n}$ of the plane

(see Equation (4.22), where $\psi_{\text{focal,XY-plane}} \in [-\pi, \pi]$.

$$\psi_{\text{focal,XY-plane}} = \arcsin \frac{\mathbf{x}_{\text{focal}} \cdot \mathbf{n}}{\|\mathbf{x}_{\text{focal}}\| \|\mathbf{n}\|} \tag{4.22}$$

The determination of $\psi_{\text{focal,XY-plane}}$ is important, when the devices is pointing to the floor e.g. by looking at the map. Furthermore, there are three thresholds defined. The first threshold t_h is needed to decide, if a pinch rotation is needed to get the camera in a horizontal orientation (parallel to the world XY-plane) or if the angles between $\mathbf{x}_{\text{target}}$ and the camera axes are needed. The second threshold $t_{h,\text{margin}}$ takes only account, if the camera needs to be in horizontal orientation. It defines a margin for the difference to the XY-plane where the current orientation is counted as horizontal. The third threshold t_{target} allows an error at the angle α between the target vector $\mathbf{x}_{\text{target}}$ and the focal vector $\mathbf{c}_{\text{focal}}$ because it is not possible to align both vectors perfectly. The guidance to the target point is done as follows:

It is checked, if the α is smaller than t_h or the absolute value of $\psi_{\text{focal,XY-plane}}$ is within $t_{h,margin}$. In this case, one of the for possible directions has to be found. To do so, the larger absolute value of the pitch-angle ψ and yaw-angle γ defines the horizontal or vertical movement. If the ψ value is larger of both, a vertical movement is needed. In this case, the user has to move upwards, if the pitch-angle ψ is smaller than zero and else downwards. In the other case, a larger γ, a turn to the right side has to be performed. In case of a yaw-angle γ smaller than zero, the indicator representing a turn to the left is shown. Additionally, the size of the indicator icon of the direction is proportional to the value of the angle α.

If the first condition ($\alpha < t_h$ or $|\psi_{\text{focal,XY-plane}}| < t_{h,\text{margin}}$) is not fulfilled, the device has to be brought to a horizontal orientation. In this case $\psi_{\text{focal,XY-plane}}$ determines the direction to rotate the device horizontally. A negative value for $\psi_{\text{focal,XY-plane}}$ will lead to display of an icon to move the device up and a downward indicator in the other case. The size of the shown icon changes proportional to the absolute value of $\psi_{\text{focal,XY-plane}}$.

4.6 Overview+Detail

Overview+detail is represented in the `AR` browser as an additional overview map positioned virtually under the user's device. For interaction purposes it is possible to zoom the given map by moving it in the world coordinate frame. This methods allows a smooth transition between the detail view and the overview. Furthermore, the framework allows to place

marker on the map and connect them virtually with their real world position.

The map in the 3D space is represented as a textured quad in the world coordinate frame. This quad is positioned under the current position (negative z-index) of the camera. In the `AR` overview+detail viewer, that means when the device points to the horizon (parallel to the world XY-plane), the map should not be visible or rather not disrupt the view. Therefore, the map quad has to be placed at a distinct distance or set invisible, if it is in the current view-port of the camera. I use the first approach in the implementation. To do so, the minimal distance of the current position (offset) along the z-axis $z_{\text{offset,default}}$ has to be determined. Therefore, an arbitrary width w_{map} for the quadratic map is defined. Given the vertical angle of view of the y-axis of the view-port of the camera AV_y respectively the half angle of view $AV_{y,\text{half}}$, the minimal offset along the z-axis can be computed as given in Equation (4.23), where $w_{\text{map,half}}$ is the half width of the quadratic map. A graphical explanation of the determination can be found in Fig. 4.6.

$$z_{\text{offset,default}} = \frac{w_{\text{map,half}}}{\tan\left(90° - AV_{y,\text{half}}\right)} \tag{4.23a}$$

$$= w_{\text{map,half}} \tan AV_{y,\text{half}} \tag{4.23b}$$

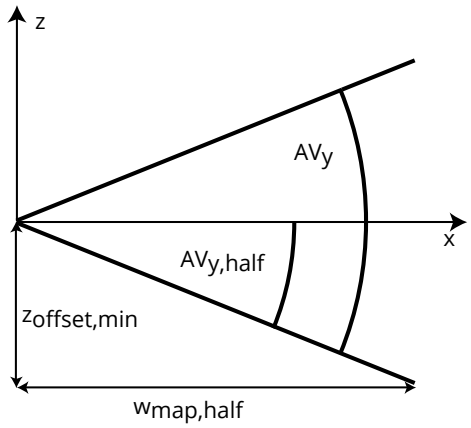


Figure 4.6: Determination of the minimal offset $z_{\text{offset,default}}$ of the map quad along the z-axis. Given the horizontal angle of view AV_y and the width of the map w_{map}.

Additionally, it should be possible to zoom the map. Here, there are two options to achieve the zooming action. The first way is to resize the quad geometry depending on a zoom factor f_{zoom}. The implemented, second option is to move the map like a

traditional map made of paper. Thus, an upper and lower bounds for the z_{offset} have to be defined. These two bounds ($z_{\text{offset,min}}$,$z_{\text{offset,max}}$) are defined in the same way as the $z_{\text{offset,default}}$ except for changing the angle AV_y to smaller or larger angles. The zoom action on the map uses pinch-to-zoom. In pinch-to-zoom, two distances between two points (the position of two fingers) on the touch sensible display of the device are compared. So, for a smooth transition between two pinch-to-zoom actions the previous zoom factor $f_{\text{zoom,prev}}$ is multiplied with the new retrieved zoom factor $f_{\text{zoom,new}}$. Additionally, minimal and maximal values (f_{min},f_{max}) for the zoom factor f_{zoom} have to be set to prevent too small or too large zoom factors. The final zoom factor is computed as in Equation (4.24):

$$f_{\text{zoom}} = f_{\text{zoom,prev}} = \min(\max(f_{\text{zoom,prev}} f_{\text{zoom,new}}, f_{\text{min}}), f_{\text{max}}) \tag{4.24}$$

If the map is zoomed in, the map geometry has either larger size or is closer to the position of the camera. This will lead to a visible map in the `AR` viewer at the exploration of `POI` and disrupt the user's view. Therefore, the map is reset to its default offset $z_{\text{offset,default}}$, when the focal vector $\mathbf{c}_{target}$ of the camera comes near the XY-plane of the world coordinate frame.

4.7 Tools

As mentioned before the implementation is running on devices with the `OS` Android. The `AR` browser supports mobile devices running Android starting with Version 2.3 Gingerbread (`API` Level 9). Applications for Android are written in Java, furthermore it is possible to run native code written in C(++) via the Java Native Interface(`JNI`).

Additionally, I use OpenSceneGraph[§] (*OSG*) for my implementation. This high performance 3D graphics toolkit allows to save and manage the 3D graphics data in a scene graph. In this scene graphs all everything is saved as nodes, like geometries, groups, transformations. It handles moreover the rendering of the scene graph by traversing the scene graph at every frame.

4.8 Testing

An early prototype to test the described clustering algorithms in Section 4.4.1 was implemented on a PC running Windows 7. Later, the code was ported and modified to work

[§]`http://www.openscenegraph.org/`

with the JNI on an Android mobile device. Although, for the graphics OpenSceneGraph is used, some modifications, especially the use of a different shading language (OpenGL ES 2.0 [¶]), had to be done to run on an Android mobile device. Finally, the code was mainly tested on a Google Galaxy Nexus running Android 4.3. Additionally, the application was tested on a 7″ tablet (Google Nexus 7, running Android 4.4). Unfortunately this mobile device has only a front camera and no rear camera. Despite this restriction the application runs on both devices. The application runs at real time at approx. 60 frames per seconds.

