FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Context-Responsive Labeling in Augmented Reality

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Thomas Köppel, BSc
Registration Number 01327052

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Hsiang-Yun Wu, PhD

Vienna, 18th November, 2020

Thomas Köppel

Eduard Gröller

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Kontextabhängige Beschriftung für die erweiterte Realität

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Thomas Köppel, BSc
Matrikelnummer 01327052

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Hsiang-Yun Wu, PhD

Wien, 18. November 2020

Thomas Köppel

Eduard Gröller

Erklärung zur Verfassung der Arbeit

Thomas Köppel, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 18. November 2020

Thomas Köppel

Acknowledgements

First of all, I would like to thank Hsiang-Yun Wu for her ongoing support. Her feedback and the resulting continuous improvements brought the project presented in this thesis to the current state. She gave me the freedom to implement my ideas while always supporting me. No matter which questions I asked, she immediately had answers and ideas. Thanks to her I could improve my scientific working skills. I enjoyed working on this project under the great guidance of Hsiang-Yun Wu.

Furthermore, I am very grateful for the detailed feedback from Professor Eduard Gröller whose in-depth comments pushed the thesis to a new level.

Last but not least I would like to thank my family, especially my mother and grandmother, for always supporting and encouraging me.

Abstract

Route planning is a common task that often requires additional information on Points-of-Interest (POIs). Augmented Reality (AR) enables mobile users to explore text labels and provides a composite view associated with additional information in a real-world environment. Displaying all labels for Points-of-Interest on a mobile device will lead to unwanted overlaps, and thus a context-responsive strategy to properly arrange labels is expected. This framework should consider removing overlaps, the correct Level-of-Detail to be presented, and also label coherence. This is necessary as the viewing angle in an AR system may change frequently due to users' behaviors. The consistency of labels plays an essential role in retaining user experience and knowledge, as well as avoiding motion sickness. In this thesis, we aim to develop an approach that systematically manages label visibility and Levels-of-Detail, as well as eliminates unexpected incoherent label movement. To achieve this, we introduce three label management strategies, including (1) *Occlusion Management*, (2) *Level-of-Detail Management*, and (3) *Coherence Management*. A greedy approach is developed for fast occlusion handling. A Level-of-Detail scheme is adopted to arrange various types of labels in AR. A 3D scene manipulation is built to simultaneously suppress the incoherent behaviors induced by the changes of viewing angles. Finally, we present our approach's feasibility and applicability by demonstrating one synthetic and two real-world scenarios, followed by a qualitative user study.

Kurzfassung

Routenplanung ist eine häufig gestellte Aufgabe. Dafür sind nähere Informationen zu den Zielpunkten erforderlich. In der erweiterten Realität (oder Augmented Reality, AR) können diese Informationen mittels Beschriftungen für den Benutzer in einer realen Umgebung angezeigt werden. Die Darstellung mehrerer Beschriftungen auf einem mobilen Gerät kann jedoch zu unerwünschten Verdeckungen führen. Hierbei sind Strategien erforderlich, welche auftretende Verdeckungsprobleme der Beschriftungen lösen, den angezeigten Detailgrad bestimmen und notwendige Aktualisierungen kohärent umsetzen, wenn sich der Benutzer bewegt oder den Blickwinkel ändert. Kohärente Anpassungen der Beschriftungspositionen spielen eine essenzielle Rolle, um die Wahrnehmung und den Entscheidungsprozess der Benutzer zu unterstützen. Unser Ziel ist die Entwicklung einer Strategie, welche die Sichtbarkeit der Beschriftungen und den angezeigten Detailgrad bestimmt sowie unerwartete Bewegungen der Beschriftungen minimiert. Um dies zu erreichen, führen wir drei Ansätze ein, darunter (1) *Sichtbarkeitsmanagement*, (2) *Detailgradmanagement* und (3) *Kohärenzmanagement*. Die schnelle Behandlung von Beschriftungsverdeckungen wird mittels eines Greedy-Algorithmus umgesetzt. Verschiedene Detailgrade für Beschriftungen werden in AR angeordnet. Eine 3D Szenenmanipulation wird durchgeführt, um inkohärente Beschriftungsbewegungen zu unterdrücken, die durch Änderungen des Blickwinkels des Benutzers entstehen. Wir zeigen die Anwendbarkeit unserer Strategie anhand eines künstlich generierten Szenarios und zweier realer Szenarien, gefolgt von einer qualitativen Nutzerstudie.

Contents

CHAPTER 1

Introduction

We schedule and make order decisions irregularly in our everyday life. For example, we visit offices, go to restaurants, or see doctors, in order to accomplish necessary tasks. In some cases, such as visiting medical doctors or popular restaurants, one has to wait in a queue until being able to continue the task. This is unfortunately time-inefficient and most people would like to avoid it especially in a pandemic situation (e.g., COVID-19 pandemic).

Normally, if a person needs to decide the next place to visit, he or she can extract knowledge about these targets of interest. Then a decision is made based on the corresponding experience or referring locations using a map. 2D maps are one of the most popular methods that describe the geospatial information of the objects, in order to give an overview of the relative positions of objects around a certain area. However, while using a 2D map for navigation, users need to remap or translate the objects on the map to the real environment, to be able to understand the relationships and distances to these objects [GM19]. This inevitably complicates the cognition process. It is also the reason why some people cannot quickly identify themselves on the 2D map or find the correct direction or orientation immediately.

Augmented Reality (AR) has been proposed to overlay information directly to the real-world environment with a lower complexity by instructing users in a more straightforward way. Based on these observations, we hypothesize that using AR for guiding users in exploring the real environment can be more effective in comparison to a 2D representation.

In AR, Points-of-Interest (POIs) are often associated with text labels [TKGS14, JZWG18, LNSG14] in order to present additional information (e.g., name, category, etc.). For example, an Augmented Reality Browser (ARB) facilitates us to embed and show relevant data in a real-world environment. Technically, POIs are registered at certain geographical positions via GPS coordinates. Based on the current position and the viewing angle of the device, the POIs are annotated and the corresponding labels are then projected

to the screen of the user's device. Naive labeling strategies may lead to occlusion problems between objects, especially in an environment with a dense arrangement of POIs. Additionally, properly selecting the right level of a label to present information can help to avoid overcrowded conditions. Moreover, retaining the consistency between key frames also enables us to maintain a good user experience and to avoid motion sickness.

Based on the aforementioned findings, we summarize that a good AR labeling framework should address:

(P1) The occlusion of densely placed labels in AR space. Occlusion removal has been considered as a primary design criterion in graph drawing and map production. It reflects user preference and also allows the visualization to present information explicitly [WTH+13].

(P2) Information provided by plain text labels is limited. As summarized by Langlotz et al. [LNSG14], labels in AR are often in plain text form rather than other richer content representation, such as figures or hybrids of texts and figures.

(P3) Label incoherence due to the move of mobile devices. During the interaction with the environment in AR, the user may frequently change positions or viewing angles. This leads to unwanted flickering that impacts the information coherence [JZWG18].

In this thesis, we develop a context-responsive framework to optimize label placement in AR. The approach contains three major strategy components: (1) *Occlusion Management*, (2) *Level-of-Detail Management*, and (3) *Coherence Management*, which are essential for the approach to be context-responsive. The *Occlusion Management* strategy eliminates overlapping labels by adjusting the positions of occluded labels with a greedy approach to achieve a fast performance. Then, a Levels-of-Detail scheme is introduced to select the right level in hierarchy and present it based on how densely placed the labels are in the view volume of the user. We construct a 3D scene to manipulate and control the movement of labels to better assist the user experience.

We introduce a novel approach to manage label placement tailored to AR. It enables an interactive environment allowing continuous changes of device positions and orientations. A survey by Preim et al. [PS18] concluded that existing labeling techniques often untangle overlapping labels once the camera stops moving or the camera position is assumed to be fixed to begin with. The approaches often project labels to a 2D plane to calculate the occlusions and perform occlusion removal algorithms. However, object movement in 3D is not obvious in the 2D projections of a 3D scene, which leads to temporal inconsistencies that could be harmful to the label readability [TKGS14].

Čmolík et al. [CPWN20] also demonstrated the difficulty of retaining label coherence in an interactive setting, if the label management is in 2D. We treat labels as objects in a 3D scene and perform the management strategies.

In summary, our main technical contributions are:

- A fast label occlusion removal technique for mobile devices.
- A clutter-aware Levels-of-Detail management.
- A 3D object arrangement that retains label coherence.
- An AR system on mobile devices to demonstrate the applicability of the label placement.

In this thesis, we present related work from map visualization, route planning, and labeling in AR (Chapter 2) to demonstrate the contribution. Then, we introduce the methodological principles of the system (Chapter 3). We show the applicability of our technique by investigating a *Tokyo Disneyland Dataset*, a *Local Shops Dataset*, and a *Synthetic Dataset* (Chapter 4) and present results for the datasets in Chapter 5. Chapter 6 highlights implementation details while Chapter 7 explains the evaluation. At the end of the thesis, we describe the limitations (Chapter 8), and we conclude the thesis and explain possible future improvements (Chapter 9).

CHAPTER 2

Related Work

Augmented Reality (AR) is becoming more and more popular as the computing power of mobile devices increases. One major benefit of AR is the possibility of showing information directly in the user's domain by annotating real-world objects. Our project considers elements and findings from various research topics. This chapter will introduce related work in the fields below and state the differences and the contribution of our approach. The main research areas are summarized as:

- Map visualization
- Navigation and Route Planning in Amusement Parks
- Labeling

The next sections will present related work in each of these areas and explain the connection to and contributions of our project. The end of this section will provide a summary of differences in comparison to the state-of-the-art.

2.1 Map visualization

Map visualization puts data into a geographical context. Different encodings can be employed to add data to maps: Scalar values can be included via color coding, textual information can be integrated via labels, or images can be used to encode information about a region or location in 2D maps. Markers can be used to highlight the location of a Point-of-Interest (POI) on maps, e.g., via geometrical shapes.

Related work in the area of map visualization focuses on encoding and providing additional information in maps depending on the tasks. Elements may get added to 2D maps in many cases to show supplementary information and, e.g., help with navigation, POI

detection, or POI selection. POIs may be different depending on the scenario and the audience. For tourists, POIs can be sights, restaurants, or hotels, etc.

The first two examples listed in this section are aimed at helping tourists to navigate in a foreign city. Grabler et al. [GASP08] dynamically generate tourist maps and add stylized buildings to help tourists recognize buildings and navigate through the city. Depending on the scenario, automatically generated maps show restaurants, buildings, or other POIs. The glyph based visualization of these buildings seen in Figure 2.1 focuses on the important information for the user - the tourist in their scenario. The stylized buildings shall help the user to spot POIs and ease navigation in the visited city.

Figure 2.1: Results of the automated tourist map by Grabler et al. [GASP08, p. 100:11]. The result shows stylized buildings to highlight POIs and help tourists navigate.

Birsak et al. [BMWW14] introduce automatically created tourist brochures. Via optimizing energy functions, detailed information about POIs is positioned at the border or center of the map as close to the respective marker as possible. In this project, markers are red or blue pins, which encode the locations of sights on the map (see Figure 2.2). The 2D map shows each marker with a number. The detailed information contains this number and provides information like the address, an iconic image, or a rating. This information was gathered from a free web database. Furthermore, they include a graph simplification to ease navigation for the user. Figure 2.2 presents the resulting brochure for Seattle with a simplified graph containing the markers. The areas at the border or in the center provide more detailed information about the respective POIs, like the address

or an iconic image.

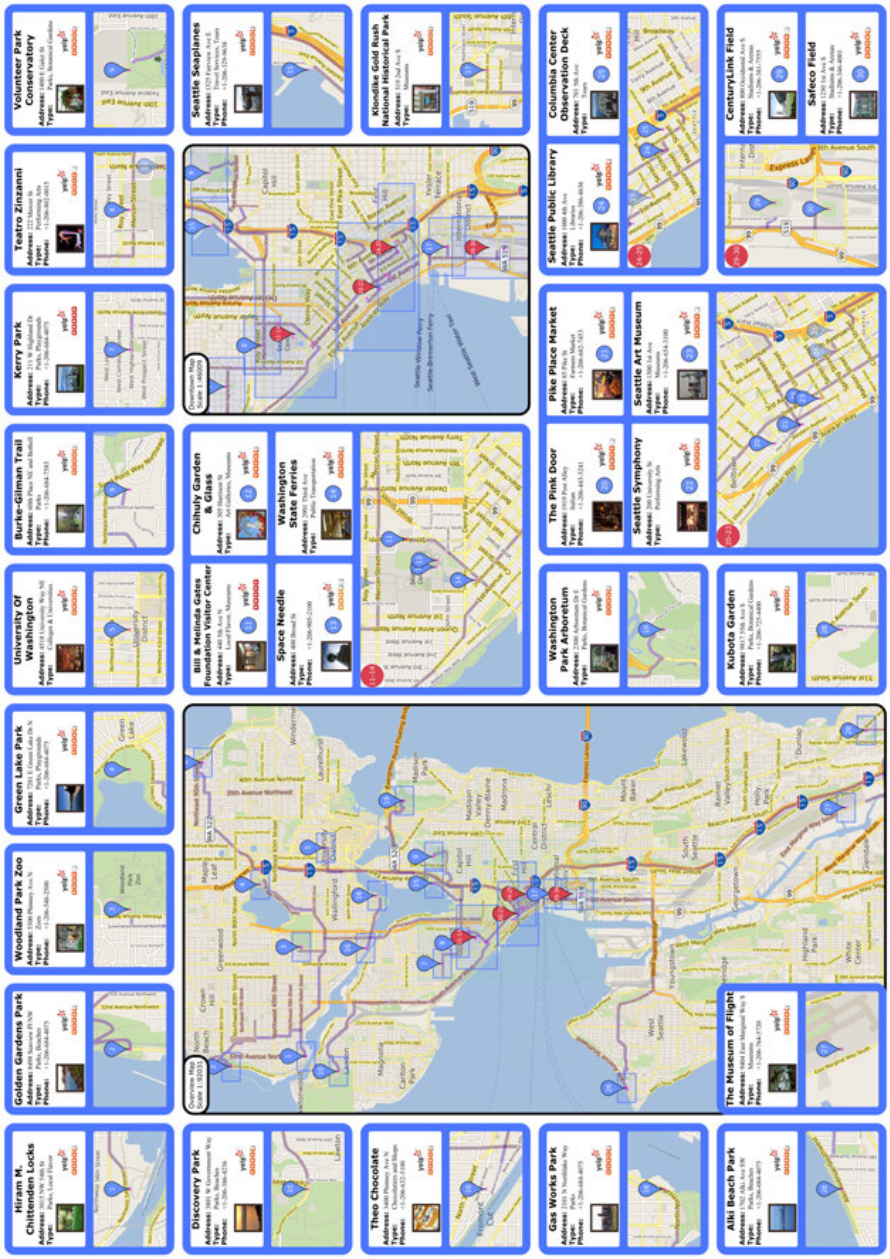


Figure 2.2: Explanatory tourist brochure by Birsak et al. [BMWW14, p. 10] of Seattle showing the simplified 2D map with markers and additional information about the POIs located at the position of the markers at the border or center of the map.

Claudio et al. [CY14] focus on tourist trip planning and overcome the mind gap between a 2D map of a city and a metro map. They use the metro map as the common possibility

to travel through cities and include POIs into the metro map. Iconic images of POIs are used to annotate sights. Based on ranking the popularity of each sight, the respective POIs are displayed. By zooming in, more space is available and more POIs can be shown on the map. Lines connect an iconic image with a metro station. Based on the zoom level, occlusions of iconic images and the metro graph can appear (see Figure 2.3). In this view of Lisbon, occlusions can be detected. In this case, the metro graph is plotted on top of the images and transparency is used to make the information behind visible. The iconic images do not occlude each other.

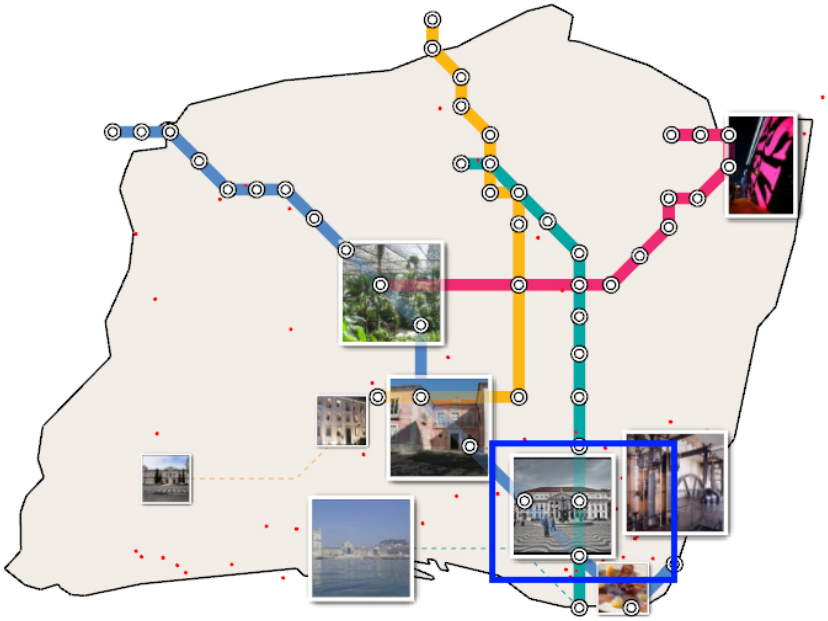


Figure 2.3: Explanatory metro map by Claudio et al. [CY14, p. 271] of Lisbon showing the simplified metro graph. The map includes iconic images of sights that are linked to the stations.

In contrast to the other projects, Ishida et al. [II19] do not focus on tourist navigation. Their goal is to overlay historical information onto old maps and visualize POIs. They scanned old maps, investigated historical information, and show this information in the visualization tool. The result can be seen in Figure 2.4. The red markers highlight historical background information at the respective position on the old map. By clicking onto the red markers, an iconic image and the name of the POI are presented. The color of the marker does not encode information. Detailed information is only displayed when interacting with the system and clicking onto the markers.

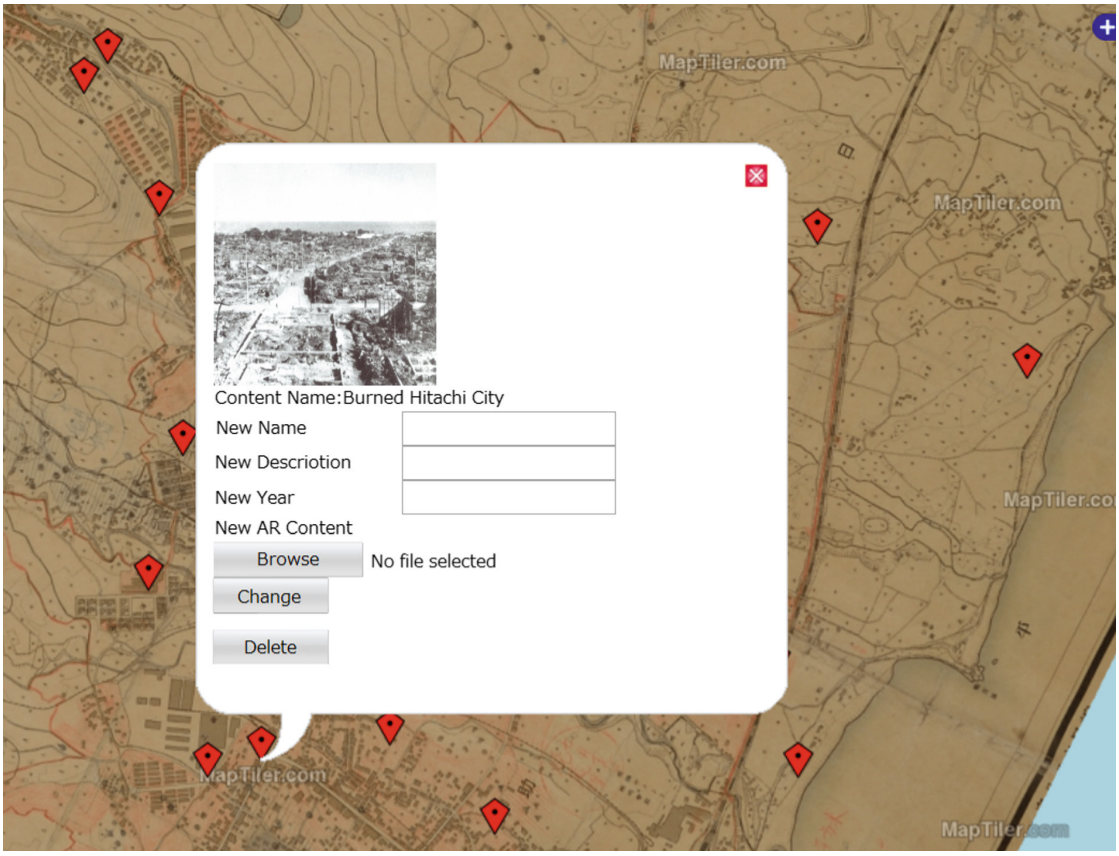


Figure 2.4: Explanatory historical map by Ishida et al. [II19, p. 317]. The red markers highlight POIs. By clicking onto the red markers, detailed information about the POIs is shown.

Similar to Grabler et al. [GASP08], Birsak et al. [BMWW14], or Claudio et al. [CY14], we focus on the important information for the user and include a stylized version of it. Our label encoding is explained in detail in Chapter 3. However, our approach works in 3D in AR and shows iconic images (e.g., photos of attractions in the *Tokyo Disneyland Dataset*). Furthermore, we encode additional information (e.g., the waiting times of attractions in the *Tokyo Disneyland Dataset*) and the visualization is based on the current location of the user. The user can explore the environment through a mobile device and the information is directly embedded in the real-world environment of the user.

2.2 Navigation and Route Planning in Amusement Parks

Navigation in AR benefits from the direct embedding of the navigational information on the user's device showing the real world. The user can look at a mobile device and sees the important information supporting the guidance to a POI. In contrast to maps, the

user does not need to translate navigation from the map to the real world, which eases the navigation process [BFH01, JZWG18, KG19]. This subsection provides an overview of navigation projects and will specifically mention route planning for amusement parks.

2.2.1 Navigation

To support users in getting to the destination, navigation projects provide tools to ease this process. The route and the current position of the user might be displayed in a 2D map or navigational elements, like arrows, are displayed in AR. Presenting navigational information directly on the user's device showing the real world eases navigation because users do not need to translate navigation from a map to the real world. The related work described in this subsection highlights different techniques of embedding navigational information into AR to support users in getting to the POI.

Schneider et al. [SBN+19] conducted a field study and focus on car navigation. They include visual cues into the windshields of cars to help during navigation. Figure 2.5 shows an example. The white arrows projected onto the windshield of the car in this example help drivers with navigation. According to the evaluation by Schneider et al. [SBN+19], the design is visually pleasing. Furthermore, users do not have to read a separate device, which leads to a higher time of looking at the street.

Figure 2.5: Explanatory result by Schneider et al. [SBN+19, p. 360]. The white arrows help the drivers to navigate towards the destination. The arrows are projected onto the windshield of a car.

Guarese et al. [GM19] show a similar result and integrate visual cues using the HoloLens by Mircosoft. In contrast to previous work, visual cues are holograms that are positioned

on the terrain. During the evaluation, Guarese et al. [GM19] compare the project for the HoloLens to classical handheld devices. Guarese et al. [GM19] conclude that it is easier for users to navigate with their tool.

Figure 2.6: Explanatory result by Knuttson et al. [KG19, p. 11]. A blue path and a green arrow indicate the route to support users during navigation tasks.

Knuttson et al. [KG19] investigate using AR for navigation. They compare navigating with 2D maps to navigation in AR. By considering GPS data, they include paths and arrows into the AR view to guide users. Figure 2.6 presents the work. A blue path and a green arrow are included into AR to help users navigate. According to the user study, participants stated that there was no need of translating a 2D map to the real world, which made navigation easier. Furthermore, the directions were clear to them and no one got lost. In contrast to the positive aspects, they stated that the overall overview of the path was less clear as the participants did not see the whole route compared to 2D maps. All in all, Knuttson et al. [KG19] showed the suitability of using AR for navigation tasks.

In contrast to the navigation-related work presented in this section, our approach does not indicate paths to specific POIs. Our main goal is to present an overview of several POIs to support the decision-making process of users. The directions to navigate towards the POIs are encoded with the position of the labels in AR. Similar to Guarese et al. [GM19], we position elements in AR according to the location of the user. The navigation, highlighting a route towards the POI, would be interesting to tackle in future work.

