Markus Tatzgern

Situated Visualization in Augmented Reality

DOCTORAL THESIS

to achieve the university degree of
Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Prof. Dr. Dieter Schmalstieg
Graz University of Technology

Referee

Assoc. Prof. Dr. Kiyoshi Kiyokawa
Osaka University

Graz, Austria, June 2015

Abstract

A common goal of Augmented Reality (AR) is to communicate additional information about real world entities or to support certain tasks of a user. AR applications achieve this by overlaying virtual data directly on top of the real world. The overlaid information can consist of simple annotations, such as the names of the restaurants that are closest to the user. The overlays can also guide a user through maintenance and assembly tasks. In these examples, AR technology is used to visualize information regarding the surrounding real world environment. This thesis refers to this kind of AR visualization as situated visualization.

Creating effective situated visualizations is challenging. While some of the involved issues are unique to AR, others have already been encountered in related visualization areas such as information visualization and scientific visualization. Consequently, situated visualization can draw inspirations from a large body of work. This thesis identifies the issues of visualization in AR and builds connections to related areas to provide a solid foundation for situated visualization. Based on this foundation, this thesis contributes a variety of solutions to problems encountered in situated visualization.

The focus of this thesis lies in addressing the most pressing issues of situated visualization that severely limit the usability of AR. Specifically, this thesis aims to provide solutions that support users in achieving overviews of the presented data. To provide overviews, the thesis addresses the issue of data overload when presenting large amounts of data. This is achieved by new view management techniques that not only reduce the amount of data and provide layouts that are free of interferes, but are also temporally coherent.

Because of the inherent ego-centric viewpoint, overviews are difficult to achieve in AR. Therefore, this thesis also investigates new techniques that allow users to compensate for their ego-centric viewpoint using transitional interfaces and multi-perspective renderings. The presented techniques also allow zooming viewpoints of situated visualizations, which is a challenging task to realize in AR.

Kurzfassung

Augmented Reality (AR) kommuniziert für gewöhnlich einfache Zusatzinformationen zu Objekten in der Welt, kann aber auch den Benutzer bei bestimmten Aufgaben unterstützen. AR-Anwendungen realisieren das, indem sie virtuelle Daten direkt über der realen Welt anzeigen. Diese überlagerten Informationen können aus einfachen Annotationen bestehen, die die Namen von Restaurants in der näheren Umgebung anzeigen. Diese Überlagerungen können Benutzer aber auch durch Wartungs- und Montageaufgaben leiten. In den genannten Beispielen wird AR-Technologie zur Visualisierung von Informationen verwendet, die sich auf etwas in der realen Umgebung beziehen. Diese Arbeit bezeichnet diese Art der Darstellung als ortsbezogene Visualisierung.

Das Erzeugen von effektiven ortsbezogenen Visualisierungen ist sehr anspruchsvoll. Während einige der dabei aufkommenden Probleme nur auf AR bezogen sind, sind andere schon in verwandten Visualisierungsgebieten, wie zum Beispiel der Informationsvisualisierung und der wissenschaftlichen Visualisierung, aufgetaucht. Die ortsbezogene Visualisierung kann sich deswegen auf einen reichen Schatz an Arbeiten aus diesen Gebieten stützen. Diese Arbeit idendifiziert zuerst die Probleme von AR-Visualisierungen und stellt Verbindungen zu verwandten Wissensgebieten her, um eine solide Grundlage für ortsbezogene Visualisierungen zu schaffen. Basierend auf dieser Grundlage steuert diese Arbeit anschließend eine Reihe von Lösungen zu Problemen der ortsbezogenen Visualisierung bei.

Diese Arbeit konzentriert sich darauf, Lösungen für die größten Herausforderungen der ortsbezogenen Visualisierung zu finden, die deren Anwendbarkeit für AR am stärksten einschränken. Um genauer zu sein, präsentiert diese Arbeit Lösungen, die es den Benutzern von AR erlauben einen Überblick über die präsentierten Daten zu bekommen. Um diese Überblicksvisualisierungen zu ermöglichen, präsentiert diese Arbeit Lösungen zur Vermeidung von überfüllten Anzeigen, falls zu viele Daten dargestellt werden. Dies wird durch neue Layouttechniken erreicht, die nicht nur die Menge der Daten verringern und Lay-

outs erzeugen, die frei von Konflikten sind, sondern die sich auch über die Zeit koherent verhalten.

Weiters schränkt der inherent egozentrische Blickpunkt von AR die Möglichkeiten von Benutzern einen Überblick über die Daten zu bekommen stark ein. Diese Arbeit präsentiert neue Techniken, die diesen egozentrischen Blickpunkt ausgleichen, indem sie Benutzern erlauben zu virtuellen Blickpunkten zu wechseln und multi-perspektivische Darstellungen anzuzeigen. Die präsentierten Techniken erlauben es Benutzern auch deren Blickpunkt zu zoomen, etwas, das in AR sonst schwer zu realisieren ist.

Statutory Declaration

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

The text document uploaded to TUGRAZonline is identical to the presented doctoral thesis.

Place	Date	Signature

Eidesstattliche Erklärung

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

Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Ort	Datum	Unterschrift

Acknowledgments

I am grateful for all the people that I worked with and that supported me over the years. First of all, I thank my scientific advisor Dieter Schmalstieg, who gave me the freedom to follow my ideas and at the same time provided guidance and support so that I did not stray too far from the path set by my thesis. I also thank Kiyoshi Kiyokawa for agreeing to be my second advisor and for the valuable feedback and suggestions for finishing this work.

My thanks also go to my colleagues Raphael Grasset, Jacob Boesen Madsen, Valeria Orso, Hartmut Seichter, and Eduardo Veas, with whom I had the pleasure to work on different research topics that make up this thesis. I especially thank Denis Kalkofen, with whom I collaborated closely over the last couple of years. I also thank all my colleagues, scientific and staff, I had the opportunity to work with over the years.

I am also grateful for the people close to me for providing their support and sticking with me, when I was busy. Gerald, who introduced me to climbing, my now favorite sport. He and Magdalena made sure that I would not forget how to get up a vertical wall. Marcel, Philipp and Michael for making me climb more. Peter, who showed me how to fly drones and then repair said drones. Birgit, for the many relaxing discussions over dinner. And especially Sylvaine, for her wonderful support and high tolerance for long working hours.

Lastly, and most importantly, my deepest thanks go to my family, for all their love and support.

Contents

Chapter 1

Introduction

Contents

Augmented Reality (AR) overlays virtual data directly on top of the real world. A common goal of AR is to communicate additional information about real world entities or to support certain tasks of a user. The overlaid information can consist of simple annotations, such as the names of the restaurants that are closest to the user. The overlays can also guide a user through maintenance and assembly tasks. In these examples, AR technology is used to visualize information regarding the surrounding real world environment. This thesis refers to this kind of AR visualization as *situated visualization* [146].

Creating effective situated visualizations is challenging. While some of the involved issues are unique to AR [79], others have already been encountered in related visualization areas such as information visualization and scientific visualization. Consequently, situated visualization can draw inspirations from a large body of work.

This thesis identifies the issues of visualization in AR and builds connections to related areas to provide a solid foundation for situated visualization. Based on this foundation, this thesis contributes a variety of solutions to problems encountered in situated visualization.

1.1 Augmented Reality

Milgram et al. [95] defined the Reality-Virtuality continuum, which allowed them to classify AR with respect to traditional Virtual Reality (VR) (Figure 1.1). The continuum spans the two extremes from only presenting the unmodified real world to presenting a purely virtual world. Everything between these extremes can be classified as Mixed Reality (MR). Within the range of MR applications, AR is situated closer to the real world, because it mainly shows a real world representation, e.g., in the form of a video stream, which is modified by virtual objects.

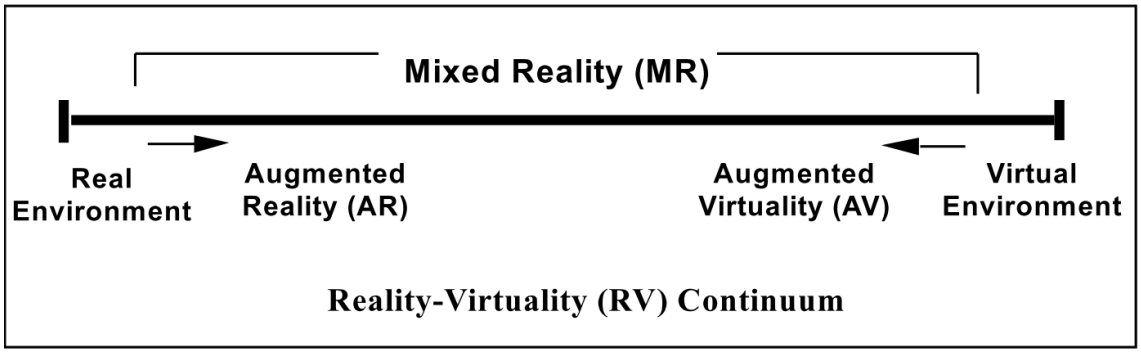


Figure 1.1: Reality-virtuality continuum (Image taken from Milgram et al. [95]).

The most accepted definition of AR comes from Azuma [6], who defined that AR applications must exhibit three characteristics:

1. They combine the real and virtual world,
2. they are interactive applications and run in real time, and
3. they are registered in 3D.

Both Milgram et al. [95] and Azuma [6] explicitly stated that AR is not bound to a certain display technology. Consequently, their definitions and classifications are device agnostic. In fact, AR applications can be displayed using very different technologies and platforms, such as a Head-Mounted Display (HMD) [60], a mobile phone [96], or by using a projector system [17].

The first AR system is attributed to Sutherland [125], who built an HMD (Figure 1.2) which could overlay virtual line renderings on top of the real world. However, it was not until the early 90s, before the term AR was coined by Caudell and Mizell [24] in a paper

that describes an HMD system which supports workers in aircraft manufacturing processes by overlaying instructions directly on top of the real world.

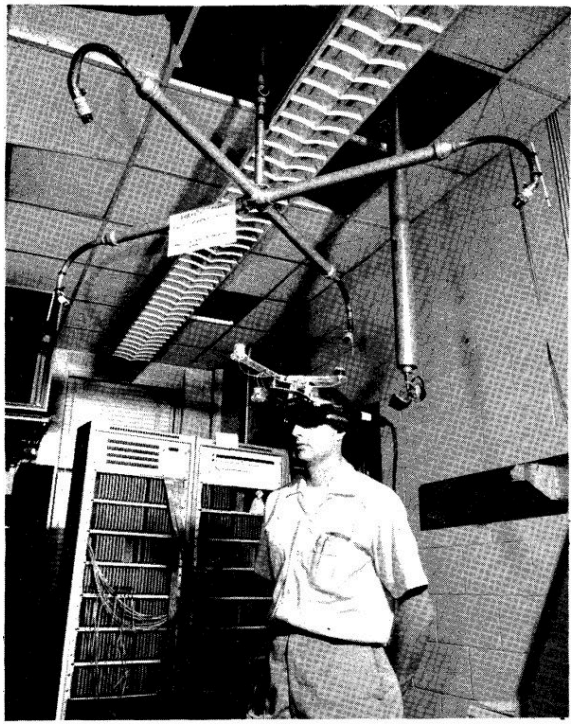


Figure 1.2: The first known HMD-based AR system by Ivan Sutherland (Image taken from Sutherland [125]).

The first stand-alone mobile outdoor AR system was built by Feiner et al. [39]. Their "touring machine" allowed users to explore the campus of the Columbia University by adding information to the real world. The system used an HMD in combination with a portable computer for creating 3D graphics. To estimate the viewpoint of the user relative to the real world, they used Global Positioning System (GPS) for positional tracking and orientation sensors to retrieve yaw, pitch and roll angles. Shortly afterwards, Thomas et al. [135] presented a similar system.

While the first mobile AR systems used expensive HMD technology, researches strived to create AR systems which could run on off-the-shelf consumer grade hardware. Consequently, handheld AR systems were created, which were deployed on a tablet PC [66], a Personal Digital Assistant (PDA) [142], or a smart phone [96].

To date, the most common and widely available platform for commercial AR applications are smart phones. Smart phones are compact mobile computers with impressive computational performance and come by default equipped with a video camera, GPS sen-

sor and compass, which can be used to estimate the user's viewpoint of the real world. This made smart phones a natural platform for AR applications and gave rise to companies[1][2] which focused on creating AR systems and applications for these mobile devices.

1.2 Visualization

The dictionary definition of "to visualize" is "to form a mental image of something" or "to make (something) visible". Visualization per se refers to a human cognitive ability where somebody gains insight into or acquires knowledge about something [122, p.5]. Visualization can be supported by pictorial representations. Such representations can reduce the complexity of data and map it to easily understandable concepts. One of the most striking examples of such a depiction is a flow map drawn by the French cartographer Charles Joseph Minard in 1869 (Figure 1.3). With this simple depiction, Minard effectively tells the story of Napoleon's fatal Russian campaign in 1812.

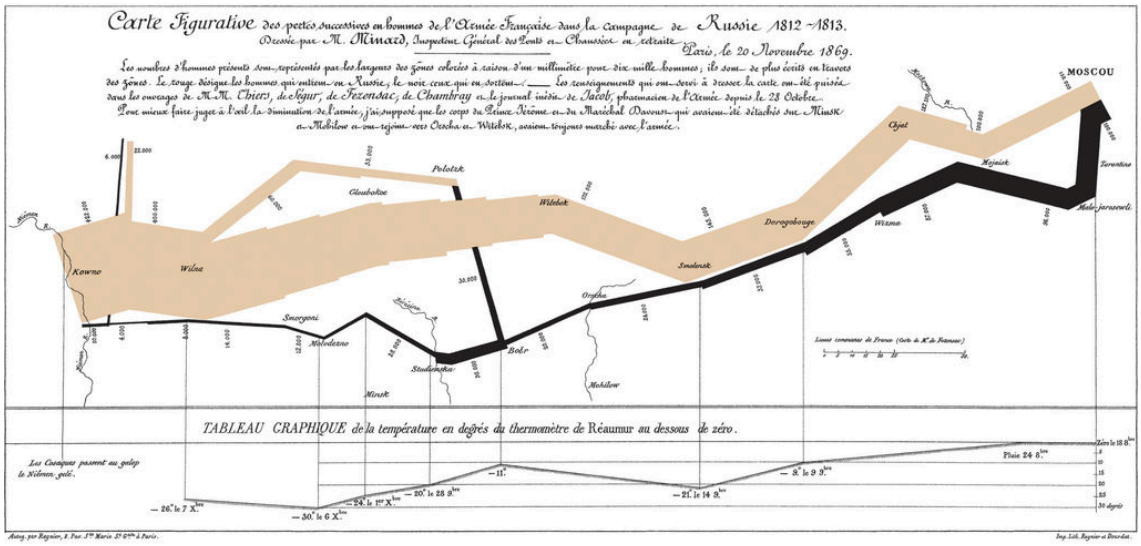


Figure 1.3: A flow map drawn by Charles Joseph Minard. The thickness of the line represents the number of men in Napoleon's army during the Russian campaign of 1812. The decreasing thickness clearly communicates the tragic loss of human life, especially on the way back to France when winter set in.

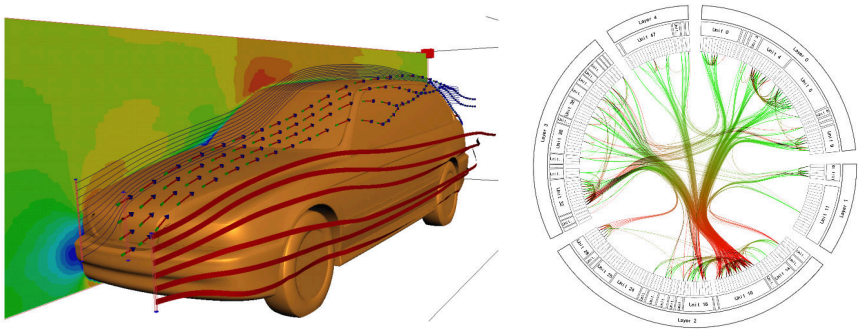
Napoleon set out to conquer Russia with an army of around half a million soldiers (brown line); he returned to France with only around 20.000 (black line). Aside from the

[1] http://www.metaio.com/
[2] https://www.layar.com/

size of the army, the map also provides a spatial representation to illustrate the troop movement and temperature information of the Russian winter the army faced on their retreat. With the help of Minard's map, a viewer can quickly and easily gain insight into aspects of Napoleon's campaign without studying bare numbers.

Researchers also benefit from such visual representations of data, which support them in gaining insights from measurement and simulation data. The investigated data sets are generally too large to create hand-drawn illustrations. However, increasing computational performance lead to the emergence of the field of *visualization* in computer science. Computers not only allow researchers to quickly and conveniently create visual representations of data, but also to interact with these representations in order to analyse the data from different viewpoints or to highlight certain aspects of the data. Card et al. [23, p.6] provide a definition of visualization in computer science, which sums it up as "the use of computer-supported, interactive, visual representations of data to amplify cognition".

Visualization is divided into different research areas. Two major areas are *scientific visualization* and *information visualization* [41]. Scientific visualization mainly deals with physics-based data gathered from measurements or simulations. Therefore, it also has an inherent spatial mapping to the physical world (Figure 1.4(a)). Information visualization focuses mainly on presenting abstract, non-spatial data, which has to be mapped to a spatial representation (Figure 1.4(b)).



(a) (b)

Figure 1.4: Scientific and information visualization. (a) Scientific visualization focuses mainly on mapping physical measurements or simulation data to the inherent spatial reference frame (Image taken from Schulz et al. [116]. (b) Information visualization transforms abstract data into spatial mappings (Image taken from Holten [62]).

1.3 Situated Visualization

Visualization is also a research area in the field of AR. A major advantage of AR is that information that relates to the real world can be displayed and explored directly in the spatial reference frame of the real world environment. White and Feiner [146] refer to this type of visualization as *situated visualization*.

In contrast to scientific and information visualization, the definition of situated visualization does not make a difference between the types of data to display. For instance, abstract data is commonly communicated using annotations [51] (Figure 1.5(a)), but also physics-based data is visualized [36, 146] (Figure 1.5(b)). Both are situated visualizations, because they relate to something that is present in the real world. The main defining property of situated visualization is this connectedness of the information to the real world. This is in accordance with White [145], who defines that the visualized data must be related to a physical context. Hence, the definition of situated visualization is based on the visualized data itself and not the system that is used to visualize this data. Situated visualizations can be created using any kind of system that allows to register data in the real world.

Note that visualizations in AR are not automatically situated visualizations, even though they are displayed by using AR technology. Figure 1.5(c) shows an example of a visualization that is not a situated visualization. Even though AR is used for exploring a scientific visualization [45], the presented data is completely disconnected from the surrounding real world. Because of this missing relation, the AR system could easily be converted to a VR system by removing the real world background.

1.4 Challenges

A shared issue among all of these visualization disciplines is the presentation of large amounts of data. However, AR researchers have encountered issues, which are unique to AR and may have to be addressed when creating visualizations for AR. For instance, a major issue is the dynamic nature of the real world environment, which makes it difficult to create a single visualization, which fits to all real world scenarios.

The following sections discuss the issues of situated visualization. While some of these issues are unique to AR, others have also been encountered in information and scientific visualization. Therefore, situated visualization designers can build on the large body of previous work from these areas. The issues described in the following will also be discussed

(a) (b) (c)

Figure 1.5: Situated visualization. Situated visualizations can communicate (a) abstract information in the form of annotations (Image taken from Grasset et al. [51]), but also (b) physical properties and measurements, such as the air pollution (Image taken from White and Feiner [146]). They are always connected to a real world entity. (c) Not all AR visualizations are situated visualizations. In this example, data is visualized using AR technology (Image taken from Fuhrmann et al. [45]). However, the visualization does not connect to a real world entity.

in the light of this previously gained knowledge. From here on, the term AR visualization also refers to situated visualization, if not indicated otherwise.

1.4.1 Visualization Challenges

Situated visualization must face the same challenges as traditional visualization. Data overload leads to a cluttered presentation when too much data is presented, which impairs its understanding. Users also must be able to explore the data interactively.

Data Overload. Presenting a large amount of data in AR quickly leads to a cluttered presentation, which makes it hard for a viewer gain insight into the data. The problem is aggravated when using an AR platform, which has only a limited amount of presentation space, such as smartphone displays in handheld AR.

Azuma et al. [7] identified this situation as the problem of increasing data density. They refer to two complementary solutions to manage the data. The first solution is to reduce the amount of data by filtering [40, 70]. The second solution is view management, which creates a layout of the data so that it is not interfering with other important information.

In the context of information visualization, Spence [122] refers to the issue of data density as data overload. Handling large amounts of data was a main motivation for the fields of scientific and information visualization. Early on, these fields developed

architectural models to cope with this problem. Most visualizations are based on such models. In the domain of scientific visualization, Haber and McNabb [55] describe a pipeline for mapping data to a visual representation. Card et al. [23] present a similar model tailored to information visualization, which also takes into account a user interacting with the visualization. Although there are slight differences in these models, they generally incorporate the following three steps to visualization (Figure 1.6):

1. Data transformation
2. Visual mapping
3. View transformation

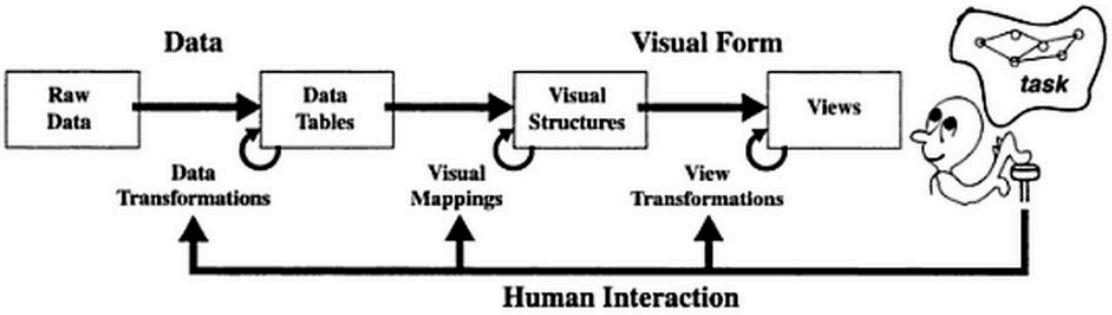


Figure 1.6: Visualization pipeline as proposed by Card et al [23] (Image taken from Card et al. [23]).

The data transformation step encompasses the reduction of the data by filtering or aggregating data points. Visual mapping refers to the creation of visual structures of the data such as color and shape. The view transformation finally determines properties such as the position and scale of the visual structures.

Coming back to the two complementary solutions, filtering and view management, for handling the issue of data density in AR [7], it becomes clear that these steps can be integrated nicely into the data transformation and the view transformation steps of the traditional visualization pipeline.

User Interaction. In contrast to hand-drawn illustrations, a core aspect of computer-supported visualization is the ability to interactively explore the data. Therefore, an effective visualization supports the exploration by providing an appropriate interface. In

the area of information visualization, the design of visualization interfaces was heavily influenced by the "information seeking mantra" presented in the seminal work of Shneiderman [119]. The mantra is based on the extensive experience of Shneiderman in designing interfaces for exploring large amounts of data and involves three main tasks that users want to perform: "overview first, zoom&filter, then details-on-demand". Shneiderman defined the tasks as follows:

- "Overview: Gain an overview of the entire collection.
- Zoom: Zoom in on items of interest
- Filter: filter out uninteresting items.
- Details-on-demand: Select an item or group and get details when needed."

While Shneiderman only intended to provide recommendations for information visualization systems and did not see the steps as being prescriptive [29], many successful visualizations have been designed by following this mantra. Therefore, designers of situated visualizations should also consider these recommendations, since they could support users getting valuable insights into the data.

1.4.2 Challenges of Augmented Reality

When embedding situated visualizations into the real world, several issues must be addressed:

1. Visual coherence: a viewer must be able to "connect" real and virtual, i.e., to spatially relate the visualization to a real world location or object.
2. Visual interference: a viewer must be able to easily discern important from irrelevant information.
3. Dynamics of AR and temporal coherence: changes of the real world environment or the viewpoint of the user can cause distracting changes of the visualization. To avoid distractions, the changes must be performed in a temporally coherent way.
4. Ego-centric viewpoint: the user is limited to an ego-centric viewpoint when exploring a visualization. This is a major limiting factor of AR.
5. Registration errors: when the registration of the visualization is erroneous, it may communicate false information.

In the following, these issues will be discussed in more detail.

Visual Coherence. Naively overlaying virtual augmentations on top of the real world does not create a very convincing rendering, because real and virtual objects do not interact with each other. The augmentations seem to float in the world. Breen et al. [20] speak about a lack of physical and visual interaction between real and virtual, referring to occlusion and illumination. This not only creates unconvincing renderings that are not visually coherent to the real world, but also impacts the understanding of an AR visualization. Important depth cues are missing, which help viewers to estimate the spatial relations between augmentations and the real world.

To improve the visual coherence between the virtual information and the real world, occlusions and illumination must be rendered correctly. For instance, invisible virtual models of real world objects can be registered to the real world and their depth values can then be used to render correct occlusion relationships [20, 74]. The virtual content can also be shaded using the real world lighting conditions [43].

Aside from occlusion and illumination, technical issues of AR platforms should also be considered for creating visually coherent AR visualizations. For instance, in video see-through AR, the image pipeline and lens of the used camera can add distortions and other artifacts to the captured image. This pipeline can be modelled and applied to the augmentations to improve the integration of virtual content into AR [78].

Visual Interference. Visualizations generally emphasize the relevant parts of data in a way that guides the viewer's attention to this information. Without this emphasis, the information can easily be overlooked, because it does not stand out from the rest of the data.

In situated visualization, the viewer's attention must be guided towards the important parts of the scene, and irrelevant aspects of the real world should not be distracting. This requires that visual interferences between the visualization and the real world must be avoided. Kalkofen [71, p.23] identified this as a focus&context visualization problem in AR. The relevant part of the scene is the focus, while the rest of the scene provides context. Figure 1.7(a) shows that the focus is emphasized to guide the viewers attention and the rest of the scene is deemphasized to avoid distractions.

Visual interferences between the visualization and the real world can also be caused by the placement of the augmentations, which can cause occlusions of the augmented real world object and other important landmarks. Therefore, view management techniques have been developed, which avoid such occlusions by rearranging the virtual content [12, 51] (Figure 1.7(b)).

(a) (b)

Figure 1.7: Visual interferences. The presentation of a visualization can interfere with the real world. (a) Therefore, the focus of the visualization must be emphasized to avoid visual interferences with the background and to guide the viewer's attention. Here, the focus is emphasized by reducing the saliency of the context (Image taken from Kalkofen et al. [72]). (b) The visualization can also occlude important landmarks. View management techniques can place content to avoid such occlusions. Here annotations are moved away from the buildings into free spaces (Image taken from Grasset et al. [51]).

Dynamics of AR and Temporal Coherence. A major issue of visualization in AR is that, in contrast to traditional visualization, the real world context is not static, but changes over time. For instance, people or cars pass through the video image, or the lighting conditions change. Hence, augmentations may not be visually coherent with the real world anymore. Recent advances made it feasible to estimate both illumination [54] and occlusions [65] in real-time, which allows AR applications to react to changes to the environment. Consequently, AR visualizations can also be updated to reflect changes to the environment.

Changes to the real world environment can also cause visual interferences, because a visualization that is designed for certain conditions may not be effective when these conditions change. Figure 1.8 shows a visualization of the internals of a car. While the internals are clearly visible in the yellow car (Figure 1.8(a)), the same visualization causes perceptual problems when the car is exchanged for a red one (Figure 1.8(b)).

To avoid such visual interferences, Kalkofen [71] suggests to create adaptive visualizations, which react to changes of the real world context they are embedded in. Computational aesthetics [59] has the goal to automatically create presentations that are also visually pleasing to humans [53].

Because of the dynamics of AR, visualizations may change over time to adapt to the

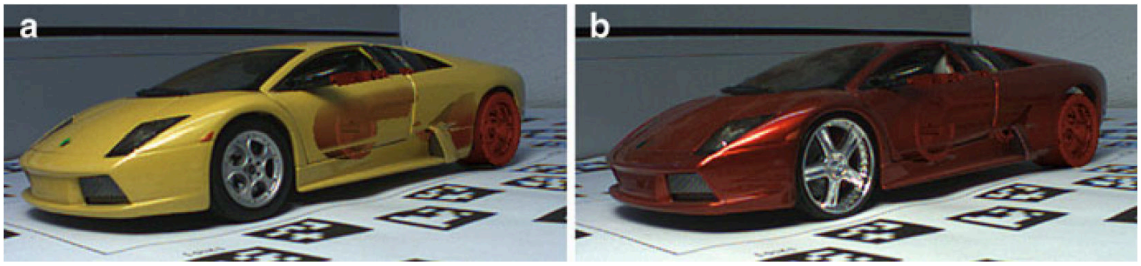


Figure 1.8: Visual interference. (a) The visualization clearly shows the internals of the car in the rear. (b) A poor choice of color severely impacts the perception of the occluded information (Images taken from Kalkofen et al. [73]).

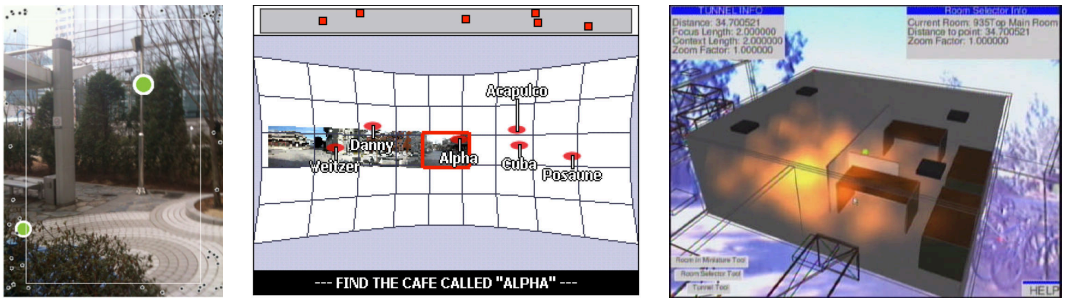
new conditions. However, frequent and strong changes in the visualization can distract a viewer from perceiving the communicated information. Therefore, visualizations must behave in a temporally coherent way.

This aspect is especially important in view management systems that create layouts of annotations in AR by continuously enforcing certain layout constraints [12]. In such systems, simple viewpoint changes may already introduce significant layout changes, which must be treated accordingly to achieve temporally coherent updates.

Ego-centric Viewpoint. Traditionally, visualization techniques are deployed in virtual environments in which users can freely change the viewpoint of the visualization. However, AR applications generally are limited to only one viewpoint: the egocentric viewpoint of the user. This severely restricts the ability of users to explore information in real world environments, because oftentimes users do not have a good vantage point on the information, e.g., when exploring a building in a city environment. This also impacts the interface design of situated visualizations, because tasks such as getting an overview [119] cannot be performed from an ego-centric viewpoint.

There are three solutions to extend the ego-centric viewpoint of the user: offscreen visualizations, multi-perspective renderings and transitional interfaces. Offscreen visualizations indicate to the viewer the direction in which information can be found outside of the current view frustum (Figure 1.9(a)). Multi-perspective visualizations integrate additional viewpoints into the ego-centric AR view. These not only communicate the general direction of the location of data, but can also expand the field of view of the user so that the actual location of the data is shown (Figure 1.9(b)). Transitional interfaces combine the AR view with VR views, which allows users to change viewpoints of the world without physically changing their location (Figure 1.9(c)). The views are often connected by

smooth transitions between AR and VR views to facilitate mentally linking AR and VR views [16, 77].



(a) (b) (c)

Figure 1.9: Extending Ego-centric Viewpoint. There are three solutions to expand the ego-centric AR viewpoint of the user. (a) Offscreen visualizations indicate information outside of the field of view (Image taken from Jo et al. [69]). (b) Multi-perspective renderings expand the field of view (Image taken from Mulloni et al. [97]). (c) Transitional interfaces provide virtual viewpoints of the real world (Image taken from Bane and Höllerer [10]).

Registration Errors. A situated visualization relates to something that is present in the real world. Hence, the visualization is registered to the real world and relies on accurate and stable registration methods. However, registration can still fail in many instances, which causes misalignments between the augmentations and the real world context they refer to.

To cope with this problem, visualizations should also take the precision of the registration into account. For instance, Roberts and MacIntyre [109] communicate the error to the viewer by integrating a virtual copy of the real world context into the visualization (Figure 1.10).

1.5 Contributions

The focus of this thesis lies in addressing the most pressing issues of situated visualization that severely limit the usability of AR. The guiding principle of the presented research is the information seeking mantra [119]. Specifically, this thesis aims to provide solutions that support the overview task by addressing data overload when presenting large amounts of data. This is achieved by new view management techniques that not only reduce the

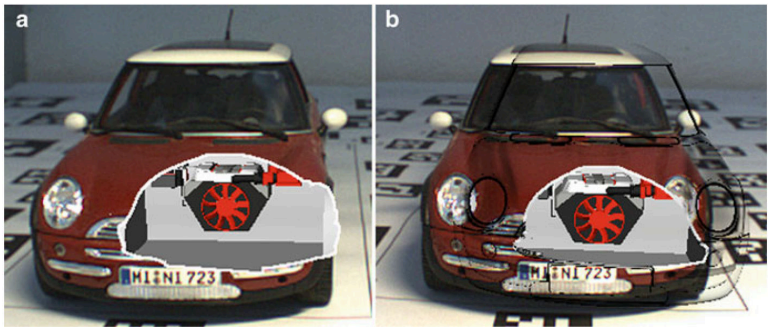


Figure 1.10: Error visualization. (a) The registration error causes a misalignment of virtual content and the real world context, here the engine compartment. (b) The registration error can be resolved by integrating a copy of the context, in this case the outlines of the car, into the visualization. (Images taken from Kalkofen et al. [73]).

amount of data and provide layouts that are free of interferes, but are also temporally coherent.

Because of the inherent ego-centric viewpoint, overviews are difficult to achieve in AR. Therefore, we also investigate new interaction techniques that allow users to compensate for their ego-centric viewpoint. The presented techniques also allow zooming situated visualizations, which is a task suggested by the mantra, but is challenging to realize in AR.

While visual coherence is important for understanding the relationship between virtual and real content, the priority of this work is to provide means for exploring information in AR. Therefore, the presented AR applications use only basic depth cues by rendering occlusions of real and virtual content correctly. Furthermore, the registration error is not investigated, as more advanced tracking systems may eventually overcome this issue.

In the following, a detailed list of the contribution regarding the issues of situated visualization is presented. This list is also reflected in the structure of the thesis.

1.5.1 Combining Filtering and View Management

Data overload has a major impact on a user's ability to understand the presented information. Common filter techniques reduce the amount of data to a sensible amount, but may remove relevant information in the process.

We present a general framework for creating a compact visualization which avoids data overload by only removing redundant data (see Section 3.1). We present a novel combination of filtering and view management that avoids information loss and thereby

creates effective overview visualizations. In contrast to previous work, this combined approach already considers the layout of information in the filtering step. We illustrate the generality of the approach with textual and pictorial annotations and assembly data [131–133].

Furthermore, we introduce hierarchies of similar items into compact visualizations (see Section 3.2.2). Hierarchies allow us to get more control over the visualization by introducing multiple Level-of-Detail (LOD).

Compact visualizations filter redundant information. If such redundancies are not available in the data set, the amount of data can be reduced by clustering data items into groups of items according to user-defined similarity criteria. We present an approach that creates a hierarchy of such clusters. Our view management algorithm chooses the appropriate hierarchy levels to fill the screen-space efficiently to create an initial overview for the user (see Section 3.2.1). The user can perform zoom&filter operations and request details-on-demand.

Aside from data overload, we also address the issue of visual interferences between virtual and real world content (see Section 3.1.6.3). We achieve this by controlling the filter output of the compact visualizations. Furthermore, we make the real world to an active part in the view management and allow manipulations of the real world scene structure to avoid visual interferences [133].

1.5.2 Temporally Coherent View Management

In contrast to traditional visualization, the viewpoint of situated visualizations in AR can change constantly. This causes frequent changes of the layout of information to avoid visual interferences between the real world and the augmented content, but also between the augmented content itself. Common approaches to achieve temporal coherence include hysteresis and positional thresholds that delay the required changes as long as possible.

In this thesis, we show how to integrate temporal coherence into the optimization of compact visualizations itself to minimize distracting layout changes during viewpoint changes (see Section 4.1.1). Furthermore, we propose to freeze layouts during camera motion and, at the same time, optimize the layout to avoid visual interferences between the augmented information (see Section 4.1.2). We illustrate the solutions with assembly and annotation data [133].

A common issue of unstable layouts in AR is the way in which the view management is implemented. Typically, view management techniques create layouts in screen-space

coordinates. Such layouts are especially prone to unstable layouts during camera motion. We present a new view management approach that both registers annotations and resolves conflicts between annotations purely in 3D space (see Section 4.2). Furthermore, we restrict the degrees of freedom of annotations to minimize the potential for conflicts. This tremendously improves the temporal coherence of the layout [129].

In order to verify our novel approach, we compare it against different view management systems. To the best of our knowledge, this is the first attempt to quantitatively compare the effect of different implementations of view management systems on the task performance of a user. Our system, which creates a static layout of annotations in 3D, clearly outperforms the reference systems, which update labels to enforce layout constraints (see Section 4.3).

1.5.3 Extending the Ego-centric Viewpoint

The usability of AR applications is severely impaired by the limitation to the ego-centric viewpoint of the user. We investigated two solutions that solve this problem in AR: transitional interfaces and multi-perspective renderings.

In contrast to previous transitional interfaces, our Object-Centric Exploration (OCE) techniques allow users to retrieve a virtual copy of a real world object (see Section 5.1). This allows users to investigate a real world without physically changing their location. Hence, users can get an overview by interacting with the object, but also can zoom into details [126, 130].

We evaluate the OCE techniques in urban settings and compare them to traditional 3D map applications. Based on our findings, we put forward design recommendations for developing such interfaces [126, 127].

To facilitate the interaction with the virtual copy, we control the transition to virtual viewpoints based on knowledge about the scene and about the task (see Section 5.2). Therefore, our system can propose virtual camera viewpoints and manipulators that are best suited for the current task of the user [130].

Aside from transitional interfaces, we explored multi-perspective renderings, that can enhance the overview of an object by visualizing otherwise occluded areas (see Section 5.3). We integrated multi-perspective renderings into the OCE techniques [126, 127], compact visualizations [132], and we also demonstrate a prototype application using a preview window for navigation tasks [128].

1.6 Publications

This thesis is the product of a collaborative research effort of a several people. This section lists the publications, which are the foundation of this thesis:

- Markus Tatzgern, Denis Kalkofen, and Dieter Schmalstieg: *Compact explosion diagrams*. Symposium on Non-Photorealistic Animation and Rendering (NPAR '10), 2010 [131] (see Sections 3.1.4 and 3.2.2.2).

- Markus Tatzgern, Denis Kalkofen, Dieter Schmalstieg: *Multi-perspective compact explosion diagrams*, Computers & Graphics (C&G '11), Volume 35, Issue 1, February 2011, [132] (see Sections 3.1.4, 3.2.2.2 and 5.3.1).

- Markus Tatzgern, Denis Kalkofen, Dieter Schmalstieg: *Dynamic compact visualizations for augmented reality*, IEEE Virtual Reality (VR '13), 2013 [133] (see Sections 3.1.2, 3.1.5, 3.1.6 and 4.1).

  The author was the main contributor to the design and development of compact visualizations. The co-authors contributed to the concept and provided valuable suggestions regarding the implementation.

- Markus Tatzgern, Denis Kalkofen, Raphael Grasset, Dieter Schmalstieg: *Embedded Virtual Views for Augmented Reality Navigation*, IEEE International Symposium on Mixed and Augmented Reality (ISMAR '11), Workshop on Visualization in Mixed Reality Environments, 2011 [128] (see Section 5.3.2).

  The author was the main contributor to the design and development of multi-perspective presentations for extending the ego-centric viewpoint of a user. The co-authors provided valuable feedback and implementation suggestions.

- Markus Tatzgern, Raphael Grasset, Eduardo Veas, Denis Kalkofen, Hartmut Seichter, Dieter Schmalstieg: *Exploring Distant Objects with Augmented Reality*, Joint Virtual Reality Conference of EGVE - EuroVR (JVRC '13), 2013 [126] (see Section 5.1).

- Markus Tatzgern, Raphael Grasset, Eduardo Veas, Denis Kalkofen, Hartmut Seichter, Dieter Schmalstieg: *Exploring Real World Points of Interest: Design and Evaluation of Object-centric Exploration Techniques for Augmented Reality*, Pervasive Mobile Computing: Special Issue on Mobile and Pervasive Applications in Tourism, 2014 [127] (see Sections 5.1 and 5.1.4).

The author was the main contributor to the design and development of the solution to use a virtual copy metaphor for extending the ego-centric viewpoint of a user. The author co-designed, implemented, performed and evaluated the user studies. The co-authors contributed to the design of the presented concepts and provided guidance regarding study design and evaluation.

- Markus Tatzgern, Raphael Grasset, Denis Kalkofen, Dieter Schmalstieg: *Transitional Augmented Reality Navigation for Live Captured Scenes*, IEEE Virtual Reality (VR '14), 2014 [130]. (see Section 5.2).

  The author was the main contributor to the design and development of the scene capturing system and the presented transitional interfaces which use scene and task knowledge to guide the virtual camera. Raphael Grasset co-designed the presented solutions. The co-authors contributed to the concept and provided valuable suggestions regarding the implementation.

- Markus Tatzgern, Denis Kalkofen, Raphael Grasset, Dieter Schmalstieg: *Hedgehog Labeling: View Management Techniques for External Labels in 3D Space*, IEEE Virtual Reality (VR '14), 2014 [129] (see Section 4.2).

  The author was the main contributor to the design and development of the 3D view management techniques. The co-authors contributed to the concept and provided valuable suggestions regarding the implementation.

- Markus Tatzgern, Valeria Orso, Denis Kalkofen, Giulio Jacucci, Luciano Gamberini, Dieter Schmalstieg: *Adaptive Information Density for Augmented Reality Displays*, submitted, 2015 [134] (see Section 3.2.1).

  The author was the main contributor to the design and development of the adaptive information density algorithm. The author also co-designed, implemented, performed and evaluated part of the user studies. The co-authors contributed to the concept and performed part of the user studies.

- Jacob Boesen Madsen, Markus Tatzgern, Denis Kalkofen, Dieter Schmalstieg, Claus B. Madsen: *Evaluating Adaptive Labeling for Dynamic Handheld Augmented Reality*, submitted, 2015 [90] (see Section 4.3).

  The author was the main contributor to the development of the view management algorithms. The author also co-designed, implemented, performed and evaluated

part of the user studies. The co-authors contributed to the design, implementation and evaluation of the user studies.

Chapter 2

Background

Contents

In this thesis, we compensate for data overload and the resulting visual clutter by reducing the number of presented items using filter and cluster techniques. Furthermore, we combine these techniques with view management to create spatial layouts that are free of overlaps. This allows us to create overview visualizations of the input data that can be used for further detailed exploration, e.g., by performing zooming and filter operations.