[¶] http://www.khronos.org/opengles/

Chapter 5

Examples

Contents

In this chapter I will present some examples for the usage of the AR browser. Therefore, I discuss and describe some use cases for the main features of the AR browser:

- the exploration of the surrounding area (see Section 5.1),
- the usage of the category filtering possibility (see Section 5.2),
- the use of the AR location search to find a specific POI (see Section 5.3)

It should be mentioned, that the position of the places may not be correct because the data of geographic coordinates and especially the altitudes are used as retrieved by the used APIs. Furthermore, it is possible that there a places with (nearly) the same name and position, again this is caused by the used data source. The correct rendering and order of objects was not the goal of this work, therefore it may be possible that the rendered objects are not displayed correctly.

The alert reader may recognize in the following images and the images in Chapter 3 the application icon in the top-left corner of the screen. This icon uses an eye with a plus instead of a pupil. This icon should symbolize the augmentation of the reality. The three letters '**AR**C' stand for the abbreviation either of **A**ngula**R** **C**lustering or **A**ugmented **R**eality **C**lustering. This expression is used because one of the main tasks of the implementation of the AR browser is the clustering of data points.

5.1 Exploration

The exploration of the nearby real world places is one of the main purposes of the AR browser. To do so, the user opens the application and a short initialization phase begins, which includes the determination of the current location by GPS or other network based methods. Then the nearby places are retrieved, either over the internet with the Google Places API or from the internal data base, if the location was visited in the past. This set up phase may take some time and depends on the network reception and the number of places to parse.

After the set up phase the user is able to use the device in landscape mode and is able to start exploring the surrounding places. This is shown in Fig. 5.1. In this image two clusters and one detailed label can be spotted.

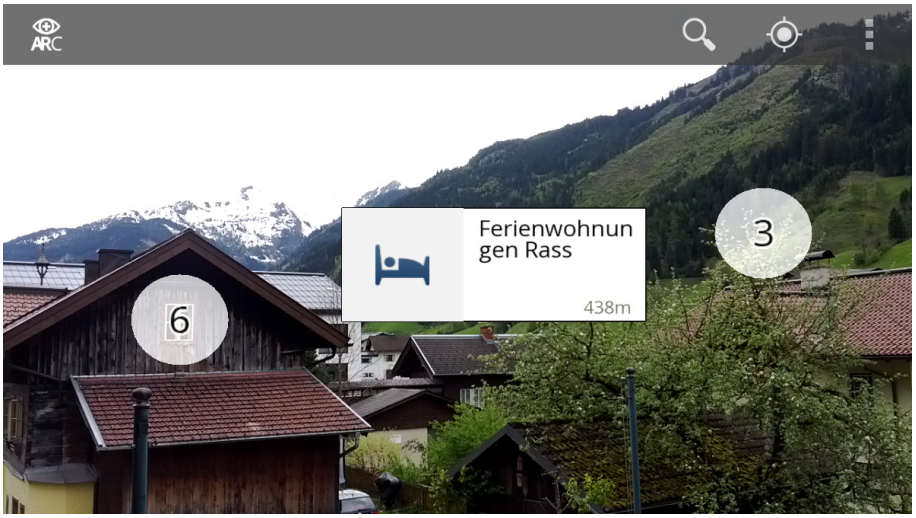


Figure 5.1: AR browser exploration example: visualization of a detailed, single-item cluster label and two multi-item cluster labels.

If the user wants to know which area is shown in front of her, she points the camera of the device on the floor. An overview map of the surrounding area is shown and the field of view of the camera is indicated as a white triangle (see Fig. 5.2).

A cluster is represented as a circle with the number of cluster members (in this case: 4). After selecting the cluster by touching the circle the cluster data detail visualization is opened (see Fig. 5.3). The detail visualization shows the members of the selected cluster: the POI are annotated at their position in the world. These annotations are connected to the labels via leader lines. The number of possible slots (labels) is restricted in the cluster data detail visualization. If the number of data points is larger than the number

Figure 5.2: AR browser exploration example: pointing the device on the ground to show the map. Only the FOV indicator is shown and nothing is highlighted by markers.

of available labels a hierarchy within the labels is created. An example of this case will be depicted later.

As example, the user gets interested in the POI labelled with the name 'Hotel Gasthof Schützenhof'. So, she wants more information and touches the label on the screen. This opens the dialog with the details of the location. This is depicted in Fig. 5.4. In the dialog the following details for the POI are shown: name, address, distance from the current position and categories. She wants to know, where to find this place on the overview map. Hence, a click on the 'Show place on map'-button (annotation (1) in Fig. 5.4) closes the dialog and a line appears on the screen, which connects the annotated position on the screen and the position on the integrated overview map. Additionally, a marker at the position of the place is added to the overview map. Both, the line and the marker can be found in Fig. 5.5. The connecting line will disappear, if the center of the screen is pointed at the position on the integrated map.

After pointing the device at the marker on the overview map, the user continues with normal exploration and opens another cluster. The visualization of the places of this cluster is shown in Fig. 5.6. This cluster has more places to label than the available slots. Hence, one label hints to an additional level within the cluster. This hint is given by an alternative label design: stack of labels and a plus-sign-icon (see Fig. 5.6 - (1)). After tapping on the plus-sign-icon, the places of this label hierarchy level are shown (see Fig. 5.7). This includes the leading POI of the previous labelling level in the hierarchy

Figure 5.3: AR browser exploration example: a cluster with less than the maximum number of slots.

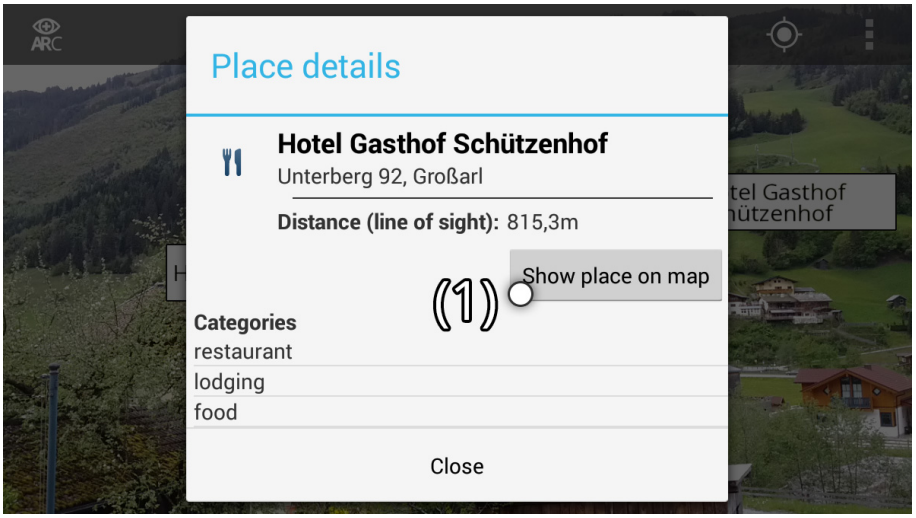


Figure 5.4: AR browser exploration example: a dialog box opens after tapping on the 'Hotel Gasthof Schützenhof' label depicted in Fig. 5.3. The place could be marked on the integrated overview map by clicking on (1). This will close the dialog.

(in Fig. 5.7 the label on the left side). If the user wants to return to the previous level, she can achieve this action by clicking on the minus-sign-icon. The minus-sign-icon can be found at (1) in Fig. 5.7. At all levels the cluster date detail visualization can be exited by touching on the highlighted area in the center of the cluster (see Fig. 5.7 - (2)) or by turning the device away from the cluster center.