2.2.2 Route Planning in Amusement Parks

To save time, routes need to be planned in advance. Factors, like the current location of a user or the distance between POIs, influence the route. Depending on the scenario, different parameters need to be considered to calculate a route, which represents a challenging task. For amusement parks, the position of the visitor, the excitement of an attraction, or the waiting time need to be accounted for to calculate a route with minimal waiting times or minimal walking distances. A special possibility provided in Disneylands is called *FASTPASS*. It allows visitors of the amusement park to have lower waiting times at specific attractions that support this system. At these attractions, it is possible to scan the ticket. The system shows a time slot when the visitor can return to this attraction to have a lower waiting time. The *FASTPASS* queue is a separate and shorter queue compared to the standard queue of an attraction. While a ticket is reserved for a specific *FASTPASS* attraction it is not possible to rescan the ticket at another attraction. For example, the visitor arrives at 11:00 am at the Tokyo Disneyland. He or she wants to ride a popular attraction, in this case the *Big Thunder Mountain* where the waiting time can exceed two hours. The visitor scans the ticket at a specific booth at the *Big Thunder Mountain* and the system tells the visitor to return at 2:00 pm. From 11:00 am to 2:00 pm the visitor cannot register the ticket for another *FASTPASS* attraction. When the visitor returns to *Big Thunder Mountain* at 2:00 pm, he or she can enter via a separate queue with a lower waiting time. After riding *Big Thunder Mountain*, the user can register the ticket at another *FASTPASS* attraction and the process works the same as described above for the *Big Thunder Mountain*. *FASTPASS* is used by Disneyland routing optimization algorithms. Other factors like excitement and relaxation (terms used by Ohwada et al. [OOK13]) describe the type of an attraction. Exciting attractions are, e.g., roller coasters, while relaxing attractions are, e.g., train rides through the park. Routes may be balanced to avoid several exciting attractions in a

row. Different factors that need to be accounted for, like the waiting time, the attraction type, the weather, or the position of the user and the attractions, indicate the complexity of route optimizations in amusement parks. The next paragraphs will introduce related work that tried to solve the route planning for amusement parks.

Ohwada et al. [OOK13] describe a route planning system specifically tailored to amusement parks. During the route computation, Ohwada et al. [OOK13] consider important information, like excitement and relaxation, waiting times, distances between attractions, and *FASTPASS*. They extend the Dijkstra algorithm to account for these parameters. Ohwada et al. [OOK13] provide a whole system for amusement parks including three different routing algorithms. According to their work, it is impossible to ride each attraction of the amusement park in one day. Therefore, the problem represents a special case of the traveling salesman problem, which they solve via branch and bound. The second algorithm considers *FASTPASS*, which represents a complex constraint according to Ohwada et al. [OOK13]. The third algorithm considers excitement and relaxation to compute balanced paths avoiding several exciting attractions in a row. Ohwada et al. [OOK13] highlight that the consideration of *FASTPASS* produces the lowest overall waiting times in the project. Figure 2.7 illustrates a route trough the Tokyo Disneyland. The attractions were pre-selected by the user. The system calculates an appropriate path and the time for the whole route. The system recommends using *FASTPASS* for the attraction *Big Thunder Mountain*.

Sakayori et al. [ST15] focus on creating a routing algorithm for amusement parks. Furthermore, they highlight the importance of considering factors like excitement, relaxation, and *FASTPASS*. Liu et al. [LGZ+19] benefit from a high precision positioning system called *Xihe positioning technology* and use the real-time position data by *Xihe* to compute routes through the amusement park. Liu et al. [LGZ+19] compute, whether the computed route covers the whole amusement park. High coverage indicates better routes according to Liu et al. [LGZ+19]. Furthermore, they include fatigue of visitors into the route planning. The evaluation done by Liu et al. [LGZ+19] shows promising results for using the tool in the Shanghai Disneyland.

In contrast to the related work described in this subsection, our tool does not compute an optimal route. We focus on providing an exploration tool allowing the user to decide on the route according to individual preferences. Furthermore, we encode the updated current waiting times during an entire day, allowing the user a higher flexibility to adapt to the current situation rather than following a pre-computed route.

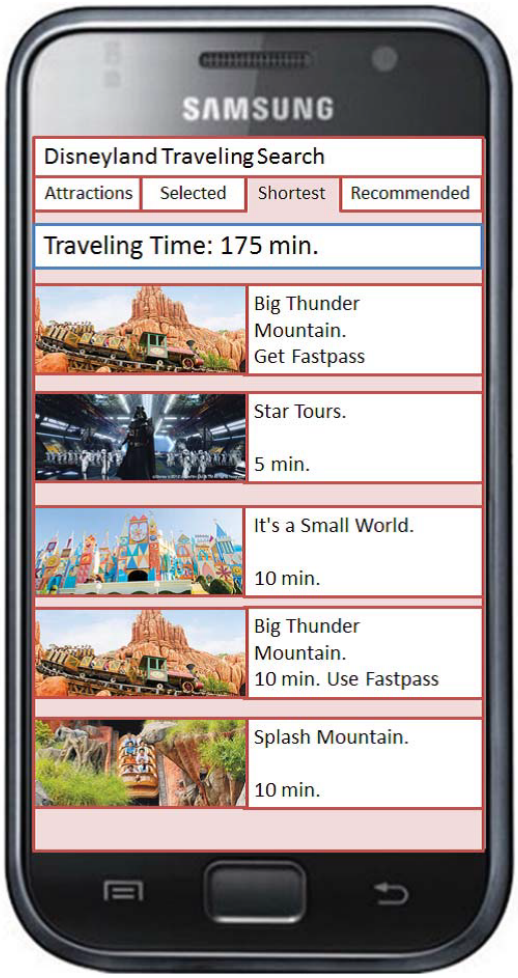


Figure 2.7: Route by Ohwada et al. [OOK13, p. 425]. The attractions were selected by the user and the system recommends an appropriate path considering *FASTPASS*, enjoyment, waiting time, etc.

2.3 Labeling

Labeling is used to annotate objects. Labels can either be placed directly in the annotated region or they can be placed outside. Figure 2.8 illustrates a labeled human heart, while both internal labels, which are placed inside of the vascular structures, and external labels, which are located on the outside of the organ, are visible. Furthermore, the internal labels in this example are deformed based on the structure beneath. The external labels are not deformed. Nowadays, labeling is used in computer programs in 2D or 3D visualizations, Virtual Reality (VR), or Augmented Reality (AR), for example.

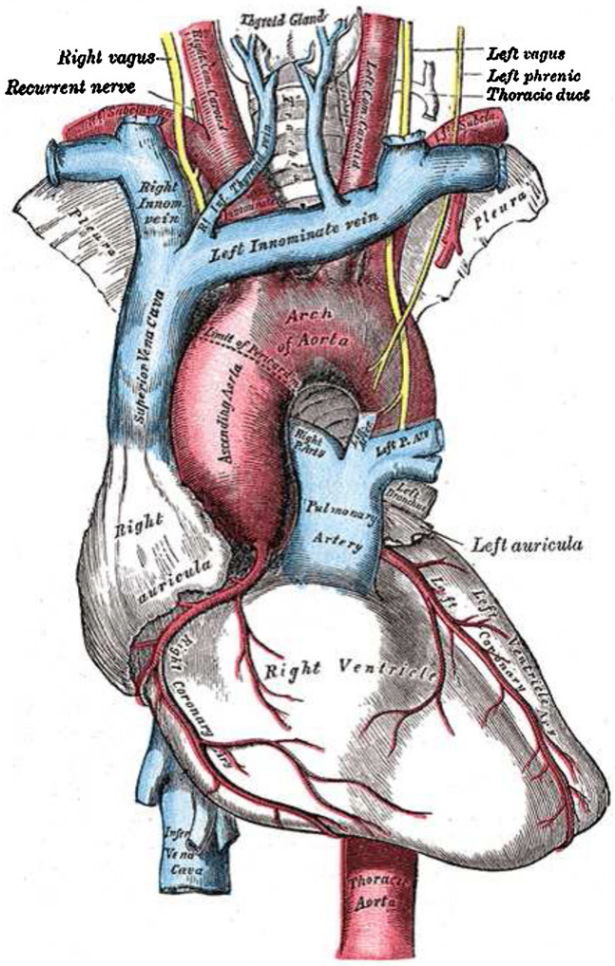


Figure 2.8: An illustration of a heart with different annotated regions by Gray et al. [Gra78]

Augmented Reality offers the possibility of showing additional information to the camera image data. Usually, real-world objects are labeled with text [LNSG14]. This section introduces different possibilities of labeling POIs.

Occlusions may occur if many labels should currently be visible and are close to each other. Occlusions make labels difficult or impossible to read. To ensure the readability of the labels, occlusions need to be resolved. Depending on the scenario, different properties are minimized during the occlusion handling to ensure the quality of the applied technique, like the distance to the labeled object.

Figure 2.9: A street with labeled buildings by Jia et al. [JZWG18, p. 1657]. The labeling is occlusion-free. Leaders are lines that connect labels with the annotated real-world entities.

Jia et al. [JZWG18] propose an occlusion-free labeling method of real-world objects, which can possibly be used in AR. *Leaders* are lines that connect labels with the annotated objects. The calculations are done in 2D. For each label, a cost function is evaluated to calculate occlusion-free positions. The lower the costs are for a label, the better is the calculated position. Jia et al. [JZWG18] analyzed the image and considered image edges for the label placement. This edge information is used in their cost function as well as the overlapping area of occluding labels, the length of the leaders, and the number of crossings between the leaders with the calculated image edges. The approach is presented using videos and images. Only a desktop prototype is implemented due to the high computational complexity of the approach [JZWG18]. Figure 2.9 shows a result of the approach by including text labels to the image of a strip mall. The labels are placed in regions containing few image edges.

Grasset et al. [GLK+12] present a similar approach and provide occlusion-free labeling for future use in AR. They include image edge information and salience maps during the minimization calculations to position the labels. Leaders connect the real-world objects with the text labels. The calculations are done in 2D. Figure 2.10 compares the initial state of the labeling on the left and the occlusion-free result on the right that considers image edge information and salience maps by Grasset et al. [GLK+12].

Figure 2.10: Result of previous work by Grasset et al. [GLK+12, p. 177]. The figure includes the labeled scene before and after the occlusion handling.

Related work presented in the survey by Preim et al. [PS18] annotates 3D objects and calculates the label positions in 2D. 2D convex hulls that surround the anatomical structures are used, e.g., to calculate the label positions. In contrast to the work presented in the survey by Preim et al. [PS18], Tatzgern et al. [TKGS14] consider 3D information while positioning the labels, which leads to a more stable positioning if the viewing angle of the device changes. Tatzgern et al. [TKGS14] estimate the 3D center position of the object and the position of the annotated part of the object to calculate 3D poles that lead from the center of the object outwards to the direction of the annotated part of the object. Labels are shifted along these 3D poles. The final label placement is calculated in 2D planes considering the pole information to avoid occlusion of leaders and labels. This technique by Tatzgern et al. [TKGS14] provides a solution for annotating an object in front of the camera. According to the report, they believe that the approach has great potential for being used in AR and VR applications [TKGS14]. Figure 2.11 presents a labeled head by Tatzgern et al. [TKGS14]. The text labels do not occlude each other and the leaders do not cross each other.

Previous work by Tatzgern et al. [TKGS14, TKS13], Grasset et al. [GLK+12], or Jia et al. [JZWG18] primarily focus on the placement of rectangular text labels and leaders that connect each label with the corresponding real-world object. The information provided by a plain text label is limited (see Figure 2.12 by Jia et al. [JZWG18]). The text labels display the names of the respective buildings. The names of the buildings and the relative locations to the viewer can be seen. More information cannot be gathered from the encoding. We provide further information in addition to text also including an iconic image (a photo of the POI), a color-coded rectangle encoding a scalar value (e.g., the waiting time in the *Tokyo Disneyland Dataset*), and icons indicating type in the POI (e.g., the attraction type - *thrilling*, *adventure*, or *children* - in the *Tokyo Disneyland Dataset*). Furthermore, our labels support various sizes because of the introduced LODs for labels in AR, which makes the occlusion handling even more challenging as the sizes

of the labels change dynamically during the program execution.

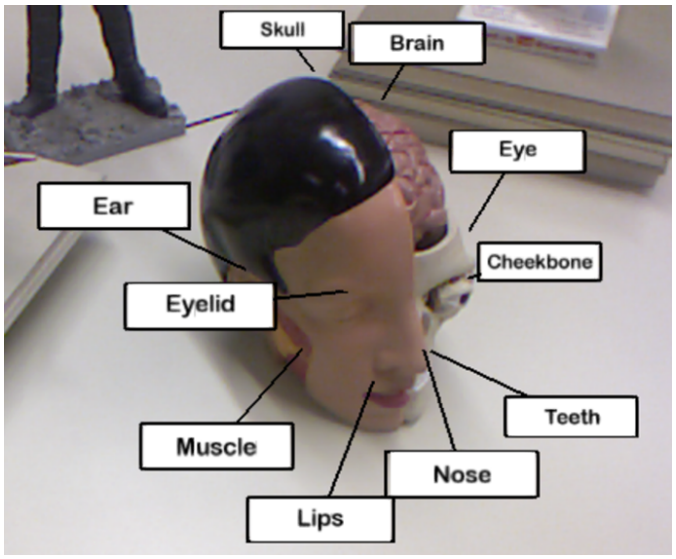


Figure 2.11: Result by Tatzgern et al. [TKGS14, p. 1]. The labels are positioned in an occlusion-free way.

Figure 2.12: A street with labeled buildings by Jia et al. [JZWG18, p. 1658]. Labels of distant POIs may become occluded by close ones. The information (text in this case) cannot be retrieved from the occluded labels.

Previous work by Tatzgern et al. [TKS13], Jia et al. [JZWG18], Grasset et al. [GLK+12],

Shibata et al. [SNS+08], Bekos et al. [BNN19], and van Garderen et al. [vGPNB17] either resolve occlusions for labels in 2D or project the labels from the 3D space to a 2D plane and resolve occlusions in 2D.

2.4 Differences to our Work

To the best of our knowledge, we introduce a novel tool to resolve occlusions of labels tailored to AR, which represents an interactive environment as position and viewing angle of the device change frequently. Previous labeling work, as presented in the survey by Preim et al. [PS18], e.g., often resolves occlusions if the camera movement stops or they assume a fixed camera position to begin with. Therefore, these techniques are not suitable for AR where the device is always moving. Furthermore, as stated by Tatzgern et al. [TKGS14], 3D movement is not obvious in 2D projections of 3D scenes, leading to temporal inconsistencies caused by moving labels. We treat labels as 3D objects in the scene. In doing so, we overcome inconsistently moving labels if the viewing angle of the AR device changes. In contrast to related work by Tatzgern et al. [TKGS14] or Preim et al. [PS18], we cannot simply project every label to a fixed 2D plane as they are located around the user. To provide stable label positions even if the user rotates the mobile device, we resolve occlusions among all the labels around the user. Our approach aligns labels towards the user in such a way that each of them is readable. The methodological basics behind the occlusion avoiding approach are explained in Chapter 3.

In addition, we provide more information than plain text by including an iconic image (e.g., a photo) of the POI as proposed by some 2D map projects described above. We also include a colored rectangle showing a scalar value of each POI (the waiting time of the attraction in the *Tokyo Disneyland Dataset*) and we include icons indicating the type of the POI (the attraction type in the *Tokyo Disneyland Dataset* or the shop type in the *Local Shops Dataset*). This encoding provides a possible solution for the problem described by Langlotz et al. [LNSG14] that labels usually display only textual information.

To reduce visual clutter, we are inspired by the LOD concept for labels [MHSG02]. We overcome the limited available mobile phone screen by providing a larger amount of information for more important POIs and a smaller amount for less important POIs. The LOD of a label can change during the exploration by the user depending on the closeness of the user to the POI and on how densely placed labels are in the view volume. These changes increase the complexity of the occlusion avoiding strategy as different label sizes need to be taken into account. In contrast to LODs for labels presented by Matkovic et al. [MHSG02], we adapt the concept by taking two parameters into account that change continuously over time, namely the closeness of the user to the annotated POIs and the label density in the view volume. These changes need to be taken into account to avoid inconsistencies caused by frequent LOD updates. Moreover, to compensate for a large number of labels, we introduce super labels to AR. Super labels are used to aggregate individual labels to reduce visual clutter.

To avoid unwanted flickering, we incorporate smooth transitions for each movement and change. These include if positions of labels change to be occlusion-free during the interaction with the system, if LODs of labels change, or if labels are aggregated to super labels.

CHAPTER 3

Our Labeling Strategy

AR is often used to facilitate user navigation [BFH01]. The effort to identify objects in AR is lower [BFH01, JZWG18, GM19] because real-world objects can be directly annotated [BFH01, GLK+12] and AR navigation requires less user focus demand compared to 2D map techniques [GM19]. Our responsive framework is inspired by Hoffswell et al. [HLL20], who proposed a taxonomy for responsive visualization design, which is essential to present information based on the device context.

In principle, our design has three major components, including (1) *Occlusion Management*, (2) *Level-of-Detail Management*, and (3) *Coherence Management*, each of which aims to solve the problems **(P1-P3)**, respectively. We will first introduce our encoding of labels beyond plain text used in the system, followed by an overview of the approach.

3.1 The Label Encoding

Previous work by Tatzgern et al. [TKS13, TKGS14], Grasset et al. [GLK+12], and Jia et al. [JZWG18] primarily focus on the placement of rectangular text labels. The information provided by a plain text label is limited. Detailed explanations and examples can be found in Chapter 2.

The label encoding is chosen to relax the limitations in existing work [GLK+12, TKGS14, JZWG18], as we introduce additional types of labels than merely text labels as concluded by Langlotz et al. [LNSG14]. The labels depicted directly in the AR domain solve problem **(P2)**. We use color to encode scalar variables as concluded by Mackinlay [Mac86] and Mazza [Maz09] of each POI. In general, the user can choose a color scale according to their preference.

Figure 3.1: Our basic label encoding including a text tag, a color-coded rectangle, an icon, and an iconic image.

In our system, a label may consist of multiple of the following components:

- A text tag containing the name of the POI,
- An iconic image (photo) of the POI,
- An icon encoding the type of the POI, and
- A color-coded rectangle representing a scalar value of the POI.

In Figure 3.1, we show a label designed by presenting one of our datasets, the *Tokyo Disneyland Dataset*. POIs are attractions in this case. For the *Tokyo Disneyland Dataset*, attractions can be categorized into three types, *thrilling*, *adventure*, or *children*, which is illustrated in the type icon. Figure 3.1 provides an explanatory label annotating an attraction of the dataset. The text tag depicts the name of the attraction and the waiting time (e.g., *Big Thunder Mountain 100 min*). The iconic image shows a photo of the train of the attraction and the type icon indicates that it is a thrilling attraction. The colored rectangle (the red rectangular background of labels in Figure 3.1) encodes the waiting time.

We visualize ordinal data via color coding. Mackinlay [Mac86] investigated which kind of visualization technique is suitable for which kind of data. Figure 3.2 presents a comparison by Mackinlay [Mac86]. This figure describes the appropriateness of different visualizing features for quantitative, ordinal, and nominal data. We visualize waiting times, e.g., which are ordinal data values, and we further classify the waiting times into average, lower than average, and higher than average. The two most appropriate features for ordinal data are position and density as Mackinlay [Mac86] stated. In our tool, the position represents the geographical locations of the attractions encoded in the AR view. The density is dependent on the positioning of the labels. Many labels close to each other would represent a high density. Therefore, the position and the density cannot be used to encode waiting times. We use the third feature of appropriately visualizing parameters for ordinal data by Mackinlay [Mac86], the color saturation. We employ a continuous color bar ranging from blue (low waiting times), to yellow (average waiting times), to red (high waiting times) in the *Tokyo Disneyland Dataset*.

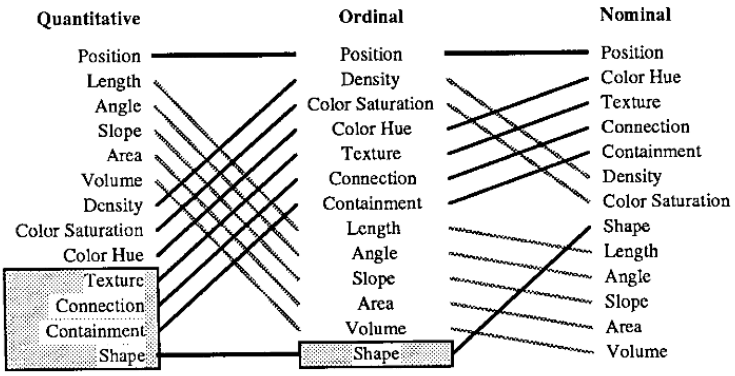


Figure 3.2: Mackinlay [Mac86] compare the appropriateness of different color encodings for quantitative, ordinal, and categorical, i.e., nominal data.

The selection of the color coding is further supported by Mazza [Maz09]. Figure 3.3 provides a comparison of the appropriateness of color for visualizations concerning quantitative, ordinal, or categorical data. According to his work, the intensity is a suitable parameter for visualizing ordinal data. Furthermore, we connect the selected

colors to a well-known metaphor - traffic lights. Many people should be familiar with traffic lights. The green color has been exchanged with blue in order to account for red-green color blindness. Men are more likely to suffer from color blindness [Ins19]. One in twelve men suffers from color blindness [Ins19]. The most common form is red-green color blindness [Ins19], therefore, we did not use a color-scheme that contains green and red. Based on the argumentation above, we selected a continuous color bar ranging from blue, to yellow, to red to encode the waiting time in the *Tokyo Disneyland Dataset*.

Figure 3.1 presents the above-described features of one POI, an attraction of the *Tokyo Disneyland Dataset*. The colored rectangle indicates that the current waiting time is around 100 minutes. The waiting time is an important decision-making factor when visiting an amusement park in order to ride many attractions within as little waiting time as possible. As explained before, our visualization technique shows more features than simply the name of the attraction, which is one of the contributions.

Attribute	Quantitative	Ordinal	Categorical
Color			
Hue	×	×	✓
Intensity	–	✓	×

Figure 3.3: Mazza [Maz09] compares different color encodings and highlights the appropriateness of each technique for quantitative, ordinal, and nominal/categorical data.

3.2 Overview of the Pipeline

Figure 3.4 gives an overview of our approach. We first position labels of POIs in AR and perform the proposed three management strategies. We compute and manipulate objects in the 3D scene using a Cartesian world coordinate system, where the xz-plane is parallel to the ground plane.

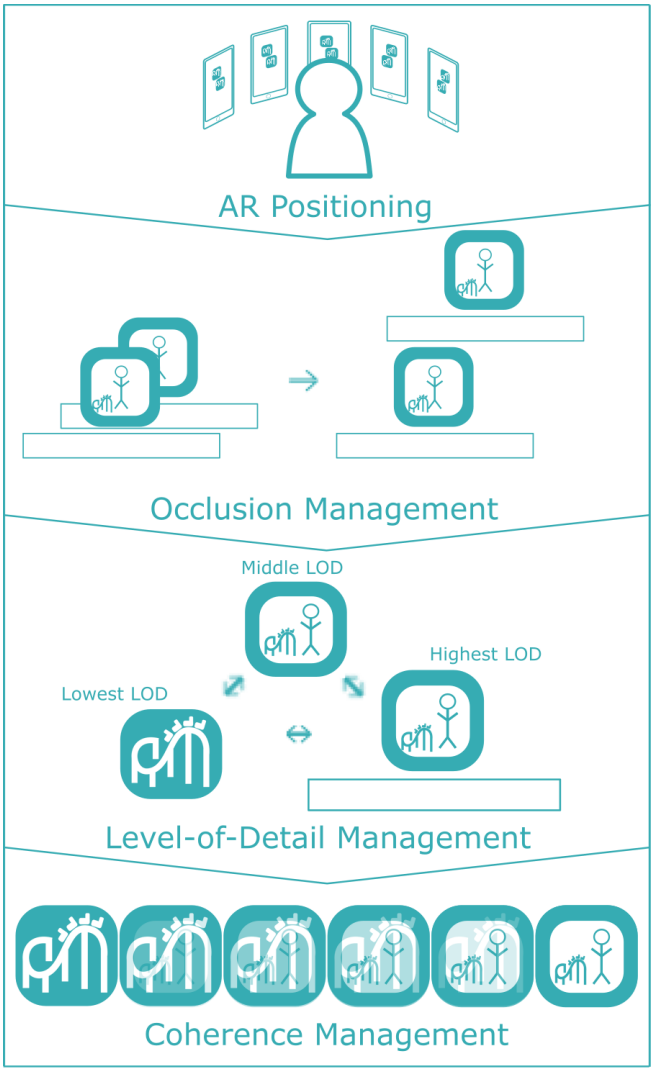


Figure 3.4: Our pipeline consists of the *AR Positioning* and our three management strategies, the *Occlusion Management*, the *Level-of-Detail Management*, and the *Coherence Management*.

The x-axis and z-axis define the ground plane and the y-axis is vertically upwards from the ground plane. The input to our system is a set of POIs $P = \{p_1, p_2, ..., p_n\}$ and a set

of labels $L = \{l_1, l_2, ..., l_n\}$, for example, manually selected by the users or downloaded from an online database. In the *AR Positioning* (Section 3.3), for each POI p_i, the corresponding label l_i is initially positioned perpendicularly to the ground plane in the world coordinate system. Currently, we assume that each POI p_i has one associated label l_i describing the attributes of the POI. We also assume that the (x, z)-coordinates of each annotated POI are more important than y-coordinate since the (x, z)-coordinates are essential to indicate relative positions of the POIs [BFH01, GM19].

Our *Occlusion Management* strategy (Section 3.4) tackles **(P1)** by resolving occlusions of labels considering the current position of the device. The labels are first sorted by distance to the device in a list S, from the nearest to the farthest positions. With this information, we resolve occlusions starting with the closest label using a greedy approach. We develop the greedy approach by arranging the lowest y-positions of the labels to be visible iteratively. This allows effective execution of the occlusion handling on mobile devices, where the computation powers are limited compared to desktop computers. The occlusion strategy provides a solution to inconsistently moving labels if the viewing angle of the AR device changes [TKGS14].