The inherent ego-centric viewpoint of AR limits the ability of users to get an overview over the presented information. Therefore, we also developed methods to compensate for this ego-centric viewpoint.

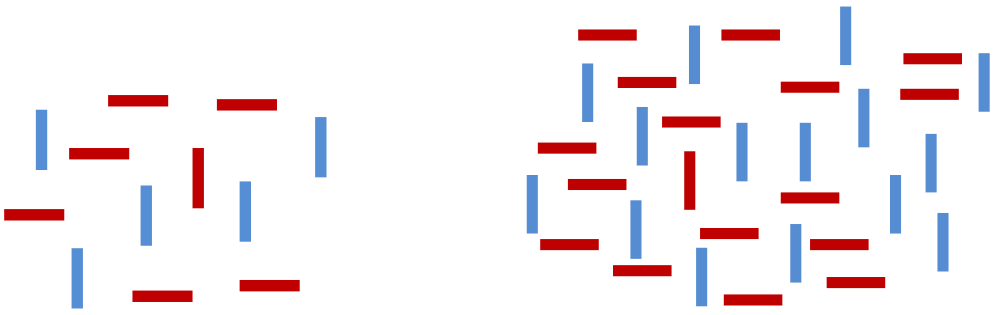
In the following, we review what visual clutter means in the context of visualization in general (see Section 2.1). We identify data overload and the spatial layout as major causes of clutter. We present related work in AR that handles data overload (see Section 2.2), before discussing view management techniques to create layouts of information (see Section 2.3). Section 2.4 discusses previous approaches that combine the filtering process and the view management. Section 2.5 presents related work regarding the extension of the ego-centric viewpoint in AR.

2.1 Visual Clutter

To achieve an effective overview of information, it is essential to avoid visual clutter. Visual clutter has always been an issue in visualization, which often states the large number of items as the main cause. Rosenholtz et al. [111] define visual clutter in visualization as follows:

> Clutter is the state in which excess items, or their representation or organization, lead to a degradation of performance at some task.

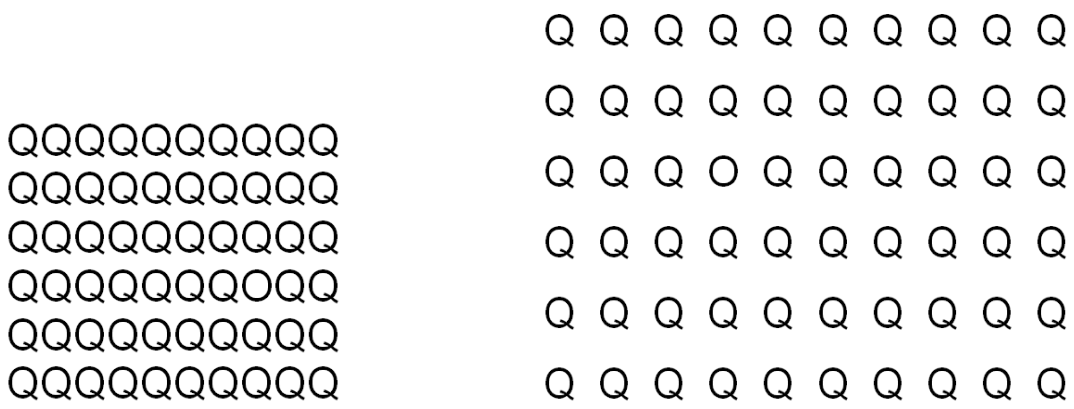
They refer to visual search tasks that are common in visualization, where a specific element must be identified. To achieve this, the element must be discernible from other elements. Typically, the performance decreases with an increasing number of presented items and users require more time to identify distinct items [149] (Figure 2.1). Aside form the number of items, the location of the elements also influences the performance. If items lie close to each other, they are harder to distinguish from each other. This is also due to an effect called lateral masking. Lateral masking impairs perceiving an item when other elements are located in its surroundings [144] (Figure 2.2). Hence, visual clutter may be caused by the number of items and their layout.



(a) (b)

Figure 2.1: Performance of visual search and set-size. The performance of visual search tasks is influenced by the amount of data. (a) In a smaller data set users will more likely find the vertical red bar, (b) than in a larger data set.

Ellis and Dix [37] presented a thorough taxonomy of clutter reduction techniques for information visualization. They distinguish between three classes of techniques: appearance, spatial distortion and temporal. Appearance refers to techniques that change the presentation of the data items. This includes changing criteria such as the size and opacity of the



(a) (b)

Figure 2.2: Performance of visual search and spatial arrangement. Surrounding elements can decrease visual search performance and distract from a certain item. This can be compensated by increasing the spatial offset between the items. (a) The "O" hidden among the "Q" is hard to spot, because of the similarities. (b) Increasing the spacing between the letters can make it easier to find.

data points, but also filtering and clustering techniques that remove items completely from the screen. Spatial distortion includes among others point displacement techniques that change the positions of data items to avoid overlaps. Finally, temporal techniques refer to visualizations that use animations to compensate for clutter and overlapping items.

The taxonomy presented by Ellis and Dix [37] can also be applied to situated visualization. Filtering techniques are also used in AR to reduce visual clutter [70], while point displacement directly relates to view management techniques that change the spatial positions of data to resolve overlapping items and occlusion of real world objects [12].

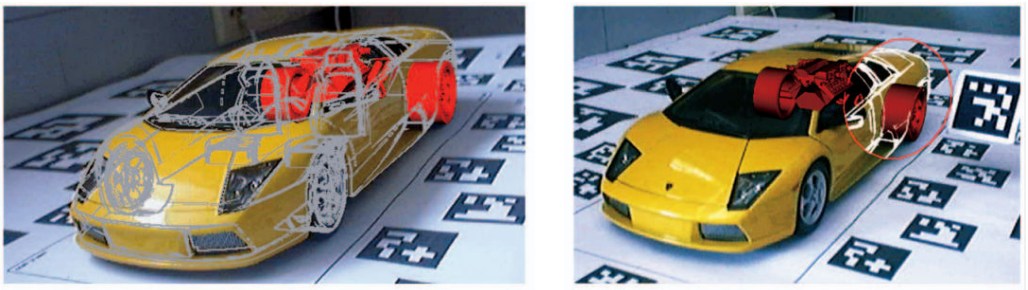
2.2 Reducing Data Overload

We use two techniques that alter the appearance of the data by removing items to compensate for data overload: filtering and clustering. While filters completely remove items from the presentation, clusters group them based on a defined distance function expressing their similarity.

2.2.1 Filtering

Augmentations can be filtered by employing spatial or semantic criteria. Feiner et al. [40] present a knowledge-based filter, which selects the appropriate information depending on the current step in a sequence of maintenance operations. Knowledge-based filters are the logical choice for showing sequences of operation, but can also be used to visualize larger data sets. For this purpose, a Degree of Interest function can be defined, which filters data based on priorities set by the user [46]. More complex filters use recommender systems [47, 107] that take user preferences and suggestions from other users into account to automatically suggest interesting items. However, such filters are not suitable for presenting an overview of unknown data, because the user has to specify a distinct goal first, or the system needs information about the user's preferences.

Spatial filtering in AR is often implemented as a magic lens [15, 93], which filters information in screen or object space (Figure 2.3). In contrast to the filter techniques presented in this thesis, magic lenses do not create overviews of data, because they work only locally in a small region. They typically require an undesirable amount of user interaction to grasp the data in its entirety.



(a) (b)

Figure 2.3: Spatial filtering using a magic lens. (a) Virtual edges are augmented onto the real car to provide depth cues for an x-ray visualization. Because of the large number of edges, the visualization is cluttered. (b) A magic lens filters the edges and reduces clutter. The edges are only shown at the spatial location of the lens (Images taken from Kalkofen et al. [72]).

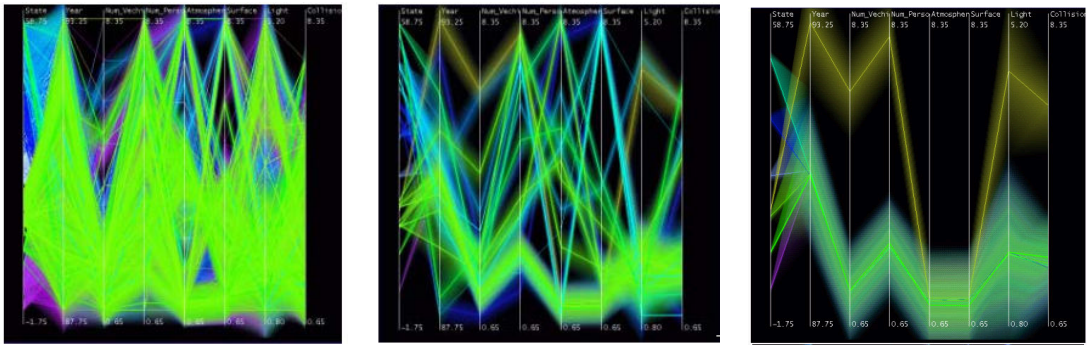
In applications for mobile phones, spatial filters are often combined with knowledge-based filters to provide location-based services [139]. Location-based services take the priorities and the current location of a user into account to present relevant items. In the context of AR, Julier et al. [70] present a hybrid filter model that filters information based

on the proximity of the user. Each item has a spatial location and a nimbus surrounding it. Once a user enters the nimbus of an object, it appears in the AR view. Knowledge about the user's task is integrated into the filtering to avoid presenting irrelevant items.

In this thesis, we present an approach to create compact visualizations that can also be seen as a hybrid filter. Compact visualizations consider the user's point of view as well as the semantics of the augmentations. However, unlike previous approaches, compact visualizations only filter redundant information and therefore do not lose any information. Furthermore, they do not rely on preset priorities for data items like [70], but provide an overview of all items, omitting only redundant data.

2.2.2 Clustering

Clustering also reduces the amount of data without any information loss. Instead of removing data from the input data set, like filtering, clustering aggregates a number of data points into groups based on a pre-defined similarity criterion. All data points of a group are then represented by a representative visualization that summarizes the content of the group. Typical representations in information visualization visualize the average or median of the items contained in a group. Hence, clustering provides the user with an overview of the available data.



(a) (b) (c)

Figure 2.4: Hierarchical clustering in parallel coordinate plots. The images visualize 230,000 data items from a fatal accident database. (a) Presenting all items at once leads to a cluttered presentation. By performing hierarchical clustering, different levels-of-detail can be calculated. (b) A higher level in the hierarchy visualizes more general trends in the data, while (c) lower levels communicate more details (Images taken from Fua et al. [44]).

User can interact with the clusters to explore the contained data items one by one.

However, with an increasing amount of data, the clusters grow too big to be explored individually. The introduction of hierarchies into the clustering process can remedy this problem [38] (Figure 2.4). A hierarchy of clusters can be created by either using a top-down or a bottom-up approach [31]. A top-down approach starts with the complete input data set and uses divisive methods to split the data into separate groups. A common clustering method is k-means clustering. A bottom-up approach based on agglomerative clustering starts with the separate data items and aggregates them into groups until all items are aggregated.

By performing different types of traversals through the hierarchy, an application can present different degrees of information to the user. Elmqvist et al. [38] differentiate above-traversal, below-traversal, level-traversal, range-traversal and unbalanced-traversal. Above-traversal show all clusters above a certain level, below-traversal shows all clusters beneath a level. Level-traversal selects all clusters that lie on the same level, while range-traversal can select multiple levels. Unbalanced-traversal is not restricted to a certain level, but can be performed in a way to show the most interesting clusters of the hierarchy.

In this thesis, we use hierarchical clustering to avoid clutter of annotations in screen-space. Using unbalanced-traversal, we select items from the hierarchy that match priorities set by the user. At the same time, our algorithm controls the selection of items in order to balance the location of the items in screen-space.

Note that in information visualization, clustering is not only used to reduce the amount of data, but also to find patterns in the data that would otherwise be hidden in the clutter of all items. Other application areas of clustering are image segmentation and object recognition [67]. In this thesis, we do not consider this notion of clustering, but use it only in a way to create groups of similar items.

2.3 View Management and Temporal Coherence

A major part of this thesis is dedicated to developing view management methods that achieve clutter-free layouts of annotations. In this context, we also investigate the creation of layouts of 3D objects in the form of explosion diagrams of assemblies. Explosion diagrams can be regarded as constrained view management problems, where the movement of 3D objects is restricted to their assembly directions.

Naturally, the aspect of temporal coherence is closely tied to the view management algorithm that creates the layout of the data. Therefore, we discuss this aspect together with the respective view management algorithms.

2.3.1 Annotations

Annotations enrich a user's visual perception with a variety of annotations such as text, image or even video data. Annotated objects appeared in the context of illustrations more than one hundred years ago [52]. However, the problem of placing labels was first discussed by cartographers [64] in the 70's. In the early 80's, computer graphics researchers started to develop algorithms that mimic manual label placement [2]. Despite the maturity of this research area, automatic label placement for interactive graphics is still an active topic of research [42].

Naively placing annotations generally leads to clutter and occlusion, impairing the effectiveness of AR visualization (Figure 2.5). For instance, occlusions among the annotations make the occluded labels unreadable (I). In addition, occlusions between the annotations and the object they refer to (II) or crossing leader lines (III) render the visualization very difficult to interpret.

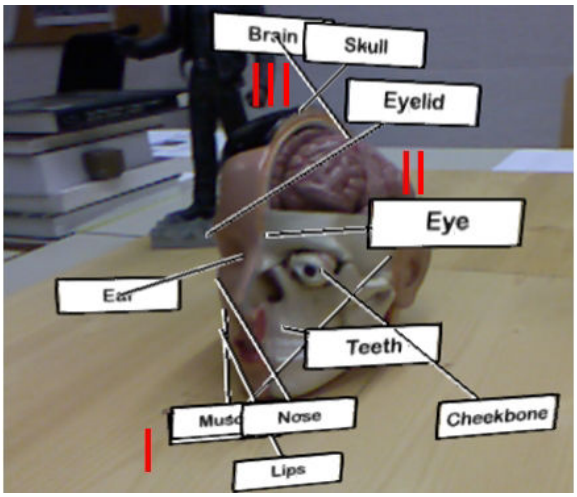


Figure 2.5: Occlusions when presenting external annotations. Without view management different types of occlusions may appear. (I) Annotations occluding each other. (II) Annotations occluding the object of interest. (III) Leader lines crossing each other.

In order to resolve clutter and occlusion issues, so-called view management techniques have been proposed for annotations [12]. These techniques automatically place annotations either directly on the surface of the object they refer to (using a so called internal label), or they place annotations outside the object of interest and draw a 2D line to its center (using a so called external label).

In traditional media, such as printed illustrations, an illustrator or graphics designer generally decides on the appropriate label type. The main factors influencing the selection

are the available space, personal preferences and intuition. Algorithms for automatically placing internal labels have been presented across several disciplines of computer graphics research, such as in volume visualization [68, 110], illustrative rendering [49, 88] and AR [12, 148]. Internal labels can be considered to operate in 3D object space, because they choose a position directly on the 3D object. Thus internal labels inherently support frame-coherent rendering, because they do not suffer from camera movements, if glued to the object of interest.

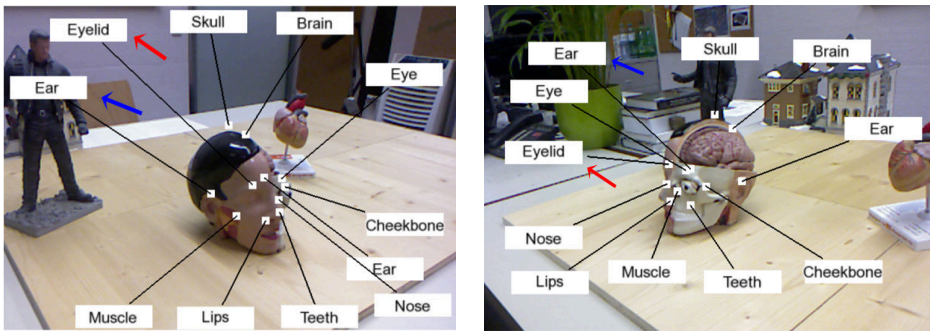
While internal labels support frame-coherent renderings, they require a certain amount of space to entirely fit on the object of interest [57]. Therefore, such approaches are usually limited to a rather small number of annotations. In addition, Coelho et al. [28] demonstrated that internal labels become ambiguous in AR when the registration error increases. This often makes them unsuitable for comprehensible information presentation in AR. We agree with this argument and focus in this thesis on view management techniques for external labels.

(a) (b)

Figure 2.6: Ambiguities when using internal labels. (a) Without registration error the labels (highlighted red) would fit into the projected area of the drawers. However, a small registration error creates ambiguities. The labels cannot be associated with certain drawers anymore. (b) The leader lines of external labels are more tolerant to registration errors and can resolve such ambiguities (Images adapted from Coelho et al. [28]).

A number of different techniques have been proposed to control the placement of external labels. They have been successfully applied to produce high quality layouts for desktop applications. However, since most of the existing techniques operate in 2D image space, they are prone to unpredictable changes over time. This happens because the screen-space distribution of the projected 3D points changes during camera movements, which can force the view management system to frequently re-order external labels in

image space. With increasing amount of label movement and re-ordering, the label motion becomes difficult to follow, and the resulting layout becomes unstable over time. This is illustrated in Figure 2.7.



(a) (b)

Figure 2.7: Problems of temporal coherence for annotations. (a) When applying view management techniques, occlusions can be resolved, but camera movements may cause unpredictable reordering of labels. (b) Rotating the camera causes the labels marked with a red and blue arrow to change their order in y-direction.

Annotations in 2D. Approaches for automatic external label placement exist as well. In their the seminal work, Bell et al. [12] propose a system for external and internal label placement at run-time. The approach is based on empty space management in image space, considering 2D screen aligned bounding boxes of annotations and objects in the scene. However, since this system does not consider leader lines, it may suffer from crossing leader lines or leader lines occluding annotations.

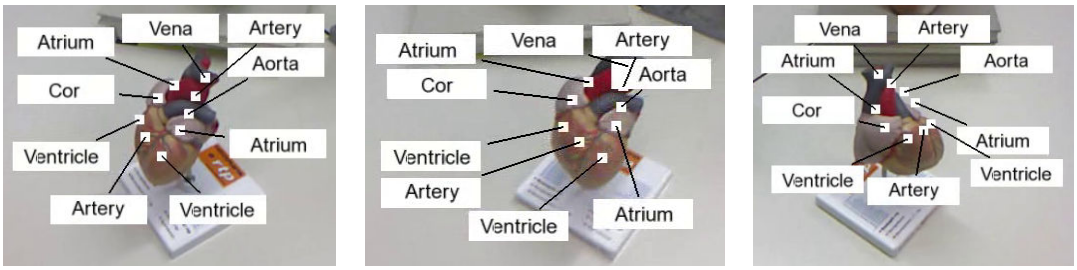
Hartmann et al. [56] propose an image-space approach, which evaluates a force field in every frame and updates the label layout accordingly. Their implementation considers leader line crossings and allows to align labels on non-rectangular shapes. This generates high quality layouts in static scenes without camera motion. However, in dynamic situations, the force field changes in every frame, causing labels to constantly move and often jump. The resulting layout is unstable over time,.

Similarly, Azuma et al. [8] apply simulated annealing to resolve colliding elements, while Rosten et al. [112] avoid covering important features in the environment by computing a force field from an analysis of the current video image. Grasset et al. [51] also use simulated annealing in combination with image saliency to guide the placement of annotations in image-space.

Force-based strategies compute a reasonable compromise, but the quality of each label's position inevitably deteriorates as the scene becomes more densely occupied. In addition, many algorithms ignore temporal coherence, which leads to unstable solutions and thus frequently changing locations of elements.

Bell et al. [12] use three techniques to achieve temporal coherence. They use hysteresis to avoid frequent changes of labels switching between internal and external modes. Furthermore, they take the previous position of a label into account when calculating its new position. Finally, to avoid jumping movements when labels change position, they smoothly animate the label to its new position.

To overcome the issue of temporal coherence for external 2D annotations, we use an image space approach which freezes the layout as long as the camera is in motion. Thereby, we avoid repositioning and also reordering labels in the visualization. Only after finishing the camera motion, we animate the annotations to their new optimal position.



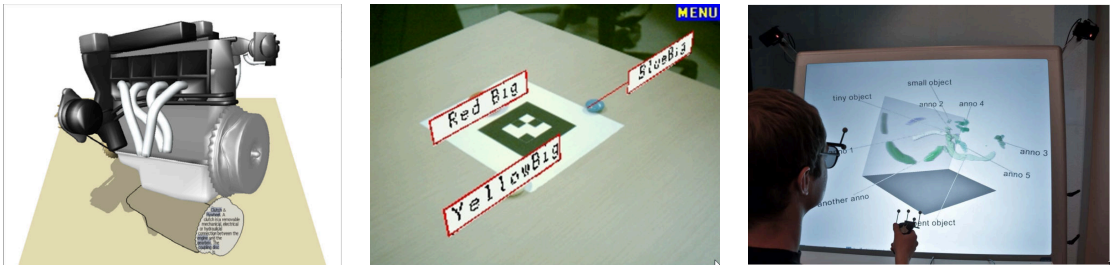
(a) (b) (c)

Figure 2.8: Positional lag when using screen-space annotations. Simulating and placing annotations in image space causes positional delays during camera motion, because the simulation has to update the annotations to their new positions relative to the object. (a) Labels are arranged around the object. (b) When the camera starts to move to a new viewpoint, labels stick to their image-space positions. Note the long leader lines to the left. (c) Annotations can only stabilize their position relative to the object when the camera stops moving.

Annotations in 3D. However, since this system still labels the object-of-interest (OOI) in image space, leader lines can easily cross. Furthermore, image-space annotations retain their pixel position during camera movement and must be moved to the new position relative to an annotated 3D object. This causes the annotation to lag behind the desired position (Figure 2.8). Therefore, we developed hedgehog labels as a novel view management approach that treats annotations as 3D objects. In contrast to screen-space

approaches, registering annotations in object space significantly reduces label motion when moving the camera.

While internal labels are often placed in 3D space on the object, related work that places external labels in 3D is rare. Chigona et al. [26] use an approach close to view management of external labels in 3D object space. The authors show annotations of the shadow of an object, which is projected to a single plane in 3D (Figure 2.9(a)). This plane can be considered as an external 3D label. However, the system is limited to the points of interest, which cast a shadow into this plane. Moreover, the extension of the shadow area restricts the amount of labels which can be placed, which is similar to the restrictions of internal labels.



(a) (b) (c)

Figure 2.9: View management in 3D. (a) Chigona et al. [26] annotate shadows of objects (Image taken from Chigona et al. [26]). (b) Shibata et al. [118] place labels on a 3D plane and apply very basic rules to create a layout. The 3D plane does not take the current viewing direction into account (Image taken from Shibata et al. [118]). (c) Pick et al. [104] create a label layout using a single 3D plane in front of the camera in a multi-display environment (Image taken from Pick et al. [104]).

Shibata et al. [118] place annotations on a 3D plane in front of the camera (Figure 2.9(b)). They resolve occlusions with other objects and annotations by displacing annotations on a circle surrounding the annotated object. This discretization of the placement limits the possible positions of labels, which can lead to overlaps. Hence, the algorithm does not scale well with the number of annotations. Furthermore, the plane orientation and, thus, the orientation of the annotations is not updated when the camera viewpoint changes. Therefore, it is not possible to freely explore a 3D object that is annotated using this method. In addition, the algorithm does not resolve crossing leader lines.

We also use 3D planes to constrain label movement in our 3D labeling approach.

However, the planes are placed relative to the annotated 3D object and updated when the camera viewpoint changes. In addition, annotations can be arranged freely in the plane, which makes our approach scale better with an increasing number of labels.

Pick et al. [104] present a view management approach that also treats annotations as 3D objects (Figure 2.9(c)). To resolve occlusion between annotations and objects, they use a 3D parameterization of a force-based approach as presented by Hartmann et al. [56]. Their goal is to provide a view management approach that works across multiple screens. Therefore, a 3D plane is introduced that is placed in front of the user. The 3D annotations then move parallel to a single plane, instead of moving on multiple screens. Plane and annotations are constantly updated, which can cause constant rearrangements of annotations, like in the 2D approaches.

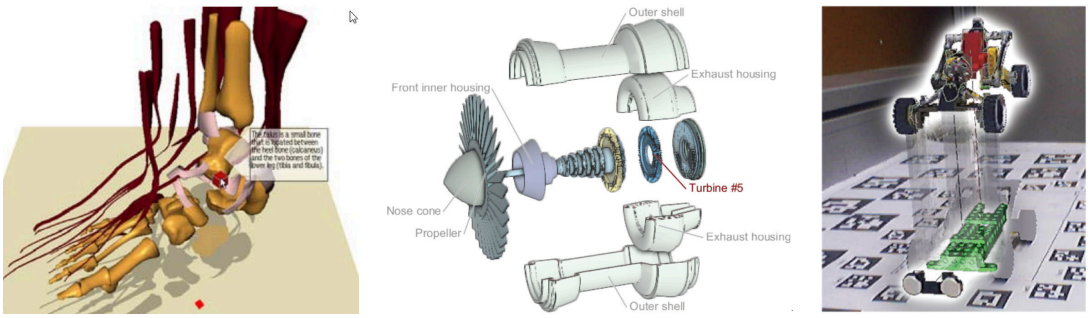
In contrast to Pick et al. [104], our hedgehog labeling approach is designed with temporal coherence in mind. In hedgehog labeling, the movement of annotations is much more constraint, to avoid rapid and constant label movement. We also avoid leader line crossing already in the initialization of the layout. A variation of hedgehog labeling also uses planes to constrain the movement of annotations. However, these planes are registered directly to the real world object. This allows us to freeze the position and orientation of annotations to achieve a stable layout that can be explored in AR.

2.3.2 Explosion Diagrams

Explosion diagrams visualize an assembly by displacing its parts against their assembly direction. In the context of 3D view management, we investigate explosion diagrams as heavily constrained layouts of 3D annotations.

Over the past 15 years, computer graphic researchers have investigated a number of different methods to automate the generation of explosion diagrams. These methods create explosion diagrams from many different kinds of data, ranging from 3D CAD data [108], triangle soups [100] and volumetric data [21] to 2D image data [84]. In addition, a number of different approaches have been presented to automatically compute the explosion's layout. Distortion techniques as presented by Raab and Rüger [106] scale occluding parts. Force-based techniques as presented by Sonnet et al.[121] and Bruckner and Gröller [21] use a set of interactively applied repelling and attracting forces, which define directions and distances for offsetting parts. Agrawala et al. [1] and Li et al. [83] use spatial blocking information between parts as well as a size analysis to automatically derive the relations and directions.

To control the visual complexity of an explosion diagram, the existing approaches mainly provide interactive techniques. For example, Sonnet et al. [121] presented an interactive system which moves parts of an object out of the 3D volume of an explosion probe (Figure 2.10(a)). Bruckner and Gröller [21] interactively define the amount and the relationships between forces to control the distances, direction and relative movements of parts. Li et al. [83] presented techniques, such as dragging or riffling of parts, to interactively explore a pre-computed explosion diagram, starting from a completely unexploded presentation (Figure 2.10(b)).



(a) (b) (c)

Figure 2.10: Explosion diagrams. (a) Sonnet et al. [121] use an interactive 3D probe to locally displace parts of an object (Image taken from Sonnet et al. [121]). (b) Li et al. [83] automatically calculate an explosion diagram from a 3D CAD model (Image taken from Li et al [83]). (c) Kalkofen et al. [74] use explosion diagrams in AR as x-ray technique to reveals the internal parts of a real world object. In this image, the explosion is computed to just reveal the base plate of the car (Image taken from Kalkofen et al [74]).

Even though research on rendering of explosion diagrams has often focused on interactive systems, few have investigated an automatic search of groups of parts to simplify the explosion layout. Thus, the works closest to our approach are the systems of Ruiz et al. [113], Kalkofen et al. [74], Agrawala et al. [1] and Niederauer et al. [100]. Niederauer et al. [100] attempt to explode the floors of a building, searching for those triangles which belong to a floor. Since different floors are usually offset at a certain distance and oriented similarly, Niederauer et al. were able to find groups of triangles by applying a statistical analysis of their locations and orientations. Ruiz et al. [113] define the thickness of parallel slabs of a volume, based on a similarity measure between neighboring slabs. The similarity values are computed using mutual information. While the former approach only performed well on structures similar to buildings, the latter is optimized for volumetric

data.

Explosions of groups of parts of a 3D CAD model have been presented by Kalkofen et al. [74], Agrawala and later Li et al. [1, 83]. While Agrawala et al. and Li et al. manually annotated their models with group information, Kalkofen and his colleagues automatically group elements based on a selected focus element, which they aim to uncover (Figure 2.10(c)). In a complete AND/OR-Graph data structure [63], they search for the largest groups of parts which can be displaced from the subassembly containing the object of interest. By recursively applying this search strategy on the AND/OR-Graph data structure, their approach is able to compute a Focus and Context explosion layout with an uncovered object of interest and a minimal number of contextual groups.

Our approach differs from Agrawala et al. [1], Kalkofen et al. [74] and Li et al. [83] in that we do not restrict ourselves to six main explosion directions, but allow all valid removal directions. Furthermore, we employ a sophisticated similarity measure [140] for identifying similar parts, which are then arranged in similar ways in the final explosion layout. Otherwise, it may happen that symmetric structures of assemblies are exploded in different ways (Figure 3.9). While Agrawala and Li et al. [1, 83] manually annotate the models with group information, we are able to find similar structures of the assembly automatically and use them to create compact exploded views. Although Kalkofen et al. [74] also group elements automatically, the found groups do not take into account any structural information of the assembly, like, for instance, similar subassemblies.

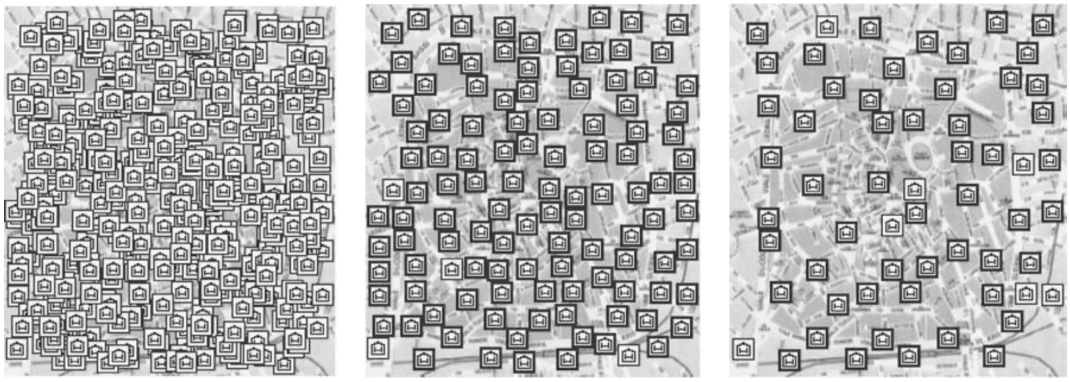
2.4 Combining Data Selection and View Management

In this thesis, we combine filtering and clustering with view management techniques to create the optimal layouts for the current viewpoint (see Chapter 3). Filtering the annotation data before arranging it in a layout allows choosing a few good locations rather than resorting to compromises. Part of this optimization is also to create layouts that make best use of the available screen space.

Maass et al. [87] and Bell et al. [12] combine filtering with view management by omitting annotations. Maass et al. [87] derive importance from depth values and can choose to not render labels beyond a certain distance. However, this approach is prone to frequently changing elements during camera movements. Bell et al. [12] filter annotations based on the visibility of referring elements and a pre-assigned importance value. Their algorithm arranges annotations with the highest priority first and stops adding annotations when layout constraints would be violated. This approach results in more stable layouts, but

manual preparation is required to assign importance values to objects, which is often impractical in dynamic AR environments. Our combined filter and layout approach automatically identifies relevant annotations based on a redundancy analysis and provides the user with a representative overview of all annotations.

In map visualizations, clutter is avoided by clustering items into groups to finally create an overlap-free layout of data. The clustering process typically aggregates items based on their distance in screen space to avoid overlapping items [22]. Note that clustering can be controlled to create layouts that increase the map legibility by increasing the spatial offset between clusters (Figure 2.11). This is also an important aspect for AR, where the presentation of an increasing amount of items also leads to an increasing amount of real world occlusion.



(a) (b) (c)

Figure 2.11: Clustering data in map visualization. (a) Presenting a large number of items in map visualizations leads to a cluttered display and overlapping items. (b) Aggregating neighboring items into clusters reduces clutter and overlaps, while filling the screen space with items. (c) To increase the map legibility, the spacing between clusters can be increased to reveal more of the map. For this purpose, more distant neighbors are aggregated into clusters (Images taken from Burigat and Chitarro [22]).

Woodruff et al. [150] take inspiration from map visualization and propose a constant density information display. Such a display tries to fill the available screen space with information. It also adapts the amount of presented information so that the information density is constant during zooming. To measure the density, the screen is divided into cells using a regular grid. Each cell can contain only a limited number of items. A placement algorithm fills the screen by combining manually defined layers of data.

We adopt a similar grid-based approach to fill the screen with items from a hierarchical

clustering approach. The approach of Woodruff et al. [150] attaches the grid to the viewport. This leads to flickering artifacts when the viewport changes, because the position of the data items in the cells change. To achieve temporal coherence, we perform a spatial subdivision around the viewpoint of the user. Therefore, data items are always assigned to the same cells of the grid.

Dix and Ellis [32] propose to use random sampling to select samples from data sets that are also balanced over the available screen-space. When returning to previously visited viewpoints, the user should see items that have been shown when visiting the viewpoint the first time. However, random sampling is not deterministic and will not select the same items again. Using a viewpoint history that stores previously selected items can solve this issue, but is hard to implement for AR applications that can exhibit random camera motion.

2.5 Extending the Ego-centric Viewpoint

One of the main goals of our view management techniques is to provide overview visualizations that are a starting point for further interaction, e.g. by performing zooming and filter operations. The inherent ego-centric viewpoint of AR limits the ability of users to get an overview o the presented information. Therefore, we also developed methods to compensate for this ego-centric viewpoint.

In information visualization, overview&detail techniques were developed to compensate for the viewpoint limitation of users exploring a large data set [27]. The current view of the user is regarded as the detail, while an additional visualization provides an overview of the surroundings. Previous work in AR also recognized the need to extend the ego-centric viewpoint of users to provide better overviews of the data and the real world.

A common solution in commercial AR browsers is the combination of the AR view with a radar or a map that visualize the Points-of-Interest (POIs) around the user. Such visualizations clearly provide an overview, but are disconnected from the real world. Hence, the user has to mentally map the location of a map to a real world location. Another solution is using offscreen visualizations in AR, which indicate POIs around the user [69]. However, offscreen visualizations only provide a directional indication and still require the user to physically change location to investigate the POIs.

In this thesis, we focus on two additional techniques to extend the ego-centric viewpoint of the user: transitional interfaces and multi-perspective renderings. Transitional interfaces provide seamless transitions between different views of AR and VR, thereby

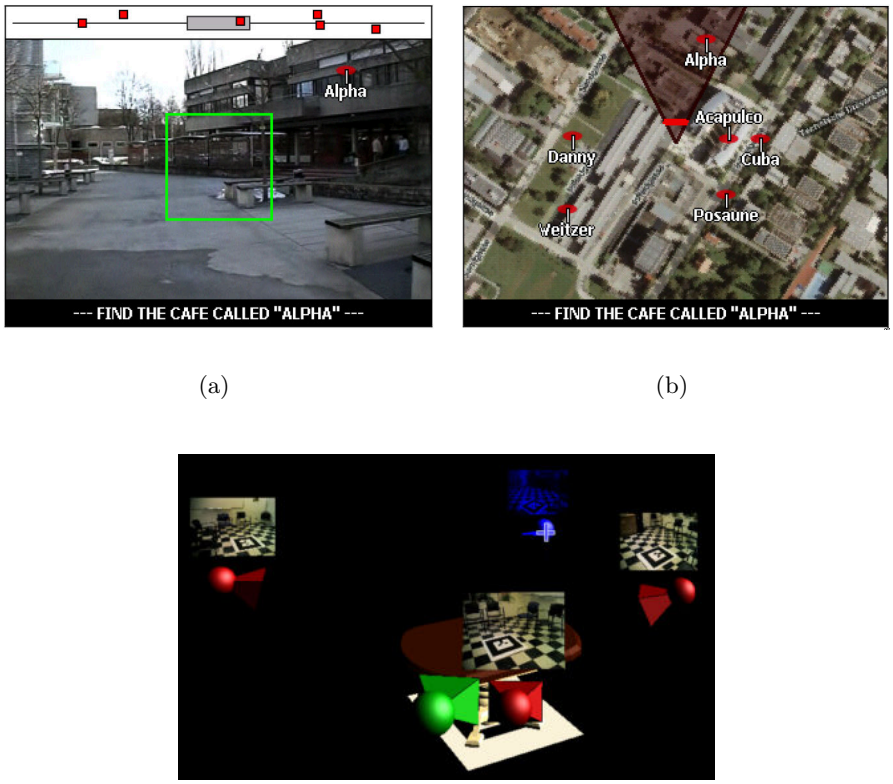
allowing a user to change the viewpoint without physically moving. Furthermore, in contrast to traditional map visualizations, the seamless transition provides a spatial cue that allows users to mentally map the viewpoints. Multi-perspective renderings extend the AR of users by integrating additional views of the real world into their current view. In both approaches, the viewpoint of users is extended without them physically changing location.

2.5.1 Transitional Interfaces

Bowman et al. [19] present instant teleportation techniques in VR environments and discover that the lack of continuous motion cues causes user to be disoriented. Consequently, in the first transitional interface used in AR, Kiyokawa et al. [77] allow users to seamlessly switch between a virtual and an augmented collaborative workspace. Seamless transitions are also provided by the MagicBook [16], which allows users to switch between an exocentric AR view on a VR scene and an immersed egocentric VR view on the same scene. In contrast to this previous work, our interfaces switch from an egocentric to an exocentric viewpoint and are designed for exploring real world objects.

Avery et al. [5] and Mulloni et al. [97] switch to exocentric viewpoints to provide an overview on the surroundings. However, the overview is focused on the user's position and does not allow viewpoint changes around a focus object (Figure 2.12(b)). Sukan et al. [124] and Veas et al. [138] allow switching to already established viewpoints and perform transitions to these viewpoints. In contrast to our 3D interfaces, they provide only access to a discrete number of views (Figure 2.12(c)).

A common solution to realize transitional interfaces is using a World-in-Miniature (WIM) [123] to complement the egocentric view of the user. Bell et al. [13] use a WIM in AR that shares annotations with the real world. Depending on the viewpoint of the user, the annotations transition from real world object in AR to the VR copy. This concept is similar to the visual links we present in this thesis. They connect the virtual and real worlds. Unlike shared annotations, in which the annotation either addresses the real world or the virtual world, visual links always connect both real and virtual worlds. Bane and Höllerer [10] present a WIM interface, in which users are able to seamlessly switch to a copy of an occluded room and interact with this copy. Similar to OCE techniques, the WIM interfaces for AR provide copies of real world objects. However, unlike our interfaces, these interfaces were designed for head-mounted displays (HMD), not handheld devices.



(a) (b)

(c)

Figure 2.12: Transitional interfaces. Transitional interfaces extend the AR viewpoint by allowing users to switch to other viewpoints without physically moving. (a) The egocentric viewpoint limits the field-of-view of the user in an AR browser. (b) By seamlessly switching to an exocentric map view, the user gets an overview of the points-of-interest relative to the current location (Images taken from Mulloni et al. [97]). (c) Transitional interfaces can also enable users to revisit previously recorded viewpoints (Image adapted from Sukan et al. [124]).

2.5.2 Multi-perspective Rendering

Multi-perspective renderings are often realized using mirror-like renderings. Bichlmeier et al. [14] use a mirror to reveal those parts of a table-sized object that face away from the user. Au et al. [4] demonstrate how mirrored views from a live video facilitate orientation in urban environments (Figure 2.13(a)). Hoang and Thomas [58] also use live video to extend the AR view by providing a zoomed view of distant objects to improve interaction accuracy.

Deformation techniques can also provide additional viewpoints on otherwise occluded

(a)

(b)

(c)

Figure 2.13: Multi-perspective rendering. Multi-perspective renderings expand the current viewpoint of the user by showing multiple viewpoints integrated into one rendering. (a) Additional viewpoints can be integrated using a mirror metaphor (Image adapted from Au et al. [4]), (b) or by applying deformation techniques to real world objects (Image taken from Sandor et al. [115]). (c) Panorama renderings also have been used to present multiple viewpoints in one image (Image taken from Mulloni et al. [98])

structures or structures that are out of view. Sandor et al. [115] deform real world buildings to enable users to investigate parts not visible from their current viewpoint (Figure 2.13(b)). Veas et al. [137] seamlessly integrate an exocentric viewpoint into the egocentric AR view to provide an overview over a large area. While these techniques extend the egocentric viewpoint, they do not allow viewpoint changes for exploring distant objects.

Panorama renderings also have been used to extend the viewpoint of the user. Mulloni et al. [98] evaluate different panoramic representations of the users' surroundings by letting them match features in the panorama and the real world (Figure 2.13(c)). In another

paper, Mulloni et al. [97] increase the users' egocentric field of view (FOV) by zooming in on a panoramic presentation of the surroundings. The use of a transition to change the viewpoint of the user also classifies it as transitional interface. This example shows that the borders between interface categories are blurred. In this thesis, we also combine a transitional interface with a multi-perspective rendering of an object.

Similarly, the transitional interfaces of Veas et al. [138] and Sukan et al. [124] register images of the available viewpoints in the real world, thus creating a multi-perspective rendering, which is similar to our transitional interface. However, their designs are not focused on exploring a single, real world object of interest, but aimed at communicating available viewpoints of the environment [138], or manipulating VR content in AR [124].

A major disadvantage of multi-perspective renderings over transitional interfaces is that the renderings typically do not allow the user to freely change the viewpoint of the rendering. However, in combination with transitional interfaces, multi-perspective renderings can be used as overview visualization for further interaction.

Chapter 3

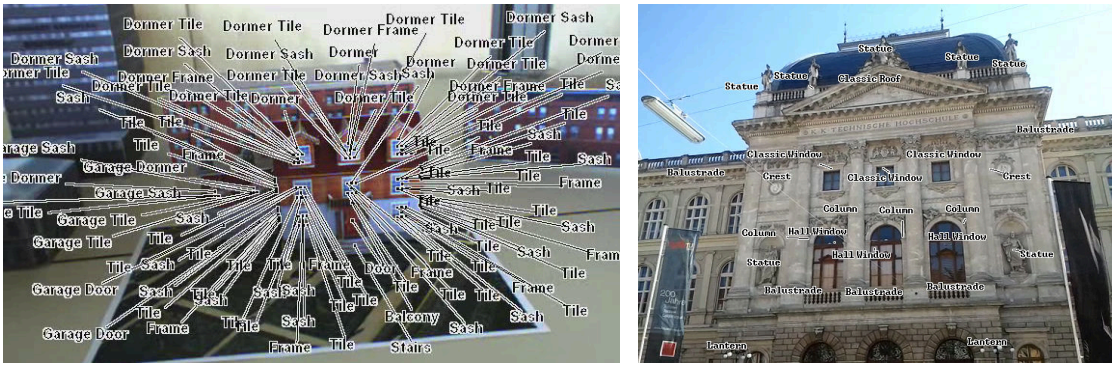
Combining Filtering and View Management

Contents

AR is a powerful tool to aid the exploration of physical objects. For example, AR explosion diagrams [74] use three-dimensional displacements to reveal the internal structure of an object assembly. AR is also frequently used to present textual or pictorial annotations of real-world objects [61], a technique now commonly used in commercial AR browsers on mobile phones.