If the user is interested in the location of all surrounding nearby POI, she change to

Figure 5.5: AR browser exploration example: a white line connects the POI annotation and the position on the overview map. This helps to find the place on the map, which is annotated with the gray flag marker.

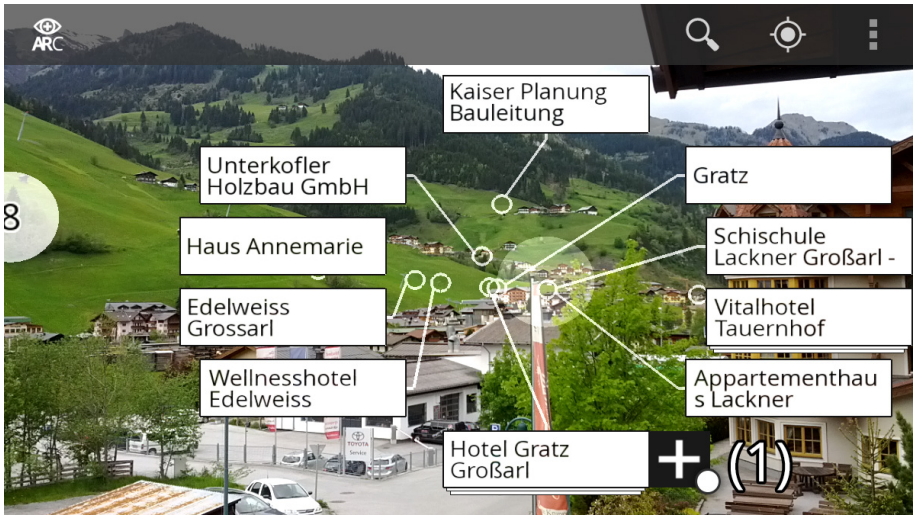


Figure 5.6: AR browser exploration example: a cluster with more than the maximum number of slots is opened. At (1) a plus-sign-icon is added to the label: this indicates an additional clustering level.

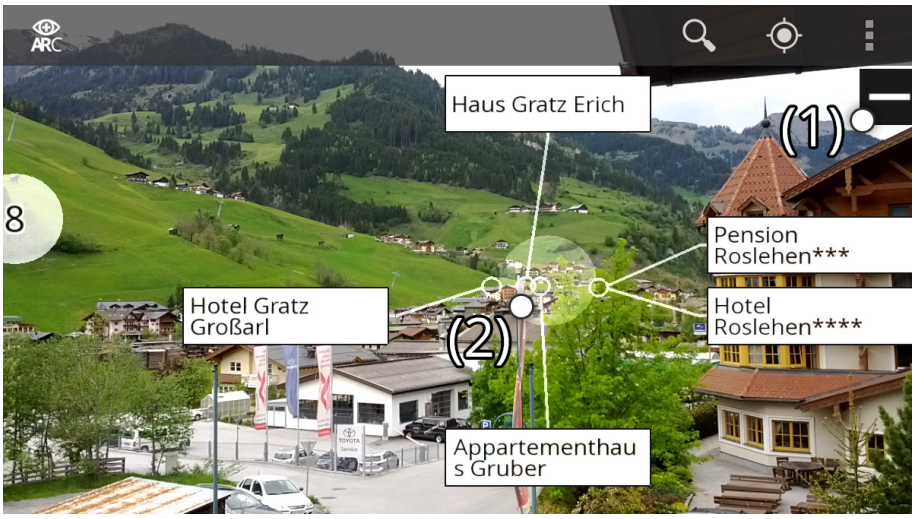


Figure 5.7: AR browser exploration example: the visualization of the cluster after clicking on the plus-icon of Fig. 5.4. A tap on the minus-sign (1) allows to return to the previous level and touching the area at (2) exits the cluster data detail visualization.

the map viewer. In the overview map in the map viewer every POI is annotated by a white round marker. The map viewer with the markers for POI can be found in Fig. 5.8.

Figure 5.8: AR browser exploration example: the map viewer shows the surrounding POI with markers on an overview map.

5.2 Filter Places by Categories

The implementation of the AR browser allows to filter the nearby POI by their categories. So, the 'Select categories'-dialog is opened (see Fig. 5.9) from the context-menu. The user can check and uncheck categories from the category filter list. By default all categories are selected and no filtering is performed. As depicted in Fig. 5.9 three categories, health, liquor-store and lodging, are selected as examples. After pressing the 'OK'-button in the dialog, the filter is applied and a new clustering is performed. The AR overview+detail viewer changes its content to the filtered data visualization.

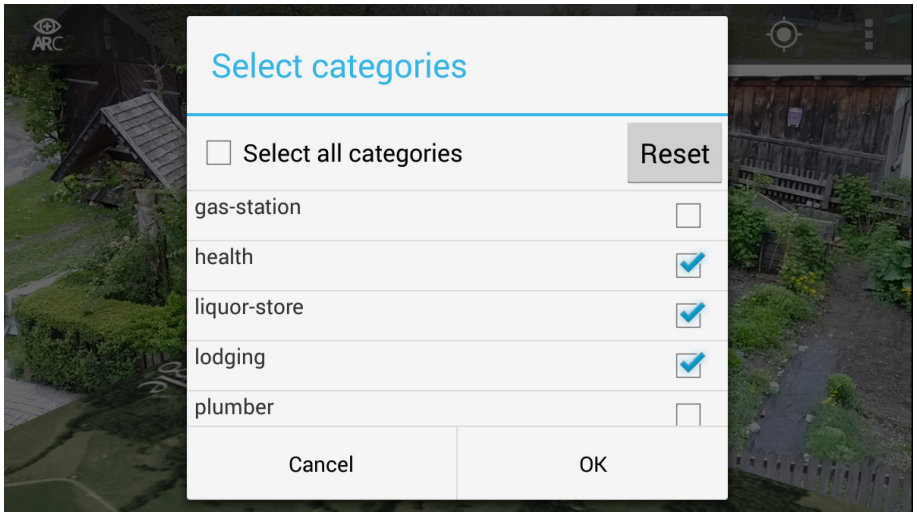


Figure 5.9: AR browser category filter example: the 'Select categories'-dialog is opened after selecting the option from the context-menu. Three categories are selected: health, liquor-store and lodging.

The labels use color coded glyphs to represent the quality of the results. Therefore, green represents a result that matches all selected categories (perfect result). Blue stands for a result with more than one match (good result) and the red color represents a results with only one matching category (bad result). This color coding for the quality will be applied to both label representations: the detailed label and the label indicating a cluster.

In Fig. 5.10 two clusters with the color indicators are shown. The distribution of the quality of the results within the cluster can be seen before opening the cluster detail visualization. As example, the cluster with 4 members ((1) in Fig. 5.10) has one good and one result with perfect matching. Additionally, two members have at least one match for the categories (bad result). The second cluster has a green indicator, which means that there is no member with a perfect result within this cluster. In contrast, there are many

results with only one match.