In the *Level-of-Detail Management* (Section 3.5), we introduce four distinct types of label encodings to represent three LODs of an individual label and one, super label, to indicate an aggregated group of labels for visual clutter reduction (Section 3.5). The *Level-of-Detail Management* depicts a different amount of information for each label. The LOD of a label l_i is selected according to the distance of the annotated POI to the device and the label density in the view volume. For convenience, we assume that close labels are more important to the user for the exploration process of POIs since it is natural to show larger objects when they are close by. For this reason, close labels in the system provide more detailed information compared to labels that are located in the distance. Super labels are labels that we use to represent a set of aggregated labels in order to reduce visual clutter of a mobile screen.

AR Positioning, *Occlusion Management*, and *Level-of-Detail Management* are smoothly updated in the *Coherence Management*. To avoid flickering that inevitably reduces coherency [JZWG18], the labels are not moved or changed immediately, but following a common animation strategy, by strategically updating changes over time (Section 3.6) to solve problem **(P3)**.

3.3 AR Positioning

One of our main goals is to provide a tool to allow an exploration of POIs for users respecting the position of the user and the positions of the POIs in an intuitive way. The system shall support the user to select a POI and help him or her to navigate towards the chosen destination by showing the label that represents the POI in AR according to the position of the user and the POI.

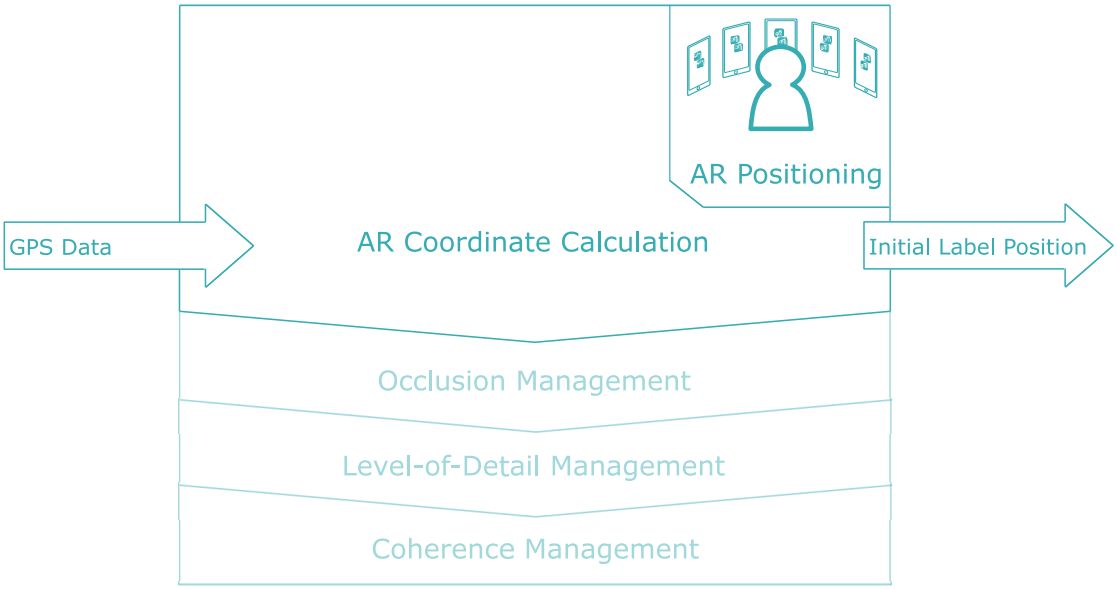


Figure 3.5: The input and output parameters of the *AR Positioning*.

In the *AR Positioning*, we need to map the geographical locations from the real world to our Cartesian AR world space. Real-world POIs can be mapped to the AR world space by considering the GPS position of the user's device, the GPS location of the POIs, and the compass orientation of the device. The labels are oriented at the user's position by aligning the normal vectors of the labels towards the AR device in the AR world space. Once this initial label positioning is done, a perspective camera projection from the AR world space to the screen space of the device is performed.

However, in principle, existing frameworks, like the AR + GPS Location SDK package [For20] or the Wikitude AR SDK package [Gmb20] can be used to map real-world objects to the AR world space. Unfortunately in our experiment, the techniques are not stable. The problem is caused by the inaccurate data of the GPS sensor [UB20] or the compass [Bow19, KGR20] of mobile devices. Moreover, the documentations also do not give an insight into how exactly this mapping is calculated. For testing and assessing the quality of the coherence strategies for the *Occlusion Management* and the *Level-of-Detail Management*, we eventually incorporated the predefined positioning of labels initially at $(x, 0, z)$-coordinates in the AR world space. This is because the existing libraries do not provide stable label positions, which would lead to a less coherent behavior that is not relying on the proposed *Coherence Management*.

3.4 Occlusion Management

This section introduces problems and challenges of labeling in AR and presents our approach, which resolves the occlusions of labels in AR.

3.4.1 Challenges when Solving Occlusions of Labels

Showing several labels at a time in AR may lead to closer labels occluding distant ones, especially, if the annotated POIs are close to each other or behind each other considering the current position and viewing angle of the user's device. Because of these occlusions, the information provided by the labels in the background can be difficult to perceive or even impossible, which indicates the need for occlusion avoiding strategies in AR.

Occlusions depend on the position of the user and the annotated POIs in AR space. Occlusions change when the position of the user changes while he or she is walking through the real world during, e.g., navigation. Due to this dynamic interaction with the environment, the system needs to be able to react to changes and provide an occlusion avoiding strategy that updates the scene regularly. Another problem occurs during navigation because of the limited computation power of mobile devices compared to laptops or desktop PCs. The system needs to be able to recalculate the occlusion handling fast enough to not hinder the exploration process for the user. It is known that point-feature labeling has been extensively investigated due to its NP-hardness when looking for an optimal solution even in 2D [CMS94]. In our setting, the condition of occlusions changes over time since the users move and their orientations also change, thus fast responsive management strategies are required to update the scene regularly.

Viewing angle and position changes of the user need to be accounted for to guarantee smooth state transitions and eliminate unwanted flickering. We calculate the entire occlusion handling in a 3D scene, overcoming the label positioning inconsistencies caused by viewing angle changes.

3.4.2 The Occlusion-Avoiding Approach

We calculate the entire occlusion handling in AR in 3D, overcoming the positioning inconsistencies caused by viewing angle changes of the user's device described in Chapter 2 to solve **(P1)**. We treat labels as objects in the 3D scene. The approach considers position and viewing angle changes of the user's device during exploration and navigation. The system recalculates the occlusion handling during the program execution, which is dependent on the position of the user. Smooth transitions are provided when positions of labels change to present the information in a coherent way to ease exploration and navigation for users. The algorithm requires two assumptions that need to be fulfilled. We assume that

- The x and z coordinate of labels (horizontal ground plane) are more important for the navigation process than the y coordinate (vertical axis), and that
- Objects closer to the viewer are either larger or as large as distant labels.

These two assumptions allow us to execute the *Occlusion Management* on mobile devices making it a suitable candidate for AR. The *Occlusion Management* consists of *Distance*

Sorting, *Occlusion Detection*, and *Shift Computation* (Figure 3.6).

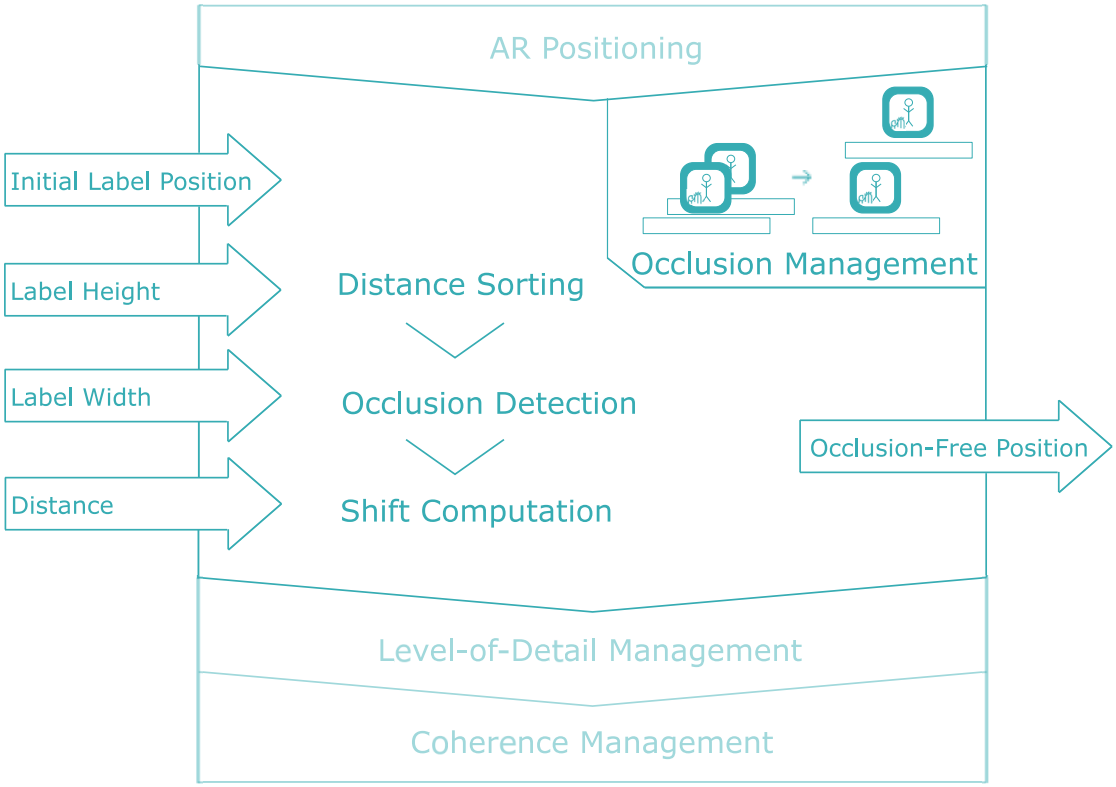


Figure 3.6: The *Occlusion Management*.

Distance Sorting

In the *AR Positioning*, the labels are placed based on their positions and the position of the user for exploration and navigation. This will lead to occlusions as described in Subsection 3.4.1. Before occlusions are resolved, we order the labels based on the distance to the user. Labels that are close to the user will be considered first and labels that are farther away will be considered afterward. Labels closer to the viewer are likely to be more important than distant labels during the exploration, e.g., if the user is interested in seeing detailed information about close POIs. The positions of close and more important labels are changed less compared to distant labels.

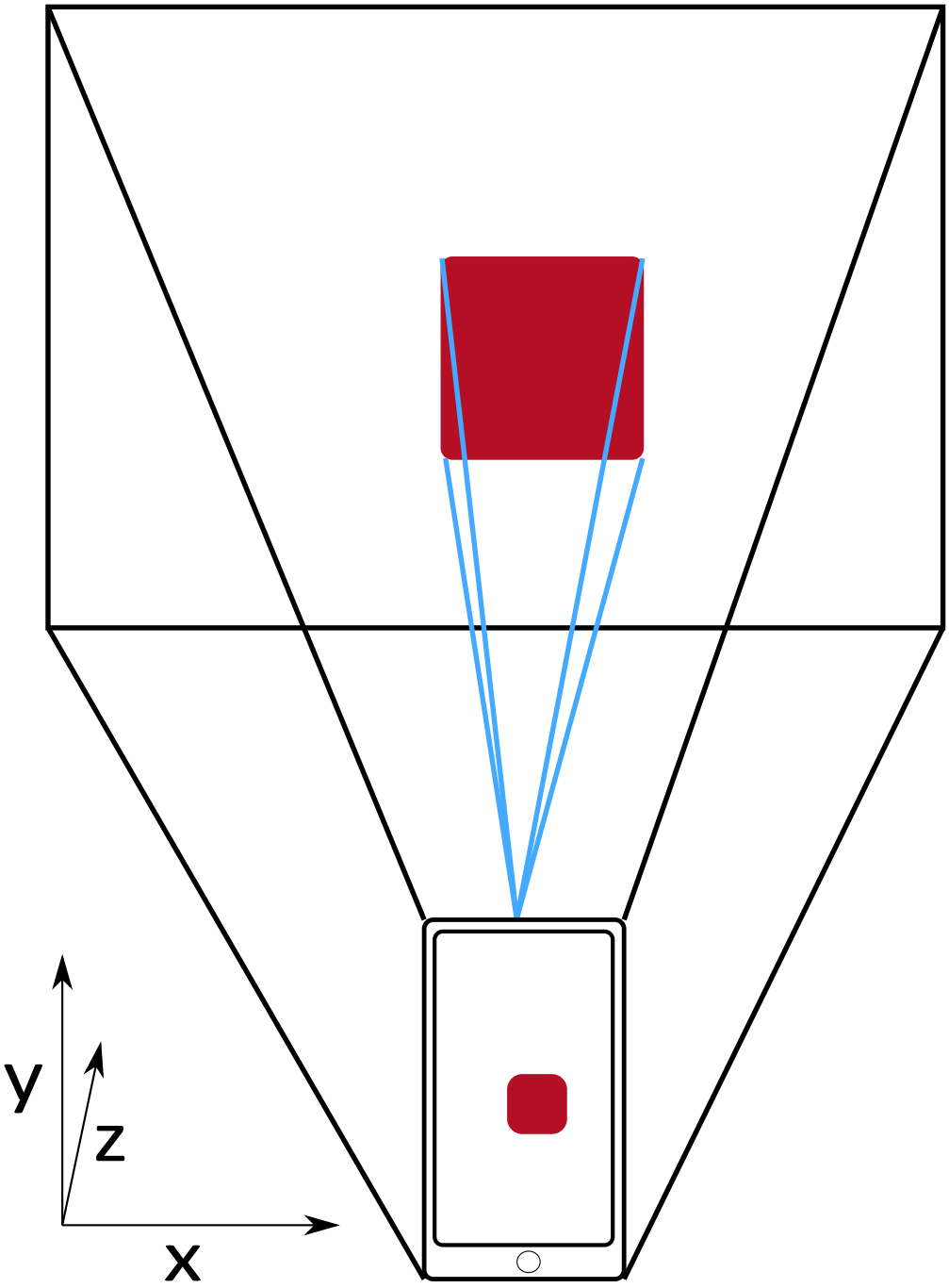


Figure 3.7: A colored label in 3D. The user's device is currently facing towards the label. The blue lines represent four rays that are traced for the occlusion detection in the *Occlusion Management*.

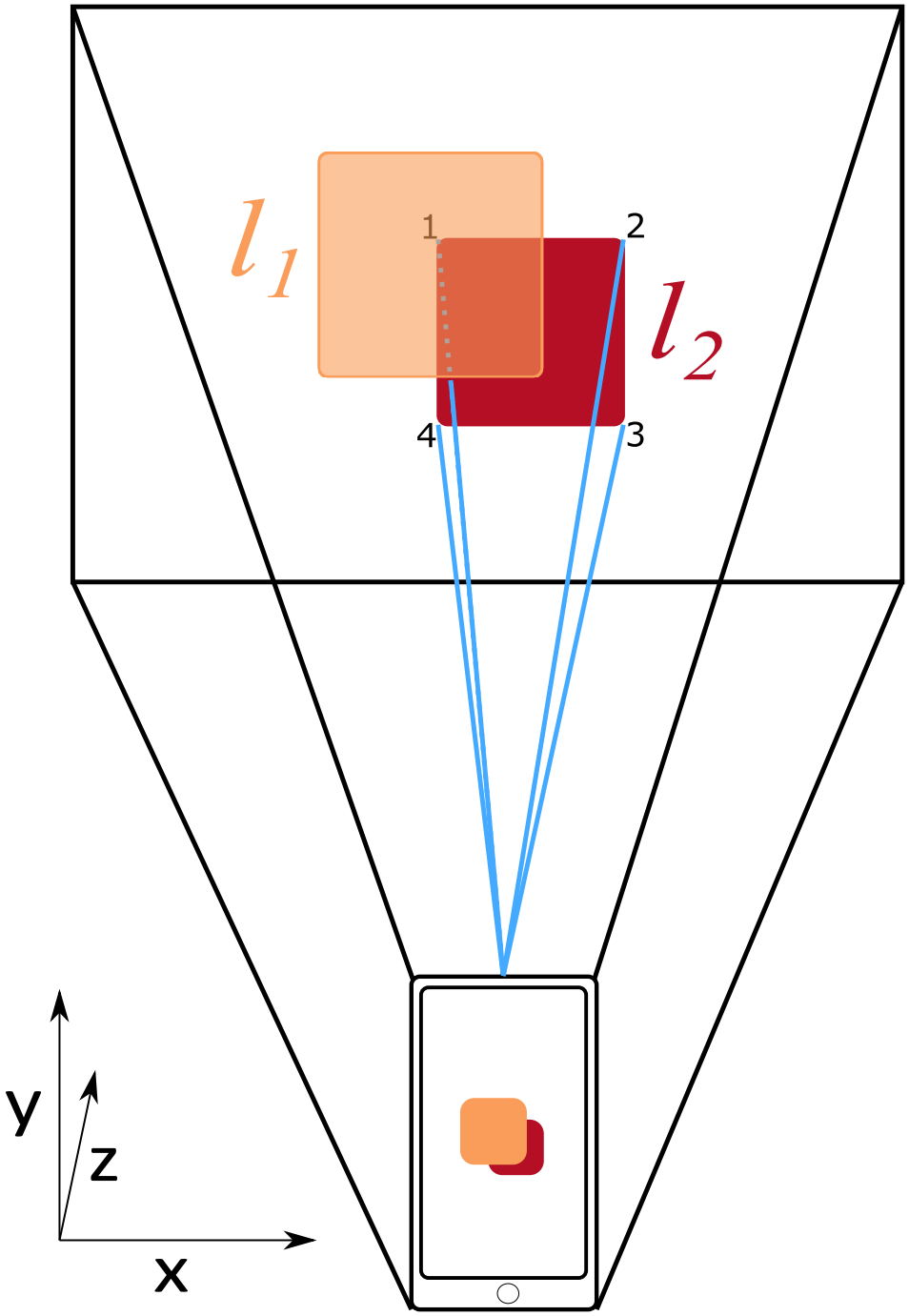


Figure 3.8: Occlusion of label l_1 (orange, transparent, and in front) to label l_2 (red). The ray of corner 1 collides with the occluding label l_1.

Occlusion Detection

Different from existing approaches [BNN19], we employ ray tracing to detect collisions. As the labels got sorted by the distance to the user, the occlusions are detected and resolved iteratively from the label l_1 to label l_n of the sorted labels in list S. For each label l_i, the origins of each four rays are set to the location of the user's device in AR and the destinations are each corner point of the label l_i (Figure 3.7). If another label is hit during the ray traversals, occlusions are detected. Each time occlusions are detected, they are resolved for the label l_i. To ensure that all possible occlusions will be detected, we assume that objects closer to the viewer are either larger or as large as far labels. It is because the *Level-of-Detail Management* selects higher LODs for close labels to show more information. This allows us to calculate four rays to detect 3D occlusions effectively. The approach works for rectangular shapes or rectangular bounding boxes of polygonal shapes, and would be extendable to polygons by increasing the number of rays.

Figure 3.8 gives an example, where label l_1 (orange) is in front of label l_2 (red) label. In this case, the ray of corner 1 of label l_2 collides with label l_1, indicating that label l_1 occludes label l_2. Since we assume that closer labels are always larger or as large as far labels, no labels will be missed.

Shift Computation

Algorithm 3.1: Simplified occlusion handling through shifting.

Data: Labels

Result: Occlusion-free labels

```
1  S = list of sorted labels;
2  for l_i ∈ S do
3      create four rays to corners of l_i;
4      while ray collides with label do
5          get occluding label;
6          shoot ray from user to top of occluding label;
7          trace ray to (x, z) position of l_i;
8          shift l_i above this point;
9      end
10     //l_i is completely visible;
11 end
```

Once the occlusions are detected, they are resolved in the next step. We keep the x and z coordinates and change the y-coordinate because of the importance of the labels being

displayed at the respective x and z positions of the real-world POIs for navigation. The labels were sorted based on the distance to the user. The closer the label is located in 3D in AR, the smaller the shift during the *Occlusion Management* will be. The basic procedure is explained in Algorithm 3.1.

In the first step, the labels are sorted by distance. Afterward, four rays are constructed that travel from the viewer to the four corner points of label l_i. If one or more rays collide with an object closer to the viewer, a new y position is calculated for the current label. Figure 3.9 illustrates the basic shift of one label. The current label, in this case, is the red one and the orange label, which is closer to the user, occludes the red one. The red label is shifted to be visually above the orange label and the occlusion is resolved.

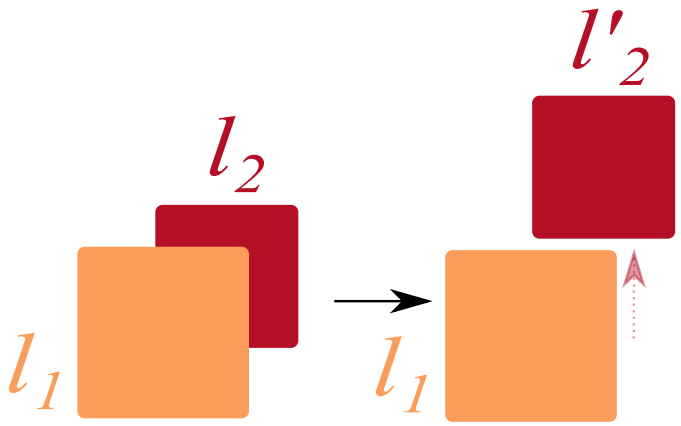


Figure 3.9: Two colored labels, label l_1 (orange) and label l_2 (red). Label l_2 is occluded by label l_1. Label l_2 is shifted upwards to resolve the occlusion.

For the calculation of the vertical shift (upward shift along y-axis), the label (the occluder) that is hit from one of the rays of the current label (the ocludee) acts as a reference. The occluder is located between the viewer and the ocludee. To calculate the correct vertical shift, a ray is cast from the viewer to the top of the occluder. This ray is traced till it collides with the plane of the occludee. This point is taken as a reference for the shift. The ocludee is shifted until the whole label is located above this point. Figure 3.10 explains this procedure. Label l_1 (orange) is the occluder and the label l_2 (red) is the occludee. The gray ray is traced from the viewer beyond the upper edge of label l_1 until it collides with the plane of label l_2. This intersection point is the reference point for the shift. Label l_2 is shifted upwards by the distance d above the intersection point. This procedure resolves the occlusion between label l_1 and label l_2. For a more detailed example see Chapter 6.2. After label l_2 is shifted, the occlusion is resolved. In Figure 3.11 label l_2 is located at its occlusion-free position. The occlusion detection rays (blue) do not intersect with a label in front. The occlusion is resolved (Figure 3.11).

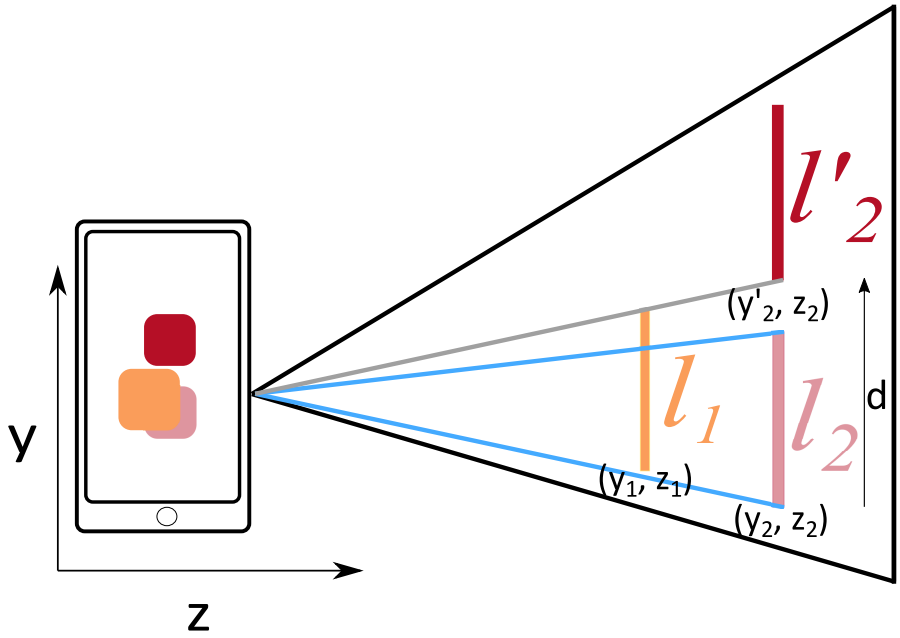


Figure 3.10: Two colored labels. Label l_2 (red) is shifted above the gray ray to resolve the occlusion by the distance d.