Such AR applications often rely on legacy databases or on crowdsourced content, which provide a high density of data for popular subjects or locations. Another source of abundant data are image recognition algorithms, which automatically detect objects in videos and create corresponding information tags. However, augmenting the environment with a large amount of visual information frequently causes perceptual problems. For instance, such augmentations may occlude important real-world landmarks, may be occluded by other objects in the environment, or may cause excessive screen clutter (Figure 3.1).

Hence, filtering techniques are necessary in order to reduce the content of the database to a manageable amount. A common approach is to use a sequence of filter and layout algorithms to visualize a reduced set of the data [33]. However, this approach often ignores the comprehensibility of the resulting layout, because filtering and layout generation are



(a) (b)

Figure 3.1: Data overload. (a) Unfiltered augmentations may quickly lead to clutter and thus decrease the comprehensibility of the resulting visualization. (b) Real world structures can be annotated automatically by applying object recognition techniques. However, the generated amount of information is unpredictable and may easily result in an overflow of augmentations. The application presents all available data including redundant labels, such as the five labels to the balustrades.

considered in isolation. This can lead to a number of problems:

- Items of the visualization may cluster in one region, thereby competing for optimal positions and degrading the overall layout. For example, a simple distance-based filter can reduce the overall amount of objects, but does not consider areas of high density.

- Ideally, the information density on the screen should be stable and therefore should not vary when changing the viewpoint. This can only be guaranteed by a view-dependent filtering approach, which takes into account that the layout should be temporally coherent.

- Most filters use static selection criteria or greedily pick objects based on priorities. Such simple selection mechanisms do not guarantee an optimal selection from the available data. Multiple objects of the same type may be annotated, thus introducing redundancies, and certain parts may not be annotated at all, leading to information loss.

We present a filter and layout technique for AR, which follows the information seeking mantra of Shneiderman [119] (overview first, zoom&filter, details-on-demand). Our

approach presents an overview of the available data, which avoids the aforementioned problems and can be used as a starting point for further interaction.

3.1 Compact Visualizations

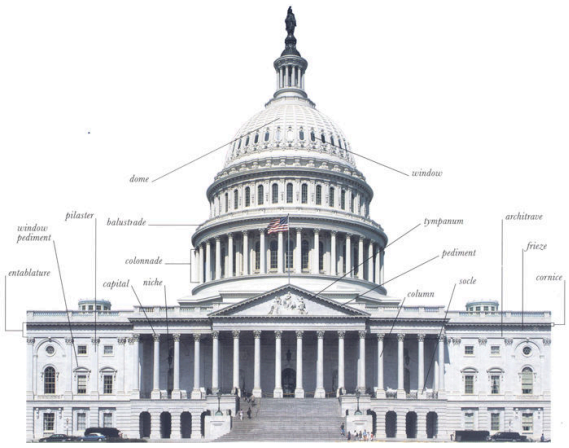


Figure 3.2: Illustration of architecture by labeling its elements. Note that from a group of similar elements, only a single representative is annotated.

Inspired by handmade illustrations such as shown in Figure 3.2, our filter preserves the information encoded in the visualization by removing only redundant elements and choosing a representative item for each object class. These representative items are able to show all object classes without information loss.

We iteratively evaluate the layouts during filtering and reject those that are not satisfying given layout constraints, thereby creating *compact visualizations*. By dividing the computations into an offline preparation and an online phase, we are able to deliver optimal compact visualizations in real time even for mobile AR platforms that may not have access to a large amount of computational resources.

In order to demonstrate the generality of our approach, we apply it to create compact layouts for explosion diagrams, textual and pictorial annotations, and a combination of explosion diagrams and annotations.

3.1.1 General Framework

In a typical information visualization pipeline [23], the input data passes sequentially through stages for filtering and mapping, before the output is presented to the user. Filtering is responsible for reducing the amount of presented data and then forwards

the filtered data to the visual mapping and then the view transformation stage, which finally creates a layout of the data. However, the filter has no knowledge about how the filtering will influence the final layout. The layout algorithm may have difficulties finding a comprehensible layout for the data, which may be clustered in one region.

Our combined filter and layout approach solves this difficulty by introducing an optimization loop between the mapping and filtering stage. This loop selects new variations of filtered items in each iteration and evaluates the resulting layout, until a comprehensible solution is found. The definition of comprehensibility depends on the desired visualization method. For instance, layouts of annotations must satisfy the condition that annotations are placed close to the annotated part [57], while in explosion diagrams all exploded parts must be visible [132].

3.1.1.1 Clustering Redundant Data

Filtering that unconditionally removes items from the input database may lead to undesirable information loss, because not all relevant information is communicated to the user. In order to avoid such information loss, compact visualizations present a minimal set of representative items from the database, which faithfully represent the input data.

We determine the minimally required amount of data through clustering the input data by similarity in a data analysis step. Then, from each cluster, one representative item is selected to be shown in the layout. Thus, we remove redundant items in the visualization.

For instance, we cluster textual annotations by string comparison or objects in the video image based on the recognition of object classes. For explosion diagrams, we use a shape descriptor for identifying similarities in 3D structures.

3.1.1.2 Layout Creation

To generate an optimized filtered layout, we first compute an initial layout, which we subsequently optimize (Figure 3.3). Both stages consist of a filter module that selects a single item item from each cluster, the algorithm for creating a layout of the selected items and finally an evaluation module, which determines the quality of the layout.

Although the structure of the two stages appears to be the same, their purpose and details differ. In the first stage we determine an **initial selection** of representatives from the previously determined clusters. For this purpose, it evaluates the quality of each single item. After all items have been evaluated, we chose the item with the highest quality measure from each cluster to be the representative of the respective cluster. To find out

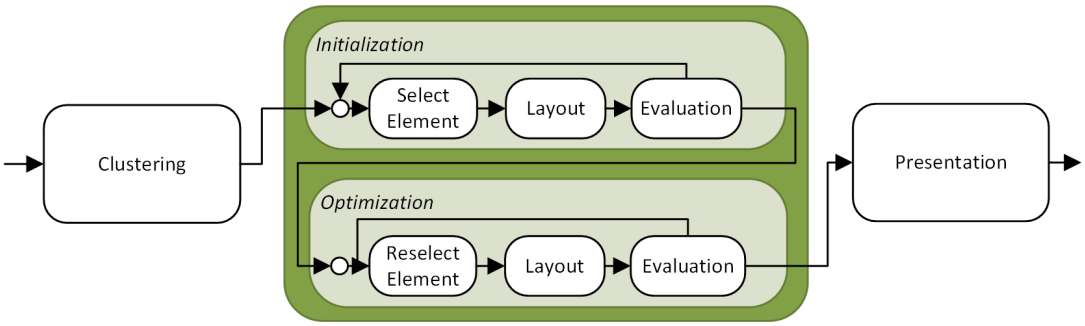


Figure 3.3: General framework for compact visualizations. After clustering the input data according to redundancies, an initial layout of representatives is created. If the initial layout is not satisfactory), the representatives are re-selected in an optimization process to find an interference free layout. Internally, the data passes through (re-)selection, layout generation and layout evaluation in both the *initialization* and the *optimization* phase. While the initialization iterates over each single item in the clusters, until all have been processed, the optimization varies the layout by a single representative in the filter stage for a defined number of cycles.

if the selected representatives influence each other, we create their combined layout and measure each item's quality again. If the quality of each representative is the same in the combined layout as the previously determined quality, the selection and the layout is optimal and the filtering process is finished. Otherwise, the selection is forwarded to the next stage for optimization.

The second stage performs the actual **layout optimization** by iteratively re-selecting and re-evaluating representatives. Iteratively re-evaluating the entire layout is a time consuming process. Therefore, in order to optimize the quality of the layout within an acceptable amount of time, a heuristic optimization method, threshold accepting [34], is applied. Threshold accepting randomly varies one item of an input configuration in each iteration to find an optimal solution. The best configuration is then stored. However, configurations of lower quality are investigated during optimization to escape only locally optimal configurations. For details on the optimization, we refer to Dueck and Scheuer [34].

To achieve a compact visualization for a certain data type, we have to specialize the implementation of the clustering, the layout algorithm and the evaluation responsible for determining the quality of each item and the layout. We define different quality criteria to achieve comprehensible visualizations, which take into account the structure of the layout. Moreover, we demonstrate how we can achieve scene-aware layouts (Section 3.1.6.3) and temporal coherence (Section 4.1) by introducing special quality criteria.

In the following, we demonstrate the general applicability of our framework by spe-

cializing it for different data types and applications. We create compact layouts for annotations, which are commonly used in AR for presenting additional information about the real world. Furthermore, we apply the framework to create compact explosion diagrams. Finally, we demonstrate that the modular structure of the general framework allows for combined optimization of different data types by combining explosion diagrams and annotations.

3.1.2 Compact Annotations

To create compact annotations, we first annotate real world objects using their registered virtual counterparts, given as a 3D CAD model. Each part of the model has been assigned a semantic tag, which we display as annotation. Semantic annotations may consist of text, icons or images, and can be created either manually or through automated recognition. The initial layout of annotations is created using the force-based approach of Hartmann et al. [56] that arranges annotations in image-space.

The **clusters** of similar annotations, from which the representatives are chosen, are currently determined by simple text comparison, but could use a more sophisticated matching such as by using a semantic network. In the **initialization stage**, the layout of each label is evaluated separately. The layout is determined by the layout algorithm, which resolves collisions between labels and geometry. Afterwards, the initial layout is created by combining representative labels from each cluster. We select a representative label from each cluster depending on the size and visibility of the referred part as well as the distance to that part. The closer a label L_i is placed to the part P_i and the higher the visibility of this part, the easier it is to understand the relation between label and part [57].

The quality criteria used during the initialization are described in (3.1). The number of visible pixel $NumVisiblePixel(P_i)$ of the labeled part is computed by projecting the part to screen space and, thereby, also considering occlusions from other scene elements. $NumTotalPixel(P_i)$ refers to the total number of pixels after projection to screen space, without considering occlusions. We introduce weight parameters w_d, w_v and w_s to control the impact of anchor point distance, visibility and size of the annotated part on the quality of the label.

$$
\begin{aligned}
Q_{Label_i} &= w_d \cdot \frac{|Position(L_i) - Position(P_i)|}{\sqrt{ImageWidth^2 + ImageHeight^2}} \\
&+ w_v \cdot \frac{NumVisiblePixel(P_i)}{NumTotalPixel(P_i)} \\
&+ w_s \cdot \frac{NumTotalPixel(P_i)}{ImageWidth \cdot ImageHeight}
\end{aligned}
\tag{3.1}
$$

(a) (b)

Figure 3.4: **Compact Annotations.** (a) Similar items have been clustered and representatives have been selected from each cluster. This allows us to reduce the amount of augmentations, while still presenting an annotation to each available object class. In addition, our system allows to control the selection of representatives according to design rules. In this case, we select those annotations which most evenly distribute around the house. (b) After filtering the data, only a few labels remain. They have been spread over the image plane to avoid interferences with other labels during camera motion.

In the **optimization phase**, the layout algorithm additionally resolves overlapping labels and leader line intersections. Thus, the optimal positions of the labels computed during the initial stage may change. To keep the mutual influence of labels to a minimum, the initial layout is refined during the layout optimization by selecting and re-evaluating different representatives. Furthermore, Hartmann et al. [57] suggest that labels should be placed at similar distances. Therefore, for each pair of labels, which are direct neighbors on the bounding geometry of the annotated geometry, we compute their distance $D(i, i+1)$ and compare it to an estimated optimal distance D_{opt} between labels, which is defined by the projected bounding box dimensions of the geometry (Dim_x, Dim_y) and the number

of labels to be placed. D_{opt} assumes a layout, where labels are placed at the periphery of the annotated assembly.

$$\begin{aligned} D(i,i+1) &= \frac{|Position(L_i) - Position(L_{i+1})|}{\sqrt{ImageWidth^2 + ImageHeight^2}} \\ D_{opt} &= \frac{\frac{1}{n} \cdot 2 \cdot (Dim_x + Dim_y)}{\sqrt{ImageWidth^2 + ImageHeight^2}} \end{aligned} \tag{3.2}$$

We estimate the quality of the entire layout (3.3). Adjusting the weights w_d, w_v and w_s in combination with the weight w_l allows us to balance the quality of a single label versus the quality of their distribution within the layout.

$$\begin{aligned} Q_{Distance} &= \frac{1}{n} \sum_{i=1}^{n-1} 1 - |D(i,i+1) - D_{opt}| \\ Q_{Layout} &= \frac{1}{n} \sum_{i=0}^{n} Q_{Label_i} + w_l \cdot Q_{Distance} \end{aligned} \tag{3.3}$$

Figure 3.4(a) shows the compact visualization of the annotations of the house from Figure 3.1(a). The redundant annotations have been removed, and only a minimal set of annotations remains, which refer to visible parts of the house. Note how these annotations are distributed around the house.

The selection strategy of the filter can be constrained to enforce certain layout styles. One strategy may group labels to annotate the same sub-structures and thus create an "annotated example" pattern. Thus, when the algorithm chooses representatives from the clusters "sashes" and "tiles", these representatives must refer to the same window. Although the selection is restricted, the optimization process distributes annotations around the object and also chooses visible parts as representatives.

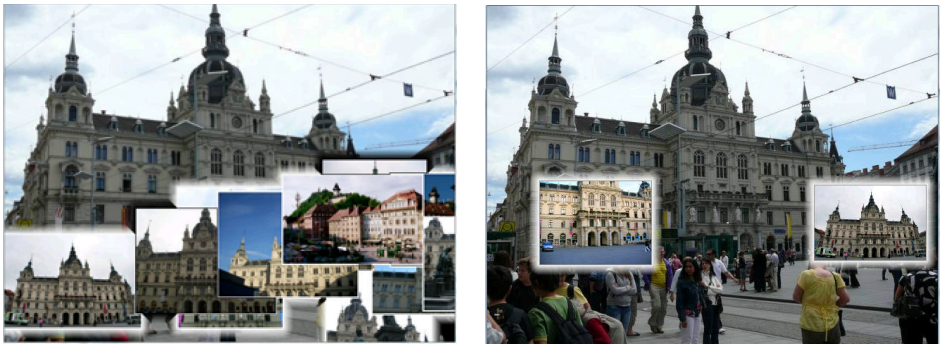
AR applications can also use databases that are the result of automatic processing, such as image segmentation and recognition, or that are created by crowdsourcing. Compact visualizations can be employed to handle this growing amount of data. In Figure 3.1(b) object classes are recognized automatically in a visual search task and converted into annotations. Without filtering the annotations, the displayed data leads to clutter in complex environments. We can easily adapt the approach of compact annotations to

handle such cases.

Within each cluster of an object class, we rank objects based on their screen size, estimated by a 2D bounding box. To avoid clustering of annotations in the video image, the optimization distributes the annotations over the image (Figure 3.4(b)).

3.1.3 Compact Photo Collections

Recent AR browsers allow exploring geo-referenced photographs. In this section, we show how our approach can control the clutter resulting from an overload of images (Figure 3.5(a)).



(a) (b)

Figure 3.5: Compact Photo Collections. (a) Geo-referenced photographs are displayed as billboards in their respective location. The unfiltered pictures clutter the display and occlude the displayed landmarks. (b) Our compact visualization algorithm clusters photographs based on their visual similarity and spatial proximity. The clutter is reduced by selecting representative images from the detected clusters.

Clustering. We assume a database of images tagged with GPS coordinates. Images are clustered by identifying similar content using the algorithm described by Li et al. [86], giving one cluster C_{L_i} per landmark. Using the GPS tag (i. e., the camera position) of the image relative to the GPS position of the landmark, we determine the orientation of each image. Within a landmark cluster, sub-clusters C_{O_j} with similar orientation are computed using k-means [89].

Initialization Stage. Since a single image requires a rather high amount of screen-space, we show only small and simple icons at the location of each visible landmark. We

furthermore allow the user to select one of the icons in order to query the images associated with the corresponding landmark.

For each landmark, we present representative images to its left and right in screen space (Figure 3.5(b)). These are selected from the orientation sub-clusters C_{O_j} based on their distance from the landmark. Images taken from a distance similar to the current distance of the user to the landmark are ranked higher than those which are further away or closer to the landmark.

Optimization phase. To evenly distribute representative images around a selected landmark, the differences between orientations of representatives are considered. We distribute representatives as evenly as possible around the object using the quality measure presented in Equation 3.4.

$$QualityDistribution_j = \sum_{i=1}^{numLabels-1} Angle_{i-1,i} - Angle_{i,i+1} \tag{3.4}$$

3.1.4 Compact Explosion Diagrams

Compact visualizations can also be created for explosion diagrams. Explosion diagrams are a powerful visualization that allow for comprehensible renderings of three dimensional objects. While other illustrative exploration techniques, such as cutaways [85] or ghostings [72] remove parts from the presentation, explosion diagrams present all parts entirely opaque and with full detail, by displacing the elements of an objects. These displacements are carefully designed to encode the assembly of the object. Artists are trained to intuitively choose comprehensible arrangements of parts. In computer science, algorithms have been investigated to compute the layout of an explosion diagram automatically. For example, the current state-of-the-art algorithms define relations between parts, which are subsequently used to control their displacement [1, 83].

However, similar to crowdsourced annotations, automatically generated explosion diagrams of complex objects can easily suffer from cluttered layouts. While artists intuitively decide which parts of a complex explosion diagram are really necessary, computer graphics applications have to resort to user interaction to control the complexity at runtime [83]. Such interaction requires a certain effort, which increases with the complexity of the 3D model. In addition, traditional media, such as textbooks, do not allow interaction with the

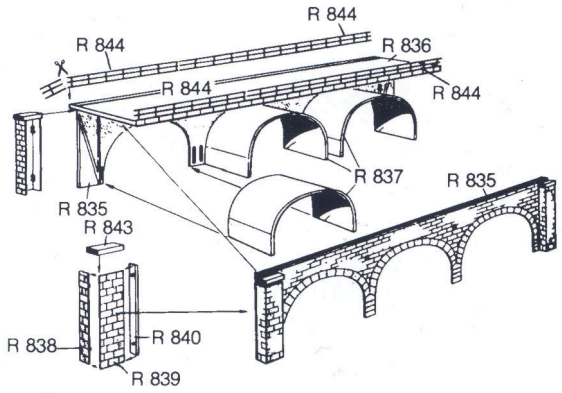


Figure 3.6: Handmade compact explosion diagrams. The assembly of the entire model is presented by a set of representative exploded views only (Adapted from [94]).

presentation at all. Illustrations targeted for non-interactive media still need to present the entire assembly of an object.

Inspired by handmade illustrations showing explosion diagrams, such as the one in Figure 3.6, we reduce the complexity of an explosion diagram by rendering an exploded view only for a subset of the assemblies of an object using our compact visualization approach. The exploded views are chosen so that they allow inferring the remaining unexploded assemblies of the entire 3D model. Note how the illustrator of Figure 3.6 uses a selective displacement multiple times to render a more compact explosion layout.

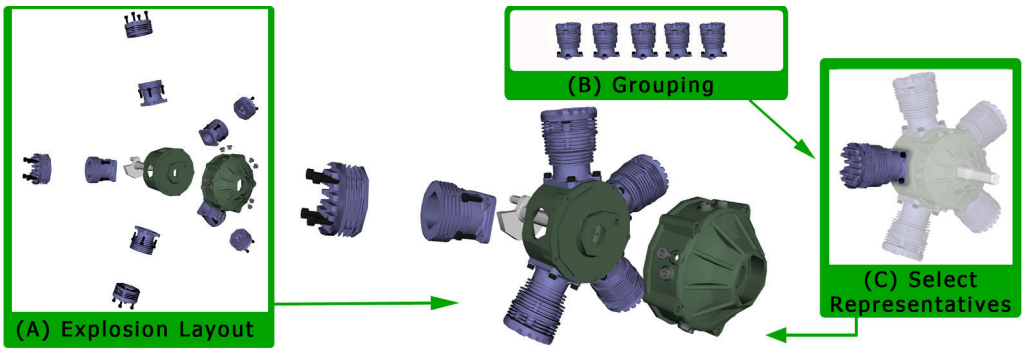


Figure 3.7: System architecture for creating compact explosion diagrams. The system for creating compact explosion diagrams system consists of three different modules which affect the rendering of compact explosion diagrams. By supplying a 3D CAD model, the system automatically computes an initial explosion layout (A), finds groups of equal parts (B) and selects a representative (C), before it initiates the rendering.

3.1.4.1 System Overview

Using the framework for compact visualizations, we can create explosion diagrams that mimic the layout techniques presented in Figure 3.6. The process for creating explosion diagrams is very complex. Therefore, we provide a short system overview in this section. Figure 3.7 illustrates the main modules of our system, which are used to render a compact explosion diagram from a single point of view. The following sections contain more detailed descriptions.

Initially, an input assembly is fully exploded (Figure 3.7(A)) using the method described in Section 3.1.4.3. The presented method identifies similar parts of the assembly by employing the shape descriptor of Vranic [140] and ensures that these similar parts are exploded in similar ways. By performing a frequent subgraph search [151] on a graph representation of the assembly, which incorporates similar parts, we are able to extend the similarity measure from single parts to similar subassemblies (Figure 3.7(B)).

In general, the creation of the explosion layout (Figure 3.7(A)) and finding similar subassemblies (Figure 3.7(B)) are independent operations. Therefore, their detailed descriptions are separated into different sections. However, we incorporated the detected subassemblies into the explosion layout generation, to ensure that not only similar parts, but also similar groups of parts, are exploded in a similar way. See Section 3.1.4.3 for a detailed discussion on the modification of the initial layout algorithm and on the variations on using and creating similar groups.

Using the compact visualization optimization, we select representatives from the set of similar groups (Figure 3.7(C)), depending on a quality evaluation of its potential exploded view, including parameters such as their visibility, their size after 2D projection and the angle between explosion direction and the view vector (Section 3.1.4.4). Moreover, our system takes into account visibility information of the remaining unexploded assemblies. This allows rendering a balanced compact explosion diagram, consisting of a clear presentation of both the exploded representatives and the unexploded remaining assemblies. Since representatives may interfere with one another, the compact visualization framework optimizes combinations of representatives using the approach of threshold acceptance.

3.1.4.2 Clustering Redundant Assembly Groups

We determine sets of similar subassemblies by performing a frequent subgraph (FSG) search on a graph representation of the assembly. Our approach is based on the gSpan algorithm of Yan and Han [151], which uses depth-first-search (DFS) codes to differenti-

ate between two graphs. A DFS code describes the order in which parts of a subgraph have been visited. Two graphs are isomorphic if their DFS codes are equal and if their corresponding node labels (which represent the parts) match. By using DFS codes and node labels, the implemented FSG algorithm finds non-overlapping sets $S = \{G_1, \ldots, G_k\}$ of the largest subassemblies G contained in the graph.

Graph Representation of Assembly The FSG requires the 3D model to be represented as a graph A_g, which contains all parts $P = \{p_1 \ldots p_n\}$, with n being the amount of parts in the assembly. The parts of the assembly p_i (with $i = 1 \ldots n$) are mapped to an equal number of nodes of the graph. Undirected edges are created between nodes, if their corresponding parts are in contact.

Nodes of parts, which are similar to each other, receive the same label. We detect similar parts by exploiting the DESIRE shape descriptor of Vranic [140]. The descriptor computes a feature vector for each part which we use to compare their shapes with. We consider two parts as being similar, if the l2-distance of their corresponding feature vectors falls below a certain threshold and if the part sizes match. The result of the part comparison is a list of disjoint sets of similar parts $P_s = \{p_g, \ldots, p_h\}$, for $g \neq h$, and $g, h \leq n$, which is used to label the nodes of the graph A_g.

Frequent Subgraph Mining Input to the algorithm is the whole graph A_g. Initially, all nodes having a label which occurs only once in the graph are removed. These nodes represent parts for which no similar parts exist ($|P_s| = 1$). For each remaining set of similar parts P_s, one set S_0 is created, containing $|P_s|$ number of groups G_0, each containing a single part $p \in P_s$. The sets S_0 define the nodes at which the FSG search will start execution.

A recursive FSG mining procedure is applied on each of the sets S_0 and iterates through all input groups G_i of an input set S_i, in order to grow the groups G_i to create similar groups of parts. In each iteration, a different group G_i is chosen from S_i to be the reference group G_r. For the current group G_r, the set of neighbors N_r is retrieved for the node which was added last to the group G_r. If all neighbors of the node added last have been processed, the neighbors of the previously added nodes are chosen. If all neighbors have been visited, the group G_r cannot be extended further.

For each $G_i \neq G_r$, the neighbors n_i similar to the ones in N_r are determined. Neighbors n_i are similar to each other if their labels and number of contact parts to the corresponding group G_i are equal to the ones of the neighbor n_r. Furthermore, the DFS codes and labels

of the contact nodes contained in the groups must be equal. This similarity measure ensures that the found groups contain nodes which have been visited in the same order and which have equal relations to their neighbors. After identifying similar neighbors for at least two groups G_i and G_j during the same iteration, a new set S_n is created. The new set contains the groups $G_{n1} = G_i \cup n_i$ and $G_{n2} = G_j \cup n_j$, which are the original input groups extended by the similar neighbors. All groups for which similar neighbors exist are extended in the same way. Note that, for each set of similar neighbors, a new set of groups is created, and these groups differ only by one part from the groups of S_i. Hence, by recursively calling the mining procedure on the new sets, a DFS is performed, growing these groups further.

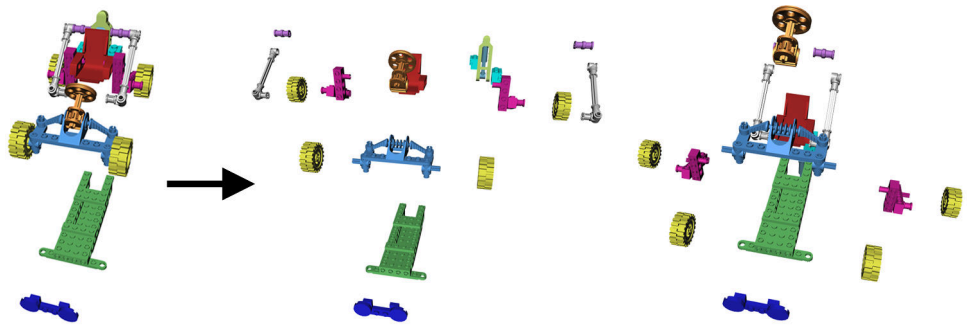
All groups G_i which have been extended by a neighbor are removed from the input set S_i, because these groups are then part of larger groups G_n. If $|S_i| \leq 1$ for a set S_i, all groups were extended, and the set is deleted. However, the mining algorithm is applied again to any parts left in the set S_i (if $|S_i| = 1$) to eventually extract smaller similar groups.

The FSG mining returns with the sets S_{out} of largest similar groups G_{out}. If the sets S_{out} do not overlap, the algorithm is finished. Otherwise, overlapping output sets must be resolved by keeping only one of the overlapping sets S_{out} and applying the FSG again to the set of $A_g \setminus S_{out}$. This operation is repeated for all results, until the output sets S_{out} do not overlap anymore. The decision on which of the overlapping groups is kept is based on the following rules. We keep the one overlapping set, which contains the groups holding the most number of parts. If this measure is ambiguous, the set having the most groups is preferred. If this is still ambiguous, the one containing the largest part is chosen.

3.1.4.3 Layout Initialization

The layout of an explosion diagram depends on the removal direction and distance chosen for each part, which set it apart from its initial position. To reduce the mental load to reassemble an exploded object, explosion directions often follow mounting directions, so collisions between displaced parts are avoided. Explosion diagrams implement this feature by introducing relations between the parts of an assembly. For example, each screw in Figure 3.7 moves relative to the purple cylinder, which it fastens. By displacing one of the purple cylinders, the corresponding black screws will be displaced implicitly.

The relationships between parts of an explosion diagram also allow parts to follow related parts. This enables a part to move relative to its initial location in the assembly,



(a) (b)

Figure 3.8: Different relationships between parts results in different layouts of the explosion diagram. (a) The stacks of parts to the left and the right in the back of the car have been related to the seat. The wheels in the front of the car follow the blue steering gear. (b) The front wheels have been related to the base-plate of the car, as have been both purple elements, which connect the wheels in the back to the car.

which also reduces the number of mental transformations to reassemble the object. For example, note how the grey bolts follow the green gearbox in the explosion diagram in Figure 3.7. However, it is often not obvious which part represents the initial location of another part best. For example, while the initial locations of the black screws in Figure 3.7 are clearly defined by the holes of the engine they fasten, the initial location of the wheels in Figure 3.8 is surrounded by a number of parts. As demonstrated in Figure 3.8a, the wheels in front of the car may follow the blue steering gear. This will result in a translation along the up-vector of the car, before the wheels explode along the x-directions of the model's coordinate system. In contrast, the explosion diagram in Figure 3.8b uses a relation between the wheels in the front of the car and the green base-plate. This results in a displacement of the wheels without a translation along the up-vector of the coordinate system. The same behavior appears for a stack of parts in the back of the car. Since, in Figure 3.8(a), the parent of the stack follows the red seat of the car, all the parts between the wheels and the seat have been moved along the up-vector, before they have been separated from each other. In contrast, the explosion diagram in Figure 3.8(b) uses a relationship between the parent of the stack and the green base-plate of the car, which reduces the number of translations of all the elements in the stack.

We define relations between parts by computing a disassembly sequence. A relationship is set up between each exploded part and the biggest part in the remaining assembly it

has contact with. To avoid collisions between exploding parts, the directions in which a part can be displaced are restricted to only those in which a part is not blocked by any other parts. This implies that the algorithm displaces parts which are unblocked in at least one direction, before it is able to explode parts which are blocked in all directions. Thus, by removing the exploded parts from the assembly, we gradually remove blocking constraints, which allows us to explode previously blocked parts in a subsequent iteration of the algorithm. Since the algorithm gradually removes parts from the assembly, the set of directions for which a part is not blocked (and thus the set of potential explosion directions) depends on the set of previously removed parts. Consequently, the disassembly sequence directly influences the set of potential explosion directions.

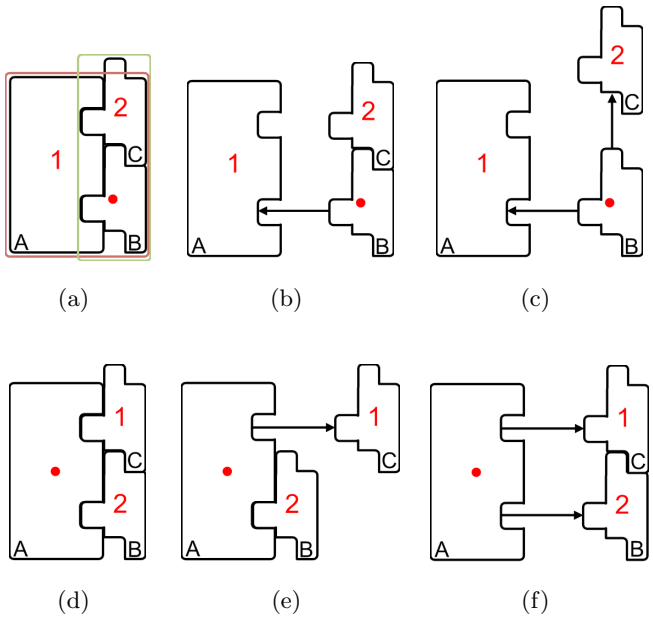


Figure 3.9: Different disassembly sequences may result in different layouts. The sequence is labeled in red. The resulting explosion diagram is illustrated in the image on the right. (a,b,c) The computed sequence is based on previous approaches, which select parts depending on the distance a part has to be moved to escape the bounding box of the remaining assembly. The bounding boxes of the remaining parts have been framed in red and green. (d) We compute the next element in the sequence based on a comparison with the previous one. (e) By removing similar parts in a row, we ensure that the remaining assemblies contain the same elements, except for one part which is similar to the next one. (f) This strategy allows us to explode similar parts within similar conditions, which in turn results in more similar exploded views of similar subassemblies.

Disassembly Sequence. Previous approaches [1, 83] compute a sequence depending on how fast a part is able to escape the bounding box of the remaining parts in the assembly. However, since this approach does not comprise any information about the similarity between exploded parts, the resulting explosion layout does not ensure similar exploded views for similar assemblies. Consequently, we encode information about the similarity of the parts in the sequence. We remove similar parts in a row, starting with the smallest. If no similar part can be removed from the assembly, we choose the current smallest part. This strategy enables us to set up relationships which subsequently allow smaller parts to follow bigger ones during explosion. Take note that, by computing a larger amount of similar explosion layouts, our system is able to choose a representative exploded view out of a larger set of similarly exploding assemblies.

Figure 3.9 demonstrates the difference between previous approaches and our new strategy to find a disassembly sequence. A sequencing based on a bounding box intersection is demonstrated in Figures 3.9(a) to 3.9(c). The algorithm first removes part A, before part B and part C will be exploded. By using this strategy, relationships between part A and part B and, subsequently, between part C and part B will be set up. The resulting explosion layout is illustrated in Figure 3.9(c). As can be seen, different explosion directions have been assigned to the similar parts B and C.

In contrast, our algorithm computes a sequence which is based on a comparison of the previously exploded part and all removable part in the remaining assembly. As demonstrated in Figures 3.9(d) to 3.9(f), our strategy will result in a sequence which supports similarly exploded views of similar assemblies. Both parts B and C have been displaced in the same direction, and both parts have been related to the same part in the remaining assembly (part A).

Relationships. Both strategies in Figure 3.9 set up relationships between the current part and the bigger one. However, since our sequence removes similar parts one after the other, the remaining assemblies are identical for similar parts, with the exception of the previously removed part (which is similar to the current one). Since almost identical conditions exist for similar parts, our algorithm is able to set up similar relationships for those parts and the parts in the remaining assembly.

In addition to the initial assignment of relationships between parts, we change the relationships for penetrating elements in a stack. For example, the black screws in Figure 3.7 have contact with the purple cylinder and the green gearbox. Since the green gearbox is the bigger item, the initial relation is set between a screw and the gearbox. However,

this would result in an explosion diagram in which the screws follow the gearbox instead of the purple cylinder.

To handle such cases, we identify stacks of parts by searching for the elements which are located between the exploded part and the one it is related to. If parts exist in-between and if these parts share an explosion direction with the currently removed part, the initial relationships are changed so that the exploded part is related to the closest part in the stack of parts in-between.

Explosion Directions Previous approaches compute the explosion direction of a part out of a set which contains only the six directions along the three main axes of the model [1, 74, 83]. However, this approach is very limited (e.g., consider the differences in directions in the explosion diagram in Figure 3.7). Therefore, we compute a non-directional blocking graph, similar to the algorithm proposed by Wilson [147], by computing blocking information between all pairs of parts. For each exploded part, we determine the set of unblocked directions by removing all blocked directions from the set of exiting 3D directions. We represent all directions by a unit sphere, and we remove blocked ones by cutting away the half sphere with a cutting plane which is perpendicular to the direction of a blocking part. By iteratively cutting the sphere, using all locking information from parts in contact with it, the remaining patch of the sphere represent all unblocked directions for a part. Thus, we output the center of gravity from the remaining patch of the sphere.

Explosion Distance. Visually similar parts should be moved by a similar distance to create a visually coherent explosion diagram. Since similar parts appear to be similarly large, we set the distance of displacement from the parent part to be proportional to the size of the exploded part. Nevertheless, since a linear mapping may easily result in overly large displacements, we introduce a non-linear mapping using equation 3.5.

$$Distance = SizeOfPart \cdot (1 - k \cdot RelativeSize^2) \tag{3.5}$$

For parts which cannot be removed at all, we compute a distance for which they can be moved until colliding with other parts. For example, the lower purple cylinder in Figure 3.7 cannot be removed before the black screws have been removed. However, the black screws will collide with the cylinder they fasten, if we explode them into a single direction. Nevertheless, we can explode the screws a certain distance, before they collide with the cylinder. Since this distance is sufficient to reveal the screws, we compute the

maximal distance they can be exploded. We explode the screws to a distance smaller than this maximal distance and are further able to subsequently explode the cylinder from the assembly.

We compute the maximal distance that a globally locked part can be moved by rendering both parts - the one which is about to be removed and the one which blocks its mounting direction - into a texture. We position the camera at the vector along the explosion direction to point at the exploded part. In a vertex shader, we use the current model-view transformation matrix to transform each vertex into camera space. The corresponding fragment shader finally renders the location of each fragment in camera coordinates into the textures. By calculating the difference between the texture values, we get a map of distances between the fragments of both parts. The maximal distance by which a part can be removed, before it collides with the blocking part, is finally represented as the smallest difference between the values in the texture.

Group-Based Layout. Our system calculates similar subassemblies independently of the initial layout of the explosion diagram. However, even though our sequence generator specifically supports similar exploded views of similar subassemblies, if their neighborhoods differ, the exploded views may be different. For example, the model in Figure 3.10(a) consists of one set of four similar subassemblies (marked by the green rectangle). Each of them contains two parts. Figure 3.10(b) shows its explosion diagram, in which each single part has been displaced. As can be seen from the initial layout, the exploded view of the subassembly in the lower right corner is different from the explosions of the other subassemblies. If we choose this exploded view as the representative of its set of similar subassemblies, the resulting compact explosion diagram lacks a presentation of the other subassemblies of this set (Figure 3.10(c)).

To prevent representatives which explode differently to other similar subassemblies, we can adjust the sets of similar subassemblies in a way that only similarly exploding subassemblies will be grouped together. Therefore, we use the layout information to modify the identification of similar subassemblies. Only those parts of the assembly are candidates for extending a group which would set up a relationship to another part in the subassembly. Figure 3.10(d) shows the result of this restriction. This strategy finds a set of only three, instead of the previously identified four, similar subassemblies (marked in green). Consequently, fewer subassemblies will be presented assembled, which results in a layout which is not as compact as in the previous case.

In order to create a more compact explosion layout, without risking to choose a repre-

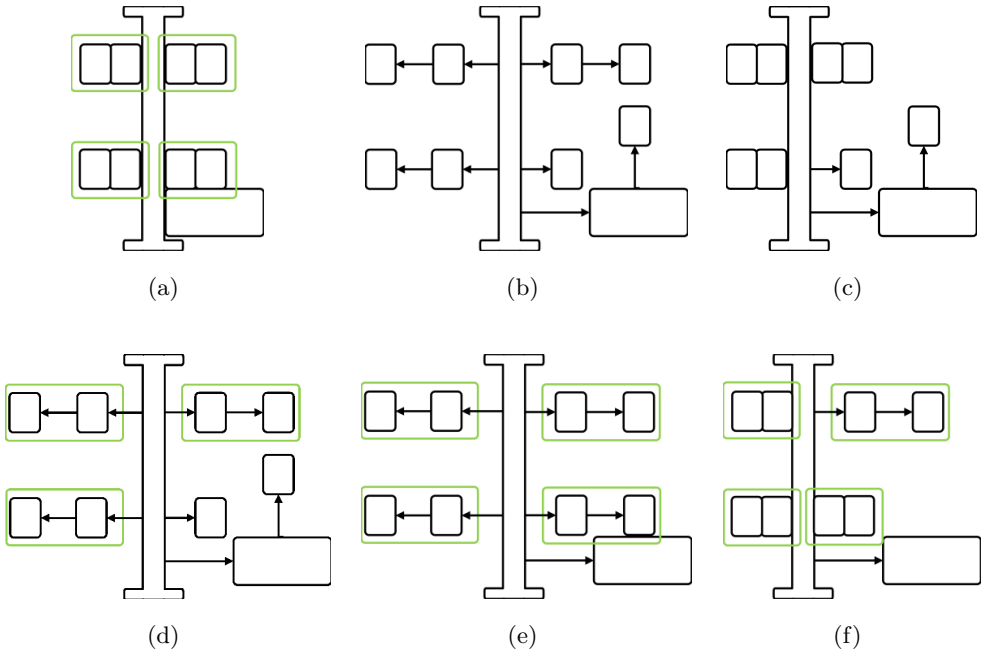


Figure 3.10: Explosion Layouts containing group information. (a) Groups have been created independently from the explosion layout. (b) Therefore, the explosion layout does not take information about similar subassemblies into account. This may generate different exploded views of similar subassemblies. (c) If we select a representative from a set of similar subassemblies which do not explode the same way, the explosion does not represent all other subassemblies. (d) By recalculating group information from the layout, the number of similar groups is reduced, which results in more exploded views. (e) Therefore, we modify the initial layout so that similar subassemblies explode in a similar way. (f) This strategy allows us to choose a representative from a larger set of subassemblies, which in turn reduces the amount of required exploded views to demonstrate the assembly.

sentative which does not demonstrate the composition of other similar subassemblies, we modify the layout of the explosion diagram instead of the information about the similarity of subassemblies. As illustrated in Figure 3.10(e), we aim to modify the layout to prevent relationships with parts outside the subassembly. We allow only one relationship between a part in the subassembly and the remaining 3D model.

This is similar to the approach of Li et al. [83], who explode a manually defined group of parts as if it was a single element in the assembly. However, we use a different approach to handle interlocking groups. Rather than splitting a subassembly, we ignore blocking parts. This allows us to keep subassemblies connected. Note that this could be at the cost of explosion diagrams which are not completely free from collisions. Nevertheless, we believe

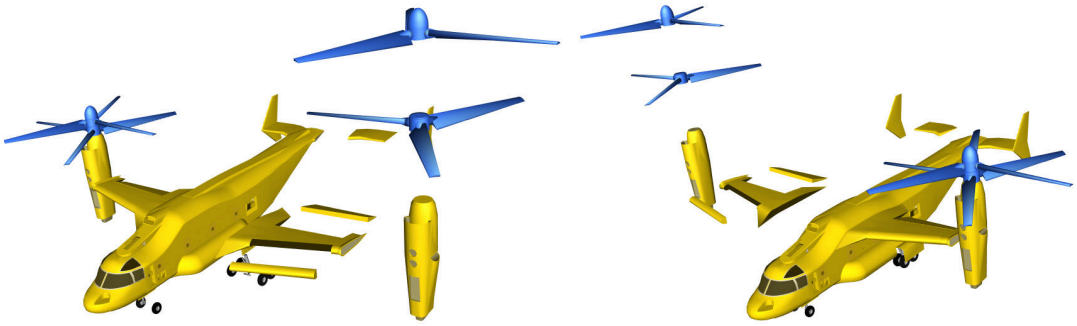
that preventing such collisions is less important for the final compact explosion layout than a larger amount of explosions or a representative which does not demonstrate the composition of its associated subassemblies. In the case of a compact explosion diagram, it is more important to select a representative from a rather large set of similar subassemblies, which additionally all explode in a similar way.