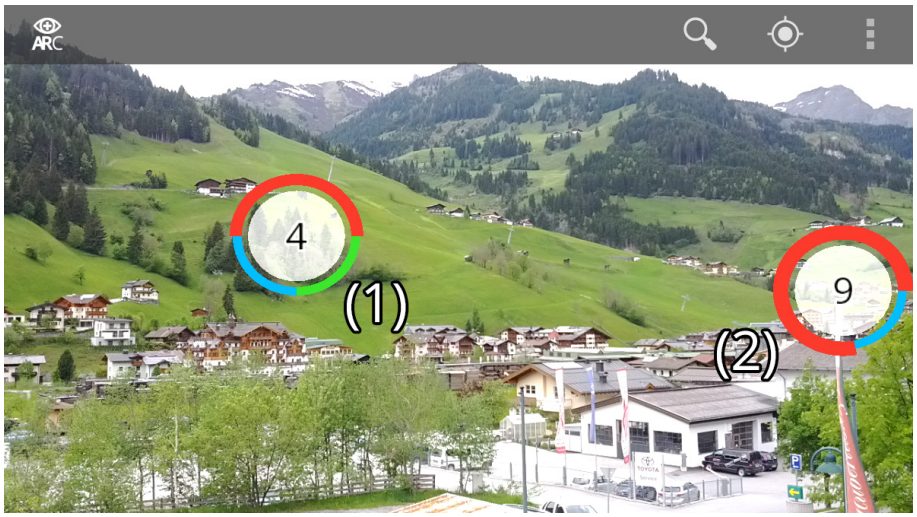


Figure 5.10: AR browser category filter example: the result after pressing the 'OK'-button of the dialog. Two clusters with different result qualities are shown. (1) has three different qualities within the cluster, whereas (2) has only two qualities. In (1) is one result which matches all categories.

The user wants to investigate the members of the cluster with a number of four POI. Therefore, she opens the cluster by touching at the label. The resulting open cluster can be found in Fig. 5.10. The quality of the results is displayed with a thick colored line on the right side of the labels for the POI.

The quality of the results after applying the categories filter is advertised on the overview map in the AR overview+detail viewer too. Therefore, the flags of the markers on the overview map use the same color coding as for the labels. An image of the markers can be found in Fig. 5.12(a), which represents the results of the filter applied on the data set. After changing to the overview map in the map viewer, the quality of the results of the category based filtering is displayed in a slightly different manner, but the color coding is maintained (see Fig. 5.12(b)).

The categories for the filter are changed in the corresponding dialog. Only two categories, food and health, are selected. This selection is depicted in Fig. 5.13. It should be mentioned that the selection of all or none of the categories results in the unfiltered data set. The selection of only two categories is a special case for the filter result quality representation. If only two categories are selected, the color red, which indicates bad results, is not used. So, there will be only good (blue) and green (perfect result) indicators.

The results after applying the filter and performing the clustering are shown

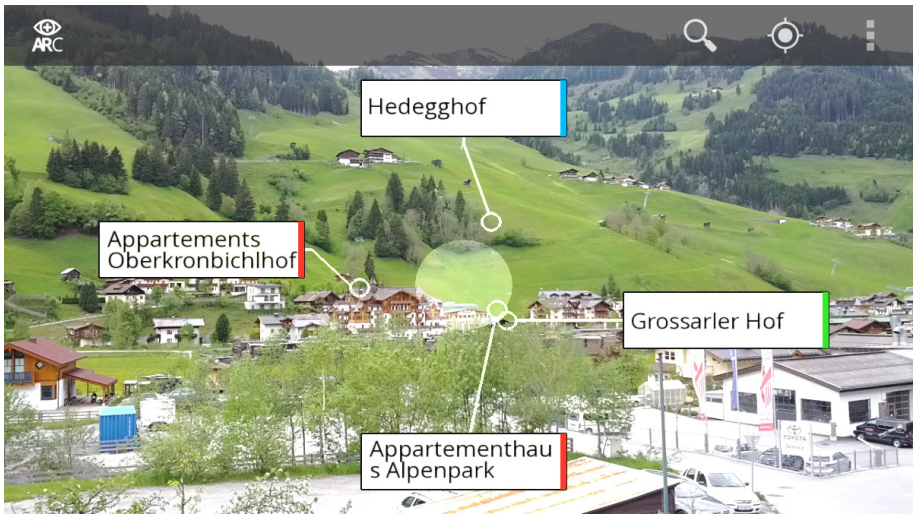
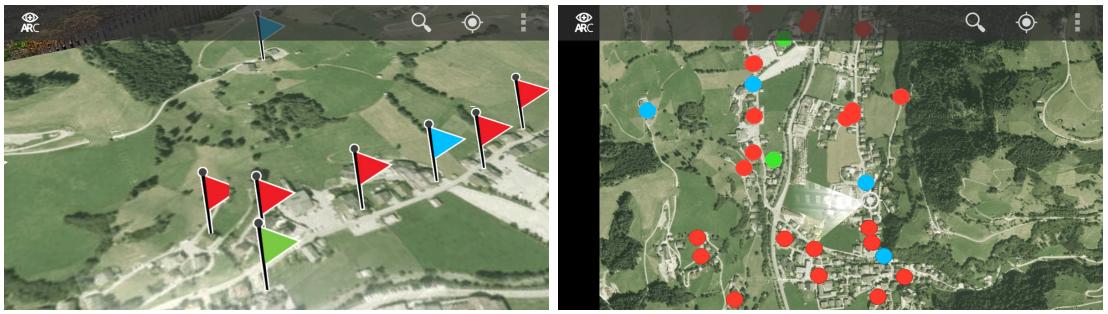


Figure 5.11: AR browser category filter example: the opened cluster of Fig. 5.10 - (1). There a two results with one match, one with a good match and one, which matches all selected categories.



(a) (b)

Figure 5.12: AR browser category filter example: the indicators for the quality of the results on the overview map in the different viewers. (a) Map in the AR viewer. (b) Map viewer.

in Fig. 5.14. In this case, the two different labels for clusters are shown. The label for a multi-item cluster (see Fig. 5.14 - (1)) indicates two different qualities of results. A blue (good) result with one match and a green match which fulfils both selected categories. The detailed label of a POI at (2) in Fig. 5.14 displays the quality with a blue indicator stripe on its right border.

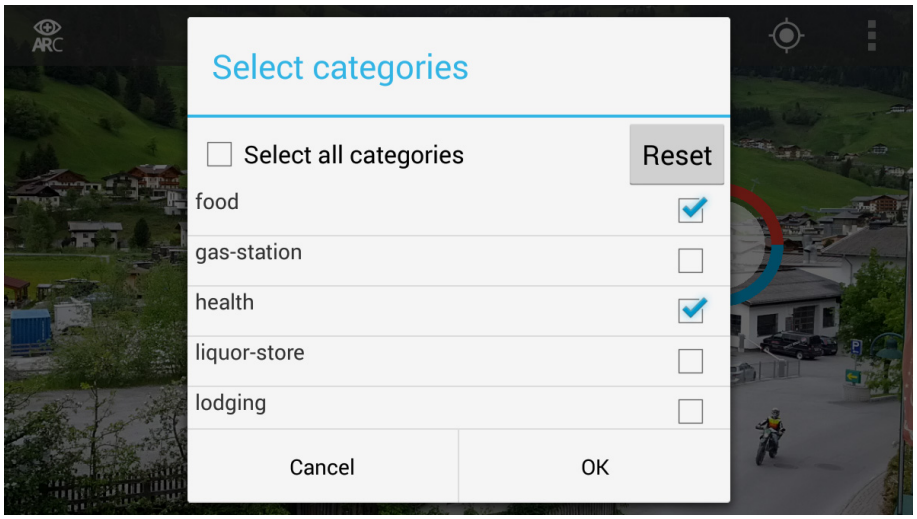


Figure 5.13: AR browser category filter example: the dialog for selecting the categories for the filter. Only two categories are selected: food and health.