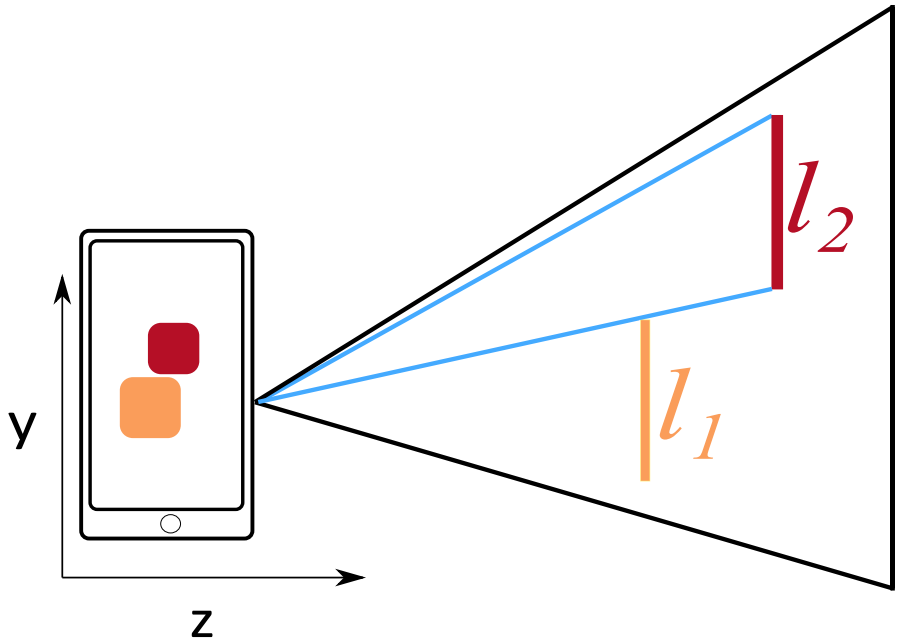


Figure 3.11: Two colored labels. The occluded label from Figure 3.10 was shifted. The occlusion is resolved.

Since the labels are shifted from the closest to the farthest one, the label l_i will be located either at its initial (x, z)-coordinates or above the previous label l_{i-1} along the y-axis.

We know that from Szirmay-Kalos et al. [SKM98], ray-tracing requires minimum logarithmic computation times in the worst case. However, modern visualization platforms facilitate real-time ray-tracing [Tec20a]. In our approach, the *Occlusion Management* takes $O(n^2)$ if labels are aligned in a sequence from the current viewing direction. It is because the current label l_i needs to be shifted above each label in front of l_i. We will show a comparison with different label alignments in Chapter 5. Our iterative, greedy label positioning allows the algorithm to terminate as soon as no other label in front occludes label l_i, reducing the n^2 costs.

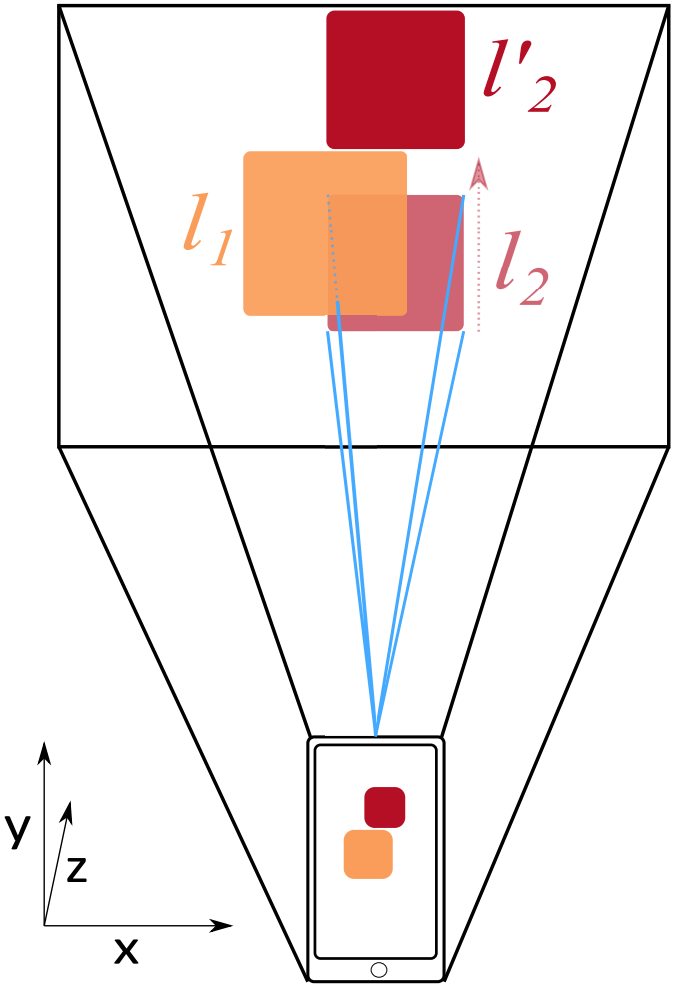


Figure 3.12: After the vertical shift of label l_2 (red) behind the label l_1 (red) is complete, both labels are fully visible on the user's device.

If several rays that are constructed from the corner points of a label to the user's device

hit another label in front, the largest vertical shift is taken in this iteration step. This could happen if two labels in front are close to each other in the user's view and the label behind is located between the two in the current view volume. To ensure an occlusion-free result, the largest vertical shift according to the calculations with all four corner rays of the current label is considered during the occlusion handling.

After the previously described steps, an occlusion-free state is achieved for the labels. Figure 3.12 illustrates labels l_1 and l_2. Label l_2 is shifted to the occlusion free position. The device in the figure presents the final state of the two labels after the vertical shift. Both labels are completely visible.

3.5 Level-of-Detail Management

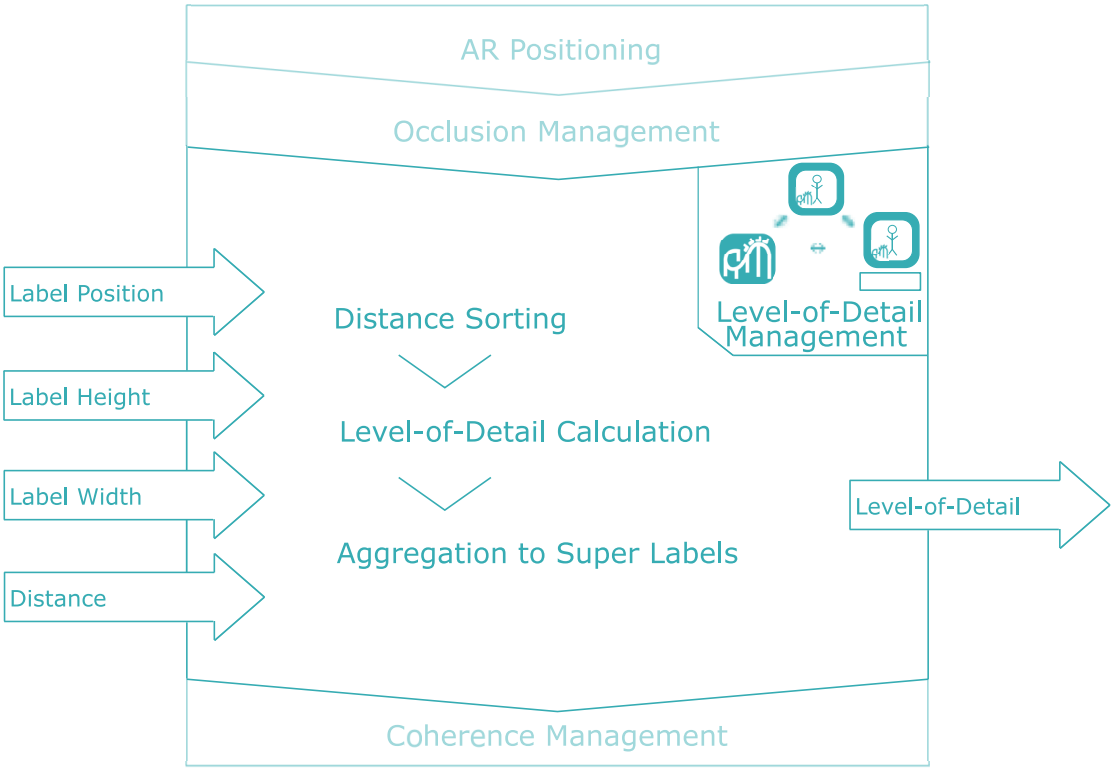


Figure 3.13: The *Level-of-Detail Management*. The steps of this management strategy are included from top to bottom and the arrows show the input and output parameters.

Our encoding (Section 3.1) occupies space that is limited on a mobile device, especially if many labels should be visible at the same time. In order to solve this problem, we employ LODs, which are selected in the *Level-of-Detail Management* (Figure 3.13). The LOD is computed based on the user position and the label density around the user (*Dynamic LODs*). Each of the steps of this pipeline is explained in the next subsections.

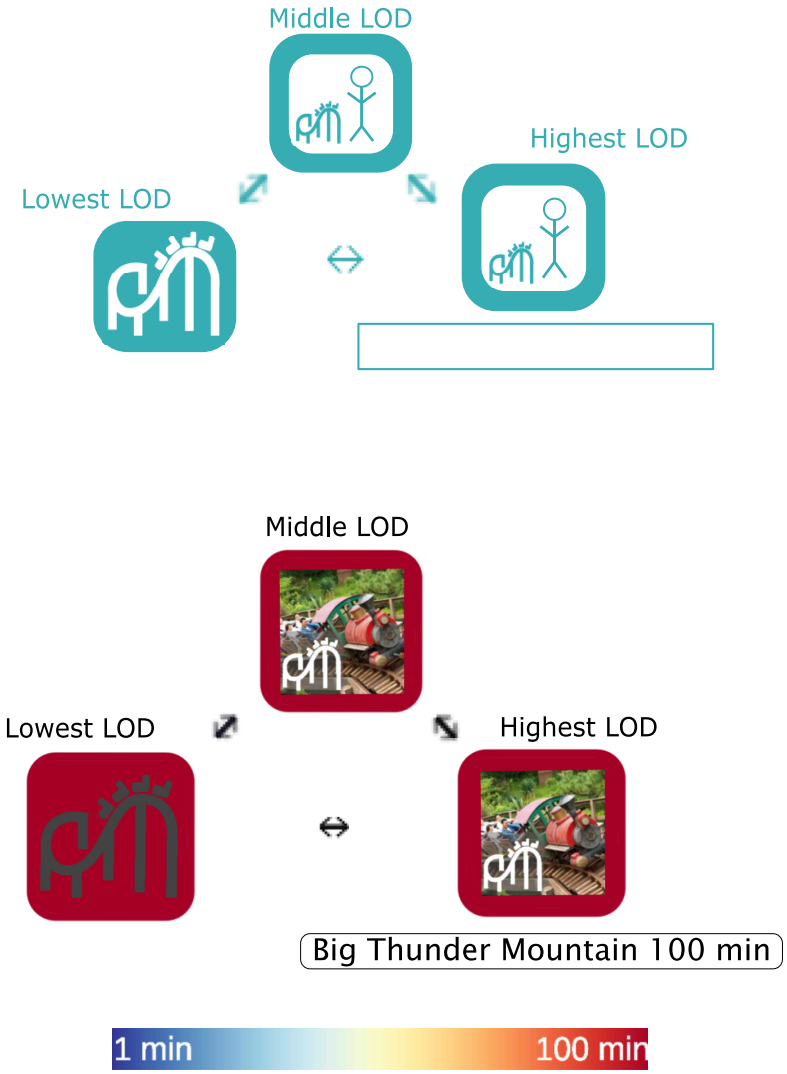


Figure 3.14: Our three LODs for individual labels. The arrows between them indicate the possible state changes over time.

The project includes three LODs for individual labels as presented in Figure 3.14. The *Lowest LOD* occupies the least space and includes a colored rectangle and an icon indicating the type of the POI. The *Middle LOD* presents the colored rectangle, the type icon, and an iconic image of the POI (e.g., a photo). The *Highest LOD* adds a text tag and occupies the most space. The text tag contains the name of the annotated POI.

Figure 3.14 depicts an example of our *Tokyo Disneyland Dataset* for the attraction *Big Thunder Mountain*. The *Lowest LOD* indicates a high waiting time based on the color coding. The type icon indicates that the attraction is a thrilling attraction. The *Middle LOD* further displays a photo of the iconic train of the *Big Thunder Mountain*. The

Highest LOD adds a text tag and shows the name and waiting time of the attraction.

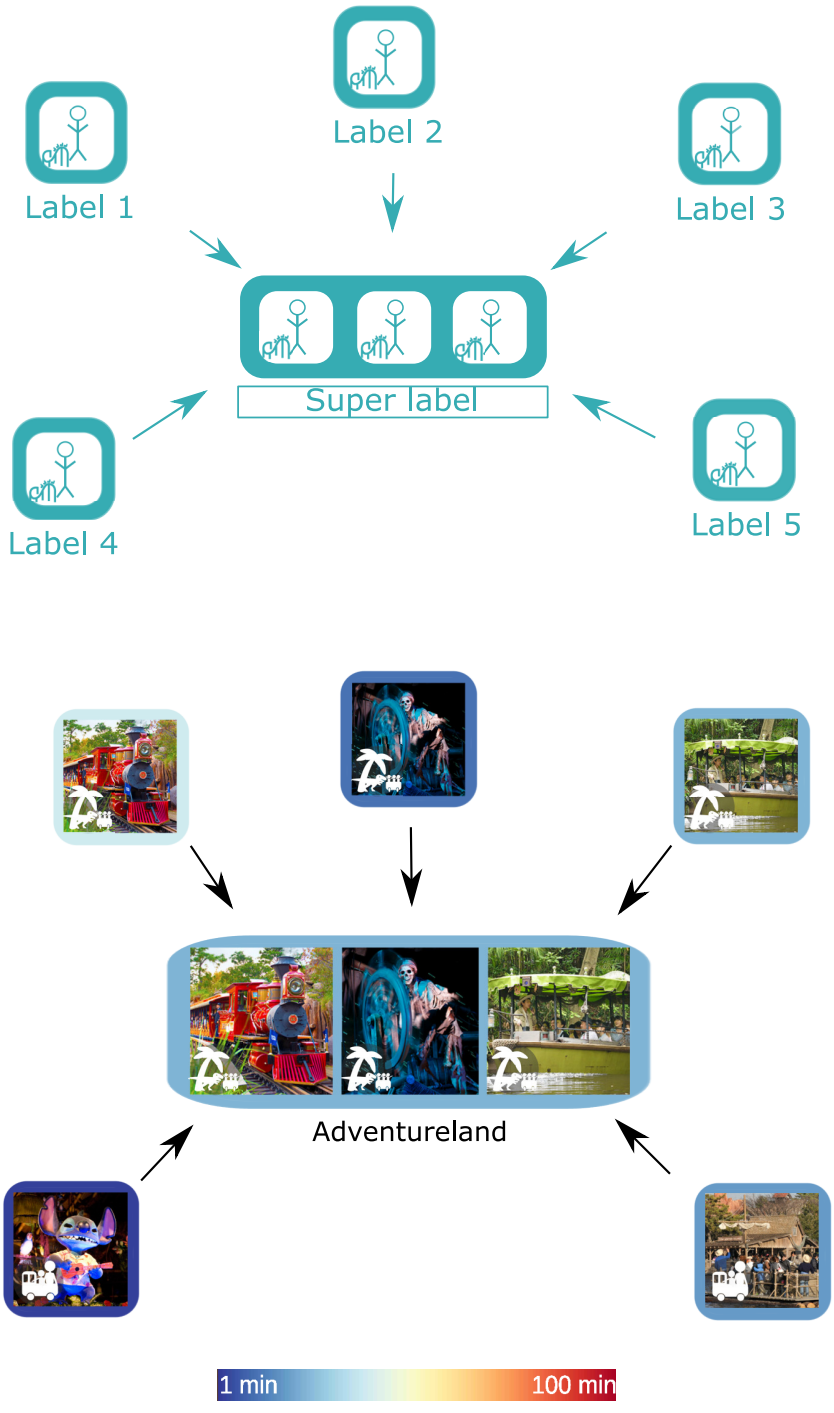


Figure 3.15: Labels with the corresponding super labels.

To reduce visual clutter, we introduce super labels that aggregate a group of individual labels. The super label displays iconic images of contained POIs. The color coding is based on the average value of the respective individual POIs. Figure 3.15 depicts five individual labels and the corresponding super label. The figure further presents an example of the *Tokyo Disneyland Dataset* showing attractions of the themed area *Adventureland*.

Based on the position of the user or the label density around the user, the labels may switch between different LODs over time or they are aggregated to super labels. The next subsections explain this process.

3.5.1 Distance Sorting

Similar to the *Occlusion Management*, the labels are sorted based on the distances of the POIs to the user. Close labels are more likely to be displayed in a higher LOD, providing more detailed information.

3.5.2 Level-of-Detail Computation

The LOD for each individual label depends on the distance to the user and the label density around the user. The more labels are too close to each other, the higher the label will be shifted by the *Occlusion Management* considering the position of the user.

The LOD is selected for one label after another starting from the closest and ending with the farthest. For each label, a virtual view volume aligned to the (x, z) ground plane is constructed to mimic that the user would look into the direction of each label.

The vector along the (x, z) ground plane and the vector from the user to the position of each label are used. If the angle between these two vectors is above a threshold t (45° by default in our system), the label is located outside the aligned view volume. We split the view volume and each label below the threshold m_1 (20° by default) receives the *Highest LOD* until one label passes the angle m_1. After that, the labels are displayed in the *Middle LOD* until m_2 (30° by default) and if a label reaches this threshold, it will be displayed in the *Lowest LOD*. These threshold angles can be changed according to user preferences.

The LODs of all labels are consistent if the viewing angle of the device changes for the current user position, like the result of the *Occlusion Management*. The *Level-of-Detail Management* provides coherent label movements when rotating the AR device. The LODs for the label are updated if the user walks through the scene.

3.5.3 Aggregation to Super Labels

To further reduce visual clutter, we introduce super labels. The super labels show representative POIs of the aggregated labels and the average value of all contained labels is color-coded. The position of a super label is calculated as the average (x, z) positions

of the individual labels that are part of the aggregation in the 3D scene. A predefined categorization of labels is necessary to compute the super labels. This categorization can be defined in the data or computed using clustering techniques. We do not aggregate labels of the closest predefined category considering the position of the user. In doing so, individual labels in the close surroundings of the user are always displayed and not aggregated supporting the user's exploration process. In other words, we only aggregate individual labels to super labels if the user is located outside of the respective predefined category.

3.6 Coherence Management

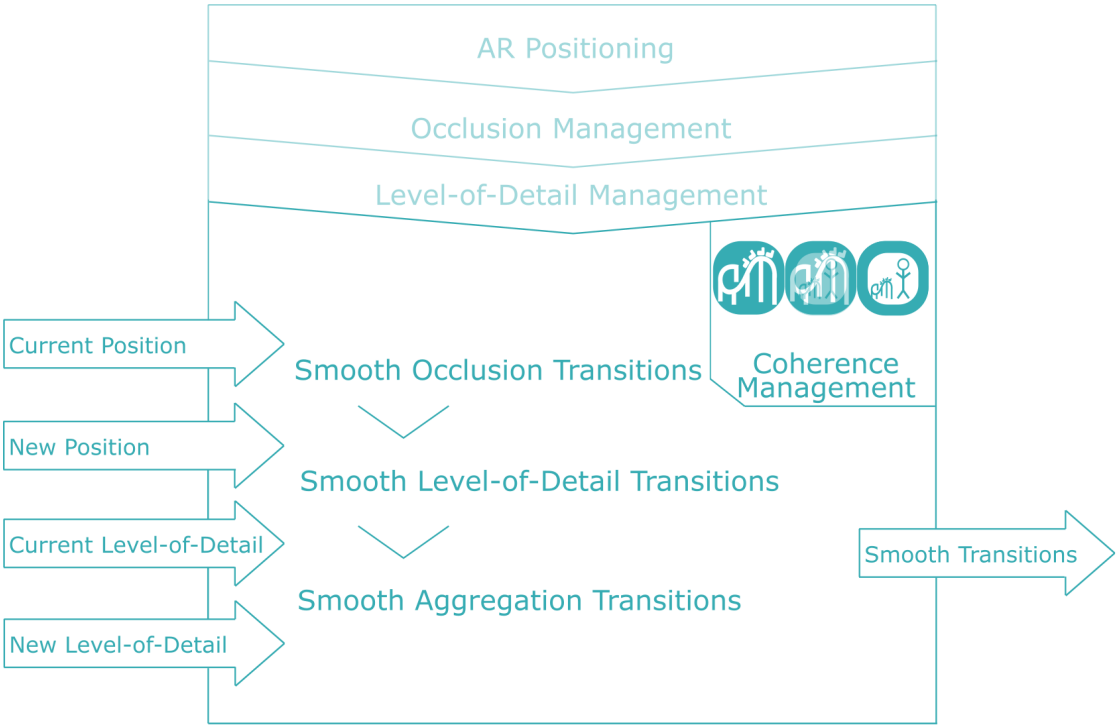


Figure 3.16: The *Coherence Management*.

To avoid unwanted flickering, we incorporate smooth transitions for each movement and change in the *Coherence Management* (Figure 3.16). These include if positions of labels change to be occlusion-free during the interaction with the system, if LODs of labels change, or if labels are aggregated to super labels.

3.6.1 Smooth Occlusion Transitions

Due to the interaction of the user, occlusion-free label positions may vary from one computation to another. If the labels would simply be displayed at the newly calculated

positions, the labels would abruptly change their positions, which destroys the user's experience since the labels do not move in a coherent way. To allow the user to better keep track of the labels, we implemented smooth transitions from the previous locations of the labels to the newly calculated ones, by interpolating the original positions and the newly calculated positions of labels. The label positions are updated in every frame until they reach the destinations. In our implementation, the positions are updated linearly using Equation 3.1, while common ease-in ease-out effects can be easily incorporated. Let $p(l_i')$ be the new occlusion-free label position and $p(l_i)$ the current label position, we calculate the speed $v(l_i)$ according to the preferred time $t_{transition}$ for the transition until each label reaches the corresponding final position. This time can be set as preferred by the user.

$$v(l_i) = \frac{p(l_i') - p(l_i)}{t_{transition}} \tag{3.1}$$

3.6.2 Smooth LOD Transitions

If the LOD for a label changes, the transition needs to be smoothed to avoid flickering and allow a coherent user experience. The LODs of labels change over time, and we adapt the alpha channel to achieve a smooth transition. In this way, the iconic images, the icons, and the text tags fade in or out using

$$\alpha(l_i) = \begin{cases} \frac{t_{current} - t_{start}}{t_{transition}}, & b = 1 \\ 1 - \frac{t_{current} - t_{start}}{t_{transition}}, & b = 0, \end{cases} \tag{3.2}$$

where $\alpha(l_i)$ is the alpha value of the iconic image, the icon, or the text tag of label l_i. The variable $t_{transition}$ gives the duration of the smooth transition. Let b be 0 if the object should be invisible or 1 if the object should be visible. The variables t_{start} and $t_{current}$ indicate the start time and the current time of the transition. The alpha value for each update in the $t_{transition}$ window is computed using Equation 3.2.

3.6.3 Smooth Aggregation Transitions

When labels are aggregated to super labels, individual labels will be moved to the respective super label positions in the scene. Simultaneously, we blend in the super labels and blend out the labels by interpolating the alpha channels. When individual labels are aggregated, the labels move towards their super label and disappear. If the aggregation is split up again, the coherency is achieved in opposite. If the alpha channel of a super label is decreased, the individual labels reappear over time and travel back to their respective positions (Equations 3.3, 3.4, and 3.5). Let l_i be a label that will be aggregated by super label l_s. The variables t_{start} and $t_{current}$ indicate the start time of the aggregation and the current time. Then the alpha values of l_i and l_s in the $t_{transition}$ window and the position of l_i are computed as follows if the individual labels are aggregated by super labels.

$$\alpha(l_i) = 1 - \frac{t_{current} - t_{start}}{t_{transition}} \tag{3.3}$$

$$\alpha(l_s) = \frac{t_{current} - t_{start}}{t_{transition}} \tag{3.4}$$

$$p(l_i') = p(l_i) + (p(l_s) - p(l_i)) * \frac{t_{current} - t_{start}}{t_{transition}} \tag{3.5}$$

CHAPTER 4

The Datasets

We investigate three datasets, a *Tokyo Disneyland Dataset*, a *Local Shops Dataset*, and a *Synthetic Dataset*. These datasets are explained in this chapter.

4.1 Tokyo Disneyland Dataset

The Tokyo Disneyland [Dis20] is an amusement park and part of the Tokyo Disney Resort, which is located in Urayasu near Tokyo. The whole resort consists of the Tokyo Disneyland, the Tokyo DisneySea, and several hotels, shops, and theaters. The Tokyo DisneySea contains many sea-themed attractions, while the Tokyo Disneyland can be compared to the conventional Disneyland parks like in Anaheim (California), Orlando (Florida), or Paris (France). The basic structures of these amusement parks are similar to each other. Iconic themed areas like *Fantasyland*, *Tomorrowland*, or *Adventureland* are present in each Disneyland. Well-known attractions are also located in several parks, like *Space Mountain*, *Splash Mountain*, or *Haunted Mansion*. If a visitor is familiar with one of the Disneylands, he or she will be familiar with the basic structures of the others and he or she will have a basic idea of which kind of attraction corresponds to which name. The basic structures of the different areas are similar across different Disneylands as well, which eases navigation if a visitor is familiar with one of them.

The Tokyo Disneyland in particular consists of main 35 attractions in seven different areas, which can be seen in Table 4.1. The iconic castle is located at the center of the Tokyo Disneyland and the themed areas of this amusement park, in particular, are arranged in a circle around the center. The number of attractions in each of the themed areas varies from 1 to 11 attractions. The attractions can be classified into three types, thrilling attractions, adventure attractions, and children attractions. The waiting times of each of these attractions are dependent on several different factors like the popularity, season, weather, time of the day, or type of attraction. Attractions, e.g., *Space Mountain*,

that are present in almost every Disneyland and therefore well known are likely to have high waiting times.