Thus, we compute an explosion diagram which ensures similar explosion layouts of similar subassemblies as explained before. However, for each part p, we determine if it is a member of a subassembly G which occurs multiple times in the model. If the algorithm is about to explode a part p which is a member of G, we choose a representative part p_r out of G, which we explode instead of p . We define p_r as the biggest part in the subassembly G which has at least one face in contact with at least one part of the remaining assembly, not considering other parts of the subassembly. In addition, the representative part p_r has to be removable in at least one direction without considering blocking constraints of parts of the same subassembly.

Even though p_r influences the explosion direction of the entire subassembly, we may not set the relationship between p_r and a part out of the remaining assembly. Since we are only able to explode each part once, and since we want to further continue to explode all frequent subassemblies in the same way, we have to choose the same part in each subassembly to set up the relation to the remaining assembly. Moreover, since we want to explode subassemblies using the guidelines presented before in this section, we want to explode the small parts before the bigger ones. Therefore, we choose the biggest part in the assembly as the main part of the assembly, and we relate it to the biggest part in the remaining assembly which the subassembly has contact with.

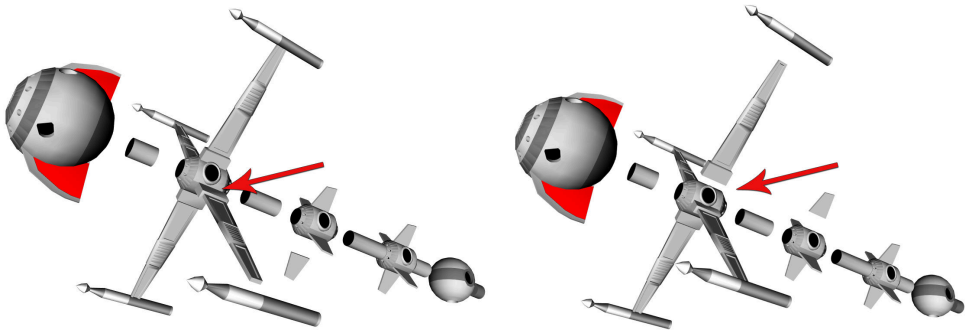
3.1.4.4 Layout Optimization

Following the idea of compact visualizations as presented in section 3.1.1, after identifying frequent subassemblies and after computing an initial explosion layout, we create a compact explosion diagram by displacing only one representative group out of a set of similar groups. We compute the impact of a representative subassembly on the explosion diagram by calculating its quality as the weighted sum of a set of measurements. Since the combination of representatives may influence the quality of a single subassembly, we need to optimize the selection by optimizing the selection of representatives. In the following, we will first describe the parameters for rating the impact of a subassembly, before we present our approach to combine representatives to the final compact explosion diagram.



(a) (b)

Figure 3.11: Local Footprint. (a) Putting emphasis on the footprint of the exploded representatives renders them in the foreground of the presentation. (b) In contrast, by putting emphasis on the footprint of the unexploded parts, the exploded representatives are shown in the background.



(a) (b)

Figure 3.12: Direction versus Size of Footprint. (a) The size of the footprint by itself does not ensure a clear presentation of the explosion. (b) By scaling up the importance of the angle between the viewing direction the explosion direction, we are able to choose a representative which better demonstrates the explosion of the subassembly.

Quality Measurements. We define the quality of a group of parts as a combination of several measurements. Therefore, for each group, we render its local explosion (which displaces only the parts of the group and parts which block the group), and we compute the following values:

- *Size of footprint of the exploded group.* The size of the footprint f describes the size of the projected area of a part of the object in screen space.
- *Size of footprint of all other similar groups without any displacements.* The size of the footprint of the parts of other subassemblies f_r describes how big similar, but unexploded subassemblies will be presented.
- *Explosion directions relative to current camera viewpoint.* Assuming that explosions which are similar to the viewing direction are more difficult to read than those which explode more perpendicularly to the viewing direction, we compute the dot product a between the viewing vector and the explosion direction for each part. The average value a is used for a group of parts within a subassembly.
- *Visibility of parts of the exploded representative.* The visibility v is a relative measure. By counting visible pixel of a part and those which are hidden, we compute its percentage of visibility from the current point of view.

$$Q_r = f \cdot f_c + v \cdot v_c + (1 - a) \cdot a_c + f_r \cdot f_{rc} \tag{3.6}$$

The final quality Q_r of an exploded view of a subassembly consists of the weighted sum of these values (see Equation 3.6). The weights (f_c, v_c, a_c, f_{rc}) indicate the importance of each single parameter to describe the quality of the group. By differently scaling these parameter, we are able to control the final presentation. For example, the Figure 3.11 shows two compact explosion diagrams generated with different intensions of the user. The compact explosion diagram in Figure 3.11(a) puts emphasis on the representative explosions, while it simultaneously shows similar subassemblies in the background as contextual information. In contrast, the image in Figure 3.11(b) presents the assembled parts of the compact explosion diagram in the foreground, while the exploded representatives are used to fill in contextual area. Both graphics were rendered by scaling up a single weight. While Figure 3.11(a) scales up the impact of the size of the footprint of the representatives, Figure 3.11(b) was generated by increasing only the values of the impact of the footprint of non-representatives.

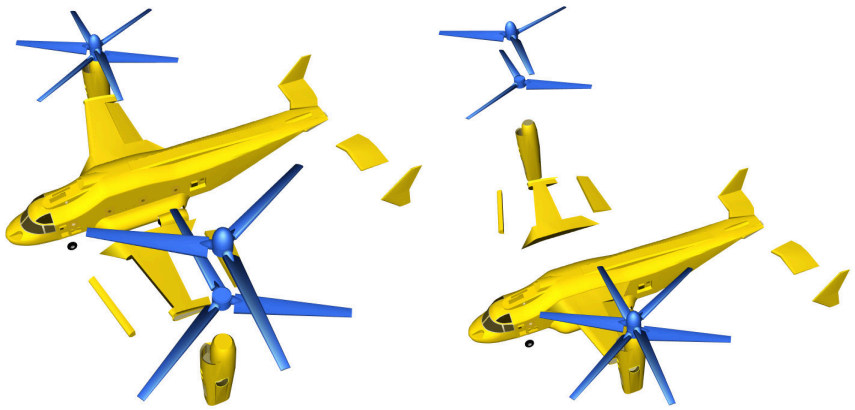
Even though the footprints of both the representatives and the unexploded elements are important parameters to compact explosion diagrams, they may fail to create easily comprehensible presentations. The explosion diagram in Figure 3.12(a) was rendered with a high impact of the footprint of representatives. However, such scaling by itself turns out to be insufficient from certain points of view. The explosion indicated by the red arrow is almost hidden. A more informative explosion diagram from the same point of view is shown in Figure 3.12(b). The presentation scales up the impact of the angle between the view vector and the average direction of explosion for each representative.

Nevertheless, a high impact of only the explosion directions leads to self occlusions which again may hinder the understanding of the final presentation (Figure 3.13(a)). As demonstrated in Figure 3.13(b), by putting emphasis on the visibility of representative parts, the system chooses a different one to explode. However, even if self-occlusions are avoided within a single representative, global occlusion between different representatives cannot be controlled by this parameter (Figure 3.14(a)).

As the examples in Figures 3.11 to 3.13 demonstrate, there is no universal rule on which parameter we have to scale up or down in order to ensure comprehensible compact explosion diagrams. However, the weights can still be used to direct the rendering towards the user's intention. The quality of the entire compact explosion diagram can only be controlled by taking combinations of explosions of representatives into account. By estimating the quality of an explosion of subassemblies independently from other explosions in the diagram, interdependent explosions and visual overlaps of representatives may change the quality of a representative explosion.

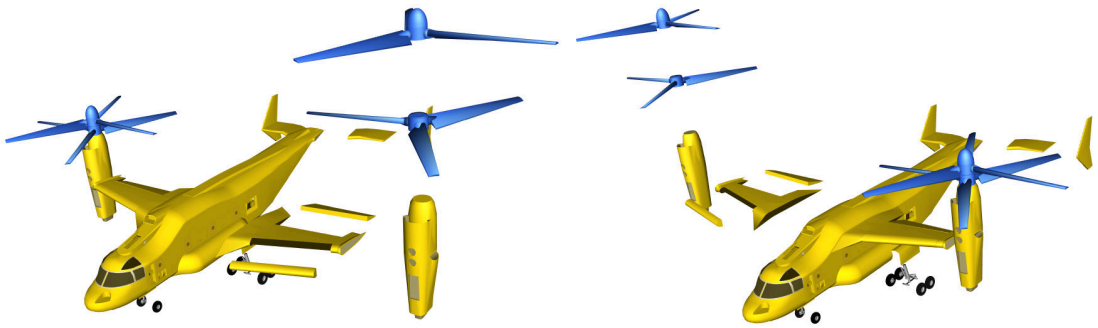
Optimization. Finding the optimal representative for one set of similar groups, as shown in the previous examples, does not ensure that the representative stays optimal when representatives of other groups are exploded. Exploded groups may interfere with each other and, therefore, decrease the quality of other representatives. In Figure 3.14(a), the representative groups are locally optimal when only these groups are exploded for themselves. However, combining all locally optimal representatives into one explosion significantly decreases the overall layout quality.

To avoid interferences of representatives with each other, we search for an optimal combination of exploded groups, using the idea of threshold accepting, as described in section 3.1.1.2. In each step of the algorithm, the quality of a combination of representative explosions is evaluated by computing the sum of their scores after exploding all of them. The initial layout consists of exploded representatives with the highest local



(a) (b)

Figure 3.13: Visibility. (a) If the visibility of the parts of an explosion are not taken into account, parts of a representative may occlude each other. (b) The visibility evaluation of representatives is able to resolve self occlusions.



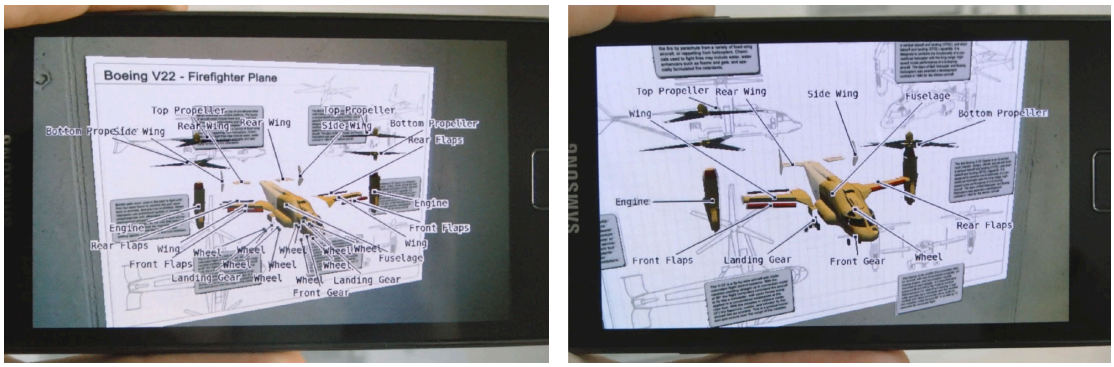
(a) (b)

Figure 3.14: Global Visibility. (a) Adding weight to the local visibility does not resolve occlusions between different representative groups. (b) Optimizing the layout using threshold accepting ensures that the overall visibility and, thus, the layout quality is maximized.

scores. Therefore, if the sum of their local scores is equal to the global score, the local representatives are global representatives too. Consequently, we do not search further for a better combination. However, if the global score is less than the sum of local scores, we change the initial layout by a single representative group and re-compute the global score of the modified layout. If the score of the changed layout is higher than the current best finding, this new one is used as the current best combination of representatives. Therefore, if the new score is equal or less than the current best score, we do not consider the current combination to be displayed. However, even if the current score is less than the best one, we compute the next tested layout based on the current one, if its difference to the best score is less than a threshold value. Otherwise, we modify the layout which the current layout was computed from. While the algorithm progresses, the threshold value decreases, which gradually allows better layouts to be the starting point for further changes.

Figure 3.14(b) shows the results of optimizing the locally scored compact explosion diagram presented in Figure 3.14(a). Since representatives in Figure 3.14(a) overlap each other, a different subassembly has been selected in the optimized compact explosion diagram in Figure 3.14(b).

3.1.5 Combined Optimization of Data Types



(a) (b)

Figure 3.15: Combined Optimization. Compact visualizations of different data types can also be combined. (a) A combination of annotations and explosion diagram can result in a large amount of clutter. (b) Our approach avoids this situation by combining the optimization of compact annotations and explosion diagrams. Clutter is reduced by exploding and labeling only representative parts of the airplane.

Until here, we created compact visualizations for different data types separately: an-

notations, photographs and 3D assemblies. However, the modular architecture of our approach allows to easily combine different visualizations types. Figure 3.15(a) shows a visualization, which uses an explosion diagram to present the structure of a firefighter plane and annotations to denote its parts. While the visualization suffers from information overflow, the one in Figure 3.15(b) demonstrates the corresponding compact annotated explosion diagram.

The combined visualization is achieved by sequentially executing the specializations for compact explosion diagrams and annotations. The explosion diagram changes the layout of the parts, which define the anchor points of the labels, but is not influenced by the annotations. Therefore, to avoid changes to the optimized layout of annotations, it is mandatory to optimize the layout of the explosion diagram, before optimizing the layout of annotations.

3.1.6 AR Challenges

In order to make compact visualizations suitable for AR, they have to be able to handle dynamic scenes with a moving camera. Therefore, the visualization must run within interactive frame rates and have guaranteed real-time performance. This is especially challenging when compact visualizations are deployed on mobile devices with limited processing and battery power. Furthermore, the small display of a handheld device only provides a limited amount of screen space for visualizations. In addition, compact visualizations must take into account the real world environment to avoid interferences with real content.

In the following, we present methods for creating compact visualizations which deal with these issues. The discussed solutions make use of the modular structure of our framework for easy integration.

3.1.6.1 Interactive Framerates

The performance of the optimization for compact visualizations mainly depends on the complexity of the input data and the involved mapping and evaluation modules. For instance, the computation of a compact explosion diagram is computationally intensive, because of the costly pixel-accurate visibility estimation and the number of iterations spent in optimization. The visualization in Figure 3.15(b) took about 30 seconds on an 2.67GHz Intel Core i7 processor.

The performance of the optimization process can be improved by tuning the algorithms

and reducing the number of iterations. However, a lower number of iterations may lead to lower quality visualizations. Additionally, when the complexity of the input assembly increases, the performance of the optimization decreases. Interactive frame rates for creating compact visualizations cannot be guaranteed for all cases and even less so when considering mobile hardware with limited CPU and battery power.

To guarantee real-time performance, we prepare compact visualizations from a finite number of viewpoints. At runtime, we present the prepared layout which is closest to the current point of view of the user. To facilitate the tracking of layout changes for the user, the changes between layouts are animated over time.

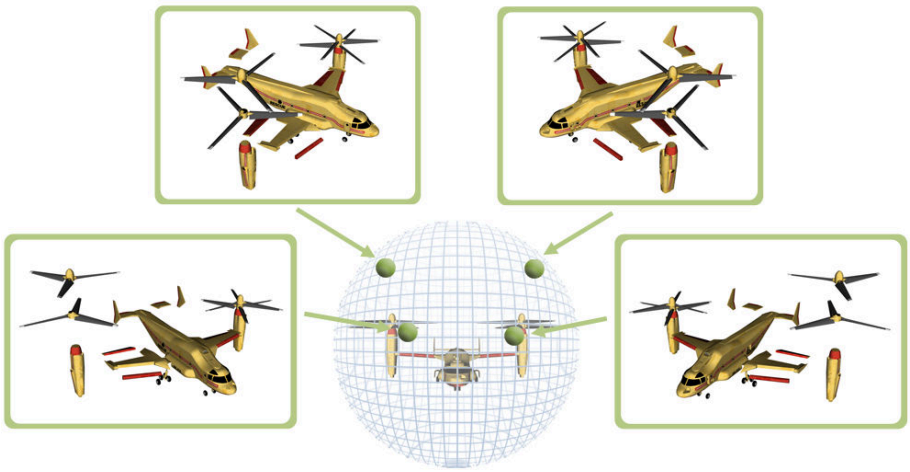


Figure 3.16: Prepared optimal layouts. We precompute the best compact explosion diagram from a finite number of viewpoints, enabling real-time layout updates by animating between layouts from neighboring viewpoints.

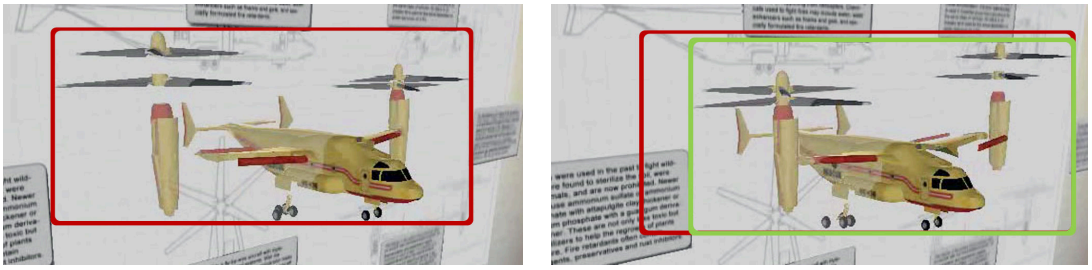
Figure 3.16 depicts different compact explosion layouts calculated using this approach. The animation between the viewpoints simultaneously collapses and expands obsolete and new representative elements, which were prepared for given viewpoints. Similarily, the transition between different compact annotation layouts is performed by changing the anchor points of the annotations and animating their movement to the new location.

We provide optimized compact visualizations for a discrete set of viewpoints by sampling them from a bounding sphere surrounding the object of interest. A virtual camera is placed at each sample point and oriented towards the center of the sphere. To cope with layouts of different dimensions, we adapt the distance of each camera to the center of the sphere so that the visualization fits into the viewport as tightly as possible. At runtime, the distance of the camera to the object center is given by the user, and the viewport may

not be centered on the object. According to our observations, these errors can be ignored in practice.

3.1.6.2 Minimizing Layout Dimensions

Up to this point, we have not explicitly considered the limited screen estate of a mobile device in the optimization of compact visualizations. In the following, we present a quality measure which allows us to optimize visualizations for minimal dimensions.



(a) (b)

Figure 3.17: Minimally extending compact explosion diagram in AR. (a) The explosion diagram of the assembly requires a large amount of screen space. The most comprehensible explosion diagram and its screen space bounding box shown in red (b) By incorporating the size of the screen aligned bounding box during optimization, we can compute a minimally extending compact visualization (green). Note the difference between the bounding box dimensions.

To keep the dimensions of the layout as small as possible, we compute the size of the screen aligned bounding box during optimization, more precisely, its diagonal $Diag_{2d,exp}$. This estimation is then incorporated into the quality estimation of the layout as additional quality parameter $Q_{Extension}$. As a result, the quality of the layout becomes proportional to the inverse size of the screen aligned bounding box. Tighter layouts are ranked higher, while large visualizations will reduce the quality value (3.7).

$$Q_{Extension} = 1 - \frac{Diag_{2d,exp}}{\sqrt{ImageWidth^2 + ImageHeight^2}} \tag{3.7}$$

Figure 3.17 demonstrates the change of required space when choosing different layouts for an assembly. Due to poor layout choices, the layout in (a) is larger than the layout in (b). Notice how the layout in (a) requires more space, making it less suitable for small screen devices. The layout is the result of optimizing the visibility and the projected size

of the exploded parts, which explodes parts into the direction of the user. To create the minimally extended layout shown in (b), a higher weight is added to $Q_{Extension}$ than to visibility and size. Note that while the extension is smaller than in (a), all parts are still visible. The smaller visualization now allows the user to zoom in closer on the object of interest, making it easier to explore fine details. Additionally, a minimally extending layout can decrease the amount of scene modifications in order to avoid collisions.

3.1.6.3 Scene-Aware View Management

AR environments are often visually very complex, and augmentations easily suffer from interferences with real world structures, if their layout does not take the background into account. For instance, in Figure 3.18(a), the best compact explosion diagram collides in screen space with the box next to the space ship, while, in the combined visualization in Figure 3.15(a), annotations interfere with the text boxes printed on the poster. To avoid such conflicts, we propose scene-aware view management, that causes not only the view management to react to the real world scene, but also the real world react to the augmented content.

To achieve scene aware view management, the straight-forward solution is to adapt the layout of the data to avoid visual interferences with the scene. We present two solutions for resolving interferences using the example of compact visualizations. First, we optimize a compact visualization not only for a single layout, but for a set of alternative layouts for each viewpoint. Second, if alternative layouts cannot be found, we adapt the layout of the scene to accommodate the compact visualization.

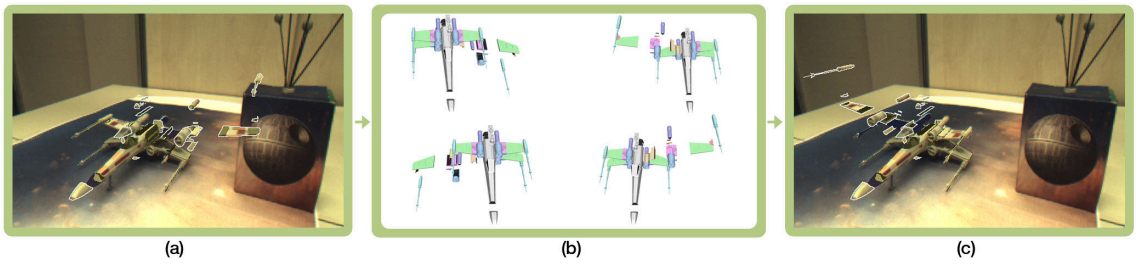
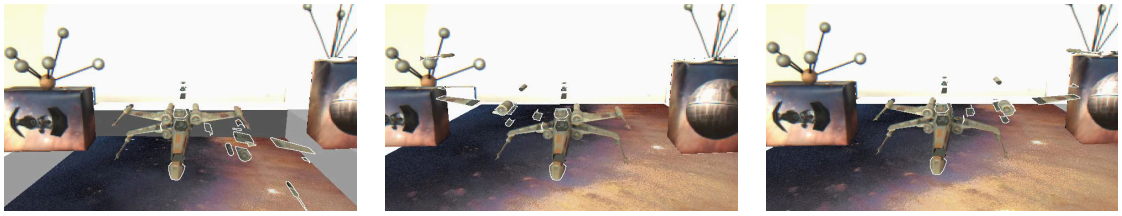


Figure 3.18: Scene-aware compact explosion diagrams. (a) The best layout may collide with important elements in the real world environment. (b) By preparing a set of alternative layouts (c) we are able to choose a layout which fits in the environment.



(a) (b) (c)

Figure 3.19: Layout-aware scene modification. As alternative layouts cannot always avoid collisions with scene objects, we displace real scene elements by applying three different strategies, which combine alternative layout selection and scene modification. (a) The best layout causes the largest amount of scene modifications. (b) To minimize the modifications, a less comprehensible layout can be chosen instead. (c) When both strategies do not produce acceptable results, a trade-off between scene modifications and comprehensibility can be computed.

Alternative layouts. In order to avoid interferences with the real world, we need layouts that fit into the spatial constraints given by the environment. Therefore, the layout optimization must not only consider compactness and frame coherence, but also potential collisions with other objects in the environment. This can easily be accomplished during optimization by integrating an appropriate collision detection method into the respective layout algorithms. The layout algorithm thereby is able to choose layouts which avoid real objects.

However, when preparing compact layouts using the method described in Section 3.1.6.1, the optimization usually has no knowledge of real objects encountered at runtime. We address this issue by precomputing alternative layouts for compact visualizations at each sampled viewpoint. To achieve temporally coherent changes for each alternative layout, we also need to prepare aligned layouts for the alternative layouts.

As the spatial constraints of the real environment are unknown during the alternative layout preparation, we precompute layouts, which vary as much as possible. Starting with the best layout calculated for a sample point, we iteratively search n alternative layouts for this same point. A new alternative layout A_{L_n} for a compact visualization is computed by optimizing its quality parameter, as described in section 3.1.4.4. In addition, all differences between the currently optimized layout and already computed alternative layouts A_{L_0} to $A_{L_{n-1}}$ of the previous $n-1$ iterations for the same sample point are incorporated into

the optimization. This ensures that the same layout is not calculated multiple times for a sample point.

This iterative optimization is easily achieved by sequentially executing the pipeline shown in Figure 3.3 n times. The resulting layout of each iteration is stored as alternative layout for the viewpoint and is also forwarded to the next iteration to calculate the quality criterion $Q_{Alternative}$, which is defined by (3.8).

$$Q_{Alternative} = \frac{1}{n} \sum_{i=0}^{n-1} Diff(A_{L_i}, A_{L_n}) \tag{3.8}$$

At runtime, we search for a suitable layout by sequentially trying all alternatives in descending order of quality, until we find one that fits into the real environment. The quality score at runtime depends on the criteria used during optimization, but without considering $Q_{Alternative}$.

The difference estimation $Diff(A_{L_i}, A_{L_n})$ between layouts depends on the type of visualization. For instance, to find alternative compact explosion layouts, which are maximally different from each other, we estimate the l_2 distance $Diff(L_1, L_2)$ of the respective part positions relative to the center part of the explosion. Figure 3.18(b) shows four different compact explosion diagrams, which have been generated by maximizing their variation. After detecting the collision with a real object (Figure 3.18(a)), we search for an alternative layout among the candidates shown in Figure 3.18(b). Note that, since we consider high quality layouts first, we select the one in the upper right corner instead of the one in the lower left corner, which would fit. However, it represents the explosion using fewer pixels and was, thus, considered less visible. The chosen layout is applied in Figure 3.18(c).

Scene modification. Preparing alternative layouts allows the integration of compact visualizations within the available space. However, in densely cluttered scenes, the space may not suffice and all alternative layouts cause collisions. Hence, if no suitable alternative layout is found, we displace real scene elements to create more room for the compact visualization. Figure 3.19(a) provides an example of a compact explosion diagram. Figure 3.15(b) and Figure 4.2 show the modification of a real poster to integrate annotations.

Displacing real world objects is a drastic measure and may be undesirable in some applications. We therefore aim to minimize such alterations by allowing three different strategies to displace scene elements. The first strategy uses the best layout without considering the amount of modifications to the real scene, and, thus, may drastically change the scene. The second strategy aims to minimize the required scene modifications by

choosing a less optimal layout. The third strategy finds a compromise between modification and layout quality to balance both. Figure 3.19 illustrates these three strategies.

We implemented the scene modifications by extending the 2D force-based approach of Hartmann et al. [56] to work with 3D objects and forces. Furthermore, we introduce motion constraints for real objects, to ensure scene-coherent modifications. For instance, the real objects in Figure 3.19 are constrained to move on the ground plane. The text boxes in Figure 3.15(b) are constrained to move on the plane defined by the wall. The constraints can also limit the overall amount of allowed modifications by restricting the distance that objects can move.

3.2 Hierarchies in View Management

Compact visualizations reduce the amount of data and, therefore, the clutter on screen by filtering redundant items and creating an optimized layout with the preserved items. However, as the data density increases, the layout will deteriorate again, until data must be removed.

Ideally, the amount of data is reduced to a presentable extent without losing any relevant information. Aside from removing data by filtering and rearranging with view management techniques, clutter can be reduced by aggregating data points into clusters. By recursively aggregating data points and the resulting clusters, an information hierarchy can be built, similar to a LOD visualization. A selection algorithm can then decide which parts of the hierarchy to visualize.

Naturally, not all clusters can be shown. Therefore, user interaction is required to retrieve all available details on demand. The aggregated data often requires a change of the representation, because a summary of the data must be presented. However, while clustering changes the appearance of the data by aggregation, the complete information space is preserved.

In the following (Section 3.2.1), we present a method that controls the information density of the displayed items based on a hierarchical data structure. Information density refers to the number of items presented on the display based on the available screen-space. Ideally, the selection is balanced over the screen-space in order to avoid clustering of items in one region. While a selection of lower information density exhibits larger gaps between the data, a higher density presents more information with less space in between. Our method selects items from the created hierarchy by taking into account user priorities and the available screen-space. The information is adapted during interaction to reduce

the visual clutter. Hierarchical representations can also be integrated into the compact visualization framework. In section 3.2.2, we show how the optimization of compact visualizations can be expanded to also handle such hierarchical data structures.

3.2.1 Adaptive Information Density of Annotations

We address the issues of data loss through filtering and clutter from data overload by creating an information hierarchy, which is conceptually similar to semantic level of detail [38]. By recursively applying clustering, an information hierarchy is built. Our clustering approach not only considers user-controlled spatial attributes (e.g., distance), but also non-spatial attributes (e.g., semantic tags). The sum of these user-weighted attributes provides a ranking of the data, which expresses its relevance to the user. To avoid visual clutter, a display algorithm shows data which is relevant for the user in more detail, while it always adapts the overall amount of information to the available screen space. It does so by solving an incremental optimization problem, deciding which nodes in the hierarchy are selected for display. Users can dynamically adjust priorities to interactively drill down on data deemed relevant, and reveal all available details on demand.

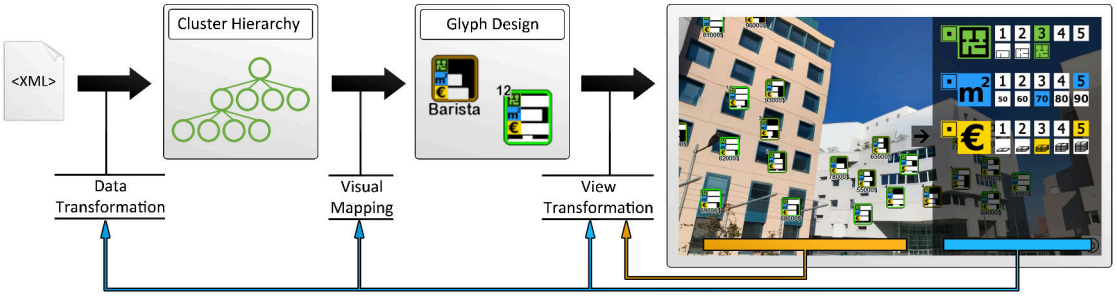


Figure 3.20: System overview. Our information density display follows the information visualization pipeline as presented by Card et al. [23]. In the data transformation step, the input data is clustered to create a hierarchical representation of the data. The data items of the hierarchy are encoded as glyphs. The user can influence each step explicitly via the user interface (blue line) or implicitly by changing the viewpoint and manipulating the glyphs directly (orange line).

Our method lets users to query geo-referenced data about their surroundings, such as restaurant information or real estate offers, from online databases such as Google Places[1]. The data points are visualized as annotations in an AR view. A typical use situation will involve several hundreds or thousands of data points, more than what can be presented in

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

full on the screen, making view management necessary.

Visualization for AR browsers has similar requirements as in classic information visualization, which involves the three main stages of the information visualization pipeline [23]: data transformation, visual encoding and view transformation (Figure 3.20). We allow user interaction in all three stages. While the first two stages change the underlying data structure and the visual encoding, the user can directly interact with the data in the third stage, the view transformation. In addition, the AR user directly controls the camera and can change the viewpoint of the data.

Data transformation and visual encoding are conceptually a pre-process, yielding visual data structures, which are presented in a temporally consistent manner during the view transformation stage. However, the pre-processing is swift, taking just a few seconds even on mobile devices, and can be repeated every time the user wishes to change a parameter.

3.2.1.1 Hierarchical clustering

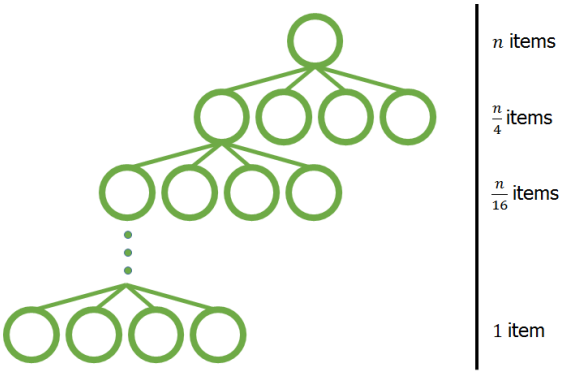


Figure 3.21: Hierarchical clustering. By recursively applying clustering on the input data, we create a hierarchy of clusters. Starting with n data points, we create four clusters that contain approximately $\frac{n}{4}$ items. The recursion terminates at the leaf nodes that contain the single data points.

In the data transformation stage, a hierarchy of clusters is computed based on similarity of data points. A flat partitioning would require knowing the number of clusters in advance and cannot reflect the structure of the data well [91, p.377]. Instead, by using a cluster hierarchy, the view management can later decide for every frame at which level the hierarchy should be cut and, therefore, how many clusters should be presented.

We consider a set of data points $\mathbf{D} = \{D_i\}$, each with a set of attributes $\mathbf{A}$. We denote the attribute set of D_i as $\mathbf{A}_i = \{A_{i,j}\}$. Each attribute can have an arbitrary data type, describing aspects such as user satisfaction rating, social tags or pricing. Both position

P_i in world space and position p_i, the projection of P_i to screen space using the current camera position, may, but need not be used as attributes. Two attribute values $A_{i,j}$ and $A_{i',j}$ of the same type $\mathbf{T}$ can be compared with a comparison function $cf_j : \mathbf{T} \times \mathbf{T} \rightarrow [0, 1]$, which yields 1 if two values are identical, and 0 if two values are most dissimilar.

Note that the comparison function can involve operations that depend on a user's situation and are, thus, subject to change over time. For example, we can use a mean or maximum value for normalizing a certain attribute, making the comparison function dependent on the current database population. We may also consider a routing algorithm that determines the time to walk to each destination from the user's current position. This estimate will become increasingly incorrect, if the user is moving. For such attributes, we assume that attributes remain approximately valid during a short duration of usage. Moreover, the user can trigger a fast re-computation at any time.

The user expresses the desired information by setting desired values $\mathbf{U} = \{U_j\}$ and weights w_j for each attribute ($\sum w_j = 1$). A weight of 0 indicates that the user does not care about a particular attribute; in this case, the desired value for this attribute can be arbitrary. With the comparison functions, we can compute the similarity $S : \mathbf{A} \times \mathbf{A} \rightarrow [0, 1]$ of two attribute sets as the weighted per-attribute difference:

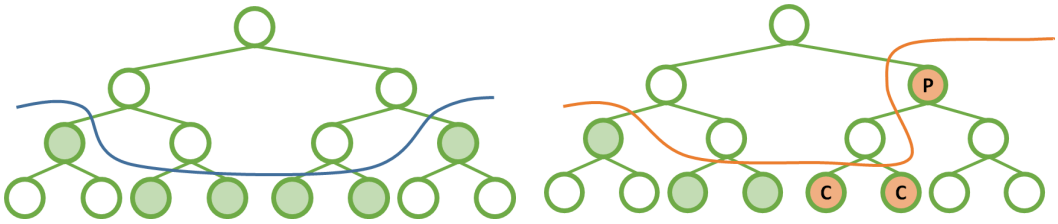
$$S(\mathbf{A}_i, \mathbf{A}_{i'}) = \sum_j w_j \cdot cf_j(A_{i,j}, A_{i',j}) \tag{3.9}$$

Once the user has set weights and desired values, the clustering algorithm can be started. We use top-down divisive k-means clustering to create our hierarchical cluster tree composed of nodes $\mathbf{N} = \{N_i\}$ (Figure 3.21). All data points initially form a cluster that corresponds to the root of the tree. We recursively perform k-means on the root, until the leaves of the tree correspond to the smallest possible cluster containing only one item, i. e., $\mathbf{D} \subseteq \mathbf{N}$. K-means creates clusters based on the similarity function S. We define the branching factor of the tree by choosing a number of clusters for each iteration of k-means. For our experiments, we set this factor to four.

The world-space position of an intermediate node is computed as the centroid of the children's positions. If P_i or p_i are used as attributes for the evaluation of S, we found it useful to decrease their weight proportional to the graph distance of a given N_i from its leaves. In this way, position has a stronger influence on clustering in the lower, more "concrete" layers of the tree, but similarity of semantic attributes has a stronger influence in the upper, more "abstract" layers of the tree.

3.2.1.2 Optimal label selection

In the following, we describe the view transformation stage of our approach to balance the available screen-space against visualizing relevant data to the user. In addition, the system must handle viewpoint changes in a temporally coherent way. A view management algorithm makes sure that any data items that might interfere are rearranged on screen.



(a) (b)

Figure 3.22: Selection from the cluster hierarchy. (a) By performing a greedy best-first search, we create a selection of the cluster hierarchy that resembles a cut through the tree. The blue line cuts the hierarchy correctly. All nodes shaded green, which lie directly beneath the cut, are placed on the screen. (b) The cut must be performed in such a way that either all children or no child of a node are placed on the screen. Otherwise, information about children would also be shown as part of the parent. Thus, the orange cut is invalid, because the predecessor (P) of two nodes (C) is included.

Initial label selection. We want to select a set of labels $\mathbf{L} = \{L_k\}$ representing a cut through the cluster tree, so that all data points have some representative. We write $children(i)$ for the set of all direct children of N_i, $children^*(i)$ for the set of all direct and indirect children of N_i (including N_i itself), and $leaves(i) = \{x | x \in children^*(i) \wedge x \in \mathbf{D}\}$. Using these definitions, we can describe the set of all possible cuts as follows (Figure 3.22):

$$\begin{aligned} cut(\mathbf{N}) = \{\{\mathbf{L}\} \mid \mathbf{L} \subseteq \mathbf{N} \ \wedge L_k \not\subseteq children(L'_k) \ \forall \ (L_k, L'_k) \in \mathbf{L} \ \wedge \\ (\exists \ L_k \in \mathbf{L} \mid D_i \in children^*(L_k) \ \forall \ D_i \in \mathbf{D})\} \end{aligned} \tag{3.10}$$

With these considerations, we can select a suitable $\mathbf{L} \in cut(\mathbf{N})$ for a given user position. By introducing a cost and benefit metric, we can interpret the label selection problem as a constrained optimization problem. It tries to fill the screen with the most beneficial labels, by optimizing a benefit function $B(k)$

$$\max_{\mathbf{L} \in cut(\mathbf{N})} \sum_{L_k \in \mathbf{L}} B(k) \tag{3.11}$$

while avoiding excessive clutter by respecting a maximum cost $C(k)$

$$C(k) \leq C_{max} \; \forall \; L_k \tag{3.12}$$

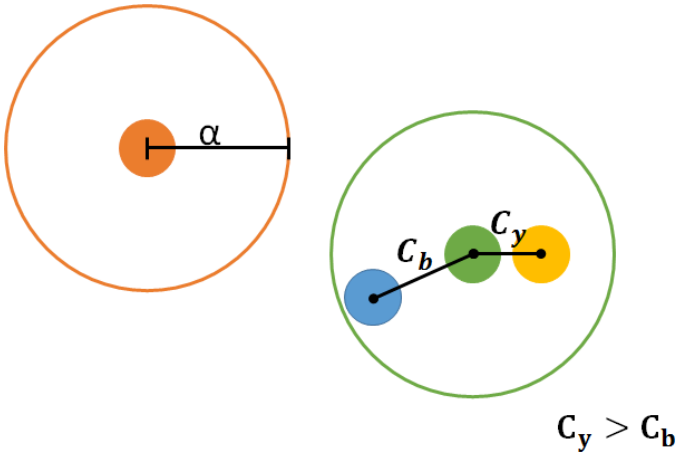


Figure 3.23: To measure the visual clutter of a data item, we consider a region with radius α around the item (see red circle). If an item falls within the radius of another item, the clutter is calculated as cost C based on the distance between the nodes. The closer two nodes lie together, the larger the clutter and, thus, the cost of placing a node. The blue and the yellow node fall within the radius of the green node. The cost for placing blue is smaller, because it is farther away from the green node and produces less clutter in the region.

The benefit of a leaf D_i is given by its similarity to $\mathbf{U}$, i. e., $S(\mathbf{A}_i, \mathbf{U})$. The benefit of an intermediate node depends how well it can represent its leaves. We account for this fact by weighting the benefit with the label's spatial displacement in screen space, w_P, and the semantic similarity of the data points represented by the label, w_S. The spatial displacement w_P gives more benefit to intermediate nodes, which are close to their data points, expressed as relative inverse distance:

$$w_P(k, k') = \frac{1}{1 + |p_k - p_{k'}|} \tag{3.13}$$

The semantic similarity w_S gives higher benefit to an intermediate node representing homogeneous data points, which have a high average similarity S:

$$w_S(k) = \frac{2 \cdot \sum_{(L_i, L_{i'}) \in leaves(k), i \neq i'} S(\mathbf{A}_i, \mathbf{A}_{i'})}{|leaves(k)| \cdot (|leaves(k)| - 1)} \tag{3.14}$$

We combine these terms in a recursive definition of a benefit metric $B(k)$:

$$B(k) = \begin{cases} S(\mathbf{A}_k, \mathbf{U}), & \forall D_k \in \mathbf{D} \\ w_S(k) \cdot \sum_{L_{k'} \in leaves(k)} (B(k') \cdot w_P(k, k')), & \text{otherwise} \end{cases} \tag{3.15}$$

The cost of including a node N_k in $\mathbf{L}$ is related to the clutter it produces. Using w_P, we can express the clutter as local density of other labels in a neighborhood of radius α around a node (Figure 3.23):

$$C(k) = \sum_{N_{k'} \in \mathbf{L},\ |p_k - p_{k'}| < \alpha} w_P(k, k') \tag{3.16}$$

This optimization problem can be approximated with a greedy best-first search (BFS) [103]. It starts with the root of the cluster tree and keeps propagating the cut through the tree towards the leaves, by unfolding an intermediate node $L_k \notin \mathbf{D}$, i. e., replacing L_k with its children. The unfolding operation changes the relative benefit $R_u(k)$:

$$R_u(k) = \left(\sum_{L_{k'} \in children(L_k)} \frac{B(k')}{C(k')} \right) - \frac{B(k)}{C(k)} \tag{3.17}$$

The $R_u(k)$ are kept sorted in a joint queue with decreasing order. In every step, the node with the highest relative benefit is chosen, provided it is eligible, i. e., $C(k') \leq C_{max}\ \forall\ L_{k'} \in children(L_k)$ for unfolding. This process terminates, if no more improvements can be found.

Greedy BFS quickly converges towards a useful result, but can get stuck in a local minimum. We therefore refine the BFS result with a random search approach based on threshold accepting [34]. Threshold accepting applies small random changes to the solution and temporarily accepts solutions that are worse than the current best configuration. We randomly select an unfolding and a folding operation (replacing a group of nodes by their common parent). The quality of the resulting configuration is determined as usual, via the cumulative benefits. Note that during this optimization step, the maximum cost given by equation 3.12 are not exceeded. The optimization terminates after a defined number of iterations, which makes performance very predictable.

Temporal coherence. After the initial selection, the labels are presented to the user. For interactive use, it is important to ensure temporal coherence and suppress jumping motion of labels. Therefore, labels are adjusted incrementally in every frame. After a change of viewpoint, the p_k are recomputed, and the queue containing the R_u is re-sorted accordingly. BFS is restarted on the re-sorted queue. Usually changes are small and continuous, so the optimization converges quickly after only a few operations.

Under typical conditions, the user will explore the information presented in the AR browser with sweeping rotational motions, while standing in a particular place [141]. Translational motion, i. e., walking while using the AR browser, is less common. We exploit the preference for rotational motion by computing p_k in *spherical coordinates* centered around the user, rather than in projective coordinates. This yields a label optimization, which is valid for any viewing direction, as long as the user remains in the same place. Rotating the camera thus selects a different viewport, but otherwise has no effect on the label selection. The label selection must only be restarted, if the user's translational motion exceeds a small threshold.