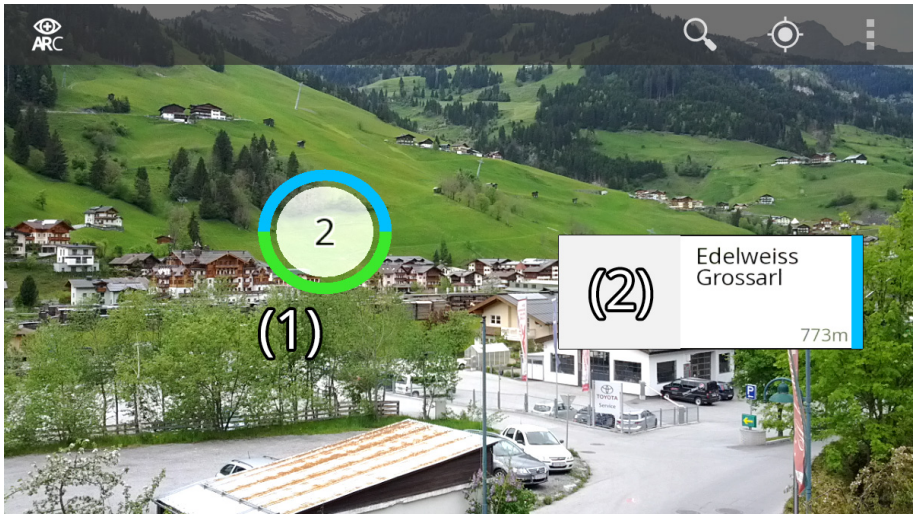
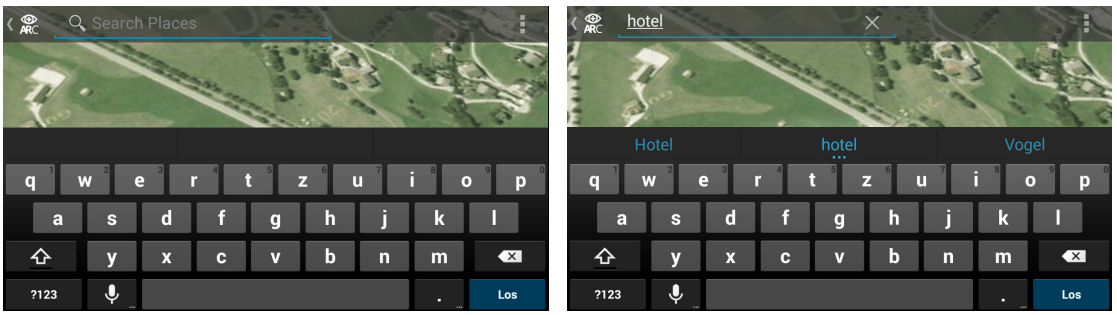


Figure 5.14: AR browser category filter example: the result after pressing the 'OK'-button of the dialog in Fig. 5.13. A cluster label with quality indicator (1) and a detailed result label (2) with the quality display on the right edge of the label.

5.3 Search for a Specific Place - AR Location Search

The AR browser allows to search for nearby places by name and address with the AR location search. Therefore, the user opens the search term input mask (see Fig. 5.15(a)) by tapping on the magnifying glass icon in the ActionBar in the AR overview+detail viewer. The input mask allows to pass multiple, space separated search terms. The order of the terms is not

important. In this example, the user wants to search for nearby POI which match the term 'hotel'. So, she enters the search term 'hotel' in the input box and starts the search (see Fig. 5.15(b)).



(a) (b)

Figure 5.15: AR location search example: input of a search term. (a) The empty input box, asking for a search term ('Search Places'). (b) After entering the search term 'hotel'.

After the search query is executed on the data set in internal data base, a dialog box with a list of the matching POI appears. This dialog box is depicted in Fig. 5.16. In this case, there are five matching locations in the list. The nearby places are listed in ascending order by distance to the current position. As example, the user wants to know where to find the first POI in the list: 'Hotel Hubertushof'. So, she taps on the left icon, which is in front of the name and address of the place. This will start the AR location search. If she only wants to show the places within this list, a click on the 'Show results'-option sets the entries of the list as the only shown data points in the AR overview+detail viewer.

In case the user started the guided search, the content of the AR viewer changes to the AR location search. In the AR location search, which is shown in Fig. 5.17, the user will spot a few differences in contrast to the previous clustered data visualization. First, there are no labels for the nearby places and cluster shown. Instead, a circle with a point in the middle appears on the center of the screen. This visual helper (crosshairs) allows to point at the target correctly. Additionally, an arrow located at one of the four borders of the view frame appears. This arrow will change its size and position to help finding the desired place. In the example in Fig. 5.17 the arrow points to the right side, which means, that the user has to turn right to find the the POI.

If she selected the POI accidentally, the AR location search can be exited by tapping on the 'Exit search mode'-button in the top-right corner.

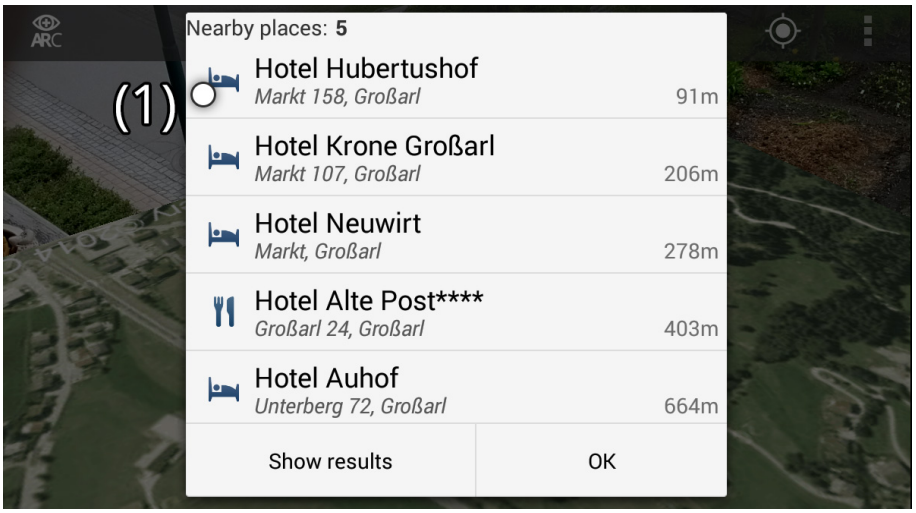


Figure 5.16: AR location search example: a list of the search results in ascending order by their distance to the current position.

Figure 5.17: AR location search example: the visualization changes from the clustered data visualization to the AR location search.

An overview map can be a great benefit at finding a specific place. Therefore, the location of the POI is marked on the integrated overview map in the AR overview+detail viewer. So, a POI may be found much faster with the overview map than using only the guidance given by the arrows on the screen. An impression of the overview map in the active AR location search can be found in Fig. 5.18. In the given figure the arrow hints to take a look upwards to find the looked-up place.

After pointing the device upwards in the direction of the target, an additional circle

Figure 5.18: AR location search example: the marker for the current POI is shown on the integrated overview map in the AR viewer.

appears on the top-right corner of the screen as depicted in Fig. 5.19. This circle has its size animated and represents the position of the looked-up POI. It can be seen, that the directing arrow points still to right side, but its size is decreased. That means, that the user is near the target.