Adventureland	Western River Railroad
	Pirates of the Caribbean
	Jungle Cruise Wildlife Expeditions
	Swiss Family Treehouse
	The Enchanted Tiki Room
Westernland	Westernland Shootin' Gallery
	Country Bear Theater
	Mark Twain Riverboat
	Tom Sawyer Island Rafts
	Big Thunder Mountain
Critter Country	Splash Mountain
	Beaver Brothers Explorer Canoes
Fantasyland	Alice's Tea Party
	It's a small world
	Castle Carrousel
	Snow White's Adventures
	Cinderella's Fairy Tale Hall
	Dumbo The Flying Elephant
	Peter Pan's Flight
	Pinocchio's Daring Journey
	Pooh's Hunny Hunt
	Haunted Mansion
	Mickey's PhilharMagic
Toontown	Gadget's Go Coaster
	Goofy's Paint 'n Play House
	Chip 'n Dale's Treehouse
	Donald's Boat
	Minnie's House
	Roger Rabbit's Car Toon Spin
Tomorrowland	Star Tours
	Stitch Encounter
	Space Mountain
	Buzz Lightyear's Astro Blasters
	Monsters, Inc.
World Bazaar	Omnibus

Table 4.1: Areas of the Tokyo Disneyland [Dis20] and the respective attractions.

Water attractions, like *Splash Mountain*, often have higher waiting times during summertime. Visitors of the amusement park have access to 2D printed or online maps, which can make it difficult to choose, locate, or navigate towards an attraction. Online waiting times are provided in some amusement parks, but they are also not embedded into the map. This makes the decision process difficult and visitors may spend a lot of time waiting at several of the 35 attractions. The reasons mentioned above show the complexity of the decision-making process and the need for tools to support the visitors.

4.2 Local Shops Dataset

The *Local Shops Dataset* provides another real-world example, where the labels are close and next to each other. The dataset includes 13 shops. The *Local Shops Dataset* contains the shop locations, the types of shops, and the number of people inside a shop per m^2 of a strip mall. We classified the shop types into clothing, shoes, and groceries. Considering the current COVID-19 regulations, we encode the number of people per m^2, to identify the current customer density in the shop. We use the number of people per m^2 measure to indicate COVID-19 safety. The higher this value, the less safe the shop becomes.

4.3 Synthetic Dataset

The *Synthetic Dataset* shows different variations of label layouts when applying our technique. We investigate three different label layouts. The labels are aligned are a circle layout (Figure 4.1), a grid layout (Figure 4.2), and a line layout (Figure 4.3). The label layouts are considered to be more computationally expensive from the circle to the grid to the line layout.

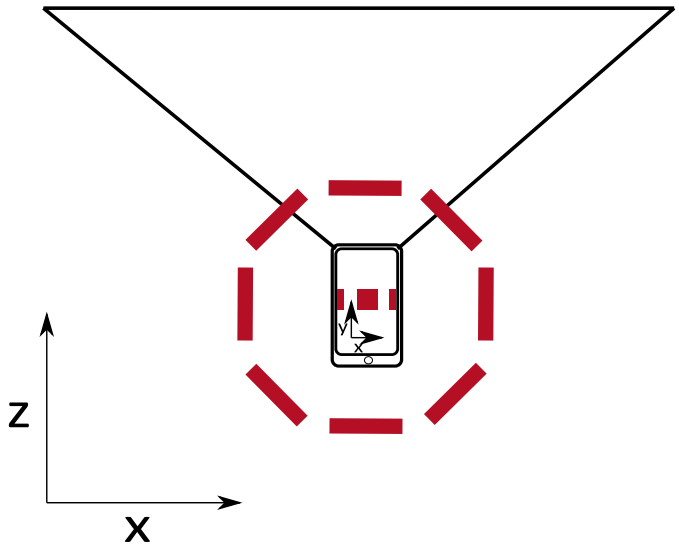


Figure 4.1: Illustration of the circle layout (*Synthetic Dataset*) in top view.

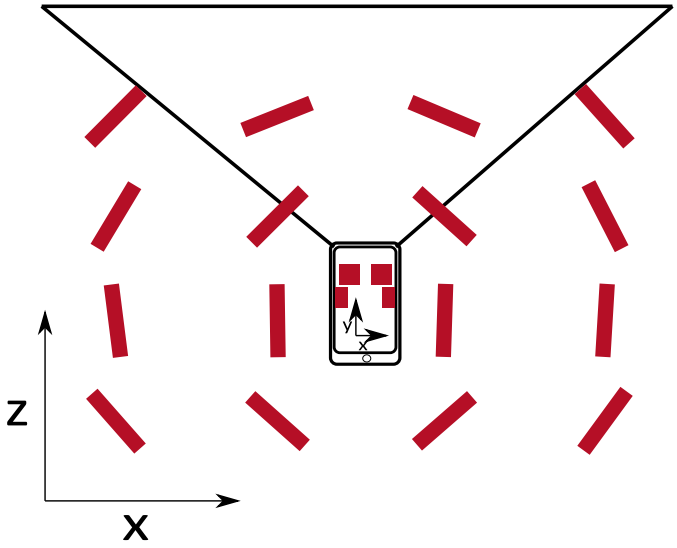


Figure 4.2: Illustration of the grid layout (*Synthetic Dataset*) in top view.

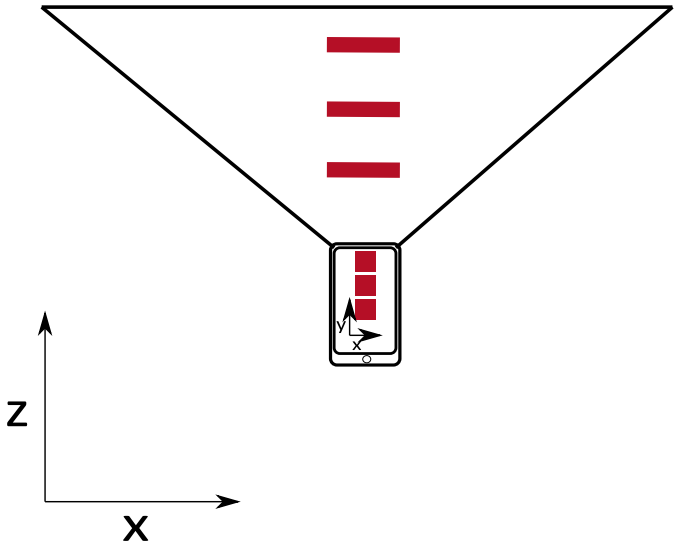


Figure 4.3: Illustration of the line layout (*Synthetic Dataset*) in top view.

Chapter 5 will present the results of our approach described in Chapter 3, specifically for the three datasets.

CHAPTER 5

The Labeling in Practice

Corresponding to the datasets described in Chapter 4, this chapter will present the results of our approach. The methodological basics are explained in Chapter 3. The POIs are labeled in AR based on the dataset to show the functionality of the tool.

This chapter will firstly depict the labels in the AR environment. Secondly, the results after the occlusion handling will be presented with and without super labels. Thirdly, the results including different LODs will be presented. Finally, coherent results are depicted and computation times of the *Synthetic Dataset* are shown. The screenshots included in this chapter were taken on a Xiaomi Mi A2 device in portrait mode. Due to privacy reasons, the background scene is blurred.

For the *Tokyo Disneyland Dataset*, the labels are conceptually colored based on the waiting time of each attraction, with blue labels indicating lower than average waiting times, yellow labels indicating average waiting times, and red labels indicating higher than average waiting times. For simplification, the resulting screenshots include waiting times from 1 to 100 minutes. For the *Local Shops Dataset*, the labels are conceptually colored based on the number of people per m^2, using a color scale ranging from white to red. The chapter will mainly present results for the *Tokyo Disneyland Dataset* to allow an easier comparison of the different management strategies.

5.1 AR Positioning Results

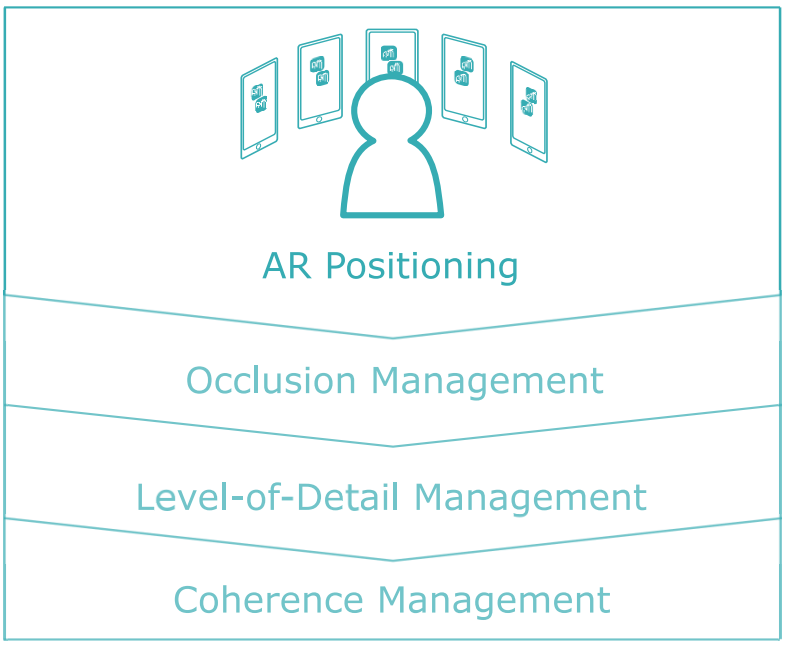


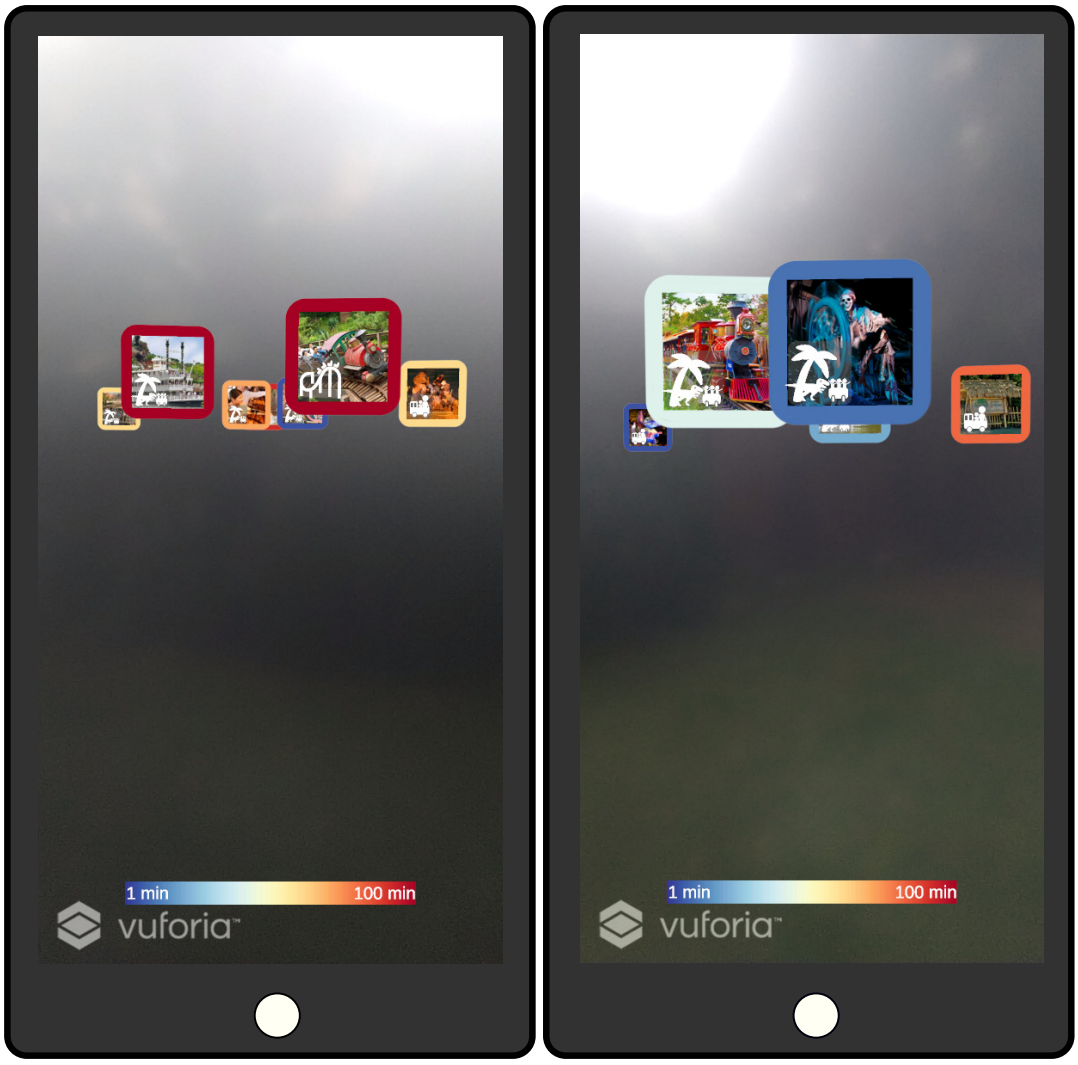
Figure 5.1: Overview of the pipeline. Results of the *AR Positioning*.

The labels are initially positioned in the *AR Positioning* (Figure 5.1). The labels are user aligned by rotating around the up-axis (y-axis) such that the user can see the information provided by the labels independent of the current position or viewing angle. For the *Tokyo Disneyland Dataset*, the screenshots in this section include a colored rectangle encoding the waiting time, an iconic image of the respective attraction, and the attraction type icon. For more information about the label encoding see Section 3.1. The initial placement of the labels is explained in Section 3.3.

Figure 5.2 provides two examples of the *AR Positioning*. Figure 5.2(a), e.g., includes labels that annotate attractions of the *Westernland* of the Tokyo Disneyland including attractions like *Big Thunder Mountain* or *Splash Mountain*. The iconic images display representative, well-known features of attractions, e.g., the iconic image of the attraction *Big Thunder Mountain* contains the well-known train. The colored rectangles behind the images encode the current waiting times of the attractions. In this example, the colored rectangle behind the iconic image for *Big Thunder Mountain* is colored red, indicating a high waiting time. Attractions closer to the user are annotated by visibly larger labels than distant ones as they are located near the user. The closer the user walks towards an attraction, the larger the respective label becomes. In this example, the attraction *Big Thunder Mountain* is currently the closest attraction regarding the position of the user resulting in the visibly largest label. After the *AR Positioning*, close attractions may occlude distant ones as seen in Figure 5.2. In this case, it is hard or even impossible to

gather information about distant attractions. This indicates the need for the *Occlusion Management* (Chapter 3). The results with resolved occlusions are presented in the next sections.

Figure 5.2(b) illustrates the result for the themed area *Adventureland*. The two attraction labels in the front occlude labels that are currently farther away in the same viewing direction. The label containing the mid-blue colored rectangle in the back, e.g., is almost completely behind the closer labels and only a small part of the color-coded waiting time is visible. The iconic image is barely visible, which means it is impossible to recognize the respective attraction.



(a) *Westernland* (b) *Adventureland*

Figure 5.2: Results of the non occlusion-free *AR Positioning* at different viewing angles of the user's device.

5.2 Occlusion Management Results

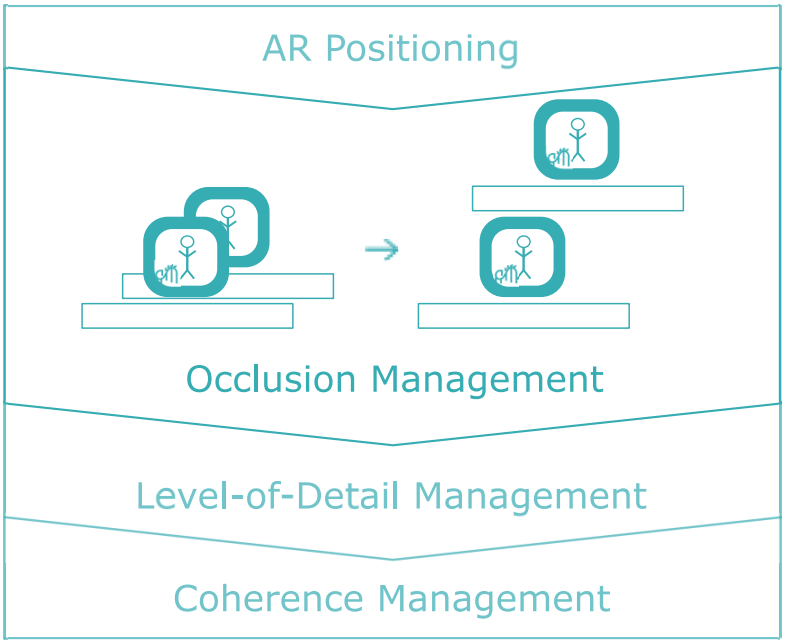


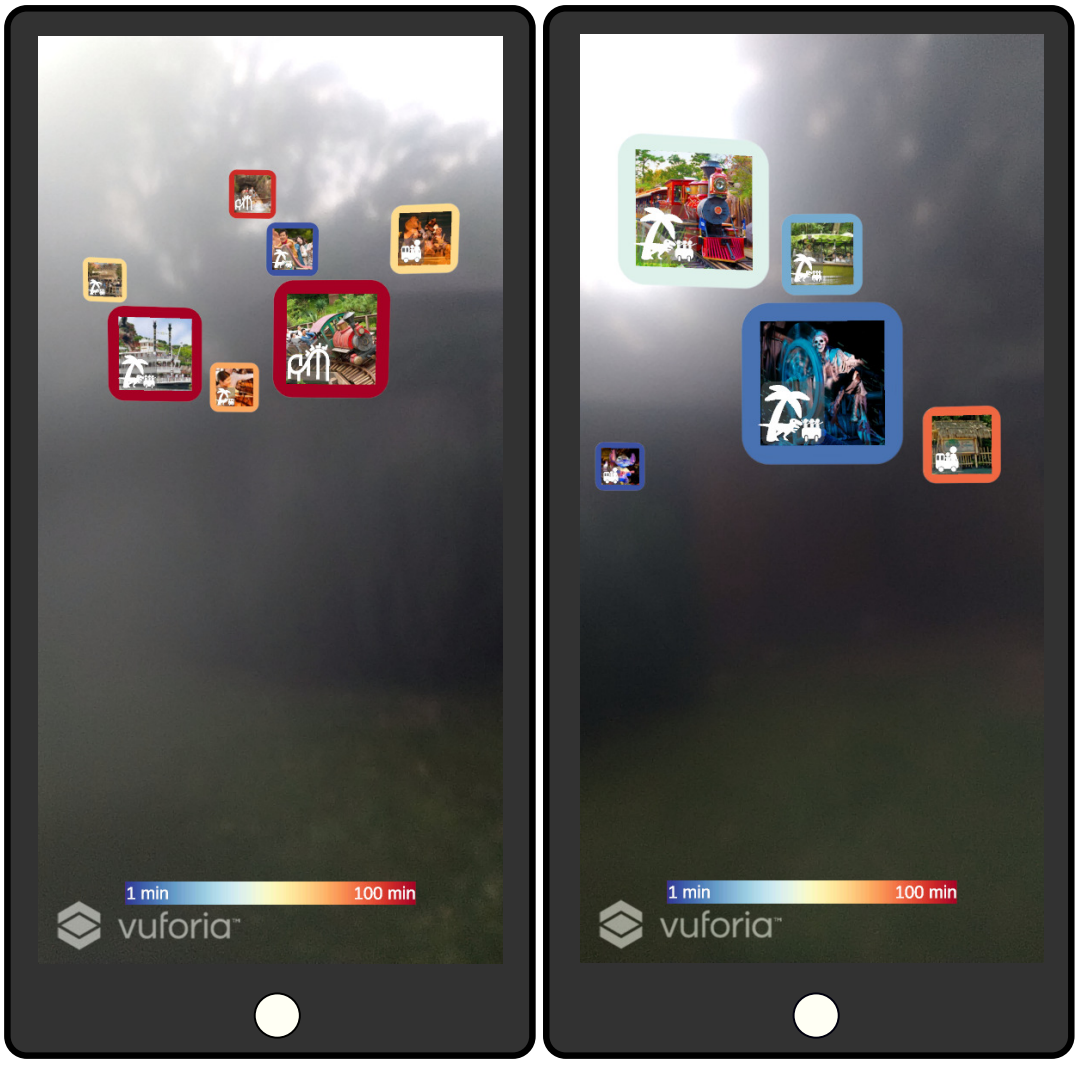
Figure 5.3: Overview of the pipeline. Results of the *Occlusion Management*.

As described in the previous section, the *AR Positioning* of the labels leads to occlusions making it difficult or impossible to gather information about distant labels. This section introduces the results of the *Occlusion Management* (Figure 5.3).

For a detailed explanation about the *Occlusion Management* see Section 3.4. Occlusions are detected in 3D in AR based on ray tracing. The required shifts of labels are calculated and occluded labels are shifted upwards because of the importance of keeping the x, z positions of the labels for navigation (see Chapter 2). Near labels stay close to the initial positions as they are considered to be more important during the exploration process. The *Occlusion Management* calculates global occlusion handling based on the users' position, each label is not occluded by another one. Viewing angle changes of the user's device that usually cause problems (see Chapter 2) do not influence the result and do not lead to flickering or incoherently moving labels in our approach.

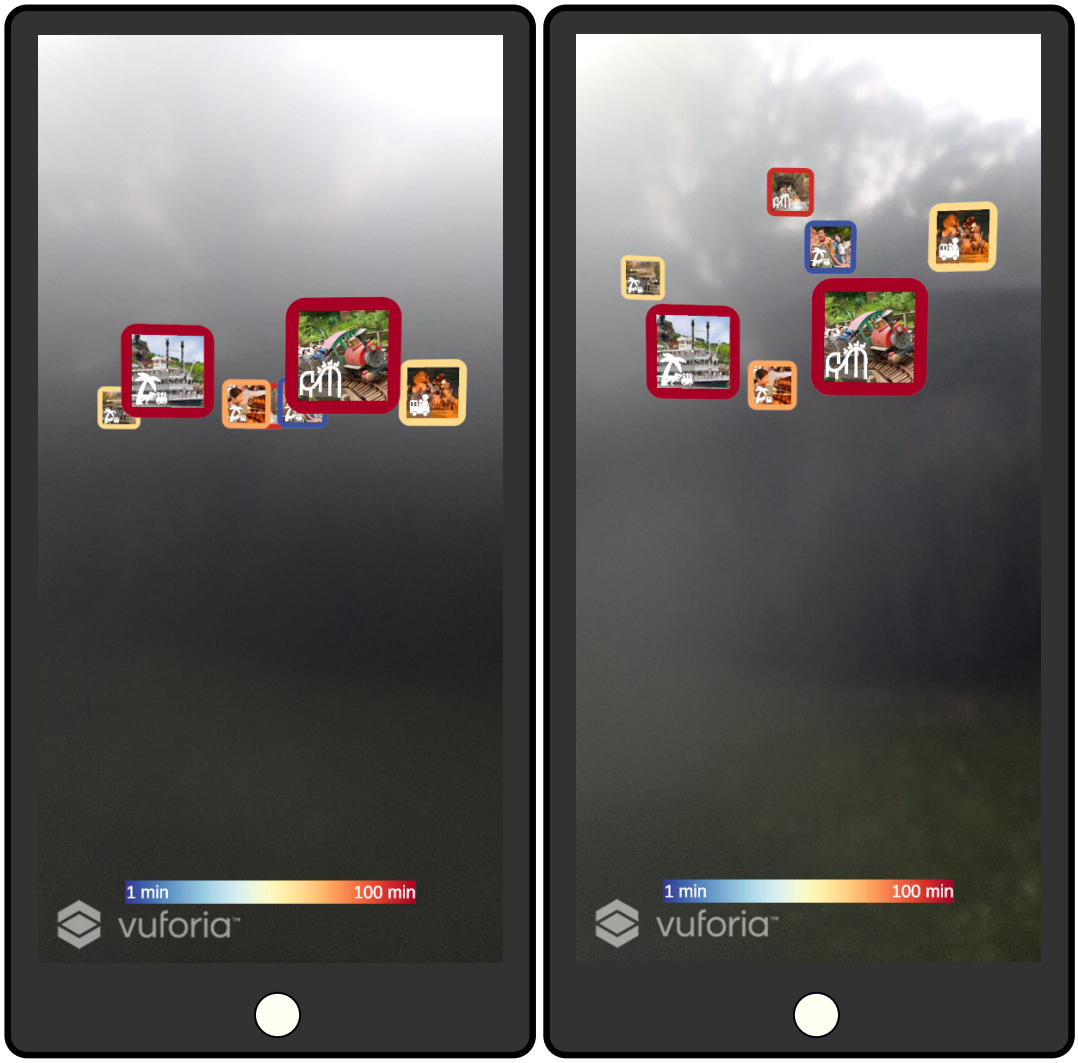
Figure 5.4 presents occlusion-free results for the *Westernland* (Figure 5.4(a)) and the *Adventureland* (Figure 5.4(b)). Labels do not occlude distant ones. All labels are visible and the iconic images and the color-coded waiting times of all attractions that are located in the current viewing direction can be seen. The label of the attraction *Big Thunder Mountain* in Figure 5.4(a), showing an iconic image of a train, is the closest one in this example. The position of this closest label is not changed. Labels of attractions that are behind *Big Thunder Mountain* are shifted upwards to be completely visible. The

user's device is facing different viewing angles in Figure 5.4. Changes in viewing angles do not lead to a recalculation of the label layout because the *Occlusion Management* resolves occlusions for all labels around the user, which avoids flickering and frequently moving labels to ease the exploration and navigation process for the user. If the position of the user changes and labels start to occlude distant ones, the *Occlusion Management* is updated for the current position and the labels are shifted in a linear, smooth way to provide coherency and avoid flickering.