Local view management. Assigning a fixed position to labels turns label placement into a discrete label selection problem with lower computational demands. However, this discretization can lead to poor results, if many important labels occupy the same region. Moreover, representing a cluster by its centroid in space is not always a good choice.

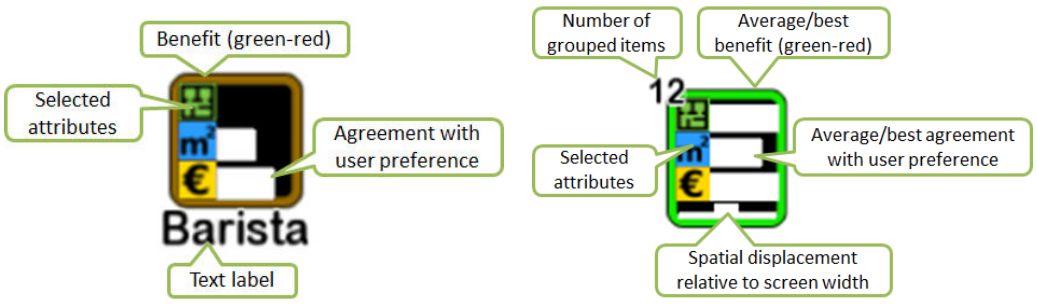
We increase the quality of view management after label selection by subjecting them to another optimization that purely considers spatial placement. This placement employs the "hedgehog labeling" technique [129] (Section 4.2), which places annotations in the 3D space to achieve stable layouts. We use the plane-based algorithm that constrains the movement of a label to a plane. In our case, the plane is placed at the geo-location of the point of interest and always parallel to the image plane. The scale is set up so that annotations have the same apparent size, independent of their distance. Alternatively, the scale can depend on the spatial distance from the user's location. This approach gives enough flexibility to compensate for poor initial placement, while ensuring temporal coherence of label adjustments. This approach can be seen as a coarse-to-fine optimization of label placement, where the label selection represents the coarse step and the label placement represents the fine step.

We use hedgehog labeling both when a node is first displayed and to compensate for local clutter after a viewpoint translation. Viewpoint rotation is already addressed by label selection based on spherical coordinates.

However, we must handle the case where a label moves off the screen. A leaf node will simply be omitted, but an intermediate node representing at least one data point on the screen must always be displayed. This problem can be handled by adding a constraint to the hedgehog labeling enforcing that only on-screen coordinates are eligible.

3.2.1.3 Glyph Design

The visual encoding stage transforms data points into glyphs encoding the relevant attributes. Glyphs are a common way to visualize multi-dimensional data in a meaningful way [143]. In our case, the glyph should inform the user about the represented data points and the relation of the different categories to the currently set user preferences. It also should have a compact visual footprint, so that it does not cover too much screen real estate. We use two variants of the glyph, one for leaves (individual data points) and one for intermediate nodes.



(a) (b)

Figure 3.24: Glyph design. (a) The glyph for a single data point encodes the underlying data by comparing it to a user defined reference value. The border color indicates the average matching quality of the selected attributes to the reference attributes (the greener, the better). The bars visualize how well selected attributes match the reference attributes (the longer, the better). (b) The glyph for grouped items shows the same information, but averaged over all items. Optionally, the user can choose to show the best contained item instead of the average. The number indicates the number of contained items. The bar at the bottom encodes the spatial extent of the contained data relative to the glyph.

The glyph for a leaf L_k (Figure 3.24(a)) should convey the relevance of the represented data point to the user directly. It consists of a square icon with a footer text describing the data point (e. g., stating a business name). The icon has a thick frame, which is color coded according to the leaf's benefit $B(k)$, where 1 is green and 0 is red. Inside

the square, there is room for up to three attributes selected by the user, arranged as a horizontal mini-barchart. Next to an icon identifying the attribute, the agreement of the selected attribute with index j to the user's preference is shown, i. e., $cf_j(A_{k,j}, U_j)$.

The glyph for an intermediate node (Figure 3.24(b)) summarizes the relevance of the leaves it represents. It is also a square icon. The number of leaves represented by the glyph is shown in the top left corner, similar to icons of popular mobile user interfaces. Like in the leaf glyph design, the frame of the glyph is color coded to show the average benefit of the contained nodes. A box extending on a line at the bottom of the glyph shows the spatial extent w_P of the cluster relative to the screen width, depicted as a fraction of the glyph width. Similar to the leaf glyph, a mini-barchart inside the square displays up to three selected attributes. However, the bar size for each attribute is proportional to the average agreement $avg(k, j)$ over all leaves with the user's preference, i. e.,

$$avg(k, j) = \frac{\sum_{L_i \in leaves(k)} cf_j(A_{i,j}, U_j)}{|leaves(k)|} \tag{3.18}$$

Averaging the content of the node provides a good general overview of the contained data values. However, when users look for the best matches to certain criteria, the averaging operation hides potentially good matches in the intermediate node visualization. To compensate for that, an intermediate node can alternatively represent the leaf node with the best benefit and adapt its appearance accordingly. The user interface contains a simple toggle switch to allow users to select the desired representation.

3.2.1.4 Interacting with Clusters

We allow user interaction in every step of the pipeline of Figure 3.20. To be able to change the structure of clustering during the data transformation, we provide the users with an interface, where they can adapt the weights of the attributes and, thus, change their preferences (Figure 3.25(a)). To facilitate the interaction, the interface allows users to specify the weights not as absolute values, but relatively to each other. Internally, the relative settings are mapped to weights that sum up to one, as required by the algorithm. Changing the weights triggers recalculation of the hierarchy (Figures 3.25(b) and 3.25(c)).

The user interface also allows users to specify the user preference values U that are used to calculate the benefit of the nodes. Consequently, this changes the selection of nodes from the hierarchy, but not the hierarchy itself. The visual mapping of the glyphs is also updated accordingly (Figures 3.25(c) and 3.25(d)).

In the view transformation step, the user must be able to manually unfold clustered

(a) (b) (c) (d)

Figure 3.25: User interface. (a) The user can change the weight of each attribute (numbers) and the attributes of the reference value (icons beneath numbers). (b) A selection of items after setting up the reference values and weights. (c) After changing the weight of attributes, the underlying groups are recalculated and the shown items are updated. Note the item with the blue frame in (b) and (c). Although the selection of presented items changes, the reference value and, thus, the white bars stay the same. (d) The display is again updated when the user changes the reference values. Note how the item with the yellow frame in (c) becomes green in (d), because it is a better match. The item with the red frame becomes red, because it is a worse match.

(a) (b)

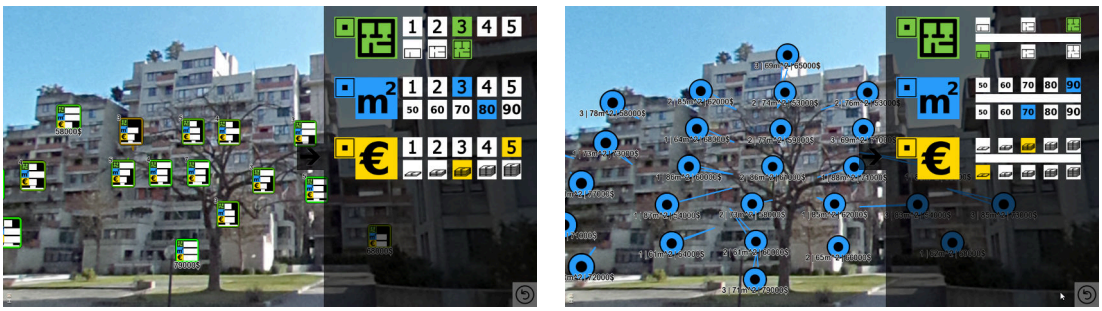
Figure 3.26: Interacting with groups. A user can unfold grouped items by clicking their glyph. (a) The user clicks on the glyph with the yellow rectangle. (b) The unfolded new elements are highlighted using a blue outline. To make room, other items are replaced by the group they belong to.

representations. We allow the user to drill down by unfolding the subsequent levels of the hierarchy step-by-step. The user simply clicks on a glyph representing an intermediate nodes to unfold the next level of the hierarchy. If the user's unfolding leads to a violation of the clutter metric, the system will first try to relax the situation locally via hedgehog label adjustments. However, it may occasionally be necessary to invoke fold operations on other labels to make room for the user's unfolding (Figure 3.26). This problem can be handled implicitly in the label selection optimization by assigning a higher weight to the benefit of the label unfolded by the user.

By default, this user-driven benefit will slowly wear off with an exponential decay. This allows the user to explore different areas on the screen or branches of the cluster hierarchy incrementally. Older user interactions will become less relevant over time and eventually make room for newer interactions. However, the user may instruct the system to remember choices indefinitely.

3.2.1.5 Evaluation: Comparing to Filter Interface

We performed a qualitative evaluation to compare our hierarchical clustering interface against a conventional filtering interface similar to the one used in similar AR browsers.



(a) (b)

Figure 3.27: Interfaces used in first study. (a) The interface that controls our information density display (HUI). The numbers change the weight of the attributes. The icons beneath the numbers change the selected attribute values. (b) The filter interface (FUI) used in the study uses the same attributes and resembles a slider interface. The user can set the range of the selected attribute values using the attribute icons. For instance, the blue attribute lies in the range between 70 and 90.

We investigated two interface conditions: our hierarchical clustering user interface (**HUI**) and a filter user interface (**FUI**). The folding and unfolding of the HUI behaved

as described before. We also allowed participants to switch the presentation of intermediate nodes between showing the average of all contained items and showing only the best contained item (Section 3.2.1.3). FUI behaved like a state-of-the-art AR browser filter interface. The data items were filtered according to parameters set up by the user. The data items were presented using a simple glyph representation, consisting of a circle and additional textual information (Figure 3.27(b)). Setting filter parameters removes information from the AR view that does not correspond to the filter parameters. If glyphs overlap, a force-based view management algorithm rearranges them to resolve the occlusion.

The interface condition was counterbalanced among the participants. We used the same amount of data in both interfaces, but changed the attributes of the data between conditions.

Hypothesis. FUI removes data that does not correspond to the current user preferences. However, the screen will still be cluttered, if too many data items are preserved. In addition, setting up the filter parameters in FUI to find relevant data points might be challenging. HUI aggregates items and, therefore, reduces clutter. HUI allows users to set a preference value to which the items are compared to, which makes it easier to identify relevant data items. Therefore, we expected our interface to be the preferred one.

Scenario and Setup. We used an accommodation search scenario in the evaluation. We gave the participants the task to find rental apartments that fulfill certain requirements. For this purpose, we performed the study outdoors in an apartment complex. The attributes of the data were created randomly and registered to the locations of the apartments.

We used the following apartment attributes: number of rooms (scale with three entries), square meters (scale with five entries), price range (scale with five entries). Participants could set one or more of the categories of each attribute. The text of the leaf node corresponded to the final price of the apartment.

We deployed the interfaces on a Surface Pro 2 running Windows 8.1 and used the front-facing camera for capturing the surroundings. We used a panorama tracker [141] to determine the orientation of the device and align the data with the real world. The resolution of the application was set to 1280x720 and corresponded to the camera resolution.

Task. The task was an apartment search task, where users are presented with a range of apartments. In order to see how participants would use the interface, we gave them an open task. We asked participants to use the respective interface to look for apartments that suited their criteria. The motivation behind this is that users identify

	Mode		Mean (SD)	
Question	FI	HI	FI	HI
I liked the visualization of data items in the interface.	3	1	3.375 (0.92)	1.375 (0.74)
I found the visualization of data items helpful.	3	1	3 (0.53)	1.25 (0.46)
The interface was convenient for finding apartments.	2	1	2.625 (0.74)	1.5 (0.76)
The interface was convenient for comparing apartments.	3	2	3.5 (1.07)	1.75 (0.71)
The interface provided a good overview of the data.	3	1	3.125 (0.99)	1.75 (1.04)

Table 3.1: Questionnaire results for first study. The mode and mean (with standard deviation) of the questionnaire results of a five-point Likert Scale (1 .. strongly agree).

interesting apartments and collect them into a list that would be revisited later to allow better comparisons and decision making.

Procedure. We met the participants at a meeting place, where they filled out the consent form and a demographic questionnaire. Then we moved on to the apartment site, where the study was performed. At the site, we explained the first interface to the participant. The order of the interfaces was counterbalanced. So participants either started with HUI or FUI.

After they were confident with using the interface, we asked them to solve the given task. We also asked the participants to speak aloud during the task. After finishing the task with the first interface, we repeated the same task with the second interface. After finishing the task with both interfaces, they filled out a questionnaire asking for feedback and rating the interfaces. We concluded the session by asking open questions regarding their experience.

Results and Discussion. A total of 8 people (3f) aged 26–34 ($mean$=31.5, sd=3.17) took part in the study. In the questionnaires, we forced participants to decide for either FUI or HUI as the preferred interface using a binary choice. Seven of eight participants preferred HUI. An exact binomial test found a significant difference ($\alpha = 0.05$) that HUI is the preferred choice ($p < 0.05$). In addition, we asked participants, if the intermediate node in HUI should show the average of all or the best item. All participants preferred the best item, because finding the best item is most relevant for a search task. The average would hide this information. In general, the questionnaire revealed that participants were in favor of our interface (see Table 3.1).

The one participant who did not prefer HUI argued that while the clustering reduces the amount of clutter, the registration of the items summarized by the cluster is lost. In

FUI, the location of an item was clearly visible, if the amount of clutter was not too high. Therefore, the participant suggested grouping items by stronger location-based criteria such as the floor of the building and also adding more options to the user interface for targeting items based on their spatial location to the user interface. Note that while we did not include stricter location-based groupings in our study, our system can easily support this by adding the respective attribute to the data.

In general, 62.5% of the participants made use of the real world registration of apartments during their search for an apartment. For instance, if several apartments had attributes of similar quality, the one that was in a higher floor was preferred. This underlines the usefulness of the spatial registration of items for this task.

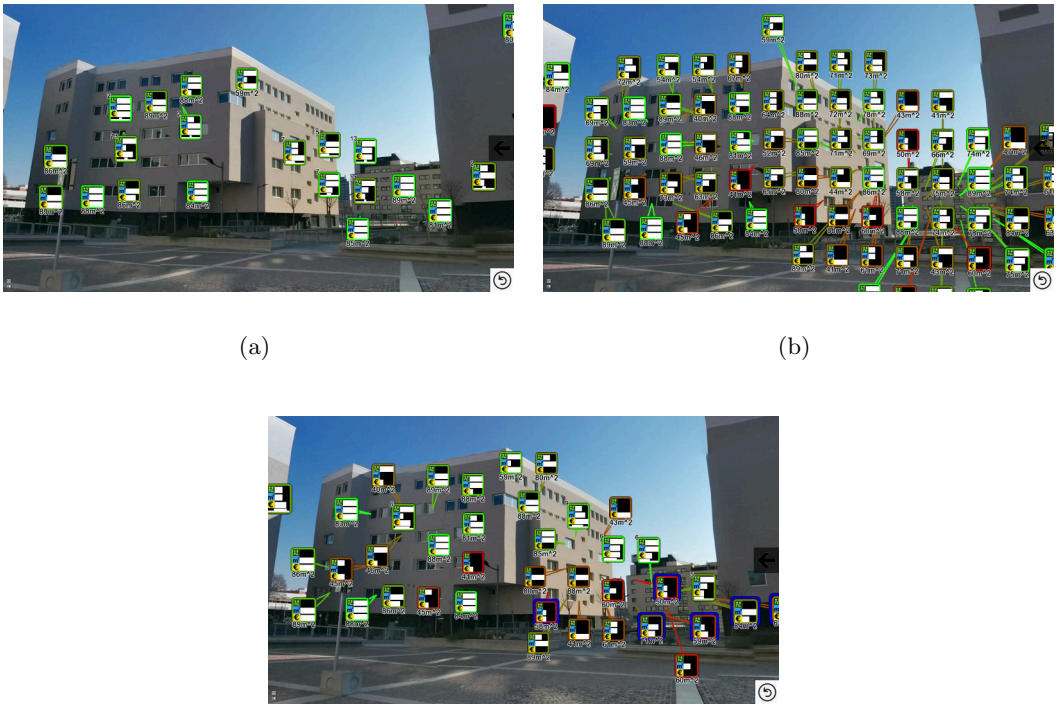
Most participants preferred HUI, because it provided a better overview of the data than FUI. In HUI, 75% of the participant did not only consider the best matches, but also checked for other apartments that were close to the set criteria. FUI reduced the amount of items to only a few, but information about other apartments was missing. 50% of the participants mentioned that increasing the search range of FUI would add more items to the visualization. This would also cause more clutter and make comparisons more difficult.

The eventual visual clutter in FUI made finding and comparing items difficult. This can probably be remedied by adding a similar glyph representation as in HUI to FUI. Therefore, we plan a follow-up study to compare HUI with respect to different degrees of spatial clustering. This study will determine if the interface benefits from the hierarchical clustering or if an appropriate glyph representation will be sufficient.

3.2.1.6 Evaluation: Different Degrees of Clustering

We conducted a comparative evaluation of three algorithms for POI clustering, to assess if the amount of information shown on the display would affect the performance in a search and in a successive recall task.

We investigated three clustering conditions in a between-subjects design. In the first condition, there was no clustering, and all the leaf nodes were displayed at the same time to the user (**LUI**) (Figure 3.28(b)). In the second condition, a clustering algorithm was introduced, in which items were grouped by proximity and the parameters set by the user (**SUI**) (Figure 3.28(c)). In this condition, the user was able to unfold groups of items. Once the items were revealed, the leaf nodes were not grouped again into clusters. Therefore, after a certain number of interactions, the display was populated by a growing number of leaf nodes, similarly to LUI. Finally, the third condition was **HUI**, as used in



(a) (b)

(c)

Figure 3.28: Interfaces used in second study. The interfaces used in the second study differed in the way items were clustered. (a) Our adaptive information density display (HUI) that uses hierarchical clustering, after unfolding a number of groups. (b) The interface using no clustering and showing all data items at once (LUI). (c) The interface using simple clustering (SUI) based on spatial proximity and similarity, after unfolding a number of groups.

the previous study (Figure 3.28(a)).

Hypothesis. LUI showed all available items, independent of their relevance to the user. After a number of interactions in SUI, a similar situation occurs, because the unfolded leaf nodes remain visible on the screen. In HUI, the unfolded leaf nodes are regrouped again into clusters, after additional groups are unfolded. We expected the leaf nodes populating the display in LUI and SUI to produce visual clutter and to affect the performance in a search and selection task and in a successive recall task of the previously selected items.

Scenario and Setup. We reused the accommodation search scenario of the first study, but conducted the second study at a different location. The interface had the same functionality as the HUI interface in the previous study. We added functionality to

perform the recall task. By pressing a button, the superimposed glyphs were removed, and only the view through the camera was shown. A participant could indicate the location of a previously selected item by tapping on the screen to mark its location with a white square. The apartment attributes remained the same, except for the text of the leaf node, which now indicated the size of the apartment in square meters. The experiment was run on a Surface Pro 2 tablet with the same characteristics as the one used in the first study.

Task. Participants were asked to perform two tasks. The first consisted of a search and selection task, in which participants had to find and select all the apartments matching the characteristics indicated by the experimenter. In the second task, they were shown the surroundings without any digital information superimposed, and they were asked to indicate the locations of the apartments that they remembered from the search and selection task.

Procedure. On the day of the test, participants were first briefed on the experimental procedure and aims. Then they gave informed consent to take part in the study. After they had filled in a brief questionnaire collecting background information, they were led to the spot where the test took place. First, participants were instructed how to operate the interface, then they were allowed to practice with the interface, until they felt confident. Next, the experimenter asked participants to search and select all the apartments matching certain characteristics: a size from 80 to 90 m2, in the highest price category and with the largest number of rooms. In total, there were 20 items matching the required characteristics with a total number of 219 items. Participants were told to alert the experimenter when they believed they had found all the items. Participants were instructed to be careful in performing the task, as they would be required to carry out a second task based on the first one. There were not explicitly told about the recall task, in order to prevent the use of mnemonic strategies. When participants told the experimenter that they had concluded the search and selection task, they started the recall task. Again, the participant was asked to tell the experimenter when the task was completed. Finally, participants were asked to complete a brief questionnaire to collect their opinions and impressions of the interfaces.

Participants. In total 36 participants (18f) volunteered in the study, 12 in each condition. The mean age was 24.27 (sd=2.5). Each subsample was composed of 50% women and was balanced for age, as confirmed by a one-way ANOVA $F(2)$=.067, p=.93. All participants had very limited previous experience with AR applications, if any.

Results. The number of items correctly selected, the number of wrongly selected

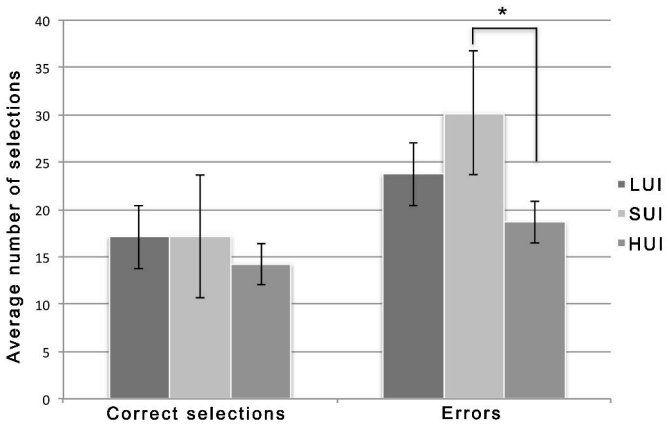


Figure 3.29: Correct and wrong selections of the search and selection task. When using the adaptive display, participants made significantly less wrong selections during the search task.

items and the task durations were compared across the three conditions with a one-way ANOVA (Figure 3.29). Statistical analysis revealed no significant differences in the number of correct selections ($F(2)$=1.88, p=.16). Similarly, the time required to complete the task did not significantly differ in the three conditions ($F(2)$=1.54, p=.22). A significant difference emerged in the number of errors users made ($F(2)$=5.04, p=.012). Post-hoc comparison with Bonferroni correction confirmed that with HUI, users made significantly less wrong selections ($median$=18.66, sd = 7.04) compared to SUI ($median$=30.18, sd=9.22). A reduction in the number of errors is evident when comparing the number of wrong selections in LUI ($median$=23.75, sd=9.6) and HUI ($median$=18.66, sd=7.04), even though the difference was not significant.

For the recall task, we computed the relative number of correct recalls as the ratio of the correctly retrieved locations and the number of correct selections of the selection task. Surprisingly, no significant difference emerged comparing the three indexes in a one-way ANOVA ($F(2)$=.26, p=.76). Regarding the post-use questionnaires, a Kruskal-Wallis test showed no significant differences in the way users evaluated the three systems in terms of information organization, ease of use, involvement, pleasantness of use and satisfaction. Even if users did not show enthusiastic evaluations of the interfaces for all the dimensions explored, a one sample t-test run against the central value of the response scale, i.e., 3, showed a trend ($t(11)$=2.22, p=.04) towards preferring HUI ($median$=3.6, sd=.9) in contrast to LUI ($median$=3, sd=.9) and SUI ($median$=3.4, sd=.79)

Discussion. The significant difference in the error rate during the search and selection

task indicates that participants were more focused on selecting the relevant items in the HUI condition. Interestingly, there was no significant difference in identifying the found items in the recall task. We believe that the reason for this is that the amount of clutter was not high enough to impact the perception of the presented data items.

For this study, the number of items was chosen in a way that the items were still clearly visible and distinguishable in the interface that did not use any clustering (LUI). However, this generally reduced the amount of clutter, and participants did not seem to have issues identifying items in the LUI and SUI interface, even though the screen was covered with items. Nevertheless, participants were slightly more satisfied with the adaptive interface (HUI).

3.2.2 Hierarchies in Compact Visualizations

The hierarchical clustering allows us to present different LOD of the information. Compact visualization can also be refined to allow for different LOD by introducing hierarchies of similar items. Hierarchies in compact visualizations allow us to gain a more fine grained control over the presented data. The data can be reduced even more or expanded with another level of information.

Hierarchies are often inherent in the filtered data. In the following, we refine the viewpoints of compact photo collections by adding an additional level of viewpoints. In a second example, we give an example where the number of representative items in a compact explosion diagram is reduced by detecting additional similarities in already similar groups of items.

3.2.2.1 Two-Level Compact Photo-Collections

For landmarks with a large number of associated images, we add a second level of representatives to each representative image. Figure 3.30 shows such a two-level compact visualization. In the middle row, first-level representatives are shown, while second-level representatives have been arranged in the upper and lower rows.

Clustering. Within each of the orientation clusters, we derive second level representatives by searching for images presenting the object in similar detail. We derive a measure of the amount of detail only by computing the distance from the GPS location of an image to its corresponding landmark, since camera zoom information is not consistently available. Thus, for each C_{O_j}, we again use k-means to create distance clusters C_{D_k} based on

Figure 3.30: Two-Level compact visualization. By additionally grouping the elements of each orientation cluster by their distance to the object of interest, we are able to select more than a single representative from each cluster. In the case of landmark presentations multi-level compact visualization allows to display a higher variation of images.

similar distance.

Selection of representatives. From each distance cluster, we select the image with the smallest distance to cluster center.

Layout optimization. Second-level representatives for geo-referenced photo collections are supposed to show the object of interest in a variety of different details. In order to maximize the variation, we optimize the layout using a measure of detail variation for a single row. To be able to show more interesting elements in more detail, we furthermore control detail variation using the number of images in a distance cluster C_{D_k}. We assume that more images indicate more interesting structure, and, therefore, we favor a more detailed visualization for representatives from those orientations.

Figure 3.30 shows the resulting two-level compact visualization. To each first-level representative, shown in the middle row, a set of second-level representatives has been added, each one of them demonstrating a set of images taken from similar distances to the explored landmark.

3.2.2.2 Explosion Diagrams

To create hierarchical compact explosion diagrams, the grouping algorithm discussed in Section 3.1.1.1 can be called recursively to detect similar subgroups.

Detecting Hierarchies. After applying the FSG search to the graph A_g of the whole assembly, a list of sets which contain the largest available non-overlapping subassemblies has been discovered. However, the selected subassemblies may even contain other frequent subassemblies. If we also identify these subassemblies, we are able to select a representative in multiple levels of the hierarchy, which in turn allows us to further reduce the number of displaced parts in a representative exploded view. To find frequent subassemblies within a previously determined subassembly, we apply the FSG algorithm recursively, until no subassembly can be determined anymore. When performing the FSG search on a set S of groups G, each group G is considered to be a separate graph to be mined for subassemblies. This means that a subsequent FSG search does not exceed the limits of the groups they are applied to.

By recursively applying the FSG search algorithm to a subassembly, we retrieve a hierarchy of frequent subassemblies. The groups of the detected sets and subsets are similar to each other, because their graph representations are isomorphic. However, subgroups of the same set may have different neighborhood relations to the group they are contained in. The reason for this is that the FSG mining algorithm removes all parts from the input graph, which do not have similar counterparts (for which only one label exists in the graph). Basically, this removes the contacts between any subgroups and the group they are contained in. By recovering this information, we are able to refine the hierarchy. This refinement allows us to choose better representatives from a set, because similar groups are then also distinguishable by their neighborhoods. Therefore, we define that similar subgroups G_l not only must be similar in terms of graph isomorphism, but also the neighborhood to the groups G_h they are contained in has to be similar. We implemented the following algorithm, which searches for similar neighbors of groups of a set.

For each neighbor of a group, the set of adjacent groups E_n is determined. Sets E_n of similar neighbors in different groups G_h are merged into the set E_s. Then, simple set operations are performed on the sets E_s to retrieve the common neighborhood for similar groups.

For a representative E_r from the sets of E_s, the following operations are performed in combination with each E_s: First, the intersection $E_c = E_r \cap E_s$ is created. If $|E_c| = |E_r|$, all groups share the same neighbor, and the algorithm continues. Otherwise, the groups of E_r share different neighbors. These groups are eliminated from E_r ($E_r = E_r \setminus E_c$). The algorithm continues, until either all E_s have been considered, or $|E_r| = 0$. Those groups left in E_r have similar neighborhoods. The algorithm finally terminates, when all sets of

E_s have been considered as representative set E_r.

Group-based Layout. If frequent subassemblies exist in an exploded subassembly, we cannot simply search for the biggest part in the main subassembly, because we also want to create a similar exploded view of all frequent subassemblies, even if they appear cascaded. Instead, we first compute a hierarchy of subassemblies, before we choose the biggest part from only the highest level of the hierarchy. The highest level ensures that no other part is similar to the chosen one, and, consequently, no conflicting explosion layout can result. Note that, once again, by removing entire subassemblies in an unblocked direction of a single representative member, we ignore collisions between parts during explosion. Even though this may result in physically incorrect sequences to disassemble the object, we are able to explode subassemblies independently of the overall model, which in turn enables to calculate a single explosion layout for all similar subassemblies.

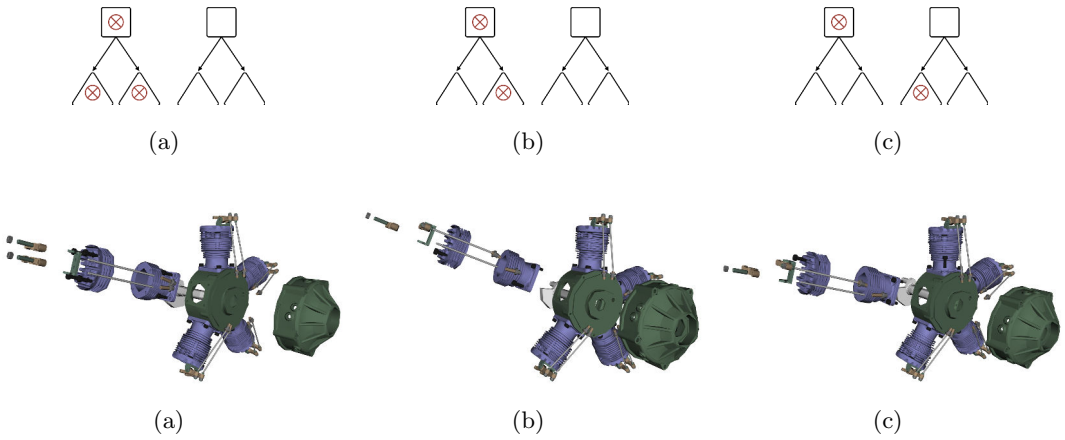


Figure 3.31: Selection Strategies in Hierarchical Groups. (a) All parts in a subassembly have been exploded. (b) Representatives have been selected in each level of the hierarchy. (c) Representatives have been selected in different subassemblies.

Hierarchical Subassemblies. If a hierarchy of groups exists, we allow to select representative exploded views using three different strategies. We allow either to choose the representative parts from a single subassembly (Figure 3.31(a), Figure 3.31(b)), or to select representative parts independently in different subassemblies of the same set (Figure 3.31(c)). If we chose to restrict the explosions to a single hierarchy, we have to decide if we want to explode the entire subassembly (Figure 3.31(a)) or only a single representative

in each level of the hierarchy (Figure 3.31(b)).

Figure 3.31 shows an example for each given situation. Since it is an open question which strategy results in the perceptually best results, our system allows selecting a strategy at runtime. The strategy shown in Figure 3.31(b) seems most reasonable, as it reduces the number of exploded parts compared to Figure 3.31(a), while representative parts are not scattered over the layout as in Figure 3.31(c). We leave a perceptive evaluation of the comprehensibility of each strategy for future work.

3.3 Conclusion and Future Work

In this section, we present methods to create overview visualizations of different data types, such as annotations and explosion diagrams. We avoid cluttered presentations by grouping data based on similarity measurements and only selecting representative items for each group. In this selection process, we also take the created layout into account and adapt the selection, if it would result in a degradation of the overall layout. For instance, overlapping annotations or annotations grouping in only one region of the image can already be avoided in the filtering step by selecting appropriate items from the input data set. Therefore, we reduce the conflicts view management algorithms must resolve by considering the layout that will be created when selecting certain items.

To achieve this, we have developed a framework which creates compact visualizations. Compact visualizations cluster redundant elements and select one representative item from each cluster to present in a comprehensible layout. However, if the number of redundant item groups grows, selecting a representative item from each can again lead to data overload. Introducing a hierarchy into the redundant data remedies this problem, because it creates additional LOD in the data structure. By visualizing different levels of the hierarchy depending on the available screen-space, the amount of presented detail can be reduced. We demonstrated such a LOD selection using hierarchical clusters of annotations. Note that while we did not use the compact visualization framework for the hierarchical annotations, the LOD selection can easily be integrated into the framework by adapting the selection process. We demonstrated hierarchical approaches for compact photo-collections and compact explosion diagrams.

The compact visualizations do not include any user interaction. However, they can be extended to support interaction. This becomes essential, when introducing different LOD into the compact visualization. A user can start exploring the data from the initial overview and then drill down to reveal more detailed representations.

Note that in the presented examples, compact visualizations use redundancies to group items that are very similar to each other. However, data can also be grouped using weaker similarities that are based on additional semantic information. For instance, we create the hierarchy of clusters for the annotations by aggregating items based on semantic information about their categories. This can easily be integrated into compact visualizations in general.

If it is not possible to detect similarities between data items, other filter methods must be investigated. One solution is using priorities that control which data items are presented in the final visualization. Such priorities can be previously set user priorities [12], but also based on geometrical properties, such as their size or visibility, if available. Furthermore, more advanced filtering methods could be introduced. For instance, personalized recommender systems can reduce the amount of data by learning the preferences of the user [107] and showing only relevant information.

For future work, several additional aspects of compact visualizations remain interesting and require further investigation. The current comprehensibility measures for compact explosion diagrams and compact annotations are based on perceptual considerations and observations derived from hand-made illustrations. These measures require a more formal evaluation. Furthermore, we want to investigate compact visualizations for other types of visualizations and their corresponding comprehensibility measures.

Another future research direction is the investigation and evaluation of the suitability of scene modifications for different AR applications. Although the modifications may be distracting the user, we believe that there are valid application cases. Scene modifications may be applicable to posters showing augmented content or table-top applications, where interferences between content should be resolved to avoid occlusions. For instance, an element of a poster may be used as control element such as a button and should not be occluded by a visualization. Using motion constraints for scene objects, the amount and type of modification can easily be controlled.

The evaluation of different degrees of clustering in the adaptive information density display was not conclusive with respect to the usefulness of clustering. We believe that the number of items presented during the study was not large enough to have an impact on the perception of the participants. Furthermore, the described studies did not relate the items shown in the AR view to the actual content of the AR view. We believe that clutter may have a stronger influence on search and recall tasks, when users must relate the items to the real world context. With an increasing amount of clutter, real world features become

increasingly occluded. Therefore, it may be more difficult to identify features of the world that items relate to.

In addition, an open issue is the integration of real world geometry into account during the hierarchical clustering phase. Using more sophisticated spatial information, the clustering can avoid grouping items that are located, e.g., in different floors of a building. Having such spatial information would also be an interesting extension of the user interface.

Until here, we also did not investigate temporal coherence for compact visualizations. This important aspect will be discussed in the following chapter (see Chapter 4).

Chapter 4

Temporally Coherent View Management

Contents

Until here, we have presented methods to create layouts for different types of information by either filtering or aggregating the input data, before presenting the filtered information. The information is presented in a layout that is created for a certain viewpoint. However, in AR, a user can constantly change the viewpoint on the data, which causes changes to the layout. These changes may be very distracting for users and also hard to follow. Therefore, to avoid distracting movement, the layout should be temporally coherent across adjacent viewpoints.

In this chapter, we extend our compact visualizations with the aspect of temporal coherence to avoid distracting changes and visual clutter of the layout. We achieve these goals by influencing the selection of the representatives that make up the final layout already in the filtering step. Furthermore, we present a new algorithm for creating layouts of annotations that completely works in 3D object space and, therefore, is much more stable during camera motion than 2D view management techniques (see Section 4.1).

Note that, for our work on compact visualizations, we relied on a 2D view management algorithm to create the annotation layouts. However, while performing this research, we

discovered that 2D view management techniques are not optimal for creating layouts of annotations in AR, as frequent changes occur due to the constantly moving AR camera. Therefore, we present a new constrained 3D view management algorithm for annotations that inherently is more stable than a 2D view management solution (see Section 4.2).

After introducing our novel 3D view management system, we perform a user study to quantitatively compare it to more traditional view management systems. The study shows that our 3D layout system, which registers labels as 3D objects relative to the annotated object, outperforms the reference systems in a task to locate and relocate labels (see Figure 4.3).

4.1 Compact Visualization: Optimizing for Temporal Coherence

Compact visualizations find an appropriate layout for presenting information from a certain viewpoint. However, the optimization results for adjacent viewpoints may differ, for instance due to changing occlusion relationships. Hence, the layout may change frequently during camera motion. Strong layout changes grasp the user's attention and, thus, may distract her from exploring the data. We show how to reduce these distractions by creating temporally coherent layouts during optimization using two different approaches. The first approach creates layouts which minimally differ from the neighboring layouts. We call these layouts "aligned". A layout is aligned, if it has both a high similarity to its neighbors and a high quality in regard to the other quality parameters. The second approach does not take neighboring layouts into account, but optimizes the layout of the current viewpoint to minimize the amount of interferences when changing the viewpoint. Both solutions can be integrated in a straight-forward manner directly into the general framework for creating compact visualizations by simply defining additional quality criteria.

4.1.1 Minimally different neighbors

By using aligned layouts, we can reduce distracting layout changes by minimizing variations between neighboring viewpoints. For this purpose, the difference between the current layout and the neighboring layouts is estimated and incorporated into the optimization using a new quality criterion. The difference between two layouts is determined by computing the difference $Diff(L_1, L_2)$. The alignment quality $Q_{Alignment}$ is then determined as follows in (4.1).

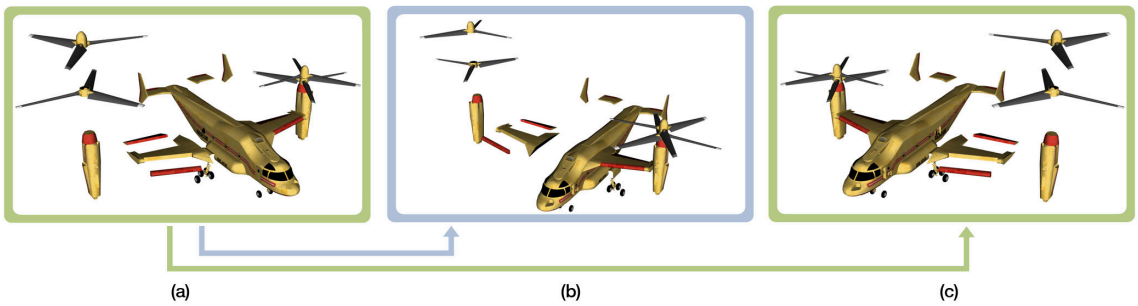


Figure 4.1: Aligned compact explosion diagrams. (a) and (c) depict the optimal layouts for the respective viewpoints from the set of prepared optimal layouts. Note the difference between the layouts, when changing between these two (*green arrow*). (b) To reduce the amount of changes during camera motion, we switch to aligned layouts and present these during transition (*blue arrows*). Note how in the aligned layout only small changes are performed to ensure that all parts are visible. After finishing the transition, we switch to the optimal layout for the current viewpoint.

$$Q_{Alignment} = 1 - Diff(L_1, L_2) \tag{4.1}$$

By incorporating the difference between two layouts into the layout optimization, we generate layouts that differ only minimally. We also consider the other quality criteria for creating comprehensible visualizations. However, aligned layouts may not always represent the absolute best layout for a viewpoint. Therefore, we optionally switch back to the optimal layout once the camera movement stops. This can be modeled by dynamically adapting the weights of the quality criteria. More weight is added to $Q_{Alignment}$ during movement, while the weight is reduced to zero, when the camera is not moving. We achieve interactive frame rates for temporally aligned layout transitions by preparing aligned layouts for each optimal layout.

The difference $Diff(L_1, L_2)$ between the exploded layouts L_1 and L_2 is determined by the l_2 distance of the respective descriptors containing the positions of the parts relative to the center part of the explosion layouts. Let $P_{i,1}$ and $P_{i,2}$ be the positions of the same parts in two layouts L_1 and L_2, and let P_0 be the origin of the model. Then, for each $P_{i,j}$, we calculate the vector $V_{i,j}$ from P_0 to $P_{i,j}$ and normalize it with the length of the 3D diagonal of the fully exploded model $Diag_{3d,exp}$ (4.2).

$$V_{i,j} = \frac{(P_{i,j} - P_0)}{Diag_{3d,exp}} \tag{4.2}$$

We calculate the difference $Diff(L_1, L_2)$ between two layouts as the Euclidean length of all distance differences between each $V_{i,1}$ and $V_{i,2}$. The more similar the layouts, the smaller the difference $Diff(L_1, L_2)$ (4.3).

$$Diff(L_1, L_2) \quad = \quad \sqrt{\sum_{i=1}^{n} |V_{i,1} - V_{i,2}|^2} \tag{4.3}$$

Figure 4.1 demonstrates the alignment of layouts in neighboring viewpoints for compact explosion diagrams. Note the differences between the best layouts of the respective viewpoints ((a) and (c)), and the smaller difference between the corresponding aligned layouts ((a) and (b)). When the camera movement starts, we switch to aligned layouts, in order to reduce the amount of variations over time. Note that, although the changes are minimal, layouts are created in which all parts are visible. Hence, $Q_{Alignment}$ is able to work in combination with other criteria.

4.1.2 Minimize potential distractions

Using aligned layouts for neighboring viewpoints works well for visualizations with a manageable amount of changes and interferences, such as with explosion diagrams. Parts can only be exploded in certain patterns, which limits the degrees of freedom of moving parts. Therefore, the amount of changes between adjacent viewpoints is limited. However, the placement of other visualizations, such as annotations, may be far less constrained. Even small camera motions may cause a change of label positions, label order and anchor points in adjacent viewpoints. Hence, it is difficult to find minimally different adjacent layouts. Instead, we can minimize the distractions during viewpoint changes by optimizing the layout of the current viewpoint for avoiding any distractions.

For compact annotations, distractions arise from order changes and representative substitutions, which change the respective anchor points. When we lock the set of anchor points during viewpoint changes, the order of annotations varies uncontrollably. In contrast, fixing the order of labels, while varying the anchor points to resolve leader line intersections, eventually leads to frequent changes in the visualization. Therefore, we lock both the order of annotations and the anchor points to ensure temporal continuity.