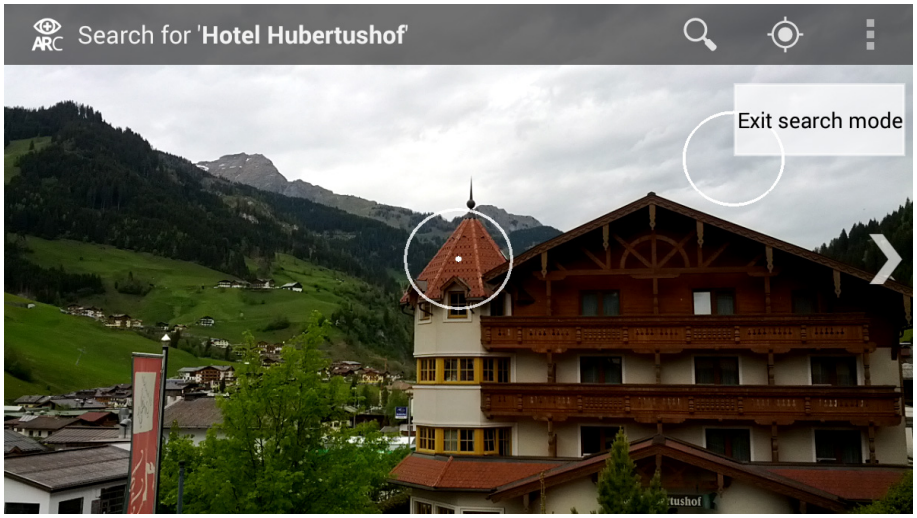


Figure 5.19: AR location search example: the user is guided towards the POI positioned at the top-corner (circle).

The AR location search is finished, when the targeted point of interest is within the targeting circle around the center of the screen. The content of the AR overview+detail

viewer changes back to the clustered data visualization. Additionally, if the target is within a cluster, the cluster detail visualization is shown after finishing the AR location search. This case complies for the given example in Fig. 5.20. The label for the looked-up place is annotated at (1). If the POI is in an upper level of the sub-cluster hierarchy, the hierarchy will be traversed to show the right level.

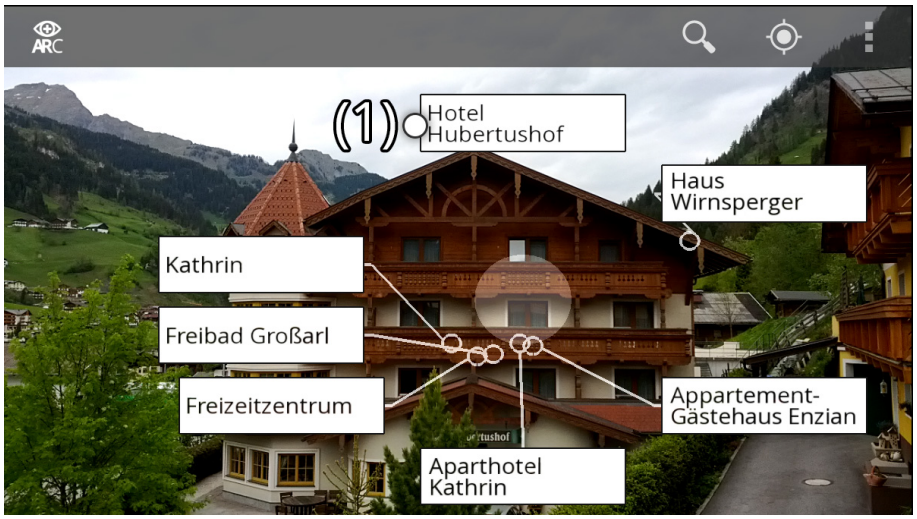


Figure 5.20: AR location search example: return to the clustered data visualization after the POI (label at (1)) is found. The POI is within a cluster, so the cluster is opened.

Chapter 6

Summary and conclusion

Contents

6.1 Summary

In this thesis, I presented a new system for an AR browser for mobile devices running on the Android platform. This AR browser allows to visualize and interact with points of interest. In the case of this thesis, the POI are real world places, which are retrieved from the Google Places API. Therefore, the sensors of the mobile device (GPS, magnetometer, gyroscope) are used for the determination of the current location on earth and the orientation of the device. The retrieved places are saved into a database for further usage. Furthermore, the geographic coordinates of the POI are converted for the use as data points in the framework. My implementation uses two programming languages. First, the Java programming language to retrieve the location and sensor data on the Android device. Second, the logic for the data representation is written as native C++ code. For the graphical visualization on the screen of the mobile device the OpenSceneGraph (*OSG*) graphics toolkit is used. The *OSG* toolkit allows to handle, generate and manipulate simple geometry up to complex models. Everything within the *OSG* graphics toolkit is managed in a scene tree, where each operation and geometry is represented as node of the scene tree. The nesting of nodes is possible and every frame the scene graph is traversed to perform the different operations and render the graphics.

The AR browser supports different visualization modes for the given data of the POI. In the general usage of the browser a simple filter mechanism by distance is used. Furthermore, a more complex filtering by the categories of the nearby places or a search of places is supported.

The main goal for the visualization methods is to present the data of the POI in a clear and non-cluttered way on the screen. Therefore, I implemented a clustering algorithm, which exploits the angles between the vector representation of the data points in the world reference frame. This, so called Angular Clustering, method is used to optimize the presentation of the labels of the POI on the screen for user interaction and avoids clutter of the labels. The advantage of the used clustering method is, that is has only be performed when the dataset is changed either by modifying the size of the dataset or changing the position of points of the dataset. Other cluster analysis methods, like the well-known k-means clustering [39], have to be performed on the projected data points (screen coordinates). This may lead to a new clustering after every change of the orientation and/or position of the device. Another drawback of other clustering methods is, that often one or more parameters have to be found, like the number k of wanted clusters in the k-means clustering method. The determination of the optimal parameter(s) for the clustering algorithm takes account to the runtime of the algorithm. This is especially important in real-time applications. Another advantage of the proposed clustering method is, that there is no need to post-process the resulting cluster for the later labelling step. The proposed Angular Clustering method takes account of the size of the later occupied space of the labels on the screen of the device. Therefore, an aggregation of labels does not occur on the rendered labels, so there is no need of a joining operation on overlapping labels after the clustering.

The Angular Clustering results in two cluster types: the single-item cluster and multi-item cluster. The multi-item cluster has an own visualization: the detail visualization of clustered data, which supports further a hierarchical structure of the POI within the clusters. Therefore, I have proposed a layout for the representation in this thesis and discussed how do build the hierarchy within the cluster representation. Due to the structure of the proposed sub-cluster layout, the labels are not always near their corresponding data points on the screen. Therefore, leader lines are used to present this connection. Taking the shortest connection between label and a data point may lead to intersections of the leader lines. Therefore, I discussed a method to select the label for a given data point along with the avoidance of intersections between the leader lines.

Another topic of this thesis is a solution for the problem of finding a way to indicate the quality of the points of interest after filtering by their categories. I have not chosen to display the quality by showing the rank nor the percentage of overlap of the result, but rather using a color coded glyph design. Thereby, the representation of the quality of a POI is rather simple at a single label for a data point. On the other hand, the design of the rank indicator of a cluster allows to detect the quality of the cluster members without visualizing the members of the cluster. The chosen approach allows to attract the user's attention with an easy to understandable color coding and without the need of much screen space.

In addition, I deal with the combination of overview and detail in a single view. The concept of overview+detail is presented in this thesis by providing the user with an overview map of the surrounding area within the AR viewer. This allows the user to extend her restricted FOV and to get a better insight of the connection between the annotation of a place on the screen and its real position in the world. The presented overview+detail approach is based on the idea of holding a physical map. The enhanced map in AR allows to change the scale, zoom in and out. Additionally, markers can be used to show specific POI or the results after a filtering.

Another problem, which I addressed in this thesis, is the problem of finding a specific location in the surrounding nearby area. Therefore, I implemented an AR location search which guides the user to a selected location. The AR guidance gives the user direction hints to change the orientation of the device. Furthermore, the presented algorithm takes account of the restricted mobility of a human individual for the guidance to the desired POI.