(a) *Westernland* (b) *Adventureland*

Figure 5.4: Occlusion-free results. The viewing angles of the user's device are similar compared to Figure 5.2.



(a) (b)

Figure 5.5: Result before and after the occlusion handling. Occlusions that occur in (a) are resolved in (b).

Figure 5.5 compares the results for the same position and viewing angle of the user's device. Figure 5.5(a) presents the result after the *AR Positioning*. Occlusions occur in this (Figure 5.5(a)). Figure 5.5(b) highlights the result of the *Occlusion Management*. In contrast to Figure 5.5(a), Figure 5.5(b) depicts occlusion-free labels. Larger labels that are closer to the user and considered to be more important are more likely to stay close to the initial positions than labels that are farther away. The two closest labels in this example (Figure 5.5) are *Big Thunder Mountain* and *Mark Twain's Riverboat* showing an iconic image of a train and a boat. The positions of these two labels are not changed. Labels that are occluded by these two labels after the *AR Positioning* are

shifted upwards. The label with the orange background in the center is farther away than the two labels with the red background. The orange label is not shifted upwards because enough space is available in between the two red labels in front and no occlusions occur considering the current position and viewing angle of the user.

5.3 Level-of-Detail Management Results

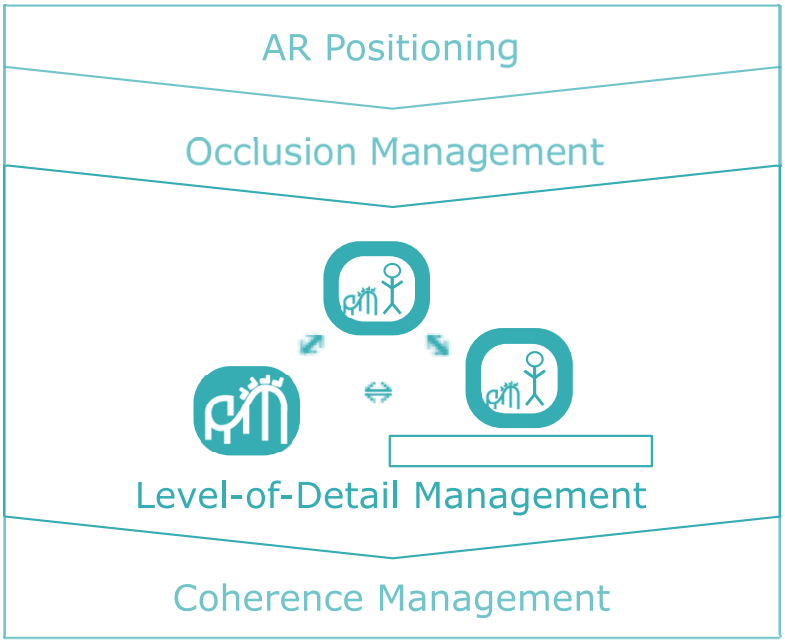


Figure 5.6: Overview of the pipeline. Results of the *Level-of-Detail Management*.

As described in Section 3.5, the system includes three LODs for individual labels. The *Lowest LOD* consists of a colored rectangle encoding the waiting time and an icon highlighting the type of the respective POI. The *Middle LOD* further presents an iconic image of the POI. The *Highest LOD* adds a text tag showing the name of the POI. Labels may be aggregated into a super label as described in Chapter 3.

The LOD is selected based on the position and the label density around the user (*Dynamic LODs*). The LODs either provide more information about POIs or reduce vertical stacking of the labels.

This section describes each LOD and highlights the respective results (Figure 5.6). One after another, the results for the *Lowest LOD*, the *Middle LOD*, the *Highest LOD*, and the *Dynamic LODs* are presented.

5.3.1 Lowest LOD

(a) *Westernland* (b) *Adventureland*

Figure 5.7: Resulting labeling with all labels being displayed in the *Lowest LOD*.

The *Lowest LOD* shows a colored rectangle and an icon for each attraction in the *Tokyo Disneyland Dataset*. The colored rectangle encodes the current waiting time of the attraction. The icons indicate the type of the attractions (thrilling, adventure, or children). Figure 5.7 presents the results for the *Lowest LOD* for the *Westernland* (Figure 5.7(a)) and the *Adventureland* (Figure 5.7(b)). If the labels are displayed in the *Lowest LOD*, users can extract the color-coded waiting time and the attraction type. This level might be most appropriate if the waiting time and the attraction type represent the most important criteria for the user. For example, if the user is currently looking into the

direction of the *Westernland* (Figure 5.7(a)), there is one blue label visible meaning that there is one attraction with a low waiting time. In this way, the *Lowest LOD* supports the decision-making process during the exploration of attractions.

The benefits of the *Lowest LOD* are the coherent appearance of the labels where the color-coded waiting time is a dominant feature at each label in the *Tokyo Disneyland Dataset*. Furthermore, this LOD encodes the least amount of information allowing a basic overview of the surroundings of the user. In addition, the square-shaped labels can be efficiently stacked compared to the results of the *Highest LOD*, which occupies more space because of the added text tag. The benefit also comes with a drawback depending on the scenario and the user's preference. The basic encoding in the *Lowest LOD* may be suitable in some scenarios but for others, users might be interested in seeing more information about each attraction than just the encoded waiting time, the type, and the location.

5.3.2 Middle LOD

Similar to the *Lowest LOD*, the *Middle LOD* consists of a colored rectangle encoding the waiting time and an icon indicating the attraction type. In addition, the *Middle LOD* includes an iconic image for each attraction. In Figure 5.8, the results for the *Westernland* (Figure 5.8(a)) and for the *Adventureland* (Figure 5.8(b)) can be seen. In contrast to the *Lowest LOD* (Subsection 5.3.1), the type icon of the attraction is embedded into the iconic image and displayed at a smaller scale at the bottom left corner. In this way, both the iconic image and the type icon can be displayed simultaneously for each attraction. The iconic images present well-known features of each attraction. For example, the iconic image of *Big Thunder Mountain* includes a photo of the iconic train and the image of *Mark Twain Riverboat* shows a white ferry.

The benefits and drawbacks of this LOD are similar compared to the *Lowest LOD* presented in Subsection 5.3.1. The layout enables a more efficient stacking than possible with the *Highest LOD* as the text tag is not included. The waiting time, the attraction type, and the location can be extracted. In addition, the iconic image provides more information compared to the *Lowest LOD*, while no additional space is needed.

As explained in Chapter 4, each Disneyland consists of several different themed areas, like the *Adventureland*, the *Westernland*, or the *Tomorrowland*. To take it one step further, even the attractions in the different themed areas are similar among the amusement parks. For example, each Disneyland has an attraction called *Big Thunder Mountain*, which is always a roller coaster with steam trains and a western-like scenery. If a visitor is familiar with the *Big Thunder Mountain* in the Disneyland Paris, the visitor knows what to expect from the *Big Thunder Mountain* in Tokyo. The rail designs of the roller coasters are not identical, but very similar. For this reason, the iconic images represent a suitable way of identifying the attractions for people that either have been to a Disneyland or if they are slightly familiar with Disneyland amusement parks in general. Even if a visitor is not familiar with the attractions, the type icons in our encoding help to identify the

pace of the attraction. Furthermore, the iconic images hint at the specific type, e.g., the iconic image of *Splash Mountain* presents the tree trunk wagon of the attraction while water splashes against the riders. This example shows that even if a visitor does not know an attraction, the iconic images give a hint of what to expect in addition to the type icon (thrilling, adventure, children in the *Tokyo Disneyland Dataset*).

(a) *Westernland* (b) *Adventureland*

Figure 5.8: Resulting labeling with all labels being displayed in the *Middle LOD*.

In contrast to the *Lowest LOD*, the type icon of the *Middle LOD* is less prominent. In the *Lowest LOD*, the type icon is almost as large as the colored rectangle. In the *Middle LOD*, the type icon is displayed smaller in the lower-left corner of the iconic images. In this way, both the type icon and the iconic image are detectable while no additional space

is necessary. The *Lowest LOD* showing the large type icon encodes less information, but in a more basic way, while the type icon and the iconic image of the *Middle LOD* present more information about each attraction. Depending on the user's personal preference, one LOD might be favored over the other (see Chapter 7). Considering the amount of information that can be displayed, the *Middle LOD* is preferred compared to the *Lowest LOD*.

5.3.3 Highest LOD

(a) *Westernland* (b) *Adventureland*

Figure 5.9: Resulting labeling with all labels being displayed in the *Highest LOD*.

The *Highest LOD* provides the most amount of information for each attraction. The *Middle LOD* design acts as a basis. In addition to that encoding, a text tag is added to each label containing the name and the exact waiting time in minutes for each attraction (*Tokyo Disneyland Dataset*). The text tags are placed below the colored rectangles. Figure 5.9 presents a similar example as the figures in the subsections before, including labels for the *Westernland* (Figure 5.9(a)) and the *Adventureland* (Figure 5.9(b)). This *Highest LOD* encoding provides the most amount of information while occupying around three times more space because of the larger, rectangular label size. In this case, the *Occlusion Management* considers a constructed rectangle that includes the text tag and the colored rectangle including the icon and the iconic image. The encoding of the *Highest LOD* leads to a larger vertical stacking of labels compared to the previously described LODs. The labels occupy more space may leading to the need for tilting the device slightly upwards to see the labels of each attraction located inside the two areas. Considering the provided information for each attraction, the *Highest LOD* is preferred. Considering the occupied space, either the *Lowest LOD* or the *Middle LOD* is preferred.

5.3.4 Dynamic LODs

The *Dynamic LODs* represent a combination of the benefits from each of the previously described LODs. In this case, the LOD for each attraction label is determined by the closeness of the user to the attraction and the label density around the user. A detailed description of the *Level-of-Detail Management* is contained in Chapter 3 and Chapter 6. The closer the labels are located in AR, the more information will be presented for each attraction. Based on the viewing angle and the position of the user, different labels will be visible. The closeness and the label density around the user are considered to dynamically choose the LOD, thus, determining the amount of displayed information. In this case, the user can benefit from the high amount of information presented in the *Highest LOD* while the vertical stacking of the labels is decreased compared to showing each label in the *Highest LOD*.

Figure 5.10 presents *Dynamic LODs* for the *Westernland* (Figure 5.10(a)) and for the *Adventureland* (Figure 5.10(b)). Close labels are presented in a low LOD showing more information compared to distant labels that are displayed in a higher LOD.

(a) *Westernland* (b) *Adventureland*

Figure 5.10: The LOD for each label is determined by considering the closeness of the user to each attraction and the label density around the user (*Level-of-Detail Management*). Important (close) labels are presented in a lower LOD, encoding more information.

5.3.5 Comparison of LODs

Figure 5.11 compares the presented LODs for the *Westernland*. Figure 5.11(a) presents the *Lowest LOD*. This LOD provides the most basic overview of the attractions, however, it presents the least amount of information. Figure 5.11(b) illustrates the *Middle LOD* adding an iconic image to the encoding. This provides more information about each attraction while including the benefits of the *Lowest LOD*. In this case, the type icon is less dominant than in the *Lowest LOD*, but the *Middle LOD* includes the type icon

and the iconic image in the same space that was necessary for the *Lowest LOD* to only show the attraction type icon. Figure 5.11(c) highlights the *Highest LOD* by adding a text tag stating the name and the exact waiting time in minutes of the attraction (*Tokyo Disneyland Dataset*). This LOD provides the most detailed information, however, the most amount of space is necessary, which leads to a higher vertical stacking of the labels compared to the *Lowest LOD* and *Middle LOD* during the *Occlusion Management*. Figure 5.11(d) highlights the result for the labels of the *Westernland* after the dynamic selection of an LOD for each label as explained in Chapter 3 (*Level-of-Detail Management*). This solution presents a compromise between the displayed amount of information and the vertical stacking of the labels, while including detailed information about close attractions and keeping the vertical stacking low compared to the *Highest LOD*. The preferred LOD might vary depending on the user's preference (see Chapter 7). Each LOD has its benefits and drawbacks, with the *Dynamic LODs* being the most versatile ones as they present detailed information about close labels and reduce vertical stacking of labels.

(a) *Lowest LOD* (b) *Middle LOD* (c) *Highest LOD* (d) *Dynamic LODs*

Figure 5.11: A comparison of different LODs and *Dynamic LODs* side by side.

5.4 Coherence Management Results

To avoid unwanted flickering, we incorporate smooth transitions for each movement and change. These include if positions of labels change to be occlusion-free during the interaction with the system, if LODs of labels change, and if labels are aggregated to super labels. The transitions are resolved in the *Coherence Management* (Figure 5.12).

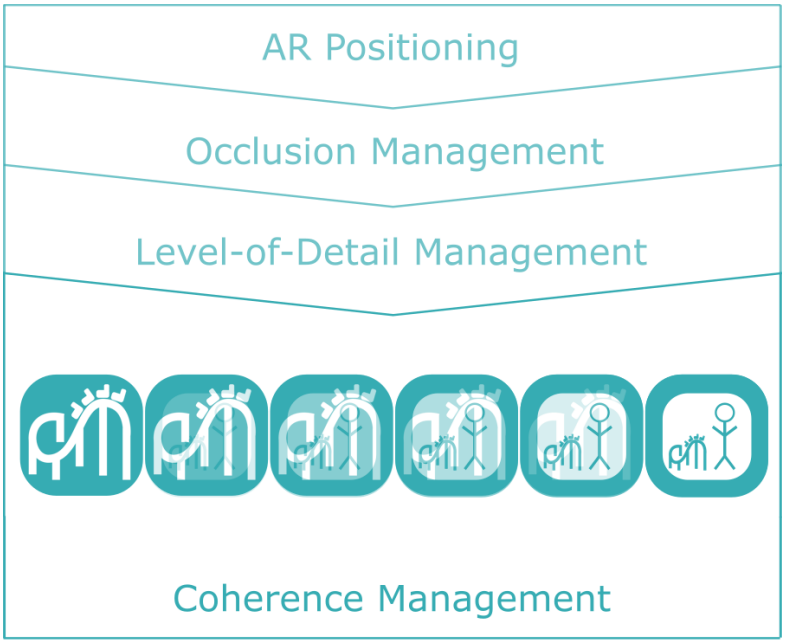


Figure 5.12: Overview of the pipeline. Results of the *Coherence Management*.

In AR, the device is always moving and, therefore, coherent label placement is expected. Viewing angle changes of the user's device do not influence the result as we treat labels as objects in the 3D scene. Figure 5.13 presents different viewing angles of the user's device showing the *Tokyo Disneyland Dataset*.

(a) −60° (b) −30° (c) 0° (d) 30° (e) 60°

Figure 5.13: Labeling of the *Tokyo Disneyland Dataset* in AR. The images present results of different viewing angles to show consistent label positions when the AR device is rotated toward different directions by the user.

Figure 5.14 presents a tilted device displaying the *Local Shops Dataset*. The tilt does

not influence the placement of the labels in general, the *Local Shops Dataset* is used to illustrate it. The labels stay perpendicular to the ground.

Figure 5.14: Tilted device showing the *Local Shops Dataset*. The labels stay perpendicular to the ground.

In addition to viewing angle changes of the device, coherent results are expected when the positions of labels change to be occlusion-free during the interaction with the system, if LODs of labels change, and if labels are aggregated to super labels. This section illustrates the transition from a super label to the individual labels as it presents both position and LOD changes. Figure 5.15 depicts an explanatory transition of a super label to the individual labels over time. In this example of the *Tokyo Disneyland Dataset*, the currently visible themed area of the amusement park is called *Westernland*. The image on the top left of the figure presents the initial state including the super label for this area. The three representative images of the *Westernland* that are included in the super label are *Splash Mountain*, *Mark Twain's Riverboat*, and *Big Thunder Mountain*. The color coding is based on the average waiting times of all attractions located in this themed area. If the user is not inside the area, the labels are aggregated into the super label. If the user is moving closer to the area, the super label is split up into the individual labels again to show the attractions and the details of the area. Figure 5.15 includes time steps of the transition from the super label of the area to the individual labels. The top left image presents the initial state and the bottom right image highlights the final state of this particular example. The columns from left to right and the rows from top to bottom depict the process over time. The transition consists of 100 steps to ensure smooth label movements as explained in Section 3.6. The resulting transition for the *Westernland* can be seen in Figure 5.15. To provide coherency during the transition, the individual labels travel from the position of the super label to their respective positions. At the beginning of the transition, the transparency of the super label is increased while the transparency

values of the individual labels are decreased to help users to identify that the attraction labels belong to the super label. During the transition, the individual labels travel to the occlusion-free positions.

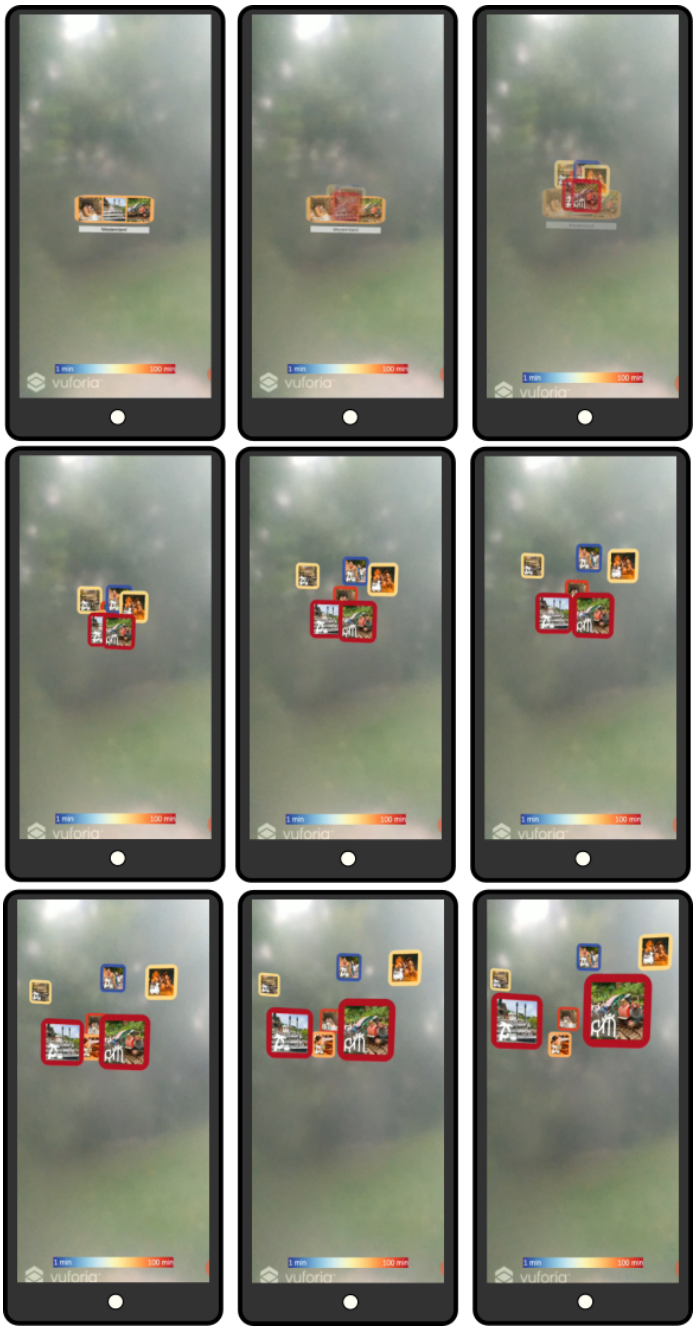
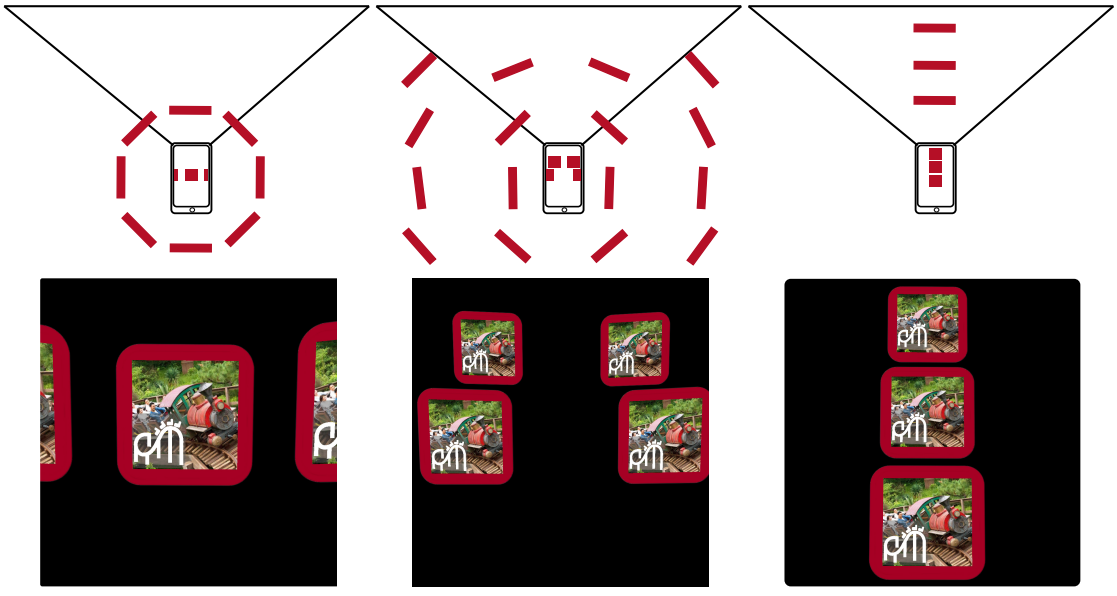


Figure 5.15: Transition from a super label to the individual labels over time.

5.5 Performance Results

We investigate the performance of our approach by analyzing three different label layouts in the *Synthetic Dataset*. The execution time is measured on the mobile device Xiaomi Mi A2. The three layouts are a circle layout, a grid layout, and a line layout. If more labels are hidden from another in the current viewing direction, more occlusion removal steps are necessary.

The circle layout (Figure 5.16(a)) requires the least computation times to resolve occlusions since many labels are initially arranged without occlusions. The grid layout (Figure 5.16(b)) distributes the labels equally leading to densely placed labels in the input scene with an increasing label number. The number of labels per row is equal to $\sqrt{n}$, where n is the number of labels mentioned in Figure 5.17. The line layout (Figure 5.16(c)) represents the computationally most expensive case for the *Occlusion Management*. The labels are located behind each other, which leads to the maximum number of $i - 1$ label shifts for each label l_i. Figure 5.17 gives a summary of the computation times of each layout in milliseconds based on a variation of label numbers. As shown in Figure 5.17, resolving occlusions in the grid layout leads to higher computation times than the circle layout and lower computation times compared to the line layout. The computation times are measured on the mobile device.



(a) (b) (c)

Figure 5.16: An example of the *Synthetic Dataset* in top view with the displayed results beneath. Labels are arranged in a circle (a), in a grid (b), and in a line (c).

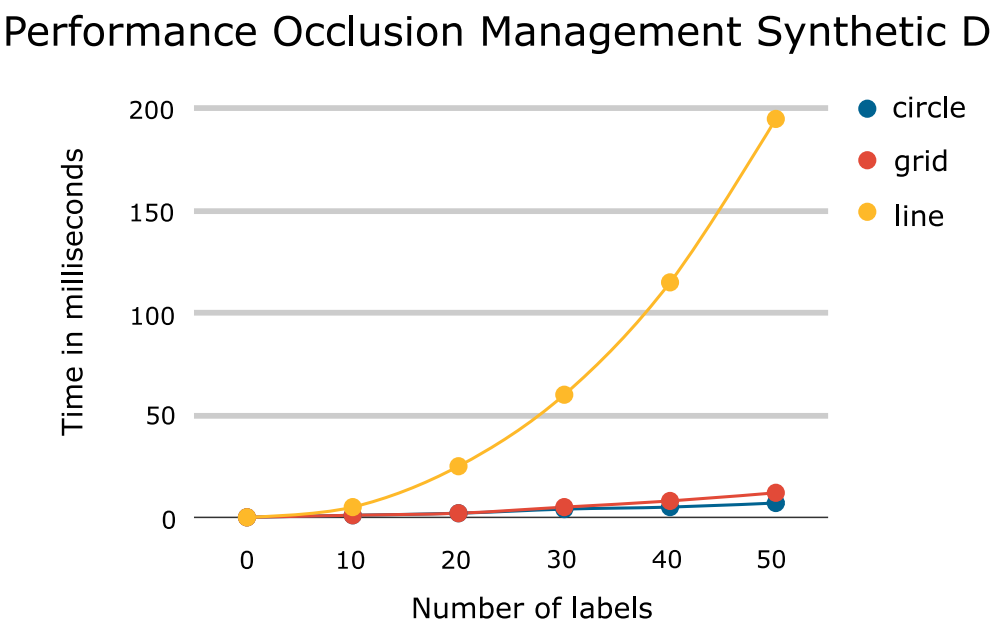


Figure 5.17: Computation times for resolving occlusions in the *Synthetic Dataset*.

CHAPTER 6

Frameworks and Implementation Details

Our project is implemented in C# supported by Unity [Tec20a] and the Vuforia framework [PTC20] for AR is used. Section 6.1 describes these frameworks while Section 6.2 explains the implementation details of our project. In contrast to Chapter 3, this chapter focuses on the technical details and highlights mathematical examples.