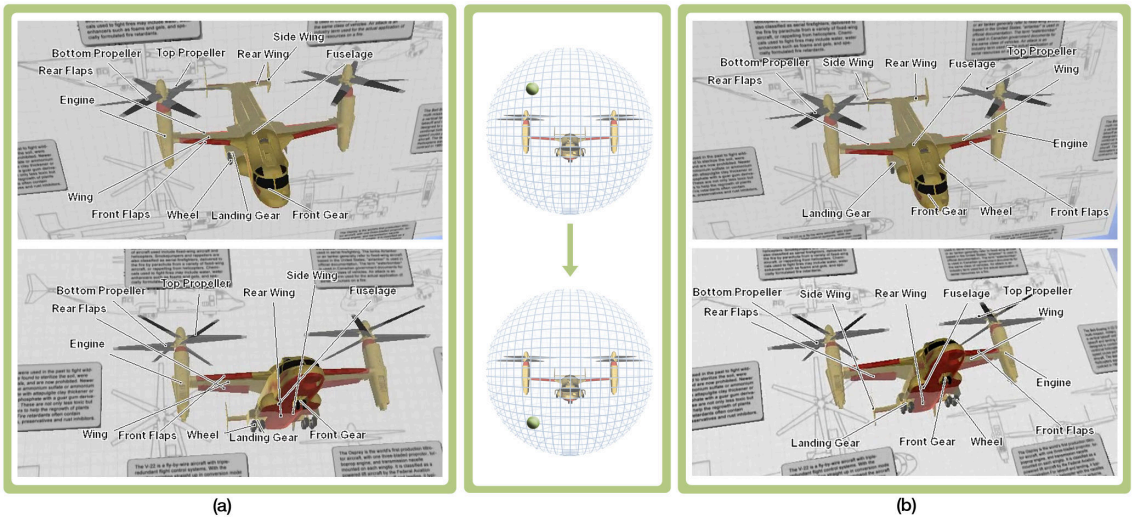


Figure 4.2: Distributing representatives. We spread labels over the layout to support frame coherent updates. (a) Fixing both the order of labels as well as their anchor points to the most comprehensible compact annotation will result in a large amount of line intersections, while the camera moves around the object. (b) By maximizing the distribution of annotations in the layout of the first viewpoint, line intersections can be reduced. Only a few line intersections appear while moving the camera to the second viewpoint.

Furthermore, we aim to minimize distracting line intersections between viewpoint changes. Figure 4.2(a) shows that after changing the viewpoint, a large number of leader line intersections may occur. To minimize such leader line intersections, we maximize the distance between labels during re-selection of representatives using $Q_{Distance}$. In addition, we add a quality criterion Q_{Anchor}, which implements the heuristic that anchor points lying farther apart are less likely to produce line intersections between the associated labels. The new criterion maximizes the minimal distances between the projected representative anchor points A_i, as seen in (4.4).

$$Dist_{Anch(i,j)} = \frac{|A_i - A_j|}{\sqrt{ImageWidth^2 + ImageHeight^2}}$$
$$Q_{Anchor} = \frac{1}{n}\sum_{i=0}^{n}\left(\min_{j=0,j\neq i}^{n}(Dist_{Anch(i,j)})\right) \tag{4.4}$$

The resulting layouts consist of elements which are less prone to leader line intersections in close viewpoints and, therefore, contain less distractions. Figure 4.2(b) shows that after

maximizing the distribution, only few leader line intersections occur. We resolve the remaining intersections once camera movement stops by animating the transition to the optimal intersection-free layout for the current viewpoint. Such an approach to updating the layout of annotations was also suggested by Ali et al. [3], who do not update the layout of annotations during user interaction, but only after the viewpoint stabilized.

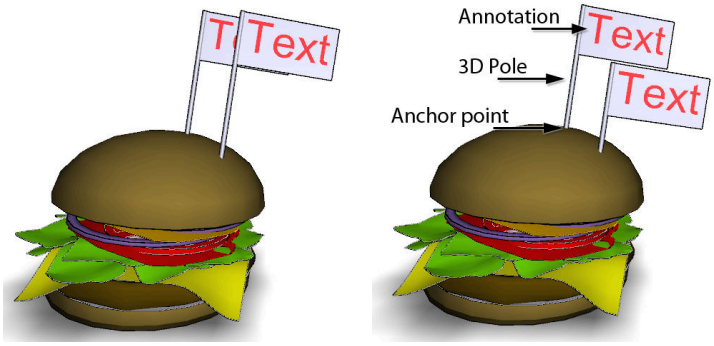
4.2 Hedgehog Labeling: Stable Annotations in Object-space

A number of different techniques have been proposed to control the placement of external labels. For instance, for our work on compact annotations, we used the floating labels approach of Hartmann et al. [56]. They have been successfully applied to produce high quality layouts for desktop applications. However, since most of the existing techniques operate in 2D image space, they are prone to unpredictable changes over time. This happens, because the distribution of the projected 3D points changes during camera movements, which can force the view management system to frequently re-order external in labels image space. With increasing amount of label movement and re-ordering, the label motion becomes difficult to follow, and the resulting layout becomes unstable over time. This is illustrated in Figure 2.7 and Figure 2.8.

To overcome the problems of such floating labels, traditional desktop applications often display external annotations only when camera movement stops. However, in AR, the camera is attached to the user, and thus it is always in motion.

In this section, we present a new view management technique for external labels in 3D space. Since frame incoherent placement of labels is often caused by unpredictable changes of the projection of 3D points into the image space, we annotate the object of interest in 3D object space using external 3D labels. During camera movements, this strategy enables us to apply arbitrary 3D transformations not only to the 3D objects in the scene, but also to their labels. Since labels follow object transformations, our approach allows to better follow label movements over time.

To further support this objective, our view management approach applies changes to the layout based on the 3D geometry of the label. A 3D label consists of a 3D annotation, a 3D pole, and an anchor point. We only allow adjustments to the length of the pole, while the orientation of the pole is fixed in object space. Figure 4.3 illustrates this strategy, which resembles a “hedgehog”. Its application to AR is shown in Figure 4.4(a).



(a) (b)

Figure 4.3: Illustration of view management in 3D space. (a) External labels occlude each other. (b) Instead of searching for an occlusion free layout in 2D image space, we adjust 3D properties of the label's geometry (in this case the length of the pole) to resolve occlusions between the annotations.

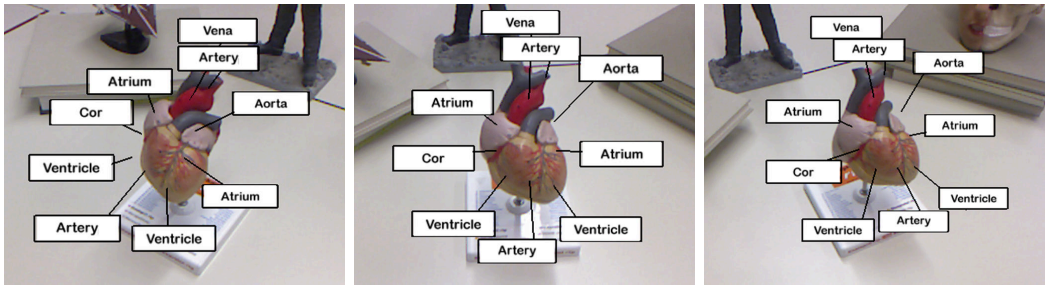
While this approach produces aesthetic and stable layouts, it may stack labels which anchor points are located close to one another. To resolve stacked label layouts, we introduce layout strategies operating on a 3D object space approach. For example, Figure 4.4(b) shows the result of a balanced label distribution, which has been computed based on a set of planes in 3D space.

Defining all elements of a label in 3D object space allows us to generate more predictable layout changes and thus less distracting label motion during camera movements. Therefore, we build an external label out of a *3D annotation*, a *3D pole* (leader line equivalent in 3D), and an *anchor point*. A 3D annotation can be a 3D or a 2D object (text, image) projected onto a billboard (Figure 4.3(b)).

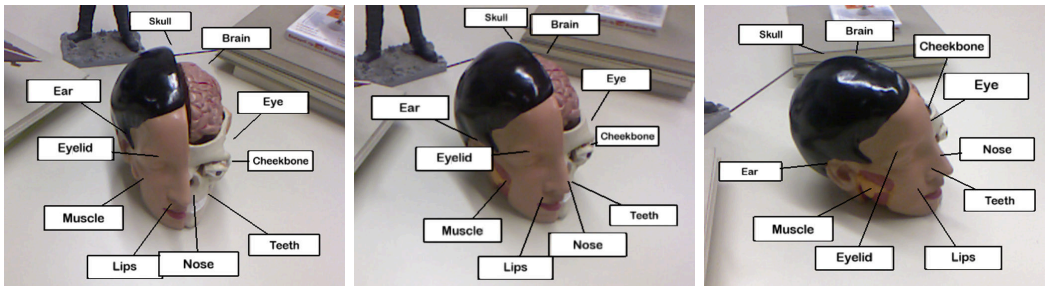
Our view management approach consists of an *initialization* phase followed by an *update* phase. In the initialization phase, we place a 3D label for each element which we want to annotate. For each 3D label, we define the position of its anchor point, the orientation and length of its pole as well as the orientation of its annotation. After initializing all labels, we start updating the layout to resolve occlusions and to maintain readability. Based on the readability, layout updates can be continuous or discrete.

4.2.1 Layout Initialization

During initialization, we position each label in the 3d scene. The behavior of a label and thus the appearance of the layout at run-time depends on the design of a 3D label and the



(a)



(b)

Figure 4.4: View Management in 3D space. (a) Label placement has been constrained by 3D poles which originate from the center of the object. To resolve occlusions, we move labels along the pole only. (b) Label placement has been constrained by a set of planes in 3D space. Labels are allowed to move within a plane, which is fixed in 3D space. To avoid constant label motion, the label positions are frozen after creating the layout for a viewpoint. The placement is updated only when the viewing angle to the plane grows larger than a threshold.

strategy to resolve occlusions. In this section, we discuss design decisions, such as where to place or in which direction to move a 3D label and which constraints this implies.

Orientation of 3D Annotation. The most common 3D annotation is a flat two-dimensional surface in 3D space, which is attached to the pole on one side. This configuration allows to rotate the surface around its pole only. While such 3D labels appear very natural, they easily suffer from perspective distortion, making the information unreadable from certain points of view. If the angle between the view vector and the 3D pole is high, a rotation of the annotation around the pole allows re-orienting the label so that its information can be easily read. However, this is not possible if the angle between the pole and the view vector is small. Such a configuration will cause the annotation to rotate

almost around the user's view vector.

Since the orientation of the view vector can change arbitrarily at run-time, we cannot guarantee a sufficiently large angle between the view vector and a 3D pole. Therefore, we allow for unconstrained rotations of the annotation. Instead of attaching one side of the annotation to the upper part of the pole, we attach the tip of the pole to the center of the annotation. This allows to rotate the annotation around all three axis of its local coordinate system which is placed in the center of the annotation. During initialization, we orient annotations parallel to the screen.

Position of 3D Anchor Point. To easily link the annotation to the 3D object, we have to place the anchor point of the label on the 3D object. The most unambiguous position is its center, which we approximate using the center of its bounding sphere.

Length of 3D Pole. The pole has to be long enough so that the projection of the annotation is not covering the 3D object of interest. Yet we want the pole length to be minimized, so that annotated scenes are compact. Since we resolve occlusions after the initialization phase, we can just ignore the length of the pole during initialization by placing the annotation at its anchor point.

Orientation of 3D Pole. When placing an external label in 3D space, the most natural orientation of the pole follows the direction of the surface normal at its anchor point. However, as demonstrated in Figure 4.5(a), this strategy easily suffers from crossing leader lines after projecting to camera space. Notice the crossing between the pole of the label of the engine and the one of the right door of the car. As the camera rotates around the object, other leader lines cross in image space.

In order to avoid crossing leader lines, we orient 3D poles using the normalized vector which originates from the center of the object's bounding sphere and passes through the anchor point of the 3D label. This strategy is illustrated in Figure 4.5(b). Note that the leader lines do not intersect anymore.

4.2.2 Layout Updates

After initializing labels, or after the camera has moved, occlusions with other labels or scene objects may occur. Since finding the optimal place for all labels has been proven to be NP-hard [92], we formulate the problem of minimizing occlusions as a force-based optimization problem. Our approach is inspired by the image space approach of Hartmann et al. [56]. However, unlike Hartmann et al., we assume constrained motion in 3D space.

One Degree of Freedom. A simple constrained motion would allow every annotation

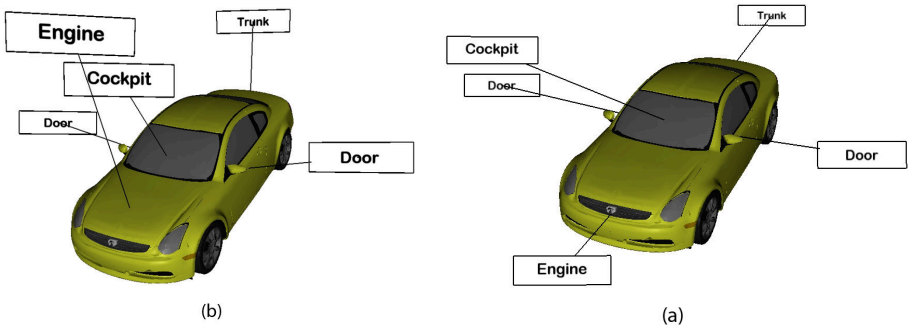


Figure 4.5: Normal vs. radial 3D pole orientation. (a) Orienting the pole using the face normal at the anchor point results in crossing poles in image space. (b) To avoid crossing poles, we orient 3D labels radially from the center of the bounding sphere of the object, i.e., aligning them with the vector pointing from the center of the bounding sphere to the anchor point of the label.

to slide along the pole direction (one degree of freedom). Figure 4.6 demonstrates the result of this strategy. Notice how occlusions have been resolved for the two annotations, which have been marked with a red exclamation mark.

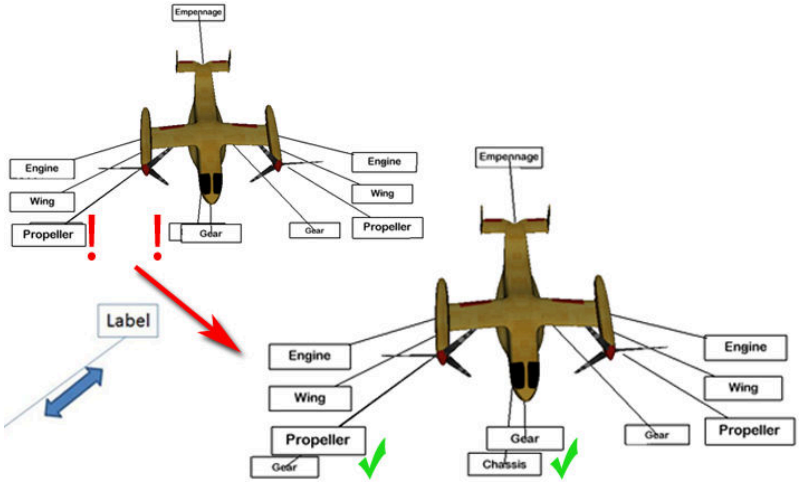


Figure 4.6: Resolving occlusions using one degree of freedom. To provide predictable movements, we allow moving the 3D label along the pole only. The upper image suffers from two occluded annotations, which we marked using a red exclamation mark. By extending the length of the pole, the system is able to resolve the occlusions.

Three Degrees of Freedom. Resolving occlusions using a single degree of freedom only produces a small amount of very predictable motion. However, this strategy tends to stack annotations if poles have similar orientation in 3D space (see the upper image

in Figure 4.7). To generate more balanced layouts, we additionally allow to move the annotation in the image plane (two additional degrees of freedom). However, to ensure that the pole is always connected to the annotation, we limit the motion in the image plane to the size of the annotation.

This strategy is illustrated in Figure 4.7. The label layout in the upper image suffers from a number of stacked labels. For example, the three annotations at the bottom (left gear, chassis and right gear) form a stack in y-direction. To resolve this stack, our system moves the annotation of the chassis in the image plane, until it fits next to the gear. Since the orientation of the pole for the right gear does not allow to place the annotation next to the gear, our system cannot resolve this stack. While a perfectly balanced layout may require more freedom of movement, this approach generates reasonably good results with a limited amount of predictable movements for a small set of stacked labels.

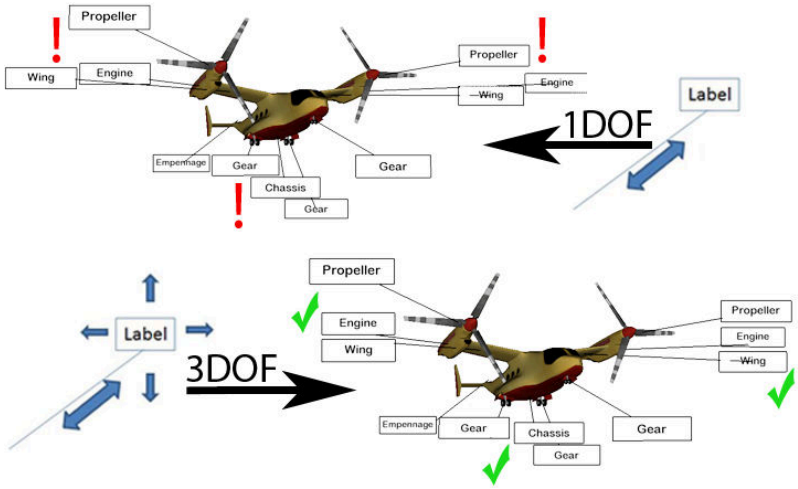


Figure 4.7: Resolving occlusions using three degrees of freedom. Stacked annotations appear for labels with poles oriented at a similar angle. Stacks have been marked in the upper image using red exclamation marks. To resolve stacks, we allow moving an annotation along the pole and within the X/Y plane of the annotations local coordinate system.

Plane-Based Occlusion Management. Our approach to stabilize the layout avoids re-orientations of 3D poles. However, this performs poorly, if many anchor points are in or close to a plane in 3D space. Such configurations lead to generate label poles inside a narrow slab and result in layouts suffering from stacked annotations, if rendered from a viewpoint perpendicular to these planes.

For example, most of the anchor points used to label the ship in Figure 4.8(a) have

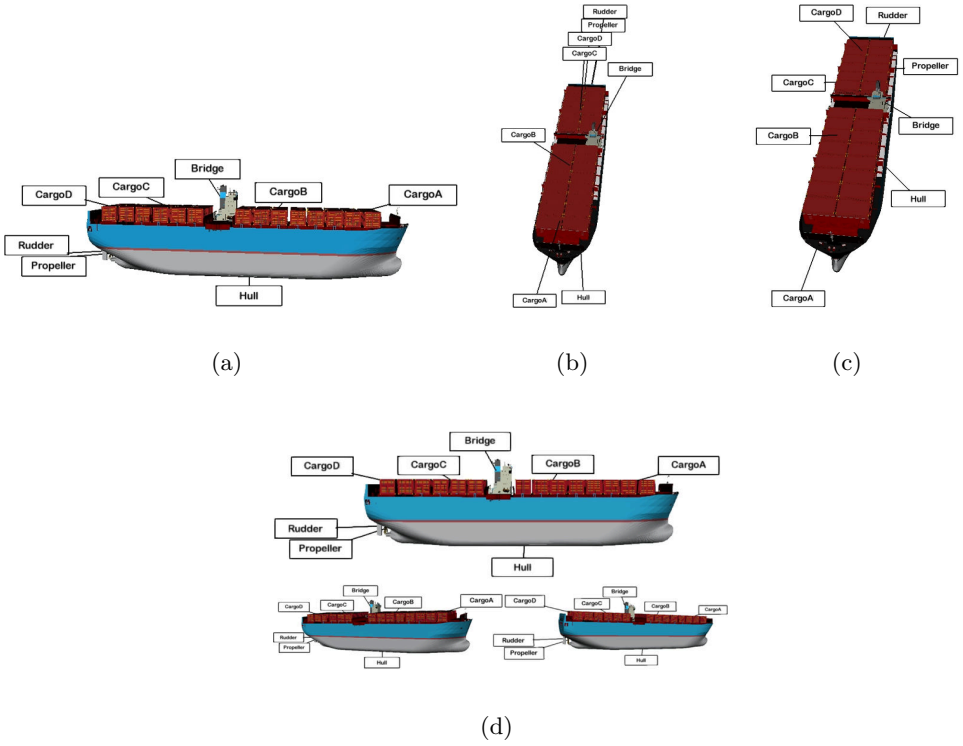


Figure 4.8: Plane-based occlusion management. The anchor points in this example have all been placed close to the x/y plane of the model. (a) The center-based labeling approach, which uses a common center to orient labels, generates a clear layout if the camera is oriented along the z-axis. (b) If the camera orientation is similar to the x- or the y-axis, the layout suffers from heavily stacked annotations. (c) Since our occlusion management approach is not able to resolve heavy stacking, we group annotations into 3D planes. This allows us to generate more balanced layout, while still providing stable layouts over time. In this example, we use three different planes, one in the front, one in the middle and one for anchor points in the back of the ship. (d) The plane-based label placement with planes generated from viewing along the z-axis.

been placed close to the x/y plane of the coordinate system of the 3D CAD model. This causes most of the label poles to lie in the close proximity of the x/y plane. When looking at the object along the z-axis, the layout generated with our approach seems adequate (Figure 4.8(a)). However, if the viewing direction becomes more similar to the x-axis, the 3D poles line up in image space, causing the algorithm to stack annotations, which cannot be resolved with the constraints we introduced.

To balance the layout, we have to relax our constraints. Therefore, we group all labels into a set of planes, and we allow searching for suitable positions for annotations

within an entire plane. We orient the planes parallel to the view plane, and we place them equidistantly in the screen-aligned bounding box of the object. Each label is automatically assigned to the plane that is closest to its anchor point. This allows us to apply any image-based view management approach from the current point of view. Specifically we achieve balanced layouts using the spring embedding approach proposed by Ali et al. [3]. The result can be seen in Figure 4.8(c) and Figure 4.8(d). Both images have been rendered with the same groupings, but from different points of view using differently oriented planes. Note that the stacking of annotations has been resolved.

This approach generates more balanced layouts. However, if occlusions will be resolved in each frame, this approach introduces frame inconsistencies which are similar to those resulting from a force field approach in image space. Therefore, this approach is best applied in combination with a discrete update strategy.

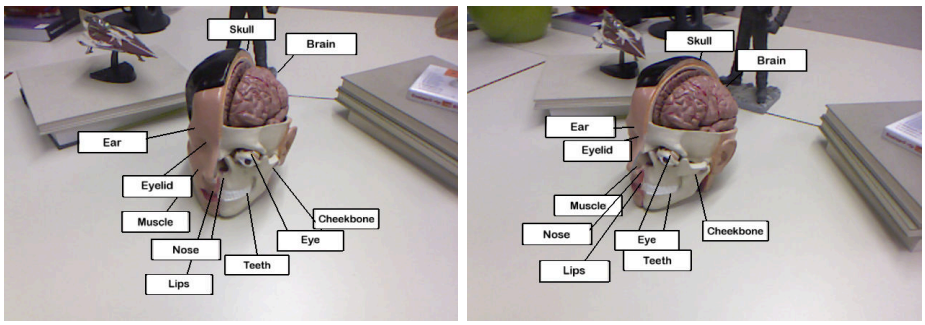
To reduce the amount of motion for plane-based occlusion management, we freeze the layout after resolving occlusions, and we update the annotations only if the angle between the view vector and the normal of the annotation's plane grows larger than a user-defined threshold. This behavior is demonstrated in Figure 4.8(d). The layout has been generated from the point of view in the upper image of Figure 4.8(d). Both renderings in the lower part of Figure 4.8(d) use the same layout from a slightly offset point of view.

While we freeze the position of an annotation within a plane, we still update its orientations in every frame so that it always faces the camera. This approach works well for sufficiently distant labels within the plane, but it may cause occlusions with other annotations, if the layout algorithm places them close to each other. For example, both Figure 4.9(b) and 4.10(a) use our plane-based occlusion management approach. However, because the annotations in 4.10(a) have been placed close to each other, occlusions between annotations appear during camera movements. We can enforce a certain spacing between annotations by adding a force to layout algorithm which maintains the spacing. However, with an increasing amount of annotations, this approach pushes annotations further outwards.

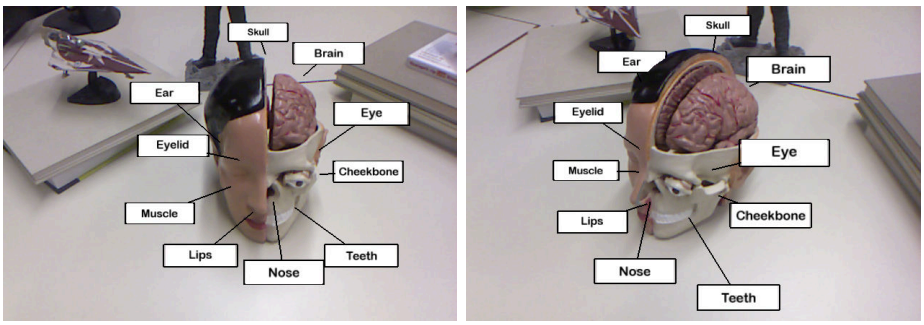
Therefore, we support an additional discrete update strategy. Since annotations cannot occlude each other if they lie in the same plane, we also support freezing the orientation of annotations (Figure 4.10(a)). This allows us to avoid occlusions between annotations which lie in the same plane, however, this strategy introduces perspective distortions of annotations. Therefore, it should be used only if necessary and in combination with a small angle between layout updates.

4.2.3 Implementation

Our view management prototype runs in real time (30Hz) for all presented illustrations and scenarios with a 640x480 rendering resolution. We deployed our application prototype on a PC running Windows 7, equipped with an Intel i7 CPU quad-core 2.66GHz, 12 GB RAM and an Nvidia 780GTX graphics card. The software was implemented in C++ using OpenSceneGraph[1], an open source scene graph library. The AR framework used in this project uses KinFu, an open source implementation of the KinectFusion approach of Izadi et al. [65], which is available through the Point Cloud Library [114]. KinFu was operated with a Microsoft XBox360 Kinect depth sensor.



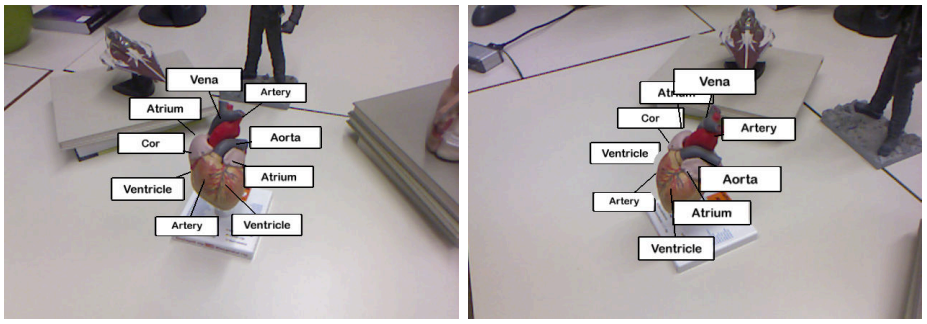
(a)



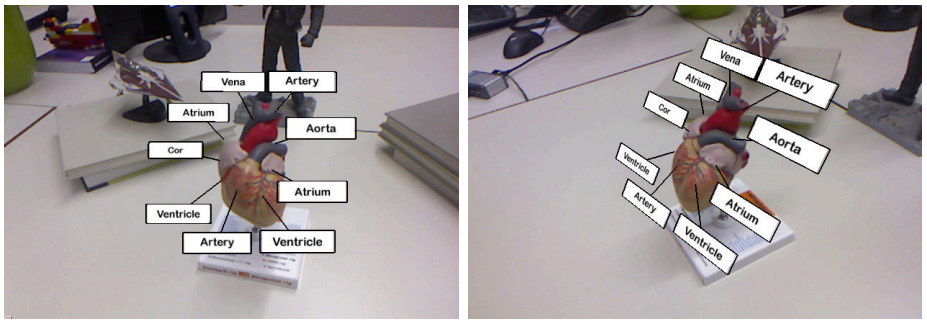
(b)

Figure 4.9: Center-based versus Plane-based Label Placement. (a) Center-based labeling causes unbalanced layouts for this configuration. (b) This can be resolved using plane-based label placement.

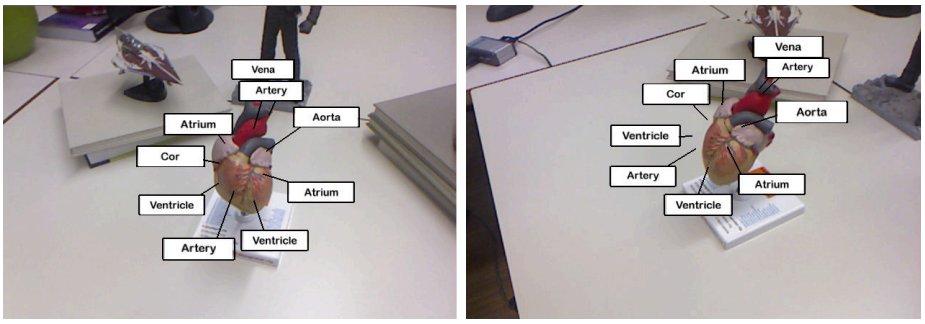
[1] http://www.openscenegraph.org



(a)



(b)



(c)

Figure 4.10: Center-based versus Plane-based Label Placement. (a) Plane-based label placement may suffer from occluding annotations if annotation have been placed close to each other. (b) To avoid such occlusions, plane-based label placement in combination with freezing the orientation of annotations can be used. However, this introduces perspective distortions and should be used with care. (c) For this configuration, center-based label placement creates appealing results which neither suffer from occlusion nor from distortion.

4.2.4 Comparison of Variations

Figure 4.10 allows side-by-side comparison of center-based and plane-based label placement. Figure 4.9(a) uses the center-based approach. Even though we chose a rather spherical object, most anchor points are placed on one side of the object, namely in the face. Therefore, our center-based approach suffers from stacking of annotations and rather long 3D poles. For this configuration, the plane-based approach creates more appealing results (Figure 4.9(b)). However, plane-based label placement requires to re-orient the label pole and, thus, should be used only if necessary.

When deciding to use a plane-based layout, one has to take into account that either occlusions between the labels occur or perspective distortions of annotations. Occlusions occur, when only the position is frozen, but the annotation is always aligned to the screen (Figure 4.10(a)). This result is visually more equivalent to previous state-of-the-art 2D view management technique [56] without their current restrictions. On the other hand, perspective distortions appear when both position and orientation are frozen and annotations are always plane-aligned (Figure 4.10(b)). Since both approaches may impair the comprehensibility of the visualization, center-based 3D label placement should be considered if anchor points are well distributed around the object of interest (Figure 4.10(c)).

4.3 Evaluating Coherence in View Management

In this section, we present a formal user evaluation, which compares view management algorithms that continuously update the layout to algorithms that only update the layout at discrete points in time. To the best of our knowledge, the temporal behavior of labels over time has never been part of an evaluation of different view management algorithms. Literature usually describes a set of constraints and methods to enforce these constraints by continuously updating the layout. An open question is if such updates have a negative impact on the performance of a user during certain tasks, because they constantly change label positions.

Our intuition was that even though discrete view management algorithms cause violations of the layout constraints during viewpoint changes, they would outperform the continuous versions in search and select tasks that are typical for AR applications using annotations. For this purpose, we chose to evaluate common force-based view management algorithms that work in 2D image space [56], but also in 3D object space [104, 129], such as the hedgehog labeler presented in Section 4.2. Based on our findings, we put forward

design recommendations for view management systems.

Our focus lies on handheld AR scenarios, which typically exhibit constant viewpoint changes during interaction. We limit ourselves to view management approaches that use external labels, because Coelho et al. [28] have shown that external labels are less ambiguous in case of tracking errors.

4.3.1 View Management Algorithms

In the following, we review the four view management techniques that we evaluated in the experiment. The implementations enforce common constraints found in hand-drawn illustrations [57]: Place annotations in the vicinity of the object (R1), avoid overlaps between annotations and the annotated object (R2), and avoid crossing leader lines (R3).

The four techniques vary in combinations two factors: the *update approach*, and the *space* in which the annotations are described and simulated algorithmically. The update approach consists of the levels continuous and discrete. Continuous means that the view management algorithm continuously updates the layout of annotations to enforce the layout constraints. Discrete means updates are applied to the layout only at discrete points in time based on certain conditions, such as the amount changes of the viewpoint. In terms of space, annotations can be described in either 2D image space or 3D object space. In 2D space, the view management algorithm treats annotations as 2D labels, which are represented and simulated in 2D screen-space. In 3D space, view management treats annotations as 3D objects that are part of the 3D scene and therefore also part of the object-space of the annotated object.

View management algorithms in AR require knowledge about the annotated scene. We provide this knowledge by registering virtual 3D geometry to the world. This geometry is then used to resolve collisions between labels and the annotated object. All implemented algorithms use the screen-aligned bounding box of the annotated objects as collision geometry. The collision geometry is used to detect overlaps between annotations themselves and also annotations and the object. Note that also for the 3D implementation, the 2D bounding box is used to detect occlusions in screen-space.

4.3.1.1 Continuous Updates

Continuous view management algorithms update the layout in each frame in order to resolve violations of the defined layout constraints.

Screen-space Labeling. Screen-space labeling approaches are the most common view management systems and are used in VR and AR applications [12, 51, 56]. They mainly define labels as 2D geometry that is projected to a 2D position in the image-plane. The layout algorithm updates the 2D position of the label in the image plane. In VR applications, the viewpoint often rotates around an object using an orbit camera manipulator metaphor, which creates stable and smooth viewpoint transitions. Although the layout updates during the viewpoint transitions, the viewpoint is stable after the interaction. Therefore, the annotations can settle into a stable position. In AR applications, the viewpoint typically changes unpredictably. Consequently, layout updates are very frequent, even though the user may perform only small unintentional adjustments to the viewpoint.

Another problem of screen-space layout approaches is inherent to the description of the labels as 2D objects that are assigned to 2D positions in absolute screen coordinates. Each time the user changes the AR viewpoint, the location of the annotated object changes, while the 2D position of the labels stays the same. Therefore, the layout algorithm must update the label positions, causing a delay in the label placement. This results in an effect, where the labels are dragging behind the object during camera movement.

View management that treats labels as 3D objects does not suffer from this problem, because annotations are placed relative to the object. To be able to make a fair comparison between 2D and 3D in our evaluation, we compensate for this drifting in the 2D case. To achieve this, we project the transformation of the current AR camera pose into 2D image space and apply the 2D translation and rotation to the label layout. Hence, despite camera pose changes, the 2D labels stay relative to the object annotations without drifting in image-space.

For this screen-space labeling approach (**L2D_cont**) we use the floating labels approach presented by Hartmann et al. [56]. Like Hartmann et al., we use separate forces to enforce the three layout constraints. One force resolves collisions (R2) by detecting collisions between labels in 2D screen-space and pushing the labels away from each other, using a direction vector that spans the centers of the 2D labels. The same applies to collisions between 2D labels and the projected 2D bounding box of the annotated 3D geometry. To avoid labels moving too far away from the annotated object (R1), another force pulls the labels back towards the point it annotates. Crossing leader lines are resolved (R3) by switching the place of the labels that exhibit crossing leader lines. This is realized by applying a force that is orthogonal to the respective leader line.

Object-space Labeling. For reference, we include a 3D space continuous layout algorithm (**L3D_cont**) in our study that treats the labels as 3D objects and assigns 3D positions that are relative to the object. Due to the continuous layout updates, the 3D algorithm exhibits the same problems of temporal coherence as the 2D version (L2D_cont). However, in contrast to L2D_cont, the 3D labels stay relative to the annotated object, because the camera pose is naturally applied to the 3D labels.

Only very few view management algorithms treat labels as 3D objects placed relative to the object. One such approach is the plane-based hedgehog labeling presented in Section 4.2, which we use for this continuous approach. Similar to L2D_cont, it enforces the layout constraints using forces. However, the hedgehog labeling does not move the labels in 2D screen-space, but on 3D planes that are placed relative to the object and oriented towards the screen. While the original approach freezes the orientation of these planes and the label position after an initial layout is calculated, we update both continuously so that they are always oriented towards the screen. Hence, the algorithm can be regarded as the application of the 2D force-based approach [56] to the 3D case.

4.3.1.2 Discrete Updates

The discrete view management approaches used in the evaluation are based on the behavior of the plane-based hedgehog labeling approach of Section 4.2. Once the initial layout for a viewpoint is calculated, layout updates are stopped. Only after the viewpoint of the annotated object changes beyond a certain threshold, the view management algorithm updates the layout to resolve violations of the layout constraints.

For the discrete view management used in our evaluations, we chose the extreme case of not updating layouts after calculating them for the initial viewpoint. Once the layout is calculated for a viewpoint, we freeze the layout, so that labels keep their location relative to the object. This is a reasonable assumption, because we designed our study in a way so that users investigate an object mainly from one dominant view direction.

Screen-space Labeling. The discrete screen-space labeling system (**L2D_disc**) is based on the continuous implementation (L2D_cont). To freeze the layout, we follow an approach similar to the one used to create temporally coherent compact annotation layouts (see Section 4.1.2).

Once the layout for a viewpoint is calculated, the annotations must keep their position relative to the object. We can achieve this by simply stopping the layout updates. Because

we still apply the camera pose to the 2D labels, as described for L2D_cont, the labels stay relative to the object. However, movements that change the scale of the annotated object cause the layout to degrade. When the user moves away from the object, the distance between the labels and the annotated object will increase; when the user moves closer, the distance will decrease and the labels will occlude the annotated object.

Therefore, the layout algorithm must update the positions of the labels so that they stay at the same relative distance to the object. For this purpose, we freeze the layout only partially by selectively deactivating forces. First, we still resolve overlaps between the annotations and the annotated object (R2). This is important to keep the relative distance of the label from the object, when moving closer to the object. Any annotations that would overlap with the geometry are pushed away from the object. When moving the camera further away from the object, the distance of the labels relative to the object increases. Therefore, we also activate the force that pulls labels towards the object (R1). To avoid that the layout algorithm resolves crossing leader lines (R3), we turn off this force, once the layout is frozen.

We also keep resolving overlaps between the annotations themselves (R2), because the 2D labels otherwise would continuously occlude each other during viewpoint changes. Note that this is not an issue in the 3D plane-based hedgehog labeling approach, where label 3D position and 3D orientation are frozen in the plane. Therefore, labels in the same plane do not occlude each other, while occlusions between labels in different places can easily be resolved with camera movement. 2D labels are always located in the image plane and always oriented towards this image plane, which can quickly cause occlusions.

Because the layout update still continues, the labels will move away from their "frozen" position. To ensure that the labels stay at the same position relative to the object after freezing the layout, we use an additional force, which pushes labels back to their relative position during viewpoint updates. For this purpose, we calculate a line from the center of the overall geometry through the position of the label when it was frozen in place. The force constantly pushes the label towards this line. Note that this force works in combination with the other forces. Therefore, the label does not strictly stay on this line, but is forced to stay close to it.

Using this implementation, we can create a discrete 2D layout that places 2D labels relative to the object and preserves their ordering. Unlike the discrete 3D layout (L3D_disc), the layout algorithm still updates the annotations, which is visible as small label movements during viewpoint changes. However, this is the closest we could come to a feasible

discrete 2D layout.

Object-space Labeling. To achieve a discrete object-space labeling (**L3D_disc**), we use the plane-based hedgehog labeling approach of Section 4.2. Once the layout has been calculated for a viewpoint, the layout algorithm stops updating and freezes the layout of the current viewpoint.

In contrast to the discrete 2D algorithm L2D_disc, the discrete 3D layout truly is frozen and does not require any kind of layout update, because the 3D labels are placed relative to the object. The camera pose is naturally applied to the 3D labels. Eventual occlusions between annotations and the annotated object can be resolved by the user due to the parallax effect of the 3D planes that place the labels.

4.3.2 Evaluation: Update Approach and Spatial Representation

In this evaluation, we investigate the task performance of the four implementations of view management systems. For this purpose, we use the previously described view management systems, which differ in the approach they update layouts (continuous, discrete) and in terms of their algorithmic and spatial description (2D space, 3D space).

Scenario. The experiment simulates an AR learning scenario, in which a user is confronted with an unfamiliar object. We assume that a user will typically first get an overview of the parts from a more distant viewpoint, before going closer and investigating the details of the annotated object. Additional labels provide further information of the annotated parts in the overview. A user can investigate a part by following the leader line to its anchor point, thereby assuming a closer viewpoint of the object.

To avoid that participants are familiar with the object of the study, we use a complex 3D object that has no resemblance with any real world object. It consists of annotated blocks of approximately equal size and uniform color. Therefore, participants do not have salient clues, with which they can associate the location of the anchor points of labels. We assume that, when using such an unfamiliar and complex 3D object, the perceived visual clutter is consistent among the participants and not a result of prior knowledge of the object, or any prior expertise, as described by Rosenholtz et al. [111].

Apparatus. The experimental code was written in C++ using OpenSceneGraph[2]. Marker tracking was performed by an in-house natural feature tracker. The trackable

[2]http://www.openscenegraph.org/

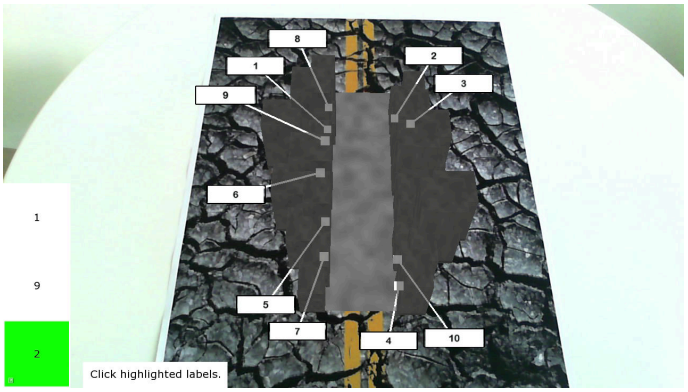
marker was printed on an A3 non-glossy paper (297mm x 420mm), and was placed as a single item on a table, with sufficient room to move around the table (Figure 4.11).

The experimental application was deployed on a Windows 8 Microsoft Surface Pro 2 tablet, which featured an Intel Core i5-4300U CPU, 4GB DDR3-1600 RAM, 10.6 inches 1920x1080 px (208 ppi) screen with 16:9 aspect ratio. We used its internal 1.2 MP, 720p rear-facing camera for tracking. The application received user input from the touch screen of the device.

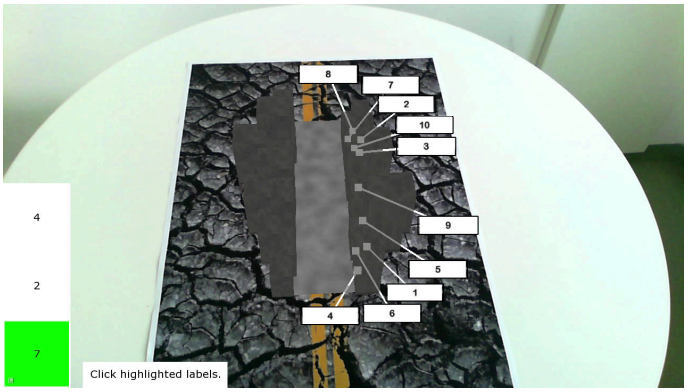
Figure 4.11: The experimental setup consisting of a free-standing table, a marker and the Surface Pro 2 tablet.

Study design. We define three independent variables for this study: the update method (continuous, discrete), the spatial description of labels (2D, 3D) and the distribution of anchor points of the object (balanced, unbalanced). The variables regarding the update method and spatial description directly refer to the previously described implemented view management systems: 2D screen-space with continuous (L2D_cont) and discrete update (L2D_disc); 3D object-space with continuous (L3D_cont) and discrete update (L3D_disc).