6.2 Conclusion and Future Work

I have bothered in this thesis the implementation of an augmented reality browser for real world data. The AR browser framework takes care of the acquisition of the needed data, the processing and the presentation of the data. For this purpose, I presented different methods and algorithms to achieve a clutter-free and intuitive interface for representing the nearby POI. One of these methods includes the clustering of the given data points. Therefore, I described a clustering algorithm using the angles between the direction vectors of the data points, which are determined by taking the current position and the position of the given data points. Further, I compared the proposed approach to the well-known k-means algorithm for cluster analysis. In future, the proposed Angular Clustering algorithm has to

be tested on larger data sets and for different label sizes and cameras with different angles of view. This may reveal possible problems and lead to improvements of the algorithm.

One of the main topics in this thesis is the visualization of the data on a mobile device screen. Therefore, I discussed the idea of a fixed layout for the visualization of clustered data. Additionally, the allocation of the labels to the coherent data points is explained in detail. Another topic concerning the visualization is discussed in detail in this thesis, which is the representation of filtered and ranked results. Therefore, I presented a way to express the quality of the content of a cluster and of a single `POI` by using glyphs of different size and color.

Further, I discussed the concept of overview+detail in the `AR` browser. The implementation uses an integrated overview map in the `AR` overview+detail viewer to visualize the surrounding area and give additional information about the position of specific places. Furthermore, the positioning of the map is chosen in such a way, that it does not to disturb the user experience at the exploration of the nearby points of interest. In addition, I occupied myself with the problem of finding the location of a wanted place. Thus, I presented a concept and the implementation of a way to guide a user to find the targeted `POI` with the help of augmented reality.

A missing chapter in this thesis is the evaluation of the proposed approaches. So, one of the main tasks for the future is the evaluation of the `AR` browser. This evaluation should include a user study to compare the user experience and performance of the `AR` browser proposed in this thesis against other available `AR` browsers.

There are many possibilities of improving and extending the `AR` browser framework. Besides to solve certain stability issues and the testing on different devices, the framework can be extended quite easily. So, e.g. the shown map could be used in an more interactive way. To do so, the map could show more information about the `POI` and the a selection of specific places and areas would be useful features. Further, the positioning of the annotations could be improved by using vision-based clues.

Appendix A

Acronyms and Symbols

List of Acronyms

AB	ActionBar
AR	Augmented Reality
API	Application Program Interface
GPS	Global Positioning System
JNI	Java Native Interface
FOV	Field of View
MARS	Mobile Augmented Reality System
OS	Operating System
OSG	OpenSceneGraph
POI	Point of Interest
UI	User Interface
URL	Uniform Resource Locator
WP	Windows Phone

List of Symbols

$< x, y >$	Tensor dot product
$\dot{x}(t)$	Derivative with respect to t
E	Expectation value
$\mathfrak{F}$	Fourier transform
$\mathfrak{F}^{-1}$	Inverse Fourier transform
j	Imaginary number, $\sqrt{-1}$
$\Im$	Imaginary part of a complex number
$\Re$	Real part of a complex number
δ	Dirac delta function
$\delta_{\triangle}$	Kroneker delta function
x $*$ y	Convolution
x^*	Conjugate complex
∇	Nabla operator
$\triangle$	Laplace operator
$\mathbf{I}$	Identity matrix
$\mathcal{N}(\mu, \sigma)$	Normal distribution
$\mathcal{N}_{\circ}(\mu, \kappa)$	Von Mises distribution (circular normal distribution)
$\chi^2(n, \sigma)$	Chi-square distribution
$\Gamma(\alpha)$	Gamma function
$R_{xx}(\tau)$	Autocorrelation
$S_{xx}(i\omega)$	Spectral density function

Bibliography

[1] Andrienko, G. L., Andrienko, N. V., Rinzivillo, S., Nanni, M., Pedreschi, D., and Giannotti, F. (2009). Interactive visual clustering of large collections of trajectories. In *IEEE VAST*, pages 3–10. IEEE.

[2] Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99)*, pages 49–60, Philadelphia, PA.

[3] Baudisch, P. and Rosenholtz, R. (2003). Halo: a technique for visualizing off-screen objects. In Cockton, G. and Korhonen, P., editors, *CHI*, pages 481–488. ACM.

[4] Baudisch, P., Zotov, A., Cutrell, E., and Hinckley, K. (2008). Starburst: a target expansion algorithm for non-uniform target distributions. In Levialdi, S., editor, *AVI*, pages 129–137. ACM Press.

[5] Bederson, B. B. (1995). Audio augmented reality: a prototype automated tour guide. In Miller, J., Katz, I. R., Mack, R. L., and Marks, L., editors, *CHI 95 Conference Companion*, pages 210–211. ACM.

[6] Biocca, F., Tang, A., Owen, C. B., and Xiao, F. (2006). Attention funnel: omnidirectional 3d cursor for mobile augmented reality platforms. In Grinter, R. E., Rodden, T., Aoki, P. M., Cutrell, E., Jeffries, R., and Olson, G. M., editors, *CHI*, pages 1115–1122. ACM.

[7] Burigat, S. and Chittaro, L. (2008). Decluttering of icons based on aggregation in mobile maps. In Meng, L., Zipf, A., and Winter, S., editors, *Map-based Mobile Services*, Lecture Notes in Geoinformation and Cartography, pages 13–32. Springer Berlin Heidelberg.

[8] Carmo, M. B., Afonso, A. P., Pombinho De Matos, P., and Vaz, A. (2008). *MoViSys - A Visualization System for Geo-Referenced Information on Mobile Devices*, volume 5188 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[9] Caudell, T. and Mizell, D. (1992). Augmented reality: an application of heads-up display technology to manual manufacturing processes. In *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*, volume ii, pages 659–669 vol.2.

[10] Chittaro, L. (2006). Visualizing information on mobile devices. *IEEE Computer*, 39(3):40–45.

[11] Cockburn, A., Karlson, A. K., and Bederson, B. B. (2008). A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.*, 41(1).

[12] de Matos, P. P., Carmo, M. B., and Afonso, A. P. (2009). Evaluation of overcluttering prevention techniques for mobile devices. In Banissi, E., Stuart, L. J., Wyeld, T. G., Jern, M., Andrienko, G. L., Memon, N., Alhajj, R., Burkhard, R. A., Grinstein, G. G., Groth, D. P., Ursyn, A., Johansson, J., Forsell, C., Cvek, U., Trutschl, M., Marchese, F. T., Maple, C., Cowell, A. J., and Moere, A. V., editors, *IV*, pages 127–134. IEEE Computer Society.

[13] Delort, J.-Y. (2010). Hierarchical cluster visualization in web mapping systems. In Rappa, M., Jones, P., Freire, J., and Chakrabarti, S., editors, *WWW*, pages 1241–1244. ACM.

[14] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistcal Society*, 39(B):1–38.

[15] Ellis, G. and Dix, A. (2007). A taxonomy of clutter reduction for information visualisation. *IEEE transactions on visualization and computer graphics*, 13(6):1216–23.

[16] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd International Conference on Knowledge Discovery and*, pages 226–231.

[17] Feiner, S., MacIntyre, B., Höllerer, T., and Webster, A. (1997). A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. In *ISWC*, pages 74–81. IEEE Computer Society.

[18] Feiner, S., MacIntyre, B., and Seligmann, D. D. (1993). Knowledge-based augmented reality. *Commun. ACM*, 36(7):53–62.

[19] Fink, M., Haunert, J.-H., Schulz, A., Spoerhase, J., and Wolff, A. (2012). Algorithms for labeling focus regions. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2583–2592.