6.1 Frameworks

Unity is an IDE by Unity Technologies [Tec20a] supporting the implementation of platform-independent applications focusing on 3D applications. The main programming language of Unity is C#. The IDE allows programmers to deploy projects for several different platforms like Microsoft Windows, macOS, Android, Nintendo Switch, and more.

As stated on the official Unity website accessed on 02-June-2020 [Tec20a], the main areas Unity is used for are

- Game development,
- Automotive, transportation, and manufacturing,
- Film, animation and cinematics and
- Architecture, engineering, and construction.

According to the Unity website [Tec20b], 50 % of all games are based on Unity [Tec20b] (2020) including PC, console, and mobile games. Apps created in Unity are downloaded

three billion times each month [Tec20b]. This was one of the reasons why we selected Unity for implementing the project.

Furthermore, Unity supports implementing VR and AR applications. Different frameworks expand the functionality of Unity, especially for AR, which was the primary reason, why Unity was selected for implementing the project presented in this thesis. The deployment process for mobile devices is done by Unity, which eases the development. Initial configurations (for Android in our case) need to be done at the beginning of the implementation phase. After the initial configurations, the deployment for mobile devices works automatically.

According to Statista [Sta20], around 75 % of mobile devices ran Android in 2019. Therefore, we implemented the project for Android devices. Using Unity, the project could be reconfigured to work on Apple devices, but it requires tools that only work on Apple devices for the implementation as well.

We use Vuforia as the AR framework for this project. Vuforia can be executed on many devices in contrast to other AR frameworks that require mobile devices to support AR Core (for Android devices). AR Core is supported by a limited number of mobile devices. This is the main reason why we use Vuforia for this project to profit from the many supported devices.

Vuforia is an SDK providing the *"fastest, easiest, and most advanced AR content development solutions"* (Vuforia Hompage accessed on 02-June-2020 [PTC20]). It uses computer vision to extract the necessary information for displaying and positioning objects in AR. One major benefit is the possibility of using markerless AR, meaning that the system does not need markers placed in the scene to position and render objects. The location of the objects in our project (labels) is dependent on the position and viewing angle of the user, thus independent of specifically designed markers, which is another reason why Vuforia is used for this project. The markerless approach allows the labels to be positioned at the respective location independent of the viewing angle of the user's device.

6.2 Implementation Details

The techniques and frameworks described in the previous section were used to realize each of the proposed management strategies of the pipeline (see Figure 6.1). The implementation details are described in this section. The right column in Figure 6.1 mentions the employed frameworks and techniques. For more detailed methodological explanations and results see Chapter 3 and Chapter 5. Basically, the algorithm initially positions labels in AR. Unity, Vuforia, and the AR GPS + Location [For20] frameworks support the initial placement. After this step, the labels have received their initial 3D, AR coordinates in the Unity world space. In the *Occlusion Management*, label occlusions are resolved. Each individual label receives one of three LODs (*Level-of-Detail Management*) providing a color-coded rectangle, a type icon, an iconic image, and a text

tag as presented in Section 3.1. Position changes and LOD changes of labels are smoothly resolved in the *Coherence Management*.

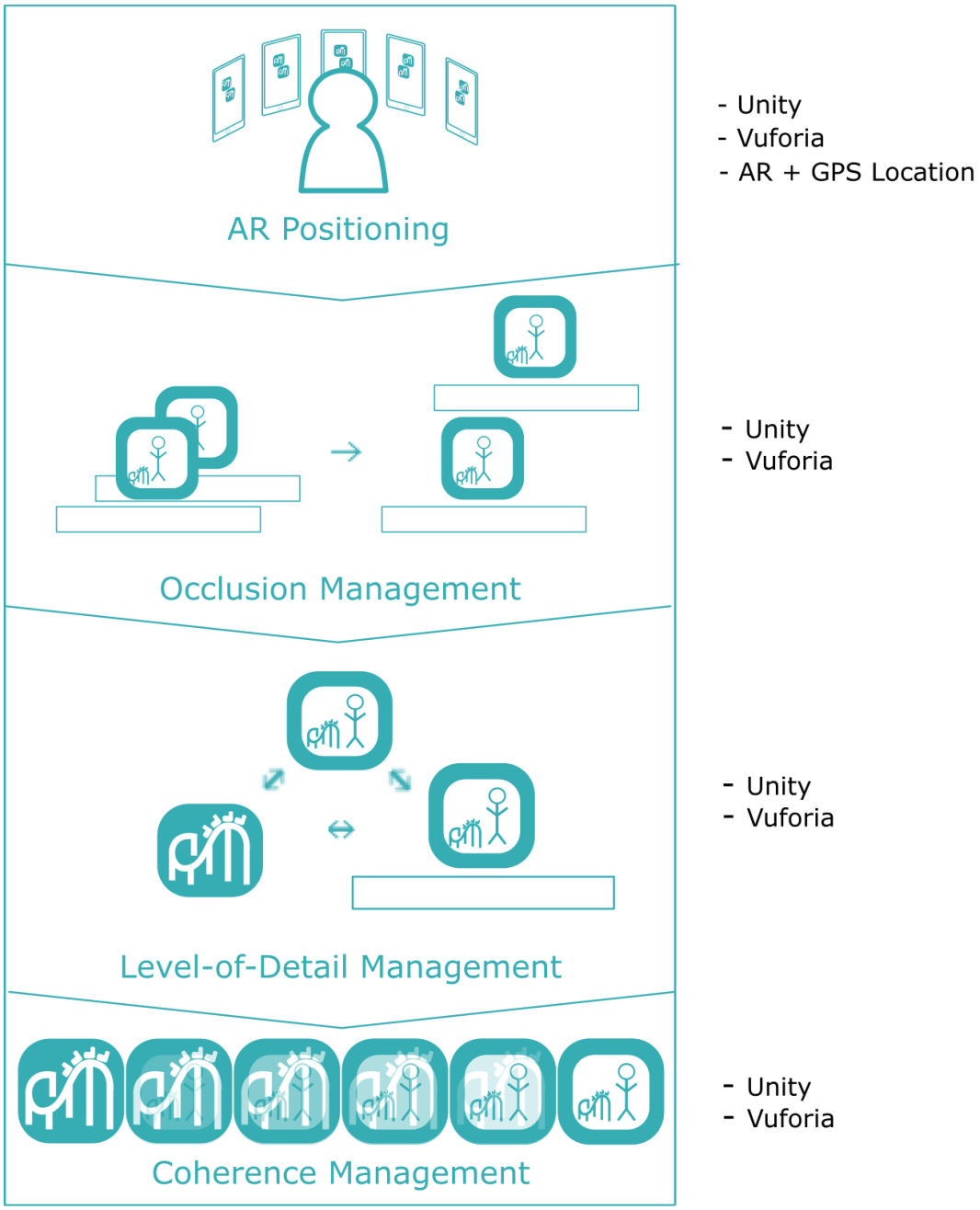


Figure 6.1: The proposed pipeline of our project linked to the frameworks that were used for implementing.

6.2.1 AR Positioning

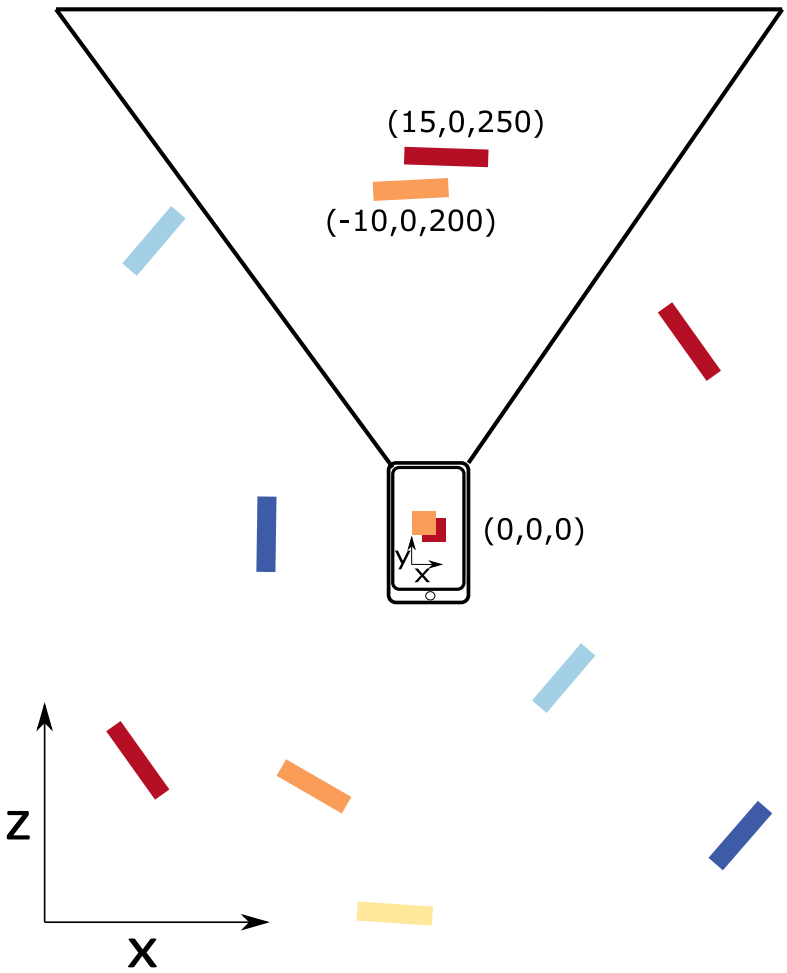


Figure 6.2: Results of the *AR Positioning*, in top view. The viewing direction of the device is the z-direction. The black cone represents the current view volume.

The *AR Positioning* describes the initial placement of the labels in AR. Each label represents a POI. The labels are placed in Unity's world space. The position of the user is in the center of the coordinate system $(0, 0, 0)$. Every label is oriented to the user, rotated around the y-axis (upward axis). Each label is oriented in such a way, that the normal vector of the label plane is oriented towards the position of the user, which is always $(0, 0, 0)$ in Unity's world space in the project. Figure 6.2 provides an explanatory initial placement. The scene is viewed from the top. The viewing direction of the device of the user is the z-direction in this example. The display in the figure shows the currently visible labels that are included in the black cone illustrating the current view volume.

Furthermore, we included the AR + GPS Location framework by Daniel Fortes [For20],

which is a Unity framework supporting the positioning of objects according to GPS coordinates. Like in our implementation, the framework keeps the position of the user at the center $(0, 0, 0)$ and shifts the objects based on the position of the user and the GPS coordinates of the objects.

Unfortunately, the available devices and techniques for computing the GPS positioning and the compass orientation do not provide the required accuracy (see Chapter 8). We tested the project on a Xiaomi Mi A2 smartphone and a more accurate Google Nexus C tablet, which provides ground plane support for AR, but the results are still not stable enough to avoid inaccuracies, frequent position changes of labels, and flickering. For this reason, we use virtual navigation to show the functionality of the tool. The virtual navigation considers the current orientation of the mobile device. By clicking onto the screen, the user virtually walks into the current viewing direction. This method avoids frequent position and viewing angle changes of labels caused by inaccurate GPS position and compass data to highlight the coherent functionality of the tool. To improve the accuracy of the positioning and the compass orientation, a collaboration with researchers who investigate high-precision positioning tools for mobile devices would be interesting in the future as stated in Chapter 9.

6.2.2 Occlusion Management

The main steps of the *Occlusion Management* are the *Distance Sorting*, *Occlusion Detection*, and *Shift Computation*.

Distance Sorting

Close labels are considered first when occlusions are resolved, which leads to the behavior that close labels are more likely to stay at the initial positions computed by the *AR Positioning*. The labels are ordered based on the distance to the user in world space. The center point of the label and the position of the user $(0, 0, 0)$ are considered for this calculation. The vector pointing from the position of the user to the center of each label is calculated. The length of this vector is measured to sort the labels based on the distance to the user. Figure 6.3 illustrates labels around the user in top view. The arrows indicate the computed vector from the position of the user to the labels. The number at each label shows the order of the labels - in this example with the closest being Number 1 and the farthest being Number 6. This ordering is used as one important decision factor in the *Occlusion Management* and the *Level-of-Detail Management*.

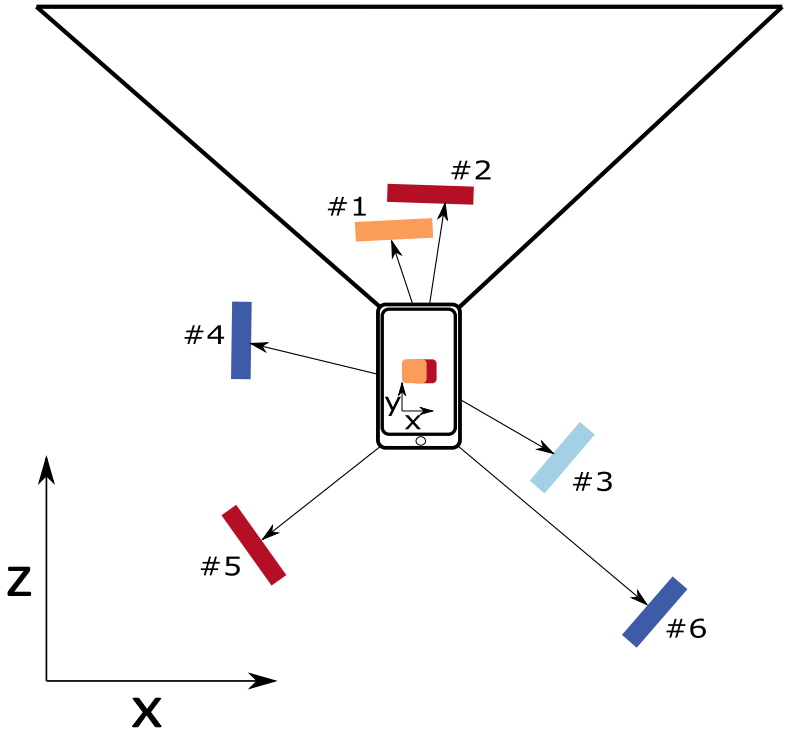


Figure 6.3: Results of the *AR Positioning* in a top view. The number of each label is the resulting position in the ordered list after the *Distance Sorting*.

Occlusion Detection and Shift Computation

During the *Occlusion Management*, the labels are considered according to the distance to the user starting with the closest label and finishing with the farthest one. Four rays are traced from the user to the four corner points of each label. These rays are traced. If another label is hit by one of the rays, the position of label l_i needs to be changed to resolve the occlusion. The detailed methodological explanation of this approach can be seen in Chapter 3.

Once an occlusion of one of the four rays with another label is detected, it is resolved by shifting label l_i upwards to be visible above the occluder. A ray is constructed from the user to the top of the occluding label (Figure 6.4). This ray is extended until it hits the plane of the occluded label. The intersection between the ray and the plane of the occluded label l_i is the lowest boundary for the occludee position to be visible. The occludee (label l_i) is shifted above this intersection point to resolve the occlusion. Figure 6.4 contains the coordinates and height values for each label in this example. Both label l_1 (orange) and the label l_2 (red) have a height of 120 units in Unity's world space in this example. The closer label l_1 is positioned at $(-10, 60, 200)$ and the distant label l_2 at $(15, 60, 250)$. These coordinates are the center points of each label. Label l_1 occludes the label l_2 in this example. To resolve the occlusion, a ray is constructed leading from

the user to the top of the occluding label. This is the gray ray in Figure 6.4. The ray is now traced until it hits the extended 2D plane of the occluded label l_2. The intersection point is located at $(15, 125, 250)$. Label l_2 needs to be positioned above this point to be completely visible. Label l_2 is shifted to $(15, 185, 250)$ and is now displayed above the label l_1 in the 2D projection on screen, which resolves the occlusion in this example.

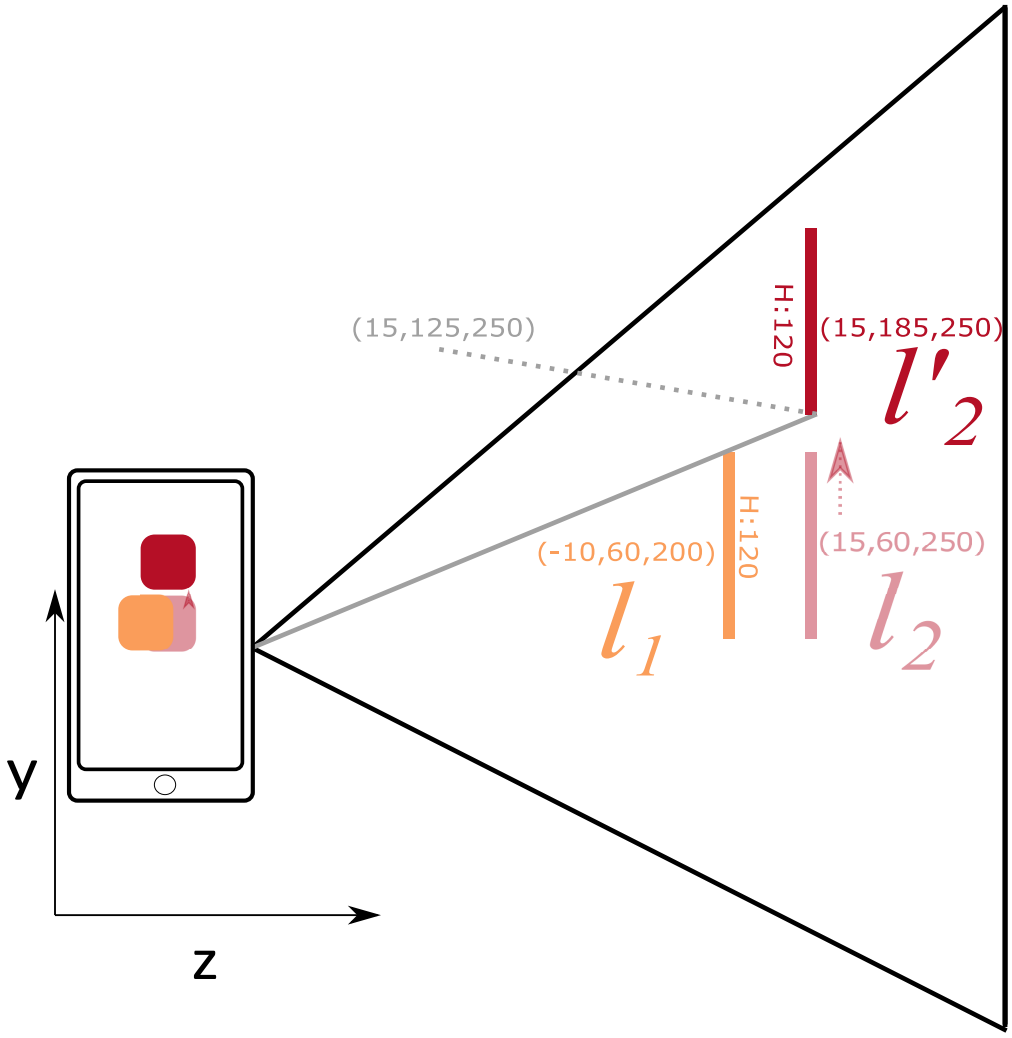


Figure 6.4: The occlusion is resolved by shifting the occluded label upwards.

6.2.3 Level-of-Detail Management

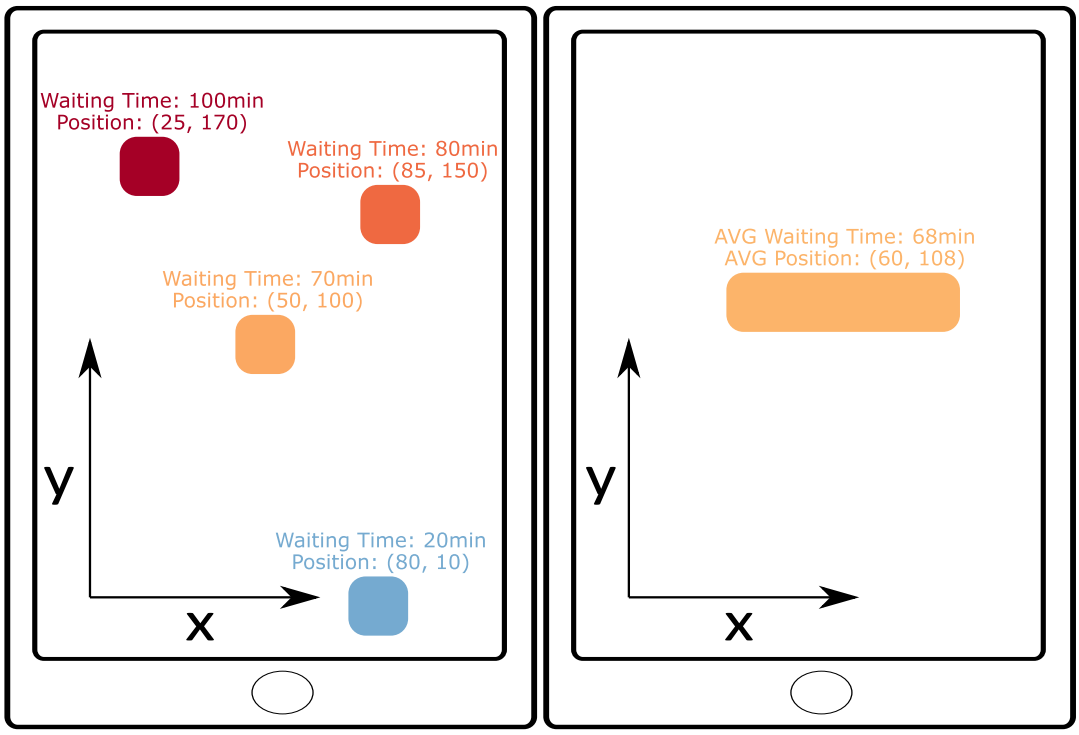
The *Level-of-Detail Management* calculates an LOD for all individual labels or aggregates a set of labels to super labels. Based on the distance to the user and the label density around the user, the LOD for each label is selected. The methodological principles behind the *Level-of-Detail Management* are explained in Chapter 3.

Level-of-Detail Computation

Starting with the first label of the before-explained ordering, the labels receive an LOD for the current position of the user's device. One of three LODs for individual labels is selected (see Chapter 3).

The labels are considered one after another based on the calculated ordering. The closest visible label is forced to be displayed in the *Highest LOD* to provide detailed information for at least one POI. For each following label, two vectors are constructed. One vector is pointing from the user's position ($(0, 0, 0)$ in Unity's world space) to the (x, z) position of label l_i along the ground plane. The second vector is pointing from the user's position to the position of label l_i. The angle between these two vectors is computed. Depending on the angle, an LOD for label l_i is selected. We currently use 0° to 20° for the *Highest LOD*, 21° to 30° for the *Middle LOD*, and angles higher than 30° for the *Lowest LOD*. These values were found heuristically providing a balanced distribution of each LOD.

Aggregation to Super Labels



(a) Individual labels (b) Super label

Figure 6.5: Individual labels (a) and the super label (b) of this example. For simplification, this example illustrates a 2D situation in contrast to the implemented 3D solution.

In the *Level-of-Detail management*, individual labels might be aggregated to super labels. The super labels show the color-coded average value of the contained individual labels (the waiting time in the *Tokyo Disneyland Dataset*). The position of the super label is calculated as an average of the positions of the contained individual labels.

Figure 6.5 provides a simplified 2D example of the super label aggregation. The left part contains four labels, each with a certain waiting time at a certain position. These four labels are aggregated to the super label presented in the right part of the figure. The (x, y) position of this super label is the average of all x and y values of the individual labels in Figure 6.5. The color coding is dependent on the average waiting times of all the individual labels from the respective super label, which is 68 minutes in this example.

6.2.4 Coherence Management

Position changes or LOD changes of labels calculated in the *Occlusion Management* and in the *Level-of-Detail Management* are smoothly resolved in the *Coherence Management*.

Smooth Occlusion Transitions

If occlusions occur, they are resolved as explained in the previous subsection or in Chapter 3. The positions of labels need to be adapted in a smooth way while occlusions are visually resolved to provide coherent label positions for the user to avoid flickering and abrupt movements of the labels.

In case of a position update, the label stays at the previous position. This position is set as the initial position and the newly calculated one is set as the goal position. Based on these two 3D positions and the maximum of two seconds for an update, the partial movement for each update iteration of the labels can be calculated (see Algorithm 6.1).

This direction vector between the previous position and the goal position is divided by the number of frames in two seconds to match the previously described threshold of two seconds ($timeForTransition$). In each update iteration, the positions of all labels are changed slightly with respect to the calculated vectors. If the user does not move, the system does not recalculate the positioning to save computation time as occlusions are resolved globally for each label around the user considering his or her location. In doing

so, viewing angle changes of the device do not influence the occlusion-free label positions.

Algorithm 6.1: One position update iteration for the *Smooth Occlusion Transitions.*

Data: Labels

Result: One position update for labels

```
1 S = list of sorted labels;
2 for l_i ∈ S do
3 |   partialMovement = (l_i.goalPosition - l_i.previousPosition) / timeForTransition;
4 |   l_i.position += partialMovement;
5 end
```

Smooth LOD Transitions

Due to the dynamic behavior of the tool caused by the interactions with AR, the LOD for each label changes during the exploration and navigation process. To provide coherence, we adapt the displayed LOD of each label in a smooth way. If a label changes the LOD from one step to another, the system initializes a smooth transition from one state to the other one. Between each iteration, the labels change the displayed LOD linearly via transparency changes. More technically, each `GameObject` (each object in Unity is a `GameObject`) has a `color` attribute either in a separate `Renderer` or in the `Image` itself. The transparency value of this `color` attribute can be set to blend in or blend out `GameObjects`. Depending on the current and on the destined LOD, different elements are blended in and blended out. If a label is currently displayed in the *Middle LOD*, the colored rectangle, the type icon, and the iconic image are visible. If the label shall be displayed in the *Highest LOD* in the next step, the transparency of the text displaying the attraction name and waiting time is decreased to blend in the text tag. Between the three LODs, we provide six different state changes.