We included the distribution of anchor points as variable, because we wanted to investigate its effect on the behavior of labels during on the viewpoint changes. We speculated that multiple anchor points grouped very closely together on the reference object would cause more violations of the layout constraints and, therefore, stronger label movements in continuously updating view management systems. In contrast to such unbalanced layouts (Figure 4.12(b)), a more balanced distribution (Figure 4.12(a)) of anchor points would



(a)



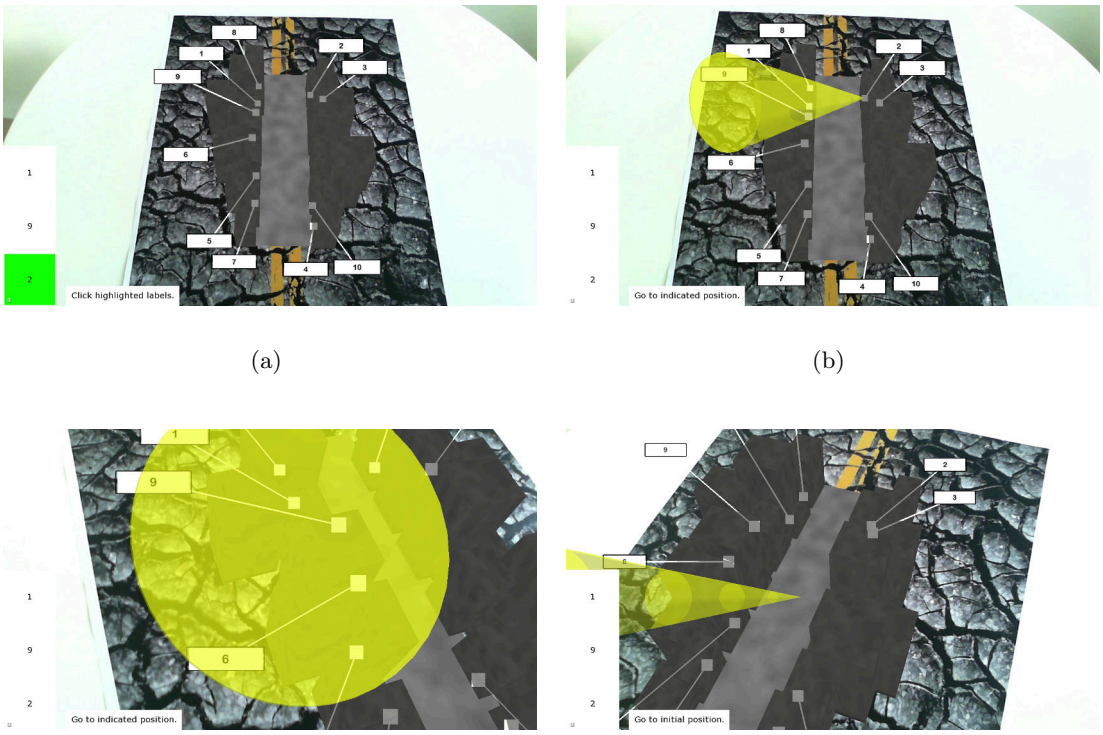
(b)

Figure 4.12: Spatial distribution of anchor points. We speculated that the distribution of anchor points would influence the amount of changes in the continuously updating view management system. We used two different distributions in our study. (a) The distribution of anchor points was balanced over the object. We expected fewer changes in this condition, than in (b) the condition, where the anchor points were clustered on one side of the model.

cause less changes. Note that the discussion of temporal coherence of compact visualizations in Section 4.1.2 follows the same argumentation of distributing anchor points over the object.

The experiment followed a mixed-methods design, using a randomized, repeated-measures design, with two factors being within-subject, and one factor being between-subject. The within-subject factors were update method (continuous, discrete) and the spatial description (2D, 3D), while distribution of anchor points (balanced, unbalanced)

was a between-subjects factor. The within-subject factors corresponded to the four view management systems, and each participant performed three repetitions, which lead to a total of 12 tasks per participant. For each participant, the combination of factors and their order was randomized using a Latin square.



(a) (b)

(c) (d)

Figure 4.13: Experimental Task. (a) The task started in the overview of all labels. Here participants had to select labels in a sequence as indicated by the system. The sequence was indicated by highlighting the corresponding number on the left. In this image, the participant had to select 2 to continue with the next number. (b) After finishing the sequence in the overview, the participant had to click on the labels of the sequence again. Each time the user clicked, a cone would appear, with which the participant (c) had to align the device. (d) After exploring each of the three labels in the closeup view, a cone would appear that indicated participants to go back to the overview, to either continue with the next iteration of the same task, or to finish the task.

As dependent variables, we measured the duration of each task and the duration of the full trial. Furthermore, we measured error rate metrics and layout statistics: the amount of wrongly identified labels, label order changes, leader line crossings, object space and screen space movement of the relevant labels.

Task. A task consists of the following steps (Figure 4.13):

1 The participant must identify three labels of interest in a certain order in the overview by clicking on them, then

2 physically move the viewpoint closer to each anchor point of the corresponding label of step 1.

3 Repeat (1) and (2) three times for each factor-level combination

The purpose of the tasks is to simulate a learning environment, in which a user gets an overview of an object from a viewpoint from which the whole object and its annotations are visible. This is simulated with step (1), in which the participant had to select a randomly generated sequence of three labels. The sequence was shown on the mobile device (Figure 4.13(a)). After identifying and clicking on all the relevant labels, the participants had to physically change the viewpoint of the device and move the viewpoint closer to each anchor identified point (2). Participants performed the second step for each label in the same order as they were presented in the first step. Before moving closer to a label, they had to click on it again to select it. Clicking on a label would force participants to look for the label by moving the device, which would trigger layout changes due to violations of the layout constraints.

After clicking on the label, the system showed a transparent yellow cone, with which the participants had to align the mobile device in order to continue the study (Figure 4.13(b)). This step also enforced movement of the mobile device. The bottom of the cone indicated the position the mobile device should be moved to, the tip of the cone pointed to the anchor point of the identified label. The participant had to align the mobile device with the bottom and look at the tip of the cone (Figure 4.13(c)).

Each cone had an angle of 45° from the ground plane. In addition, cones indicating items on the left side of the model were extruded to the right, while the ones for the right side were extruded towards the left side. Cones were also oriented 45° towards the participants. For instance, the cone in Figure 4.13(b) indicates a location on the right side and, therefore, is extruded to the left.

The cone disappeared when the alignment was correct, which indicated the participant could continue with the task. We introduced a positional and angular tolerance to the alignment, to avoid that participants spent too much time aligning the view. During the trials, we did not experience issues with participants having alignment problems.

After aligning the device with the cone, the task continued with the next label, until the task would force participants to go back to the overview to trigger the next iteration of

the task (Figure 4.13(d)), starting again with step (1). Overall, each participant repeated the task twelve times, three times for each investigated view management system.

Hypotheses. We had two main hypotheses:

- **H1**: Task completion time differs between the view management systems.
- **H2**: Anchor point distribution has an effect on task completion time between view management systems.

Regarding H1, we expected that the properties of the view management system influences the task performance during viewpoint changes. When a user changes the viewpoint, a continuously updating system constantly resolves layout constraints. Therefore label positions and their relative relationship to each other and the annotated object may change. We reasoned that such changes have a negative impact on repeatedly locating labels, as required by the task of this evaluation. On the other hand, in the discretely updating setups, the label layout ideally does not change, which makes it easier for users to keep track of the locations of labels during viewpoint changes and consequently improves task performance.

Note that the implementation of the investigated discrete 2D case (L2D_disc) allows labels to move to avoid excessive occlusions between them. However, the labels will never change their relative order to each other. In addition, similar to the discrete 3D version (L3D_disc), L2D_disc is also prone to layout violations in the form of crossing leader lines. We expected L2D_disc to perform better than the continuous 2D version L2D_cont. Furthermore, we generally expected the discrete update methods L2D_disc and L3D_disc to outperform both continuous update methods L2D_cont and L3D_cont.

Regarding H2, we expected that the distribution of labels around the object influences the view management systems in different ways. For this purpose, we defined balanced and unbalanced distributions of annotated object, which also lead to unbalanced distributions of labels around the object. The unbalanced layout grouped anchor points and therefore labels closely together. We speculate that during viewpoint changes this setup would cause more layout violations and, consequently, more label updates, than a balanced setup, where anchor points and labels are well distributed. Accounting for both balanced and unbalanced layouts, we hypothesized that there would be a performance difference between the four systems, because the relative locations of labels would change to a different degree.

Procedure. Prior to starting the experiment, the participant was asked to fill out an information and consent form along with a demographics questionnaire. We introduced the participant to the experiment and gave a thorough explanation of the purpose of the study and the used apparatus.

Before starting the experiment, the participant performed a set of training tasks with the view management system of the current condition. During this task, the participant was free to ask any questions regarding usage and control of the system. The training task was a simplified version of the real task with only four labels, two of which were part of the selection and identification task. The configuration of labels of the training task was different than the configuration of the actual task, to avoid unintended learning effects. Following the training task, the participant started the experimental task and completed it without interruption. After finishing the task, participants were allowed to take a break, before moving on to the next view management condition, which again started with a training task. Between each view management system and after completing all trials, the participant filled out questionnaires to collect qualitative feedback. The participant was also incited to give additional verbal feedback in an interview after finishing the experiment.

Participants were instructed to pay attention to solving the task to the best of their abilities, and that the amount of time spent on each task was not relevant. As we log metrics for both completion time and error rate metrics, it is generally considered that they are inversely correlated. However, the aim was to not have the user rush through the system to achieve best time, but emphasize a balance between time and quality.

Participants. A total of 24 participants (6f) were part of the experiment, aged 24-36 (M=29.3). They were recruited from both on and off the campus area. All participants self-reported normal or corrected-normal vision. Familiarity with AR was self-reported to be average, and familiarity with handheld mobile devices, above average on a 5-point scale. Data collected from 24 participants $\times$ 12 tasks resulted in 288 tasks in total. The average total completion time of the experiment per user was 24.5 minutes (SD=3.68min).

Results. The analysis was performed using a significance level of $\alpha = 0.05$. The main analysis method was a type III ANOVA and the Friedman test. Pairwise comparisons in post-hoc tests were performed using Tukey's test. The collected task performance time violated normality and homogeneity. Therefore, we transformed the task time data logarithmically.

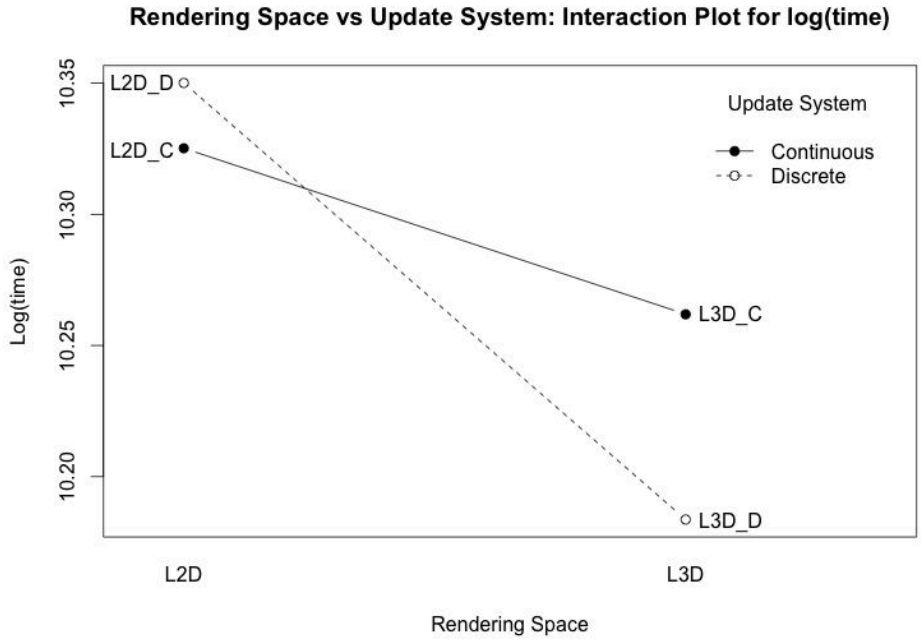


Figure 4.14: Interaction plot of factors rendering space and update method. There is a significant difference between the discrete plane-based approach (L3D_disc) and both 2D approaches. There is a near significant difference between the continuous 3D approach L3D_cont and the discrete 2D approach L2D_disc. Although there are no other significant differences, the discrete 3D approach L3D_disc appears to have a better task performance than the continuously updating system L3D_cont. The data indicates that the 3D approaches generally perform better than the 2D approaches.

A type III ANOVA found a significant main effect on spatial description (2D, 3D) ($F_{1,22} = 15.79, p = 0.00064, \eta_G^2 = 0.084$). This means that participants showed significantly slower completion time in the 2D screen space rendering condition. No other main effects or interactions were found in the analysis of $log(time)$. However, the two-way interaction between spatial description (2D, 3D) and update method (continuous, discrete) was near-significant at ($F_{1,22} = 3.90, p = 0.061, \eta_G^2 = 0.018$). This near-significant interaction between rendering space and update method is illustrated in Figure 4.14.

A follow-up Tukey test on the within-subjects factors revealed that the 3D discrete system (L3D_disc) significantly differs from both 2D systems (L2D_cont, L2D_disc) (both $p < 0.001$). Furthermore, L3D_cont showed near-significant difference from the discrete 2D system (L2D_cont) ($p = 0.085$).

4.3.3 Evaluation: 3D Continuous and Discrete

We performed a followup study to collect more qualitative feedback on selected view management systems. Although participants of the previous study already filled out a questionnaire to collect qualitative feedback, the data did not yield any reliable results regarding the preferences of systems. Based on the feedback gathered from participants, we believe that participants could not distinguish between the four systems after completing the experiment.

To avoid users mixing up the different systems, we focused on only two view management systems. The first study identified L3D_disc as the one achieving best task performance. Therefore, we removed the 2D condition and compared L3D_cont and L3D_disc in this study. Furthermore, the data from the first experiment indicated that unbalanced layouts cause stronger layout changes than balanced layouts. Therefore, we removed the independent variable regarding the distribution of anchor points of the object by focusing only on the unbalanced scenario.

The apparatus, task and procedure were identical to the first experiment.

Study design. The study had a randomized, within-subjects design with one independent variable: update method (continuous, discrete). In this study, we only used 3D view management systems. Therefore, the update method directly refers to the continuous 3D system (L3D_cont) and the discrete 3D system (L3D_disc). Participants performed the same task as in the first study. For the two conditions, this produces a total of six tasks per participant.

We collected participant satisfaction data using a 5-Likert scale questionnaire, which queried the participant's satisfaction with the system, immediately after a condition was finished. Overall preference for a system was queried after the tasks for both systems had been finished. In this questionnaire, we forced each participant to choose either one or the other. Due to the random ordering of the first and second task, any imbalances due to order effect should be avoided.

Participants. A total of 10 participants (all males) were part of in the second experiment. All participants were recruited from the same pool of participants as in the first experiment. No participant took part in both experiments. All self-reported normal or corrected-normal vision.

Results. The analysis used a significance level of $\alpha = 0.05$. The main analysis method for the user satisfaction data was a Wilcoxon signed-rank test. Analyzing participant preference scores was performed using Exact Binomial test method.

We found a significant effect when analysing the difference in the responses of the 5-Likert scale question of participant preference. The mean ranks of discrete and continuous were 14 and 7, respectively: W = 3.5, Z = -2.21, $p = 0.02734, r = -0.49$. This is a strong indication that participants were overall more satisfied with the discrete update system, despite there being no significant differences in task performance.

For participant preference, participants were asked to make a binary choice of preference for update system, choosing between either the continuous or discrete update system. One participant did not have any preference and would not prefer a single system. The outcome was eight participants preferring discrete (80% of participants) and one preferring continuous (10% of participants). This indicates a strong preference towards the discrete system. However, due to the small sample size, the Exact Binomial method did not yield a significant difference. If undecided is counted as not preferring discrete updating, the resulting proportion 0.8 of preference for discrete updating is near-significantly higher than expected $0.5, p = 0.055$ (1-sided).

4.3.4 Discussion

The first study clearly showed that the view management system, which treats labels as 3D objects and creates a static layout (L3D_disc), significantly outperforms the 2D continuous view management system (L2D_cont). This is in line with our expectation, because the continuously updating layout seems to make it difficult for users to keep track of the labels. However, L3D_disc also outperforms its 2D counterpart L2D_disc, which avoids strong label motions by preserving the order of labels. This difference can be explained by the fact that due to the limitations of the 2D description of labels, the implementation of L2D_disc does not freeze the discrete 2D layout as L3D_disc does. L2D_disc is constantly updating the positions of labels to avoid occlusions. Furthermore, similar to L2D_cont, the labels are always oriented towards the screen, thereby lacking the static 3D representation of L3D_disc. By inspecting the performance data (Figure 4.14), we can also see that both L2D_disc and L2D_cont exhibit very similar performance. This indicates that even though L2D_disc enforces a certain label order, the small motions of the simulation running in the background and the lack of a static 3D representation has a negative impact on the ability of users to locate and interact with labels.

The difference between the 2D systems and L3D_disc could also be explained by the way the systems are implemented. Despite both implementing similar force-based approaches, the spatial representation of labels clearly influences the implementation of the systems and thus can also have an impact on the label behavior. To isolate this factor, we also included a continuously updating 3D layout (L3D_cont) in our study. Indeed, the near-significant difference between L2D_disc and L3D_cont hints at implementation specific differences. The better performance of L3D_cont supports our argument that the 3D implementations view management systems should be preferred.

Although there was no significant difference between L3D_disc and L3D_cont, a visual inspection of Figure 4.14 shows a difference in performance. In addition, the follow-up study revealed that participants preferred a discrete 3D layout (L3D_disc) for the given task. Therefore, we can recommend frozen layouts 3D as the most suitable view management system. A follow-up study should investigate the performance aspect with a larger sample size.

Overall, we accept hypothesis H1. A combination of update method (continuous, discrete) and spatial description (3D, 2D) has an influence on task performance. The static layout of the discrete 3D view management system significantly outperforms the 2D versions. A visual inspection of the performance data also shows better performance, when compared to the continuous 3D version, which should be part of further investigations.

Regarding the second hypothesis H2, we did not find any significant difference in task performance between balanced and unbalanced. Therefore, we reject hypothesis H2. Nevertheless, in the collected data itself, we noticed a larger number of label changes when the label layout was unbalanced. To reasonably compare this data across the systems, we limit ourselves to investigating the data in which users were in the overview mode, i.e., the part of the task, where all labels visible on the screen and in focus, and the participant must identify three labels correctly. Pooling all label order changes of the unbalanced condition yields 1151 changes, the same for the balanced condition 457 changes. This leads to approximately five changes per overview in the unbalanced condition and two changes per overview in the balanced condition (20 participants $\times$ 4 systems $\times$ 3 tasks $=$ 240 overviews). Even though the label order changes had no negative impact on performance, it supports our assumption that layouts are prone to changes when anchor points are not distributed well over the object and cluster in one region. Nevertheless, it appears that label motion and the relative position of labels in 3D have a larger impact on performance than the amount of label changes.

4.4 Conclusion and Future Work

In this chapter, we discussed approaches to integrate temporal coherence into the optimization process of compact visualizations. Furthermore, we presented a novel 3D view management technique that is more suitable to create layouts of external labels in AR than similar 2D approaches. 2D view management techniques place annotations in image space. While they allow to create high quality label layouts for still images, they fail when the viewpoint changes regularly.

With hedgehog labeling, we propose to place external labels in 3D object space and use geometric constraints to control their motion. Our approach fulfills the desired objectives of layout algorithms (e.g., avoiding overlapping labels) and also behaves consistently over time during viewpoint changes, thus improving temporal coherence. We use two geometric constraints: a "hedgehog" constraint, where labels originate from a common point and move along a 3D pole stuck into the annotated object, and a plane constraint, where annotations move in a plane that is either parallel to the viewing plane or user-defined in world space.

While the "hedgehog" constraint restricts the movement of labels along one line, the plane-based approach allows labels to move freely along the plane. To avoid constantly changing annotations during camera movement, our view management approach optimizes the layout for the current viewpoint of the user and then freezes the layout to avoid changes in adjacent viewpoint. We use a similar approach to achieve temporal coherence for compact visualizations. More specifically, we optimize the layout of annotations for the viewpoint of the user and freeze the layout. Only after the viewpoint changes and the movement stops, we update to the new optimal layout. Note that for the compact annotations examples, we still use a 2D view management approach [56]. However, we can easily exchange this approach with the 3D view management approaches presented in this work.

Freezing the layout once it is optimized stabilizes the layout during camera motion. However, when reaching the new viewpoint, the switch to the newly optimized layout can completely rearrange the layout, thus causing again distracting changes. To reduce the amount of changes, the layouts could be updated only partially. Hence, instead of allowing the algorithm to rearrange all labels, only a subset of labels is rearranged. This subset can consist, for instance, of those labels that violate certain layout criteria strongest, e.g., by having the longest leader lines or causing the most leader line crossings. We already use a similar method for switching between layouts of compact explosion diagrams. For compact

explosion diagrams, we reduce the amount of changes between adjacent viewpoints by enforcing that the new layout must be similar to the previous layout. This method can be expanded and combined with the approach to freeze the layout between viewpoint changes.

Until now, researchers have based the aesthetic criteria for label layouts mainly on experience and observation of illustrated examples [57], but, to the best of our knowledge, the impact of these criteria on the perception of the visualization has not yet been formally evaluated. Based on the assumption that violating aesthetic criteria to a certain degree may not have a negative impact on the perception of the visualization, we performed a user study, in which we compared our plane-based hedgehog labeling approach to view management systems that continuously update labels. In our study, our approach clearly outperformed systems working with 2D representations, even though it froze the 3D layout of annotations relative to the object, thereby violating the given layout constraints. Based on this result, we are confident in our recommendation to use 3D layouts that do not have to be continuously updated. Furthermore, labels should be treated as 3D objects and part of the scene. Integrating 3D labels into the scene allows the AR system to naturally apply the camera pose to the labels. It also has other advantages with respect to 3D interaction methods. For instance, a method for manipulating a 3D object in AR can be directly applied to manipulating a label. For instance, simple manipulations such as picking and transformations (scale, translation, rotation) can be used without modifying the used interface. In addition, transitional interfaces, which switch from an AR view to a virtual view of the scene can treat annotations as scene objects and, therefore, do not have to treat annotations different than any other object.

An additional advantage of a static 3D layout is that it can be calculated by optimizing the overall layout for a single frame. After the initial computation, no additional computational resources are required, because the layout is not continuously updated. This is beneficial for the battery life of mobile devices.

In the following, we discuss additional open issues regarding the 3D view management as future work. Assuming that labels in different planes can be clearly distinguished using parallax effects, we can further use plane-based label placement to reduce the amount of label fighting by resolving occlusions between labels only if they are grouped into the same plane. To reduce clutter from conflicting labels in the other planes, it may be sensible to allow for transparent labels.

We can also improve the definition of the plane constraints. While the current imple-

mentation of our plane-based approach requires the user to set the amount of planes at run-time, we plan to develop more advanced plane fitting techniques that take into account the extents of the object and the current viewpoint of the user. Hence, an elongated object is split by more planes than a shorter object.

To further enhance plane-based label layouts, we can use other planes than those which are parallel to the current image plane. This is particularly useful, if there are dominant surfaces of an object, such as a building facade. In such a case, we may want to use labels that are not automatically rotated towards the viewer, but rather remain in the plane of the main surface, and are also displaced in this plane to avoid occlusions.

For concave objects, our center-based hedgehog approach may generate label poles which penetrate unrelated parts of the object between the anchor point and the label's annotation. To handle such situations, we can decompose the object into a set of convex shapes, for which we apply our approach separately. However, this may cause heavy fighting between labels. Such situations can also be handled by rendering labels after scene objects with disabled depth test. In this case, a penetrating pole always appears in front of the object.

Chapter 5

Extending the Ego-centric Viewpoint

Contents

Mobile devices such as smartphones allow users to access location-based information anywhere and at anytime. For instance, tourists can query information about surrounding points of interest in a foreign city, a task which can also be supported by mobile tourist guides [117]. The information is commonly presented using a spatial representation, such as 2D or 3D maps. 3D maps even allow exploring real world objects freely, since they are not bound to the egocentric viewpoint of the user. However, mobile map solutions are not optimally designed for urban exploration [9] and provide limited capabilities to access the data and to relate it to the real world. For instance, users of 3D maps often try to align the virtual viewpoint of the map with their egocentric viewpoint for easier orientation, a strategy that is not well supported by the interface [102]. The only alignment feature 3D maps offer is to align the exocentric top-down view with the general viewing direction of the user. Another issue of currently available 3D maps is that the camera view of the object is often occluded by nearby structures, which is especially problematic in densely built-up areas (Figure 5.1).

AR is a natural choice for exploring location-based information of real world objects,

Figure 5.1: A 3D map (here: Google Earth) allows users to explore surrounding real world objects. However, the user first has to identify the corresponding virtual object in the map and then relate it to the current position. Furthermore, in densely built-up areas, neighboring buildings will cause occlusions of the virtual viewpoint during exploration.

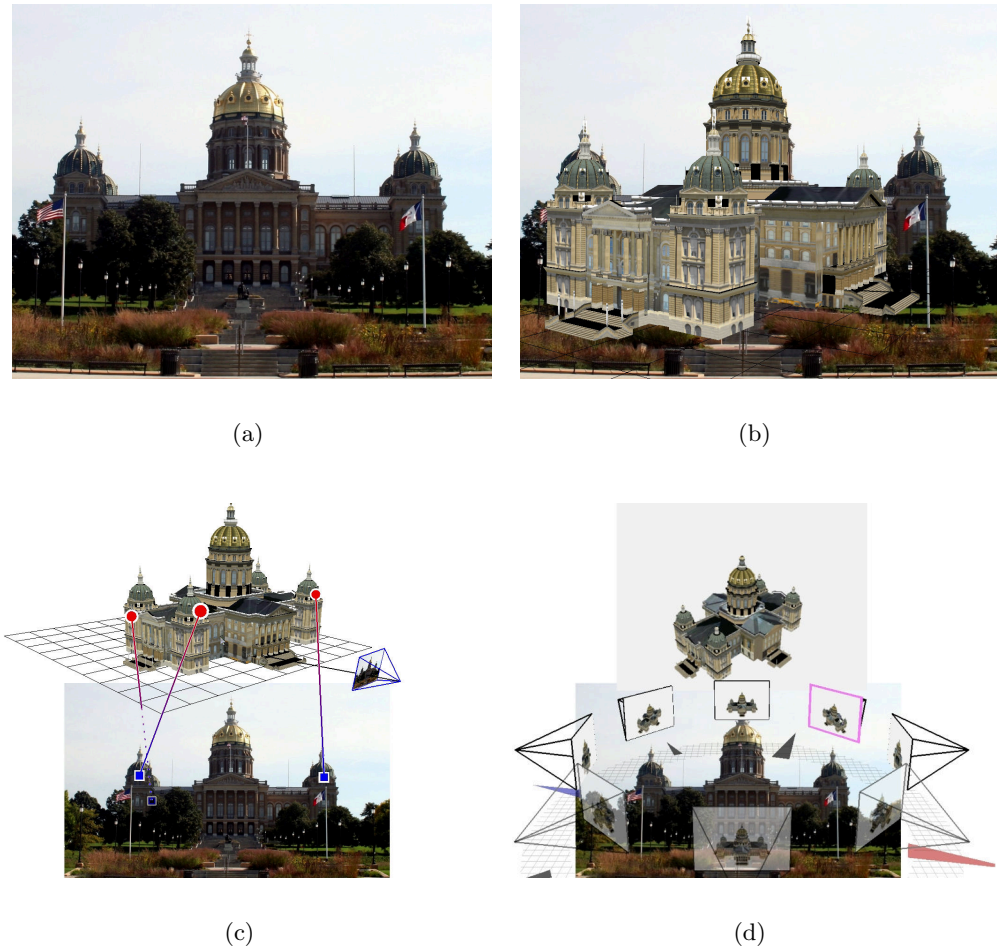
because AR overlays information directly into the user's surroundings. For instance, a user can easily access additional information about a building in an urban environment by pointing an AR-enabled mobile phone into its direction. However, in contrast to a map interface, users are limited to the inherent egocentric reference frame of an AR interface, which becomes an obstacle, once the user wants to explore objects that are out of reach. The user would need to physically move to a new position, which might be too cumbersome or even impossible.

To deal with these limitations, we introduce Object-Centric Exploration (OCE) techniques for handheld AR, which use a virtual copy metaphor [105] to gain access to distant viewpoints of a real world object in the user's AR view (see Section 5.1). The OCE techniques use a simple seamless transition that zooms the virtual copy for further exploration. However, such a transition can also assume specific viewpoints that also match the current task of the user. In Section 5.2, we present a system that automatically assumes target viewpoints that better match the scene geometry.

Aside from transitional interfaces, we also explored the usage of multi-perspective renderings to extend the ego-centric viewpoint of the user in Section 5.3.

5.1 Object-centric Exploration Techniques

We explore different designs of OCE interfaces for the exploration of buildings in an urban setting. In contrast to 3D maps, OCE techniques allow users to focus on a single object they are interested in. OCE techniques also do not suffer from occlusions from



(a) (b)

(c) (d)

Figure 5.2: (a) How can I explore the Iowa State Capitol without physically moving? We present Augmented Reality interfaces using a virtual copy metaphor to access additional views, e.g., (b) uses an in-place, (c) a separated 3D copy with visual links between virtual and real world object. (d) We also present a spatially separated interface, which uses a 2D copy of the real world object. The available viewpoints are arranged as a circle around the real world object. The current viewpoint is highlighted.

neighboring structures, because a virtual copy of only a single object is presented. To present additional viewpoints of this real world object, our OCE interfaces separate the virtual copy (focus) from its real world counterpart and from its surroundings provided by the AR video (context). We consider spatial and temporal techniques for combining focus and context [27]. While the former separate focus and context in space, the latter do so over time, thus removing the context from the interface. Figure 5.2 shows spatial OCE techniques that preserve the context by either overlaying the copy on the context

(Figure 5.2(b)) or separating the copy from the context (Figure 5.2(c)).

We perform a series of studies to evaluate our initial designs and the ability of the user to relate the virtual information to the real world. We perform studies under controlled conditions and collect real world experiences with our interfaces in a real world pilot study. Based on the results from the real world pilot study, we evaluate the performance of our designs and compare them to a more common 3D map interface. We summarize our findings in design recommendations that should be considered when developing OCE interfaces for potential application areas such as future generations of location-based AR browsers, 3D tourist guides, or situated urban planning. Relevant real world objects could be annotated with additional information that can easily be explored using OCE interfaces.

5.1.1 Design Space

Our goal is to create OCE techniques for mobile AR that enable users to remain at a physical location and explore a real world object from arbitrary viewpoints taken on its virtual copy. For instance, the 3D model of the building in Figure 5.2(b) is virtual copy of the physical one.

The design aspects of OCE techniques derive from different steps involved in the exploration of real world objects through a virtual copy in AR. OCE techniques must enable users to identify a selectable real world object and provide means to select and interactively explore this object using appropriate input controls and feedback on a display device. OCE techniques also require spatial cues to facilitate mental linking between the virtual copy and the real world. Figure 5.3 outlines the aspects discussed in this section.

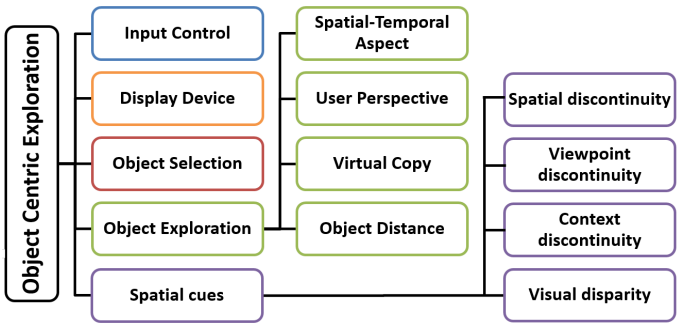


Figure 5.3: A hierarchical diagram of the discussed design space for Object-centric Exploration techniques in AR.

Input Control. The input control is defined by the input type, the mapping of the input data to programmatic functions, and the feedback to the user on how to perform an action. The input type for mobile AR may be one or more of the following: single and multi-touch, locomotion, sensors and speech input.

Display Device. The presentation medium impacts the overall presentation of OCE interfaces. An interface designed for a handheld device, such as a mobile phone, might not work for a HMD. For instance, when using video see-through HMDs, manipulations of the video background also influence the real world view of the user. A user with a handheld device still has an unmodified view on the world by looking past the device.

For the moment, we will only consider handheld devices as target medium, because these devices are widely available and a major platform for AR applications. The design of an OCE interface is mainly influenced by the available screen-space and the orientation of the device (landscape, portrait).

Object Selection. Object selection connects the user's intent with a specific virtual or real entity for subsequent tasks. In AR, not all of the objects of a scene can be selected and interacted with, unless a full, dense and semantically connected 3D reconstruction of the scene is available. Because such a reconstruction is hardly ever available, users require guidance to recognize interactive objects in the AR scene. Hence, this aspect requires selection guidance to highlight [136] which objects are interactive.

Object Exploration. To categorize the design of the exploration technique, we identified four aspects that consider the relationship between the real-world object and its virtual copy: the type of separation, the user's perspective, the properties of the virtual copy and the size of the object on the screen.

Spatial-Temporal Aspect. Additional viewpoints of an object can be spatially or temporally separated from the original AR viewpoint. Spatial techniques preserve the egocentric viewpoint of the user in the video image, and show additional viewpoints of the object. Temporal techniques also provide additional viewpoints, but do not preserve the original viewpoint of the object.

User Perspective. In outdoor mobile AR, users explore the world from an egocentric perspective. OCE techniques can provide the user with exocentric viewpoints of an object. In indoor environments such as tabletop setups [16], the user already has an exocentric view of the object. In this case, OCE techniques can complement an egocentric perspective.

Virtual Copy. The virtual copy of the real object depends on the available data. The object can be represented using different media (3D model or 2D picture) and can be placed either in a 2D image space (e.g., map) or within a 3D coordinate system.

Object Distance. The projected size of the real object depends on the screen size and its position relative to the user. The object may either be too large (too close) or too small (too distant) to effectively combine virtual and real views with a spatial technique. In this case, a temporal technique can be employed, which replaces the real object with one taken from a more appropriate viewpoint.

Spatial Cues. Spatially or temporally separating the virtual copy from the real world object creates discontinuities between the virtual and the real world. Users have to be able to link both worlds in order to transfer the spatial knowledge gained in one representation to the other representation. Spatial cues facilitate this linking to overcome the following discontinuities.

Spatial discontinuity. Moving the virtual copy out of its original position in the context creates a spatial discontinuity.

Viewpoint discontinuity. Changing the viewpoint of the virtual copy causes a misalignment of this viewpoint with respect to the real world viewpoint.

Context discontinuity. A context discontinuity occurs when the video image of the real world is modified. In extreme cases, an interface zooms in on the object and removes its context completely.

Visual disparity. The degree of visual disparity depends on the quality of the virtual copy. There is no disparity, when the virtual copy perfectly matches the real object. Note that not only the virtual copy can be adapted to become more similar to the real world, but also the representation of the real world can be changed. In this case, the context discontinuity may increase, but at the benefit of decreasing the object discontinuity. Hence, spatial cues can resolve certain discontinuities, but aggravate others.

Spatial cues that address all of these discontinuities are transitions between real and virtual spaces [16, 50]. Hence, we use transitions as a standard cue, when switching between the copy and the real world. For instance, when switching to the copy, the virtual copy is gradually faded in (addresses visual disparity) and seamlessly rotated to a bird's eye view (addresses viewpoint discontinuity). At the same time, the virtual copy and the context are rearranged on the screen using an animated transition (addresses spatial and context discontinuity).

5.1.2 Interface Design

In this work, we explore a limited subset of the design space. Our focus is the design of spatial-temporal representations of the interface and evaluating these with respect to the spatial awareness of the user. Therefore, we explore the aspects of *object exploration* and *spatial cues* in detail.

In our designs, we only consider handheld devices as *display device*, because these devices are widely available and a major platform for AR applications. We assume that the mobile device has a large screen, to be able to experiment with screen-space demanding designs. Our interfaces are designed for landscape mode, which is the default mode of currently available AR browsers. Furthermore, we only use a common single-touch interface for *input control*, and we *highlight* selectable objects with a simple frame.

The *user perspective* is defined by our application case, where we focus on large-scale outdoor exploration. Hence, in accordance with the design space, the user perspective is always egocentric and extended by exocentric viewpoints. For the *object distance*, we assume the ideal case where the real world object is presented at a sufficient scale so that all of the features relevant to our studies are clearly visible. For the *virtual copy*, we assume that we have access to a 3D model of the object. In the following, we refer to the initial view containing only the real world object as AR mode, and to the mode containing the copy of the object as VR mode.

Spatial separation techniques seem to be the most relevant choice for exploring large objects in an outdoor setting, because they preserve the real world context. We expect that spatial separation techniques create an artificial bridge for mapping content in the virtual copy to the real world. To investigate this aspect, we developed a 3D interface and a 2D interface with spatial separation between focus and context.

In the **3D separation interface** (3DSEP) (Figure 5.5(b)), a 3D copy is presented, which allows for the continuous exploration of different viewpoints of the object. The user interacts directly with the 3D copy through a virtual orbit metaphor. When entering the VR mode, the copy is viewed from a bird's eye perspective. We integrated common spatial cues into the interface, to allow users to mentally link the viewpoint of the copy to the original viewpoint of the context. A grid shows the ground plane of the copy and a camera icon, located in the coordinate frame of the copy, indicates the original egocentric viewpoint relative to the object. A radar icon in the top right shows the same information in a more abstract visualization and from a top-down view. The copy is in the center of the radar, while a dot rotating around the center indicates the camera position relative to

the object.

The **2D separation interface** (2DSEP) (Figure 5.5(a)) uses images as copy. These images could be pictures taken from the real world object. To avoid visual disparity of the focus between both interfaces, we render them from the same 3D model used in 3DSEP, taken at equidistant positions (45°) on a horizontal circle around the object, with the camera pointing towards its center. The viewpoints are elevated to bird's eye views. The user can replace the zoomed image at the top of the interface by using an explicit one-finger tap, or by swiping over the set of images. The ground plane is rotated upwards around the x-axis, so that the images do not occlude each other or the object. In contrast to 3DSEP, 2DSEP does not provide continuous viewpoint updates.

We included corresponding spatial cues from 3DSEP in 2DSEP. We did not include the cues in the rendered images, but only applied them to the image circle, so that we could investigate if the circle is sufficient for users to orient in the interface. We added a grid to visualize the ground plane on which the images are placed and removed its center, to avoid occlusions of the real object in the video image. Each image in the circle received a camera icon representation. A radar-like cue is achieved by the relation of the currently selected highlighted image to the image showing the frontal view.

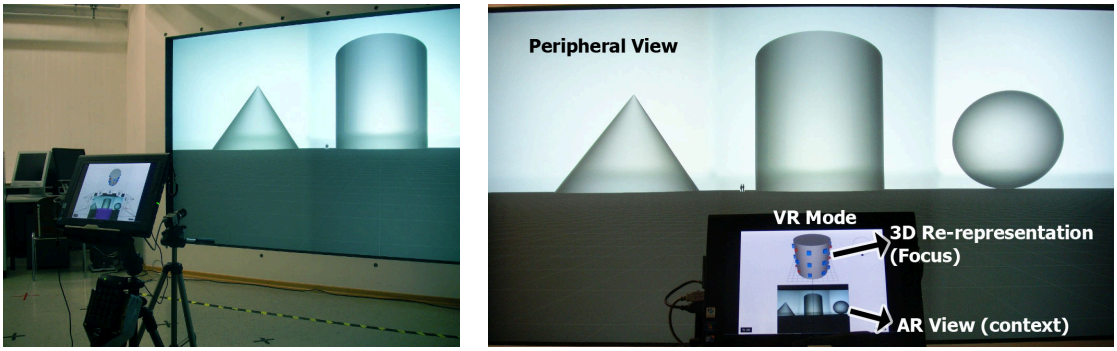
Aside from these spatial cues, both interfaces provide a smooth transition between AR mode and VR mode to connect these spaces [16]. When entering the VR mode, the video image is scaled down and moved to the bottom of the screen, while the copy is moved to the top of the screen. Spatial separation fully preserves the context at the cost of introducing a spatial offset between focus and context. Spatial discontinuity is alleviated by seamlessly animating the transition of focus and context. Visual disparity is addressed by gradually fading the copy in and out.

To address both viewpoint and spatial discontinuity,, we added a switchable spatial cue called visual links (as shown in Figure 5.2(c)) to 3DSEP, thus creating interface 3DSEP+L(inks). Links provide a visual connection between the copy and the real world object. By tapping on a location on the 3D copy, a user can create a 2D line to the corresponding location in the video image. The line style is adapted to communicate occlusion with the focus object, and color coded to communicate the end points.

5.1.3 Evaluation: Abstract Scenarios

We explored the usage and usability of OCE techniques in a series of user studies. We focused on how users interact with our techniques independently of the semantics and

salient content of the real world. Therefore, we evaluated the interfaces using abstract scenes with basic geometric shapes.



(a) (b)

Figure 5.4: Indoor apparatus. (a) The apparatus used during the laboratory studies. We placed a tablet PC in front of a back projection wall. Users were seated in front of the tablet PC looking at the wall. (b) The view of an abstract scene from the participant's position showing the peripheral view in the background an the used tablet PC in the foreground. The tablet PC shows the VR mode of the 3DSEP interface.

5.1.3.1 Evaluation Testbed

To avoid confounding factors from the real world, we used a simulation testbed for AR. A testbed allows us to present artificial environments and structures with which the participants are not familiar. These scenes can represent real world environments, or can be purely abstract. Testbeds for simulating AR scenarios have already been used to control the registration error [80] or variable lighting conditions [81]. Testbeds were also used to overcome technical limitations of currently available hardware [11].

In our scenarios, a user has already found a real world object of interest and is looking towards it. We assume that the user remains stationary, while exploring the object with our interfaces, and thus does not require an immersive 360° view of the environment. Therefore, we simulate the peripheral view of the world with a back-projection wall (4×2m, 4000×2000 Pixel) used in daylight conditions (Figure 5.4(a)).

We seated participants in front of the wall and mounted the AR device on a tripod in front of them, to simulate holding a handheld device, while at the same time removing the associated physical fatigue. The AR device, a tablet PC (Motion Computing J3500, 12.1″), showed a static snapshot of the environment (1066×800 Pixels) that simulated the

view through a video camera. Figure 5.4(b) shows the view of the participant when seated in front of the wall.

5.1.3.2 Experimental Design

The following studies are within-subject and share the same experimental design and apparatus (Section 5.1.3.1). They differ only in their interface conditions.