[20] Fröhlich, P., Simon, R., Baillie, L., and Anegg, H. (2006). Comparing conceptual designs for mobile access to geo-spatial information. In Nieminen, M. and Röykkee, M., editors, *Mobile HCI*, pages 109–112. ACM.

[21] Google Inc. Google maps coordinates. Available online at `https://developers.google.com/maps/documentation/javascript/maptypes%23MapCoordinates`, last visited 6.6.2014.

[22] Google Inc. Google places API types. Available online at `https://developers.google.com/places/documentation/supported_types`, last visited 15.5.2014.

[23] Guo, D., Peuquet, D., and Gahegan, M. (2002). Opening the black box: interactive hierarchical clustering for multivariate spatial patterns. In Voisard, A. and Chen, S.-C., editors, *ACM-GIS*, pages 131–136. ACM.

[24] Hamerly, G. and Elkan, C. (2002). Alternatives to the k-means algorithm that find better clusterings. In *CIKM*, pages 600–607. ACM.

[25] Hartmann, K., Ali, K., and Strothotte, T. (2004). Floating labels: Applying dynamic potential fields for label layout. In Butz, A., Krüger, A., and Olivier, P., editors, *Smart Graphics*, volume 3031 of *Lecture Notes in Computer Science*, pages 101–113. Springer.

[26] Höllerer, T., Feiner, S., Hallaway, D., Bell, B., Lanzagorta, M., Brown, D. G., Julier, S., Baillot, Y., and Rosenblum, L. J. (2001). User interface management techniques for collaborative mobile augmented reality. *Computers & Graphics*, 25(5):799–810.

[27] Höllerer, T., Pavlik, J. V., and Feiner, S. (1999). Situated documentaries: Embedding multimedia presentations in the real world. In *ISWC*, pages 79–86. IEEE Computer Society.

[28] Höllerer, T. H. (2004). *User Interfaces for Mobile Augmented Reality Systems*. PhD thesis, New York, NY, USA. AAI3115354.

[29] Hohl, F., Kubach, U., Leonhardi, A., Rothermel, K., and Schwehm, M. (1999). Next century challenges: Nexus - an open global infrastructure for spatial-aware applications. In Kodesh, H., Bahl, V., Imielinski, T., and Steenstrup, M., editors, *MOBICOM*, pages 249–255. ACM.

[30] Jo, H., Hwang, S., Park, H., and hee Ryu, J. (2011). Aroundplot: Focus+context interface for off-screen objects in 3d environments. *Computers & Graphics*, 35(4):841–853.

[31] Julier, S., Baillot, Y., Brown, D. G., and Lanzagorta, M. (2002). Information filtering for mobile augmented reality. *IEEE Computer Graphics and Applications*, 22(5):12–15.

[32] KHARMA (2014). KARML reference. Available online at `https://research.cc.gatech.edu/kharma/content/karml-reference`, last visited 15.5.2014.

[33] Kähäri, M. and Murphy, D. J. (2006). MARA - sensor based augmented reality system for mobile imaging. In *Proceedings of the 5th Annual IEEE and ACM International Symposium on Mixed and Augmented Reality*.

[34] Kooper, R. and MacIntyre, B. (2003). Browsing the real-world wide web: Maintaining awareness of virtual information in an ar information space. *Int. J. Hum. Comput. Interaction*, 16(3):425–446.

[35] Kuhn, H. (1955). The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

[36] Layar (2014). Layar. Available online at `https://www.layar.com/`, last visited 15.5.2014.

[37] Lee, R., Kitayama, D., Kwon, Y., and Sumiya, K. (2009). Interoperable augmented web browsing for exploring virtual media in real space. In *LocWeb*, ACM International Conference Proceeding Series, page 7. ACM.

[38] Lehikoinen, J. and Suomela, R. (2002). Accessing context in wearable computers. *Personal and Ubiquitous Computing*, 6(1):64–74.

[39] Lloyd, S. (2006). Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137.

[40] Maass, S. and Döllner, J. (2006). Efficient view management for dynamic annotation placement in virtual landscapes. In Butz, A., Fisher, B., Krüger, A., and Olivier, P., editors, *Smart Graphics*, volume 4073 of *Lecture Notes in Computer Science*, pages 1–12. Springer.

[41] MacIntyre, B., Hill, A., Rouzati, H., Gandy, M., and Davidson, B. (2011). The argon ar web browser and standards-based ar application environment. In *ISMAR*, pages 65–74. IEEE.

[42] metaio GmbH (2014). Junaio. Available online at `http://www.junaio.com/`, last visited: 15.5.2014.

[43] Mulloni, A., Dünser, A., and Schmalstieg, D. (2010). Zooming interfaces for augmented reality browsers. In de Sá, M., Carriço, L., and Correia, N., editors, *Mobile HCI*, ACM International Conference Proceeding Series, pages 161–170. ACM.

[44] Munkres, J. R. (1957). Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38.

[45] National Imagery and Mapping Agency (2000). Department of defense world geodetic system 1984: its definition and relationships with local geodetic systems. Technical Report TR8350.2, National Imagery and Mapping Agency, St. Louis, MO, USA.

[46] Nokia (2014). HERE city lens. Available online at `http://www.windowsphone.com/en-us/store/app/here-city-lens/b0a0ac22-cf9e-45ba-8120-815450e2fd71`, last visited 15.5.2014.

[47] Rantanen, M., Oulasvirta, A., Blom, J., Tiitta, S., and Mäntylä, M. (2004). Inforadar: group and public messaging in the mobile context. In Raisamo, R., editor, *NordiCHI*, pages 131–140. ACM.

[48] Rekimoto, J., Ayatsuka, Y., and Hayashi, K. (1998). Augment-able reality: Situated communication through physical and digital spaces. In *ISWC*, pages 68–75. IEEE Computer Society.

[49] Schmalstieg, D., Langlotz, T., and Billinghurst, M. (2008). Augmented reality 2.0. In Coquillart, S., Brunnett, G., and Welch, G., editors, *Virtual Realities*, pages 13–37. Springer.

[50] Schwerdtfeger, B., Reif, R., Günthner, W. A., Klinker, G., Hamacher, D., Schega, L., Böckelmann, I., Doil, F., and Tümler, J. (2009). Pick-by-vision: A first stress test. In Klinker, G., Saito, H., and Höllerer, T., editors, *ISMAR*, pages 115–124. IEEE Computer Society.

[51] Snyder, J. P. (1987). Map projections – a working manual. Technical Report 1395, U. S. Geological Survey.

[52] Spohrer, J. C. (1999). Information in places. *IBM Systems Journal*, 38(4):602–628.

[53] Sutherland, I. E. (1968). A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), pages 757–764, New York, NY, USA. ACM.

[54] Tatzgern, M., Kalkofen, D., and Schmalstieg, D. (2011). Multi-perspective compact explosion diagrams. *Computers & Graphics*, 35(1):135–147.

[55] Tatzgern, M., Kalkofen, D., and Schmalstieg, D. (2013). Dynamic compact visualizations for augmented reality. In Coquillart, S., Jr., J. J. L., and Schmalstieg, D., editors, *VR*, pages 3–6. IEEE.

[56] Ward, M. O. (2002). A taxonomy of glyph placement strategies for multidimensional data visualization. *Information Visualization*, 1(3-4):194–210.

[57] Wikitude GmbH (2014). Wikitude (world browser). Available online at `http://www.wikitude.com/`, last visited 15.5.2014.

[58] Yang, Y., Gong, Z., and U, L. H. (2011). Identifying points of interest by self-tuning clustering. In Ma, W.-Y., Nie, J.-Y., Baeza-Yates, R. A., Chua, T.-S., and Croft, W. B., editors, *SIGIR*, pages 883–892. ACM.