Figure 6.6: State change in LOD between the *Lowest LOD* and the *Highest LOD*.

Algorithm 6.2: Smooth LOD change of a label

Data: Current and destined LOD for a label, currentLabel

Result: LOD for each label

```
1  if firstUpdateStep then
2  |   currentLabel.currentChangePercentage=0.0f;
3  if currentLabel.LOD = Lowest And currentLabel.goalLOD = Highest then
4  |   currentLabel.icon.transparency = currentChangePercentage;
5  |   currentLabel.iconicImage.transparency = 1.0f - currentChangePercentage;
6  |   currentLabel.smalltypeIcon.transparency = 1.0f - currentChangePercentage;
7  |   currentLabel.textTag.transparency = 1.0f - currentChangePercentage;
8  ... ifs for the other possible combinations of LOD state changes ...
9  currentLabel.currentChangePercentage+=0.01f;
10 repeat in each update iteration;
```

Figure 6.6 illustrates LOD changes from the *Lowest LOD* to the *Highest LOD*. In this case, many items change the transparency. The *Lowest LOD* depicts the colored rectangle encoding the waiting time and an icon. During the smooth transition from the *Lowest LOD* to the *Highest LOD*, this icon becomes gradually invisible while the iconic image, the small type icon in the iconic image, and the text tag beneath become gradually more visible. Algorithm 6.2 explains this transition from the *Lowest LOD* to the *Highest LOD*. This algorithm provides a basic example of a single update step during the smooth state changes. In each update step, the transparency values are decreased or increased accordingly to achieve a smooth transition when labels dynamically change the LOD during the program execution. After the final execution of Algorithm 6.2, the label is displayed in the destined LOD.

Smooth Aggregation Transitions

Individual labels might be aggregated to super labels. To provide coherence, the displayed aggregation is implemented in a smooth way. The individual labels travel to the position of the corresponding super label and are blended out via transparency when they are aggregated. The super label is blended in via transparency, simultaneously.

During the aggregation, the individual labels move a little closer to the calculated super label position in each update iteration, while the transparency values of the individual labels are slightly increased and the transparency of the super label is slightly decreased. If the aggregation is split up again, the whole process takes place in a reversed order.

The smooth transitions for the super label aggregation are a combination of the beforementioned *Smooth Occlusion Transitions* and the *Smooth LOD Transitions*.

CHAPTER 7

Evaluation

We conducted an online survey to evaluate the effectiveness of our system. 13 participants who are experienced with visualization techniques and graduate students of visual computing participated in the survey. The age of the participants ranges from 24 to 64 years with the majority of participants being in the late twenties or the early thirties. The hypotheses we investigated are the following:

(H1) The occlusion-free label placement eases the information retrieval process.

(H2) Our LOD design in AR leads to a better exploration and selection of POIs in contrast to plain text labels.

(H3) Users can perform common route planning tasks faster using our system in comparison to conventional 2D maps.

The hypotheses **(H1)** to **(H3)** are linked to the problems **(P1)** to **(P3)**.

We conducted an online survey to assess the effectiveness of our tool. The survey is implemented as a Google Web App [LLC19] and Materialize [Mat20] is used for the CSS design. The user input is stored in a Google Sheet [LLC20]. The result of each participant is stored in a separate Google Sheet table. All participants agreed that personal information will be gathered including education levels, age, and experience with visualization, info-vis, and amusement parks. The data is treated strictly confidential and may be used in reports, presentations, and publications, but we as the researchers will not be able to identify the participants based on the collected data.

The survey is conducted using the *Tokyo Disneyland Dataset* and consists of four main tasks of which each question requires participant input. Time and accuracy are measured and general feedback is collected at the end of each task. Table 7.1 summarizes the tasks

Tasks	Investigation
Task 1	Influence of partially occluded and occlusion-free labeling
Task 2	Adoption of Levels-of-Detail
Task 3	Effectiveness of 2D maps and our AR encoding
Task 4	Combinatorial features in our system

Table 7.1: Overview of the tasks in the user study.

in the user study. We perform the tasks sequentially, while the questions in each task are randomized to avoid a learning effect.

(H1) presents the importance of resolving the label occlusions in an AR environment. As described in Chapter 2, existing work concludes the importance of solving occlusions in AR to support the decision making process by users [GLK+12]. In task 1, we show participants a few images synthesized using our system. Participants are asked to detect the waiting time of specified attractions and then select the attractions with minimal waiting times. None of the participants managed to select the correct waiting times if occlusions occurred. Participants stated that the waiting times were not recognizable if occlusions are present. 92.3 % of the participants could successfully identify minimal waiting times if the occlusions were resolved. When we asked the participants to select attractions with minimal waiting times, no participant was able to select the correct attractions if occlusions were present. Four participants guessed the attractions and nine stated that they could not recognize them. If the occlusions were resolved, all participants selected the respective attractions with minimal waiting times. 12 participants explicitly stated that it was difficult or impossible to select the correct answers if information is occluded, and 11 participants agree that the occlusion-free positioning eases the decision-making processes when investigating the labels.

To investigate hypothesis **(H2)**, we design task 2. Participants need to take several labels into account to answer the questions. The participants are asked to select a themed area of the amusement park with the lowest average waiting time. In this task, we presented participants a set of images with labels of different themed areas. We prepared three settings here, where we only show text labels, icon labels, and super labels, individually. If the waiting time information is depicted using text labels or icon labels, the accuracy of the answers differs from 76.9 % to 69.2 % of the participants who selected the correct answers. 92.3 % of the participants selected the correct answers if the super labels were shown.

The time needed to answer questions in task 2 is summarized in Figure 7.1. Each line corresponds to one participant. The measured data points for each user are connected to indicate how the necessary time to answer the questions changes if text labels, icon labels, or super labels were shown. The x-axis encodes three label categories, and the y-axis represents the completion time in seconds.

The participants, in general, spent more time if only text labels are present, with an average of 72.7 seconds since they probably needed to calculate the correct number to answer the questions properly. If we present information using icons, a shorter time, 23.4

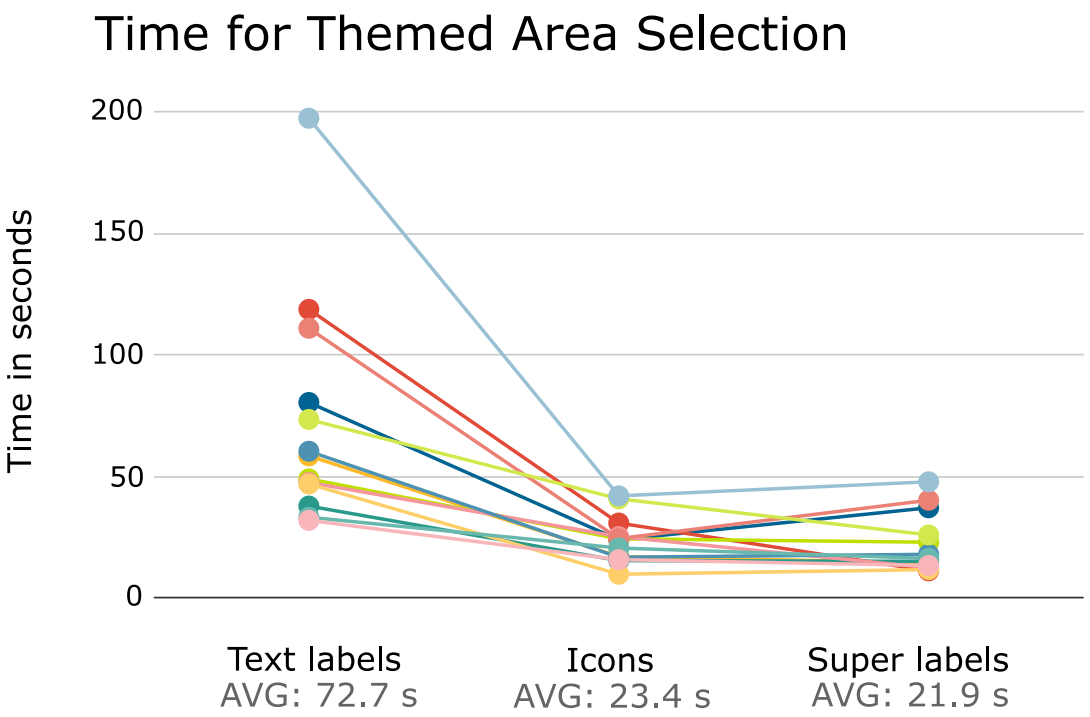


Figure 7.1: Time needed by the participants to solve task 2. Each line represents one participant.

seconds on average, was required in comparison to pure text labels. If we present the average time using super labels, participants needed only 21.9 seconds to answer the questions.

For hypothesis **(H3)**, we investigate and compare the decision making effectiveness when using 2D maps and our AR encoding in task 3. We again compare the task completion time and accuracy between the Tokyo Disneyland map and our visualization. We removed other POIs than the 35 main attractions from the map, like shops or restaurants, to increase the comparability.

76.9 % of the participants selected attractions with minimal waiting times of a themed area when using the 2D map and 84.6 % if the AR encoding was presented. The average time that the participants needed to select an attraction using the 2D map was 71.6 seconds while they spent 26.7 seconds on average when using the proposed approach, which clearly shows a reduced effort for the POI selection process.

Besides the tasks connected to the aforementioned hypotheses, we also include task 4 to demonstrate the appropriateness of the design choices and general feedback of features in our system. In this task, we present six videos with different settings to the participants. The videos include (1) non-occlusion-free labeling, (2) occlusion-free labeling, (3) all labels in *Highest LOD*, (4) all labels in *Lowest LOD*, (5) *Dynamic LODs*, (6) *Dynamic*

LODs with super labels.

The *Dynamic LODs* as computed according to the *Level-of-Detail Management* were chosen as their favorite LOD strategy by 53.8 % of the participants. 38.5 % of the participants preferred the *Highest LOD*. Participants, who selected the *Dynamic LODs* as their favorite design, emphasized that they prefer *Dynamic LODs*, as the vertical stacking of labels is reduced while detailed information about close attractions is presented. However, the participants who chose the *Highest LOD* as their favorite design appreciated the detailed information that can be used for decision making. It surprised us that they were not disturbed by the high vertical stacking of the labels, which occupies limited screen space. The *Dynamic LODs*, however, reduce vertical stacking of labels while presenting more information about close labels and less information about far labels.

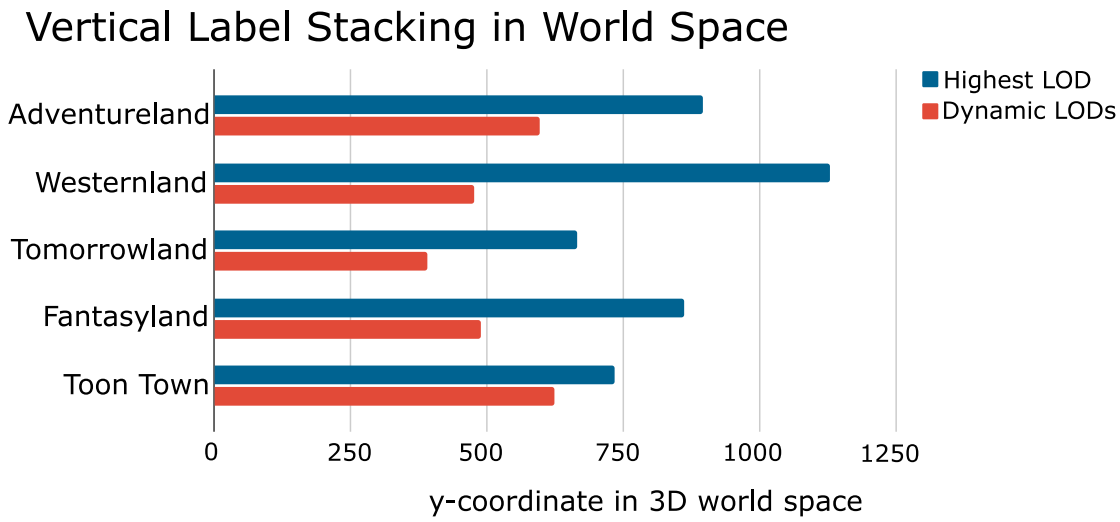


Figure 7.2: y-coordinates of the highest shifted label after the *Occlusion Management* for the *Highest LOD* and the *Dynamic LODs* as proposed by the *Level-of-Detail Management*. The lower the y-coordinate, the smaller is the required shift, which represents lower vertical stacking of the labels.

Figure 7.2 depicts a comparison of the *Highest LOD* and *Dynamic LODs*. The more information is included for each label, the higher the labels need to be shifted upwards leading to higher vertical stacking of the labels. The y-coordinate of the highest label of the two methods is used as a representative value for each themed area in Figure 7.2. The vertical stacking of the labels can be effectively reduced using the *Dynamic LODs* as proposed in the *Level-of-Detail Management*.

Two participants mentioned that they prefer 2D maps compared to AR since 2D maps show the global scenario. Both the 2D map and the AR encoding have strengths and weaknesses. Two participants suggested combining the 2D map and the AR encoding if the tool would be used for route planning. Two participants stated that they would prefer the super labels combined with the *Highest LOD*, which would be a possibility to

reduce visual clutter but this could potentially lead to a larger vertical stacking of labels compared to the *Dynamic LODs*.

The occlusion handling and the smooth transitions incorporated in the approach were mentioned by participants in the general feedback. For example, *"I really like the Occlusion Management, to my eyes, it's almost seamless."* or *"Active occlusion handling is much superior to no occlusion handling."*. Another popular feature that was mentioned specifically was the super label aggregation. Participants appreciate the overview of each of the areas by giving feedback like, *"I like the super label transitions if there are many attractions because it gives a good overview of an area."*, or *"I like the super label transitions the most."*.

CHAPTER 8

Limitations

One limitation of the system is inherited from the accuracy of mobile GPS. The position and especially the rotation of the available Xiaomi Mi A2 smartphone and the Google Nexus C tablet are not consistent. This leads to positioning inconsistencies of labels as the returned data of each of the two devices is not stable. Depending on the distance of the user to the attraction, the label placement accuracy is influenced by the hardware accuracy. This limitation unfortunately limits the capability to fully utilize the application.

However, we envision this will sooner or later be solved by scientists in the field. For this reason, we present the results in this work using predefined label positions in 3D world space in AR. It is because frequent changes of device position and viewing angle influence the coherent label positioning of our tool. It would be interesting to collaborate with researchers focusing on high-precision GPS positioning systems.

Another limitation is the loss of the global overview when using AR compared to 2D maps as already pointed out by related work [GLK+12, BFH01, GM19]. The user needs to interact with the system and look in different directions to see all the labels. The AR view only depicts the labels that are currently in front of the user in the respective view volume. We could introduce additional labels on the sides of the screen to eliminate this drawback.

CHAPTER 9

Conclusion and Future Work

In summary, we present a context-responsive labeling framework in Augmented Reality, which allows us to introduce rich-content labels associated with POIs. The label management strategy suppresses label occlusion and incoherent label movements caused by the transitions and rotations of the device during user interaction. It presents an alternative approach for spatial data navigation. The *Occlusion Management* resolves label occlusions. The *Level-of-Detail Management* takes the position of the user and label density in the view volume into account, and thus, the computed Levels-of-Detail for each label decrease vertical stacking of labels, while still retaining basic information based on the object distance. To further reduce visual clutter, we introduce the concept of super labels, which group a set of labels. Smooth transitions have been implemented in the *Coherence Management* to avoid flickering and enable stable label movement. The evaluation shows the applicability of the proposed approach.

As our future direction, techniques will be investigated to overcome the drawbacks of seeing only the labels that are in the current view volume of the user's device, so that the user can still anticipate POIs outside the view volume and retain the global overview of the annotated scene as 2D maps. One possibility would be including the technique presented by Lin et al. [LLT+17] to depict labels that are currently outside the view volume at the border region of the display of the device. Considering the positioning accuracy, it would be interesting to include Dual-Frequency GPS [EER19] or Continuous Operating Reference Stations (CORS) [DDP19] as investigated by related work to improve sensor accuracy of mobile devices [UB20, KGR20, Bow19].

List of Figures

List of Tables

List of Algorithms

Bibliography

[BFH01] Blaine Bell, Steven Feiner, and Tobias Höllerer. View management for virtual and augmented reality. In *Proceedings of the 14th annual ACM symposium on user interface software and technology*, pages 101–110, 2001.

[BMWW14] Michael Birsak, Przemyslaw Musialski, Peter Wonka, and Michael Wimmer. Automatic generation of tourist brochures. *Computer Graphics Forum*, 33(2):449–458, 2014.

[BNN19] Michael A Bekos, Benjamin Niedermann, and Martin Nöllenburg. External labeling techniques: A taxonomy and survey. *Computer Graphics Forum*, 38(3):833–860, 2019.

[Bow19] David S Bowers. Augmented reality smartphone compasses: opportunity or oxymoron? In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, pages 13–16, 2019.

[CMS94] Jon Christensen, Joe Marks, and Stuart Shieber. Placing text labels on maps and diagrams. *Graphic Gems*, 4:497–504, 1994.

[CPWN20] Ladislav Cmolik, Vaclav Pavlovec, Hsaing-Yun Wu, and Martin Nöllenburg. Mixed labeling: Integrating internal and external labels. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–14, 2020.

[CY14] Pio Claudio and Sung-Eui Yoon. Metro transit-centric visualization for city tour planning. *Computer Graphics Forum*, 33(3):271–280, 2014.

[DDP19] Paolo Dabove and Vincenzo Di Pietra. Towards high accuracy GNSS real-time positioning with smartphones. *Advances in Space Research*, 63(1):94–102, 2019.

[Dis20] Tokyo Disneyland. Tokyo Disney Resort Official Website. `https://www.tokyodisneyresort.jp/en/index.html`, 2020. [Online; accessed 01-November-2020].

[EER19] Abdelsatar Elmezayen and Ahmed El-Rabbany. Precise point positioning using world's first dual-frequency gps/galileo smartphone. *Sensors*, 19(11):2593, 2019.

[For20] Daniel Fortes. AR + GPS Location. `https://assetstore.unity.com/packages/tools/integration/ar-gps-location-134882`, 2020. [Online; accessed 09-June-2020].

[GASP08] Floraine Grabler, Maneesh Agrawala, Robert W Sumner, and Mark Pauly. Automatic generation of tourist maps. *ACM Transactions on Graphics (TOG)*, 27(3):1–11, 2008.

[GLK+12] Raphael Grasset, Tobias Langlotz, Denis Kalkofen, Markus Tatzgern, and Dieter Schmalstieg. Image-driven view management for augmented reality browsers. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 177–186, 2012.

[GM19] Renan Luigi Martins Guarese and Anderson Maciel. Development and usability analysis of a mixed reality gps navigation application for the microsoft hololens. In *Proceedings of the Computer Graphics International Conference*, pages 431–437, 2019.

[Gmb20] Wikitude GmbH. Wikitude Cross Plattform Augmented Reality SDK. `https://www.wikitude.com/products/wikitude-sdk/`, 2020. [Online; accessed 25-October-2020].

[Gra78] Henry Gray. *Anatomy of the human body*, volume 8. 1878.

[HLL20] Jane Hoffswell, Wilmot Li, and Zhicheng Liu. Techniques for flexible responsive visualization design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, page 1–13, 2020.

[II19] Tomoyuki Ishida and Hayato Ito. Construction of a historical information visualization system by superposition of old maps. In *Proceedings of the International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 310–318, 2019.

[Ins19] National Eye Institute. Color Blindness. `https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/color-blindness`, 2019. [Online; accessed 10-October-2020].

[JZWG18] Jianqing Jia, Yu Zhang, Xiaoce Wu, and Wei Guo. Image-based label placement for augmented reality browsers. In *Proceedings of the 4th International Conference on Computer and Communications (ICCC)*, pages 1654–1659, 2018.

[KG19] Pontus Knutsson and Oskar Georgsson. Augmented reality navigation compared to 2d based navigation. 2019.

[KGR20] Tim Kuhlmann, Pablo Garaizar, and Ulf-Dietrich Reips. Smartphone sensor accuracy varies from device to device in mobile research: The case of spatial orientation. *Behavior Research Methods*, 2020.

[LGZ+19] Yan Liu, Chi Guo, Manli Zhang, Xiaoqun Wu, and Ye Zhou. How to tour Shanghai Disneyland Park. *Journal of Internet Technology*, 20(1):25–38, 2019.

[LLC19] Google LLC. Web Apps. `https://developers.google.com/apps-script/guides/web`, 2019. [Online; accessed 26-August-2020].

[LLC20] Google LLC. Google Tabellen: Kostenlos Tabellen online erstellen und bearbeiten. `https://www.google.com/intl/de_at/sheets/about/`, 2020. [Online; accessed 26-August-2020].

[LLT+17] Yung-Ta Lin, Yi-Chi Liao, Shan-Yuan Teng, Yi-Ju Chung, Liwei Chan, and Bing-Yu Chen. Outside-in: Visualizing out-of-sight regions-of-interest in a 360 video using spatial picture-in-picture previews. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 255–265, 2017.

[LNSG14] Tobias Langlotz, Thanh Nguyen, Dieter Schmalstieg, and Raphael Grasset. Next-generation augmented reality browsers: rich, seamless, and adaptive. *Proceedings of the IEEE*, 102(2):155–169, 2014.

[Mac86] Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.*, 5:110–141, 04 1986.

[Mat20] Materialize. Materialize. `https://materializecss.com/`, 2020. [Online; accessed 26-August-2020].

[Maz09] Riccardo Mazza. *Introduction to information visualization*. Springer Science & Business Media, 2009.

[MHSG02] Kresimir Matkovic, Helwig Hauser, Reinhard Sainitzer, and M Eduard Gröller. Process visualization with levels of detail. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 67–70, 2002.

[OOK13] Hayato Ohwada, Masato Okada, and Katsutoshi Kanamori. Flexible route planning for amusement parks navigation. In *Proceedings of the 12th International Conference on Cognitive Informatics and Cognitive Computing*, pages 421–427, 2013.

[PS18] Bernhard Preim and Patrick Saalfeld. A survey of virtual human anatomy education systems. *Computers & Graphics*, 71:132–153, 2018.

[PTC20] PTC. Vuforia: Market-Leading Enterprise AR. https://www.ptc.com/en/products/augmented-reality/vuforia, 2020. [Online; accessed 02-June-2020].

[SBN+19] Matthias Schneider, Anna Bruder, Marc Necker, Tim Schluesener, Niels Henze, and Christian Wolff. A field study to collect expert knowledge for the development of AR HUD navigation concepts. In *Proceedings of the 11th International Conference on Automotive User Interfaces and Interactive Vehicular Applications: Adjunct Proceedings*, pages 358–362, 2019.

[SKM98] László Szirmay-Kalos and Gábor Márton. Worst-case versus average case complexity of ray-shooting. *Computing*, 61(2):103–131, 1998.

[SNS+08] Fumihisa Shibata, Hiroyuki Nakamoto, Ryoichi Sasaki, Asako Kimura, and Hideyuki Tamura. A view management method for mobile mixed reality systems. In *Proceedings of the 14th Eurographics Conference on Virtual Environments*, pages 17–24, 2008.

[ST15] Kkyohei Sakayori and Satoshi Takahashi. An optimal routing problem for attraction with a fast pass constraint. In *Proceedings of the International Conference on Computer Application Technologies*, pages 156–157, 2015.

[Sta20] Statista. Mobile operating systems' market share worldwide from January 2012 to December 2019. https://www.statista.com/statistics/272698, 2020. [Online; accessed 02-June-2020].

[Tec20a] Unity Technologies. Unity Website. https://unity.com/de, 2020. [Online; accessed 01-June-2020].

[Tec20b] Unity Technologies. Unity Website Our Company. https://unity.com/de/our-company, 2020. [Online; accessed 02-June-2020].

[TKGS14] Markus Tatzgern, Denis Kalkofen, Raphael Grasset, and Dieter Schmalstieg. Hedgehog labeling: View management techniques for external labels in 3D space. In *Proceedings of the 21st IEEE Virtual Reality (VR)*, pages 27–32, 2014.

[TKS13] Markus Tatzgern, Denis Kalkofen, and Dieter Schmalstieg. Dynamic compact visualizations for augmented reality. In *Proceedings of the 2st IEEE Virtual Reality (VR)*, pages 3–6, 2013.

[UB20] Marcin Uradziński and Mieczysław Bakuła. Assessment of static positioning accuracy using low-cost smartphone gps devices for geodetic survey points' determination and monitoring. *Applied Sciences*, 10(15):5308, 2020.

[vGPNB17] Mereke van Garderen, Barbara Pampel, Arlind Nocaj, and Ulrik Brandes. Minimum-displacement overlap removal for geo-referenced data visualization. *Computer Graphics Forum*, 36(3):423–433, 2017.

[WTH+13] Hsiang-Yun Wu, Shigeo Takahashi, Daichi Hirono, Masatoshi Arikawa, Chun-Cheng Lin, and Hsu-Chun Yen. Spatially efficient design of annotated metro maps. *Computer Graphics Forum (Special Issue of EuroVis 2013)*, 32(3):261–270, June 2013.