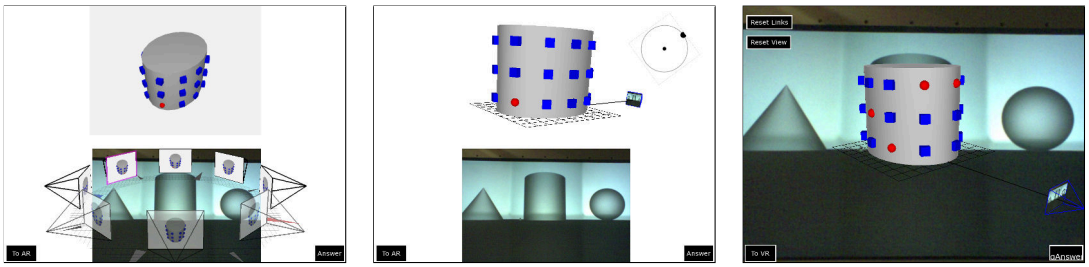
Scenario. We rendered a virtual scene consisting of only basic geometric shapes (cone, elliptic cylinder, sphere). The scale and position of these were chosen to resemble real buildings (e.g., elliptic cylinder, $35m$ in height, half-axes length x=$17m$ and z=$22m$). The peripheral view was rendered using a virtual camera (60° FOV), placed $120m$ from the scene at eye level of the participants. A human scale icon was used as a reference. The AR view was taken with a camera (60° FOV) mounted on the tablet PC.

Tasks. The tasks are representative of interaction with real world 3D objects: (T1) a counting task, where users navigate the copy to find particular figures and count them; (T2) a mapping task, where users search the copy for a single object and point to its location in the peripheral view. For both tasks, the scene included distractors (blue cubes) and targets (red spheres), which were placed on the cylinder in a regular pattern (10 angles, 3 elevations). For T1, five to seven spheres were randomly distributed around the object. For T2, only one sphere was placed at a random location on the grid.

Procedure. For each task and interface, the participants had one practice trial without time constraints. T1 trials were completed by entering the number of counted spheres on an auxiliary keypad, T2 trials by point-and-click to the location of the sphere in the peripheral view with a laser pointer. Participants completed questionnaires regarding different aspects of the interface such as ease of use or intuitiveness between each interface and task, and after the experiment. We recorded task completion time for both T1 and T2, counting error for T1, and a pointing error for T2. The latter was estimated using a vision-based method, which provided the Euclidian distance for images with resolution 640×480.

5.1.3.3 First Study: Varying Copy and Cues

This explorative study compared our first interface designs (3DSEP, 2DSEP, 3DSEP+L) to evaluate 2D and 3D copy representations and the spatial cues. Figure 5.5(a) shows 2DSEP and (b) 3DSEP as used in this study. 3DSEP+L is the same as 3DSEP with the option to create visual links.



(a) (b) (c)

Figure 5.5: Interface designs for studies in abstract conditions. Spatially separated interfaces using (a) images (2DSEP) and (b) a 3D copy (3DSEP), as used in the first study. (c) The 3D in-place interface (3DINP+L), as used in the second study.

Table 5.1: Mean completion times in seconds and point errors in pixels, both with SD, for the first and second study.

		T1	T2	
	Interfaces	Time	Time	Error
Study 1	2DSEP	22.8 (7.3)	18.3 (7.3)	41.5 (28.0)
	3DSEP	15.1 (4.5)	16.7 (7.6)	48.1 (26.3)
	3DSEP+L	16.9 (5.6)	20.5 (9.6)	39.4 (21.2)
Study2	3DSEP+L	19.7 (8.1)	25.5 (12.9)	22.5 (9.9)
	3DINP+L	20.8 (7.7)	25.7 (12.9)	23.4 (7.8)

Participants. A total of 24 people (12m/12f), 16−35 years old ($mean$=25.9, sd=4.2), performed five repetitions (720 trials) for each task and interface. The presentation order of interfaces and tasks was counterbalanced.

Results. For each interface×task condition and participant, we calculated the mean completion time and error from the five repetitions (Table 5.1). We performed non-parametric tests, because our sample violated normality. A significant effect of interface on time was only observed for T1 (Friedman, $X^2(2)$=22.3, p<0.001). A post-hoc Wilcoxon signed-rank test with Bonferroni corrected α=0.0176 showed that 3DSEP (p<0.001) and 3DSEP+L (p<0.001) were significantly faster than 2DSEP. Otherwise, the performance data revealed no significant effects.

Discussion. Both 3D interfaces outperformed the 2D interface in the counting task T1. However, the questionnaire data revealed only a significant preference of 3DSEP over 2DSEP for T1 (Table 5.2). This is reasonable, given that participants did not require the visual links of 3DSEP+L to solve this task. For the mapping task T2, both 3D interfaces were preferred over the 2D interface. The interview revealed that participants had difficulties with orientation using the discrete image switches in 2DSEP. Participants

Table 5.2: Significant effects regarding interface preference (5-point Likert scale). Study 1 was tested with Friedman (not reported) and Wilcoxon signed-rank tests with Bonferroni corrected α=0.0167; Study 2 with Wilcoxon signed-rank tests with α=0.05.

	Study 1			Study 2
Task	2DSEP& 3DSEP	2DSEP& 3DSEP+L	3DSEP& 3DSEP+L	3DINP+L& 3DSEP+L
T1	**p=0.001**	p=0.067	p=0.035	p=0.713
T2	**p=0.006**	**p=0.008**	p=0.512	**p=0.029**

found this especially challenging in T1, where they had problems keeping track of multiple neighboring spheres.

In general, visual links were well received and found to be intuitive. Nevertheless, only 58% of the participants used the links, because, according to their feedback, the task could easily be solved without them. We did not find any significant difference in point error between 3DSEP and 3DSEP+L. However, when dividing the trials of 3DSEP+L and 3DSEP into those with (n=68, $mean$=31.2, sd=14.4) and those without (n=172, $mean$=48.8, sd=40.58) visual link usage, the results indicate that participants made less errors, when they used links.

The interviews showed that the camera icon was a strong cue for communicating the starting point of rotation. Based on the interviews, we believe that participants were unsure when rating the radar cue. The arrangement of images in 2DSEP was well perceived. Participants also stated that it provided a good overview of the object in T2, because the single red sphere was very salient.

5.1.3.4 Second Study: Varying Spatial Separation

Since 3DSEP and 3DSEP+L were the preferred interfaces, and both performed better during the exploration task (T1), we focused on investigating 3D interfaces further. We kept 3DSEP+L as representative 3D mode, because the visual links showed value as spatial cue in the mapping task (T2). Based on our observations, we introduced a reset button, which automatically realigns copy and context viewpoint. We also removed the radar cue from the interface. Aside from these changes, 3DSEP+L corresponded to the interface used in the first study (Figure 5.5(b)).

In this study, we explored two variations of spatial separation. We created an in-place interface (3DINP+L) that is similar to 3DSEP+L, but which has the copy overlaying the real-world object (Figure 5.5(c)). We included the visual links in 3DINP+L, even though their end points are occluded by the 3D copy. Our assumption was that participants would need to switch between AR and VR modes to remove the occlusion and to mentally

connect focus and context.

Participants. Twelve participants (6m/6f), aged between 19 and 30 ($mean$=24, 7, sd=3.3), performed five repetitions (240 trials) of each task and interface. The presentation order of interfaces and tasks was counterbalanced.

Results. In the analysis, we used the same methods and statistical tests as in the previous study. Time and error measurements are summarized in Table 5.1. Statistical analysis did not reveal any significant effects.

Discussion. For T2, participants significantly preferred 3DINP+L over 3DSEP+L (Table 5.2). In the interview, participants stated that they preferred 3DINP+L because it was a more natural and intuitive approach to not separate focus from context. They also mentioned the increased size of the object in 3DINP+L. The lack of significance of 3DINP+L for T1 may be explained by the comments of participants, who stated that they only focused on the 3D copy and did not consider the video background for this task. Therefore, the placement of the virtual copy was irrelevant.

Interestingly, the visual links still served as orientation cue in 3DINP+L, even though they penetrated the copy and the endpoints were occluded. Participants noted that visual links showed the misalignment between the copy and the context. As before, trials in which links were used showed smaller point errors (n=97, $mean$=19.6, sd=10.6) than trials without visual links (n=23, $mean$=36.9, sd=26.4), which underlines their value as spatial cue.

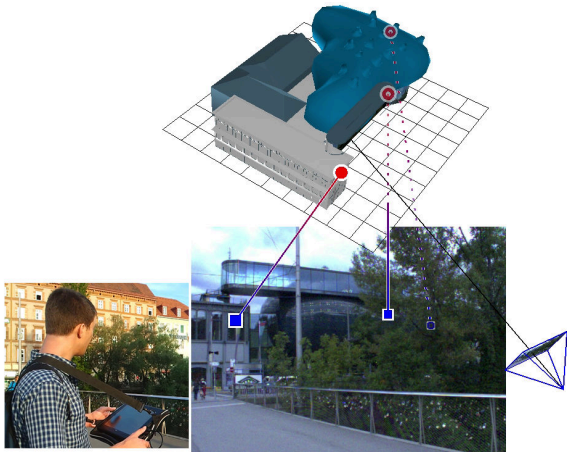


Figure 5.6: Pilot Study. A spatial technique (3DSEP+L) applied in a real urban environment. The small inset shows a participant using our system.

5.1.4 Evaluation: Real-World Scenario

In the previous studies, we focused on general properties of our interfaces and avoided confounding factors from real world scenes by using only abstract scenarios. In the following study, we introduce the real world into our interface design. We first performed a pilot study in a real world setting to collect qualitative feedback and identify issues with the interfaces. Afterwards, we performed a more thorough and controlled study in our laboratory testbed.

5.1.4.1 Pilot study: Real-world Setting

We conducted a study in a popular urban area of our city center with the 3DINP+L and 3DSEP+L interfaces. Figure 5.6 shows the 3DSEP+L interface with one of the target buildings.

Task and Methodology. Participants had to find and point to the real world location of a sphere located on the copy of the focus object. Participants were bound to a fixed location, but could rotate with the mobile device (InertiaCube3 sensor). The task was repeated with three visible distinctive cultural buildings located in varying distance around the participant: an art gallery ($40m$), a building floating on the river ($200m$), and a tower ($370m$). Pointing was estimated roughly by visual and verbal assessment. After the experiment, participants completed a questionnaire.

Participants. Ten people (7m/3f) aged between 16 and 32 ($mean$=24.2, sd=4.3) participated. They were recruited among local pedestrians and familiar with the surroundings.

Discussion. All participants were able to solve the task easily and, generally, gave positive feedback. All of them could imagine to use such an interface as a tourist, for exploring unknown landmarks and sights (5-point Likert, 1=strongly agree: $mean$=1.3, sd=0.48). Visual links were regarded useful as orientation cue. In contrast to the previous study, we did not find any significant difference in preference between 3DINP+L ($mean$=4.0, sd=0.82) and 3DSEP+L ($mean$=3.8, sd=1.1). Participants who preferred 3DINP+L again stated that it was more intuitive and natural; the ones who preferred 3DSEP+L stated that it provided a better overview and that the copy was clearly visible due to the spatial separation from the video context. Hence, a main issue seems to be the visual interference of the copy with the real world.

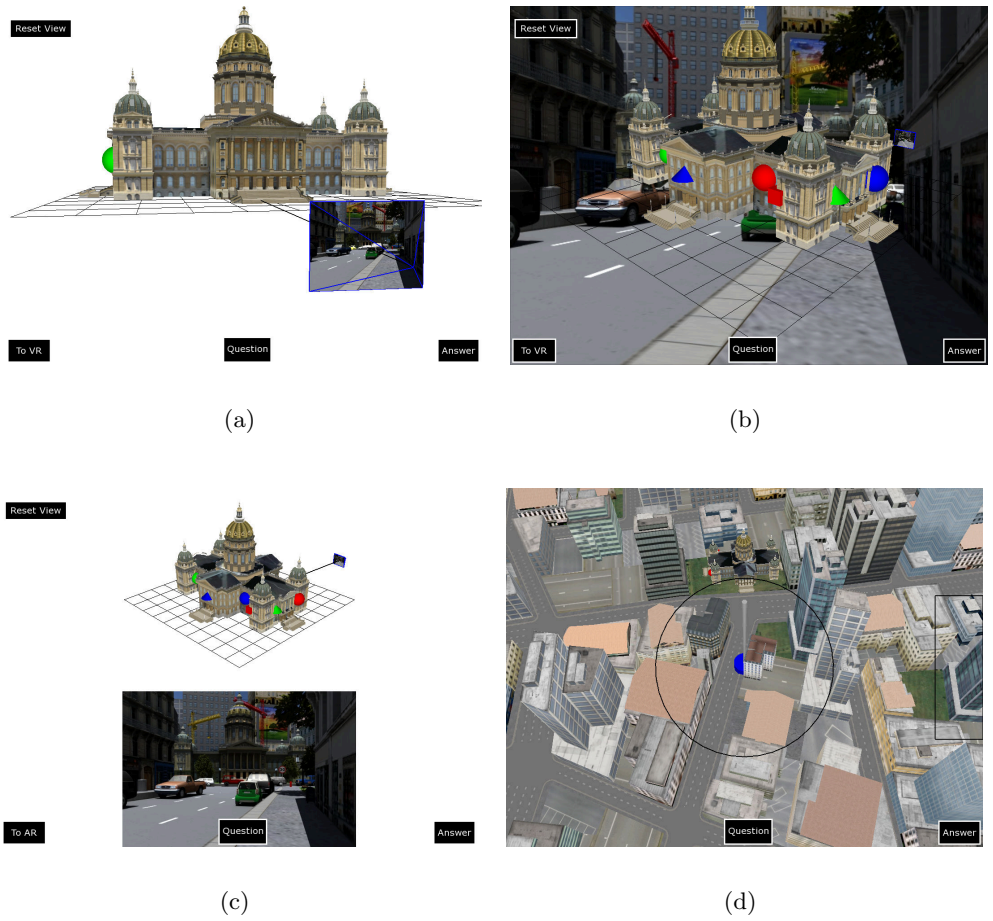


Figure 5.7: Interface designs for study in real scenarios. (a) Temporal (3DTMP), (b) in-place (3DINP) and (c) separation interface (3DSEP) applied to the most complex scene. Note the lack of contrast between virtual copy and context in (b) and the reduced size of the virtual copy in (c). (d) A map interface (MAP) applied to the most complex scene. The map is centered on the user's position and oriented in viewing direction. The circle in the center indicates the interaction area for the rotation, the rectangle on the right the one for the zoom. The translation interaction area is located outside of the other areas.

5.1.4.2 Experimental Design

Although users preferred 3DINP+L in the laboratory setup, we could not reproduce this when deploying the interfaces to a real world setting. According to participants' feedback, the main issue was the visual interference between video background and the 3D copy in 3DINP+L. Therefore, we decided to investigate the influence of different real world scenes on our interface design. We used the apparatus described in section 5.1.3.1.

Condition: Interfaces (4). The studied OCE interfaces only differ in terms of the

spatial-temporal aspect. We reused the spatially separated 3D interface (3DSEP) without visual links (Figure 5.7(c)) and the in-place interface, from which we removed the links (3DINP) (Figure 5.7(b)). We also developed a temporal interface (3DTMP) which, similar to 3DINP, shows the focus object registered to the real object, but does not preserve the video when switching to the VR mode (Figure 5.7(a)). Hence, this interface not only exhibits high visual contrast to the background similar to 3DSEP, it also exhibits the natural behavior of 3DINP and its increased size of the 3D object.

As a baseline condition for our interfaces, we included a 3D map interface (MAP) (Figure 5.7(d)). The map shows buildings and terrain without additional contextual information from the real world, such as trees and cars. The view is centered on the location of the user, which is indicated with a blue cylinder, and oriented towards the real focus object. The user can translate, zoom and rotate the view on the map. The rotation and zooming center is defined by the screen center and indicated by a grey cylinder.

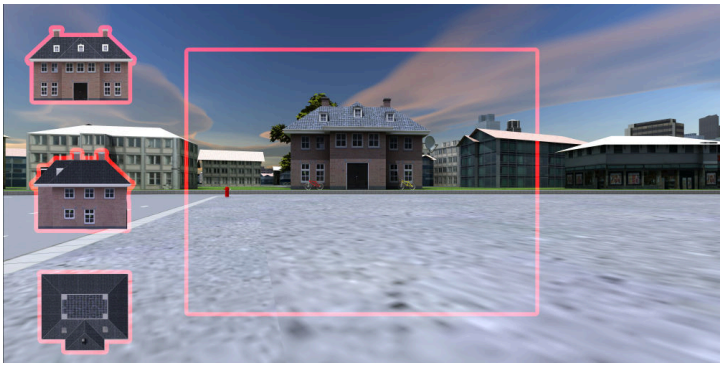
We did not include the visual links in this study, because they may be a confounding factor due to the clutter added to the presented real world scenes. Furthermore, in MAP and 3DTMP, the end point of visual links do not connect to a visible real world context.

*Condition: Scene Complexity (*3*).* We prepared three artificial scenes with a 3D city modeling software (Esri CityEngine). Unlike when using realistic pictures created with image-based modeling techniques relying on photographs, this approach allowed us to have control over the presented scenes. It also removed the effect of scene knowledge from the study, because participants were not familiar with the buildings or their locations.

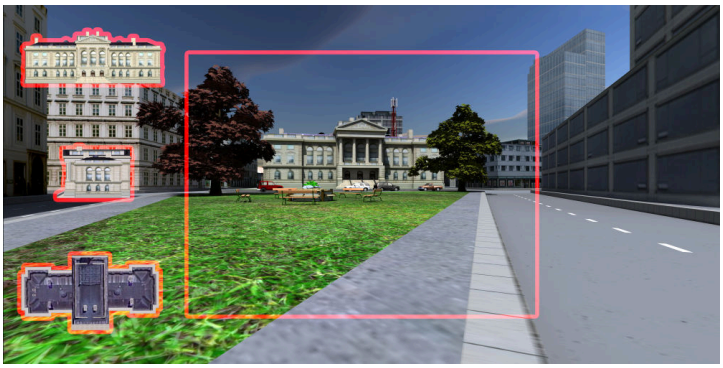
We rendered two views of each scene using a third party software (Lightwave'12): one simulated the periphery, one the AR view. The periphery was rendered using a camera (100° FOV) in a resolution that matched the projection wall. The AR view was rendered with a camera (60° FOV) in a resolution that matched the one of the display device. All views were taken at the eye level of the participant.

The generated scenes exposed different degrees of real world complexity. In contrast to Lee et al. [81], who define complexity with different levels of visual realism, we define it as the number of unique objects classes, the geometric complexity of the focus buildings and the density of the neighboring buildings. Figure 5.8 shows the peripheral views, the outtake for the AR view (red rectangle), and the buildings with an outlined silhouette.

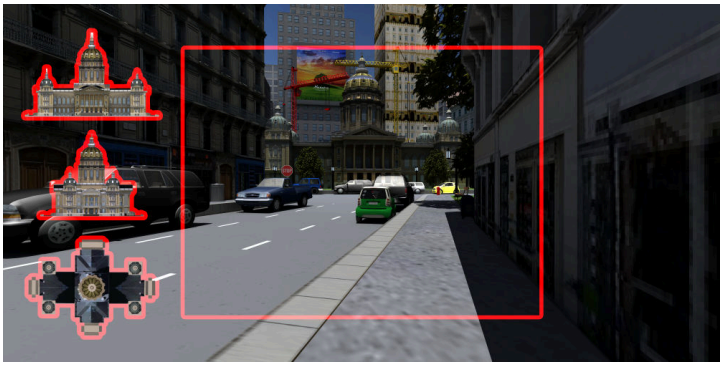
Task. We included the context in the task and asked the participants to find an object of a certain color and shape, which was close to an object visible in the context. We used a total of nine objects: three cubes, three cones, three spheres. Each of the objects of



(a)



(b)



(c)

Figure 5.8: Scenes with (a) low, (b) medium and (c) high complexity. The estimation of complexity considers the number of unique object classes, the density of neighboring buildings and the geometric complexity of the focus buildings. To the left, the silhouette of the buildings is outlined. The center rectangle is the content visible in the AR view. (Best viewed digitally and zoomed.)

Table 5.3: Excerpt of questionnaire. Q1 and Q2 use a 5-point Likert scale (1=strongly agree), Q3 a grade system (1 to 5; 1 = worst).

Q1	It was easy to solve the task using the interface.
Q2	The interface was intuitive to use.
Q3	Rate how you liked the interface.

a shape were colored in either red, green or blue. We arranged the shapes at a medium height on the copy of the focus. To force participants to navigate the copy, we placed the objects only sideways and in the rear (Figure 5.7). The colors were randomized, and the shapes were placed pseudo-randomly such that the answer to the posed question had a clear solution. The position of the object in question was consistent among the trials between participants.

The question was: "Which color does the *shape* closest to the *object* have?" *Shape* refers to one of the three shape types, *object* to an object visible in the periphery as well as the AR view. To avoid that participants learn the location of *objects*, we varied their placement in the scene between each interface.

Procedure. The interface condition was counter-balanced, scenes in each interface condition were presented in random order. Before using an interface, participants solved a trial task without time constraint in a trial scene. Participants finished all scenes with an interface, before progressing to the next. The question was shown at the bottom of the screen when starting the task. It disappeared, when interaction started, and reappeared, when the corresponding button was pressed. The task was finished, when participants selected a color on screen. We recorded task completion time and color error.

When using MAP, the participants had to take the mobile device in their hands to simulate map usage behavior. In the other interface conditions, the device was mounted in front of the participants to simulate an AR view. Participants completed questionnaires after each interface (4) and after finishing the study (1). An excerpt of questions is shown in Table 5.3.

Hypotheses. H1. Our hypothesis was that the interfaces outperform each other in terms of task completion time as follows: MAP $<$ 3DTMP $<$ 3DINP $<$ 3DSEP. We considered that MAP is not designed for object-centered exploration and requires the most interaction effort. Furthermore, in MAP, the investigated buildings are occluded by neighboring buildings. 3DINP and 3DSEP outperform 3DTMP, because, in 3DTMP, participants cannot use the video context in the VR mode and must look into the periphery. 3DSEP outperforms 3DINP, because the video is not occluded by the copy. *H2.* Our

Table 5.4: Third Study. Mean completion times and SD in seconds.

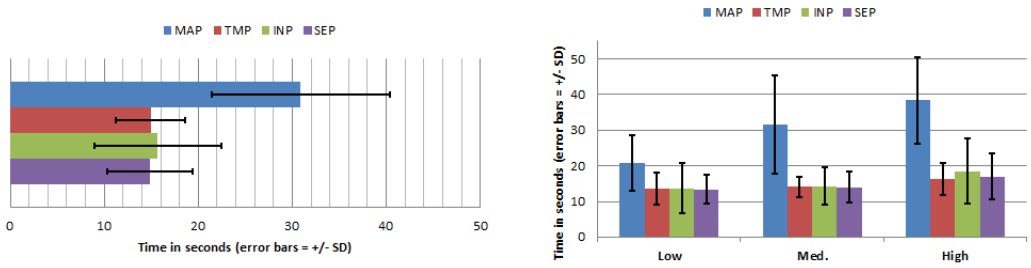
		Low	Medium	High
	Mean (SD)	15.36 (4.43)	18.51 (4.67)	22.57 (5.9)
MAP	30.90 (9.45)	20.69 (7.77)	31.56 (13.85)	38.34 (12.12)
3DTMP	14.92 (3.68)	13.66 (4.51)	14.16 (2.80)	16.43 (4.51)
3DINP	15.67 (6.75)	13.74 (6.95)	14.32 (5.32)	18.48 (9.20)
3DSEP	14.87 (4.56)	13.35 (4.04)	14.05 (4.32)	17.04 (6.36)

Table 5.5: Third Study. Significant effects between interfaces per scene, tested with Friedman ($\alpha = 0.05$) and Wilcoxon signed-rank tests with Bonferroni corrected α=0.0083.

	Low	Med.	High
	$X^2(3)$=17.1 **p < 0.001**	$X^2(3)$=26.8 **p < 0.001**	$X^2(3)$=29.7 **p < 0.001**
MAP&3DTMP	**p=0.006**	**p<0.001**	**p<0.001**
MAP&3DINP	**p=0.001**	**p=0.001**	**p<0.001**
MAP&3DSEP	**p=0.002**	**p<0.001**	**p<0.001**
3DTMP&3DINP	$p = 0.836$	$p = 0.918$	$p = 0.535$
3DTMP&3DSEP	$p = 0.796$	$p = 1.0$	$p = 0.836$
3DINP&3DSEP	$p = 0.959$	$p = 0.642$	$p = 0.569$

second hypothesis was that the scene complexity has a negative influence on the task completion time. We believe that more complex scenes lead to higher task completion times.

Participants. The experiment followed a within-participants design, with five repetitions for each interface and task (960 trials). A total of 16 people (14m/2f), 24-46 years old ($mean$=30.06, sd=5.62), took part in the study. The participants were recruited from university staff and on the campus.



(a) (b)

Figure 5.9: Third Study. Task completion time (a) per interface condition and (b) per interface and scene.

Table 5.6: Third Study. Significant effects between scenes (Low=L, Med.=M, High=H) per interface tested with Friedman ($\alpha = 0.05$) and Wilcoxon signed-rank tests with Bonferroni corrected α=0.0167.

		L&M	L&H.	M&H
MAP	$X^2(2)$=22.875, **p < 0.001**	**p=0.001**	**p=0.001**	p=0.026
3DTMP	$X^2(2)$=14, **p <0.05**	p=0.148	**p=0.001**	**p=0.003**
3DINP	$X^2(2)$=16.625, **p < 0.001**	p=0.163	**p=0.001**	**p=0.005**
3DSEP	$X^2(2)$=7.875, **p < 0.05**	p=0.5	**p=0.003**	**p=0.007**

5.1.4.3 Results

For each interface × scene complexity condition and participant, we calculated the mean time and error from the five repetitions. Based on this, we calculated the mean of the scene complexity and interface conditions for each participant. The values are summarized in Table 5.4 and Figure 5.9. We performed non-parametric tests, because our sample violated normality. We do not report on the error, because it was practically non-existent.

Interface. A significant effect of interface on time was observed (Friedman, $X^2(3)$=18.075, p<0.001). A Post-hoc Wilcoxon signed-rank test with Bonferroni corrected α=0.0083 showed that all AR interfaces were significantly faster than MAP (p<0.001). This significant effect between MAP and AR interfaces was present in each scene complexity condition. For better readability, these test results are presented in Table 5.5.

Based on these results, we *partially accept H1.* The AR interfaces outperform MAP in terms of task completion times. The performance of the task completion time is also consistent between all scenes and, thus, applies to different scene complexities.

Scene. A Friedman test between scene complexities revealed a significant effect on completion time ($X^2(2)$=22.875, p<0.001). A post-hoc Wilcoxon signed-rank test with Bonferroni corrected α=0.0167 showed significant differences between scene A and B (p=0.003), A and C (p=0.001) and B and C (p=0.002). For better readability, the significant effects between the scenes of each interface are summarized in Table 5.6.

Based on these results, we *accept H2.* Scene complexity had an overall negative impact on the task completion time of all interfaces.

Questionnaires. The questionnaire data is summarized in Table 5.7, significant effects, in Table 5.8. Participants were asked to rank the scenes according to their perceived complexity, and rated the scenes as follows: high as high (100%), medium as medium (94%), low as low (94%). One participant switched medium and low.

Table 5.7: Questionnaire data with mean and SD (rounded) for the study using the real world content (3).

	Study 3			
	T3			
	MAP	3DTMP	3DINP	3DSEP
Q1	3.0 (1.2)	1.4 (0.5)	1.6 (0.6)	1.3 (0.5)
Q2	2.9 (0.9)	1.5 (0.8)	1.3 (0.5)	1.3 (0.5)
Q3	2.1 (1.0)	3.9 (1.0)	4.4 (0.6)	3.8 (0.9)

Table 5.8: Third Study. Significant effects in questionnaire data. The data was tested with Friedman (not reported) and Wilcoxon signed-rank tests with Bonferroni corrected $\alpha = 0.0083$.

	Q1	Q2	Q3
MAP&3DTMP	**p=0.001**	**p=0.001**	**p=0.002**
MAP&3DINP	**p=0.003**	**p=0.001**	**p=0.001**
MAP&3DSEP	**p=0.001**	**p<0.001**	**p=0.001**
3DTMP&3DINP	p=0.317	p=0.527	p=0.070
3DTMP&3DSEP	p=0.414	p=0.206	p=0.869
3DINP&3DSEP	p=0.096	p=0.655	p=0.078

5.1.4.4 Discussion

Participants significantly preferred the AR interfaces over MAP (Q3), and all AR interfaces performed better than a traditional 3D map interface. Even 3DTMP that did not preserve the video context in the VR mode performed better, which indicates that the transition between AR and VR and the simple camera cue are sufficiently strong cues for connecting real and virtual world. A major limiting factor of MAP was occlusions from neighboring buildings in scenes of higher complexity. This was also supported by corresponding statements in the interview. Generally, the task was significantly easier to solve with the AR interfaces (Q1). Another factor which might have negatively influenced MAP performance is that the objects referred to in the task were only visible in the periphery and not in the map itself. For this reason, in MAP, participants were forced to redirect their view between the mobile device and the periphery, while in the AR conditions the relevant objects were presented in the context displayed on the mobile device either sequentially (3DTMP) or in parallel (3DINP, 3DSEP).

Contrary to what we expected, we did not find any differences between the AR interfaces. Participants could quickly find the queried object in the context at the beginning of the task. In the VR mode, participants could always look into the periphery by glancing past the mobile device mounted in front of them. This may have been sufficient to also achieve good performance in 3DTMP, where no context was available after switching to the VR mode. When using an HMD without the option of looking at the periphery

directly after switching to the VR mode, 3DTMP may perform differently.

The participants' comments regarding the interfaces were in line with those of previous studies. Participants noted the good visibility of the copy in 3DTMP and 3DSEP and the intuitive arrangement of focus and context in 3DINP.

Q2 revealed that the AR interfaces were significantly more intuitive than MAP. One reason for this rating might be that the interface was not well suited to the task of object-centric exploration. Another reason may be that MAP used a single touch interface instead of a more common multi-touch interface. A multi-touch interface might improve the intuitiveness and even the performance of MAP, but will still exhibit problems with occluding structures. Hence, we are confident that our results also hold up against MAP interfaces available on current mobile devices.

The performance and questionnaire data confirmed our estimation of the scene complexity. Performance generally degraded with increasing scene complexity, and the ranking of scene complexity by the participant was in line with our estimation. The impact on performance can be attributed to different factors in the AR and MAP conditions. In MAP, our observations and the interviews revealed that the density of neighboring buildings had a major impact on performance. In the AR conditions, the decrease in performance can mainly be attributed to the geometric complexity of the focus object. In the AR conditions, there was only a significant effect between the scene with highest complexity and all other scenes (Table 5.6). In contrast to the buildings in the other scenes, the building in the most complex scene exhibited concavities, which occluded the queried shapes and thus required more navigational effort.

The map interface can be classified within the frame of our design space. It is a temporal interface, which does not provide any strong cues to resolve the viewpoint discontinuity between the egocentric viewpoint of the user and the exocentric map view. There is no seamless transition between these views, but, similar to the camera icon in the AR interfaces, the position indicator of the map interface represents the current viewpoint of the user. The map interface exhibits a large context discontinuity, because the real world context is replaced with only a virtual representation showing buildings and terrain. In comparison, the context in the three AR interfaces shows only an egocentric 2D view, but is richer in information, because it contains details such as cars or street signs.

5.1.5 Design Recommendations

In the following, we put forward design recommendations for OCE techniques based on the findings of the previous studies. We also outline potential future research directions. The recommendations are structured based on the aspects outlined in the design space presented in section 5.1.1.

Spatial-Temporal Aspect. We did not find performance differences between the in-place and the spatially separated interface. However, under controlled laboratory conditions, participants significantly preferred the in-place interface (3DINP+L), because the arrangement of focus and context was more natural. Occlusion of the context by the copy did not seem to be an issue. In the real world setting, we did not find a significant preference, because participants also preferred 3DSEP+L, because of the higher contrast between the copy and the white background. Therefore, an in-place interface may have to adapt the *context* to always achieve a good contrast to the overlaid focus object (e.g., desaturation of video background).

Generally, participants used the interfaces to get a quick *overview* of the focus object. Using 3D interfaces, participants switched to a top-down view to quickly look at the different sides of the object. In 2DSEP, participants used the small images as multi-perspective visualization to quickly identify the view containing the queried red sphere in T2. Therefore, OCE techniques should offer modes to explicitly get an overview of an object. When using images as overview, the relevant items on the object should be emphasized in an authoring step beforehand (e.g., by labels), due to the small size of the images, especially on mobile phones.

Input Control. An overview can easily provide *shortcut navigation* to quickly access viewpoints. In 2DSEP, participants used the small images to quickly navigate between viewpoints in non-sequential order by accessing 90° and 180° offset views. This is also a main motivation for similar interfaces, such as SnapAR [124] or the one of Veas et al. [138].

Based on our experience with MAP, a simplified *camera navigation* model with few degrees of freedom (e.g., orbit metaphor) was sufficient for the investigated structures. However, future designs should also consider zooming and the exploration of more complex structures, which require more sophisticated navigation metaphors. For instance, the HoverCam [76] allows to explore complex objects with few degrees of freedom.

Virtual Copy. Our findings are in line with Bowman et al. [19], who found out that instant teleportation causes disorientation. In our studies, continuous 3D viewpoint changes outperformed discrete 2D switches. Therefore, a 3D interface is the most sensible choice

for presenting viewpoint changes to the user.

Spatial Cues. When designing our OCE interfaces, we focused on the presentation of a single point of interest and provided only limited contextual information through the video background (3DSEP, 3DINP). In one design, we even removed the context completely and presented only the virtual copy and the camera icon (3DTMP). Participants could easily solve the given tasks with 3DTMP, the most basic OCE interface design. All OCE interfaces also outperformed the map interface, which also showed structures of surrounding buildings and thus provided more contextual information about surrounding structures. Hence, the *transition* between AR and VR mode and the *camera icon* seem to be sufficiently strong cues to connect separated views and address spatial, viewpoint and context discontinuity. The radar icon was not considered helpful. This indicates that cues should be connected directly to the spatial reference frame of the copy.

Based on the collected data and participants' feedback, we can say that *visual links* are a valuable spatial cue. We designed visual links with spatial separation between focus and context in mind (3DSEP+L). However, participants considered spatial separation as inferior to a more natural in-place interface, due to the smaller size of the zoomed focus object. Hence, the links could be redesigned to better support in-place interfaces. We observed that participants mainly used links during the mapping task (T2). An intermediate mode could be introduced that switches from in-place to a spatially separated presentation, when users want to map information back to the real world.

5.2 Smart Transitions using Scene Semantics

The OCE techniques showed that the transition between virtual and real content seem to be sufficient so that users can stay oriented when switching between modes. However, the transitions are rather straightforward and do not take into account the actual goal of the user. A user that annotates an object might have another viewpoint in mind than a user that just explores the object. Furthermore, often users may not want to completely switch to the virtual viewpoint, but only preview certain aspects. Therefore, we refine the transitional aspect of the interfaces to be more reactive to the context of the application.

Another aspect of the interfaces presented in this section is that they are better suited to handle unknown environments. Augmented Reality (AR) applications traditionally rely on predefined, rigidly modeled content that only applies to those conditions of the real world that were present when the application was developed. Hence, often an AR application can be deployed only in a single physical location, which limits its flexibility.

This problem is aggravated when changes in the environment cause misalignments between the previously recorded data and the real world. Such a situation can impair the correct functioning of the application.

Fortunately, there has been tremendous progress in the area of real-time Simultaneous Localization and Mapping (SLAM) [99]. SLAM enables users to deploy AR in unprepared and unknown environments and allows them to capture the geometry and the visual appearance of the environment. Furthermore, changes to the real world scene can be captured, and AR applications can react to these changes. This allows users to engage into AR applications anywhere and at anytime. However, real world scenes are not modeled beforehand anymore, and users need to interact with and manipulate unprepared environments that are unknown to the application.

We combine rapid scene acquisition and extraction of basic semantic information from the captured scene with new transitional interface techniques that allow users to navigate and manipulate unknown real world environments. Our techniques provide natural transitions between AR and VR viewpoints and strategically place the virtual camera in locations that are better suited for the current task of the user. For this purpose, we capture a real world scene using KinectFusion [65] and analyze it to extract semantic information to determine the virtual camera viewpoint. Note that the presented techniques can be used independently of the real-time capture system and can be extended by more sophisticated semantic information.

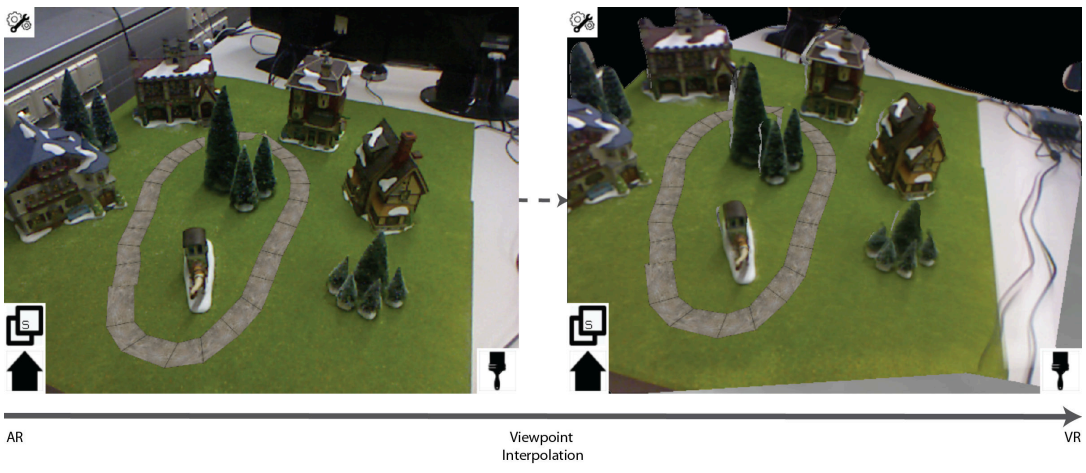


Figure 5.10: After placing physical objects on a table, our system offers the possibility to structurally navigate an unprepared scene with a set of new transitional navigation techniques in AR (Left) and VR (Right) modes. The techniques seamlessly switch from AR to VR modes.

5.2.1 Interface Design

We present four navigation techniques that can be used for dynamic scenes. Each technique allows a seamless transition between the AR view and VR view of the captured scene. The design of the presented transitional interface corresponds to a temporal interface as evaluated before (3DTMP) (see Section 5.1.4.2). Consequently, the interfaces transition from an AR view to a VR view that shows only the copy of the real world.

We classify the presented transition techniques into two categories: context-aware transitions that use the knowledge gathered from the scene analysis and intermediate transitions that provide VR views but are still connected to the AR view.

The user can switch between AR and VR views at any time using one of the presented techniques. To quickly move from VR back to AR, we provide a home button located in the bottom left of the view. We provide spatial cues in the VR view to communicate the current position of the tracked AR camera and, thus, the position of the user relative to the virtual model. We visualize the tracked camera position in VR by rendering a camera frustum (Figure 5.11(a)). We also visualize the viewing direction of the tracked camera by projecting the borders of its viewport onto the scene geometry (Figure 5.11(b)). Furthermore, we allow the user to open an AR window inset in the VR view, which shows the AR view of the scene (Figure 5.11(c)). This also allows a user to see changes that were performed in the VR view in the current AR view without switching back to AR.

To demonstrate our techniques, we use the scene shown in Figure 5.13(a) and render the virtual scene using Image-based Rendering (IBR). Semantic input is provided by classifying the scene into *ground*, *object* and *top*.

5.2.2 System Overview

Our system consists of four components: capturing a real world scene, analysis of the scene, retrieving knowledge about the user's task and camera navigation that fuses these components to create viewpoint transitions. Figure 5.12 provides a schematic overview of the components and their relation to the traditional AR pipeline.

In a traditional AR pipeline, the tracking component controls the viewpoint of the rendering. We expand this pipeline with our scene navigation component, which seamlessly switches between live tracking and virtual camera viewpoints. This component also selects camera manipulators to allow changes of the virtual viewpoints. The navigation component strategically selects viewpoint and manipulator based on the knowledge about the user's task and knowledge about the scene. In our system, we gather scene knowledge

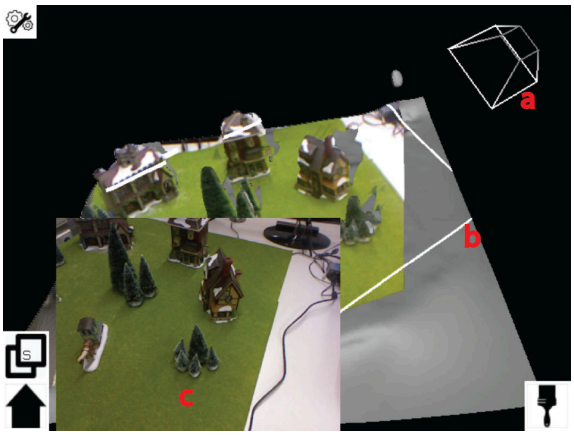


Figure 5.11: Spatial Cues. We provide spatial cues to facilitate orienting in the VR view. (a) A camera icon shows the pose of the AR view, which is still tracked when the user switches to the VR view. (b) The viewport of the AR view is also projected into the scene. (c) The user can optionally activate a small window showing the current AR view.

automatically by analyzing a virtual representation of the scene, which we capture and reconstruct at run-time. The update frequency of the capturing process can be adjusted to the reconstruction algorithm and its computational demands. The navigation runs in real-time and operates independently of the capturing.

In the following, we present the details of three components: scene capture, scene analysis and scene navigation. The task knowledge component depends on the application scenario (authoring, games, etc.). We do not discuss this component in detail.

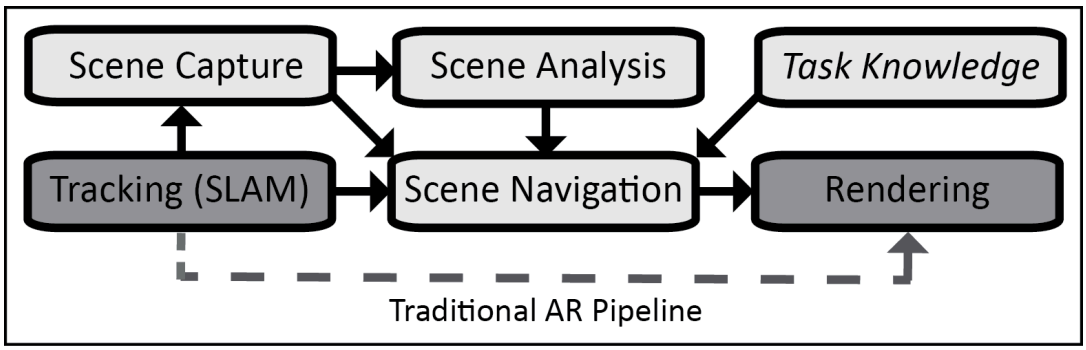


Figure 5.12: Overview. In a traditional AR pipeline (dashed line) the tracking system controls the viewpoint of the rendering. We expand this pipeline by a scene navigation component, which allows to switch from AR to VR views and chooses viewpoints based on scene and task knowledge. We gather scene knowledge from analyzing a scene captured at run-time, e.g., by using a SLAM system.

5.2.3 Capturing Scene Semantics

To enable modifications of the scene, we need scene knowledge. Therefore, we developed a system that allows us to capture a 3D model of the real world environment and that automatically identifies the 3D objects of the scene.

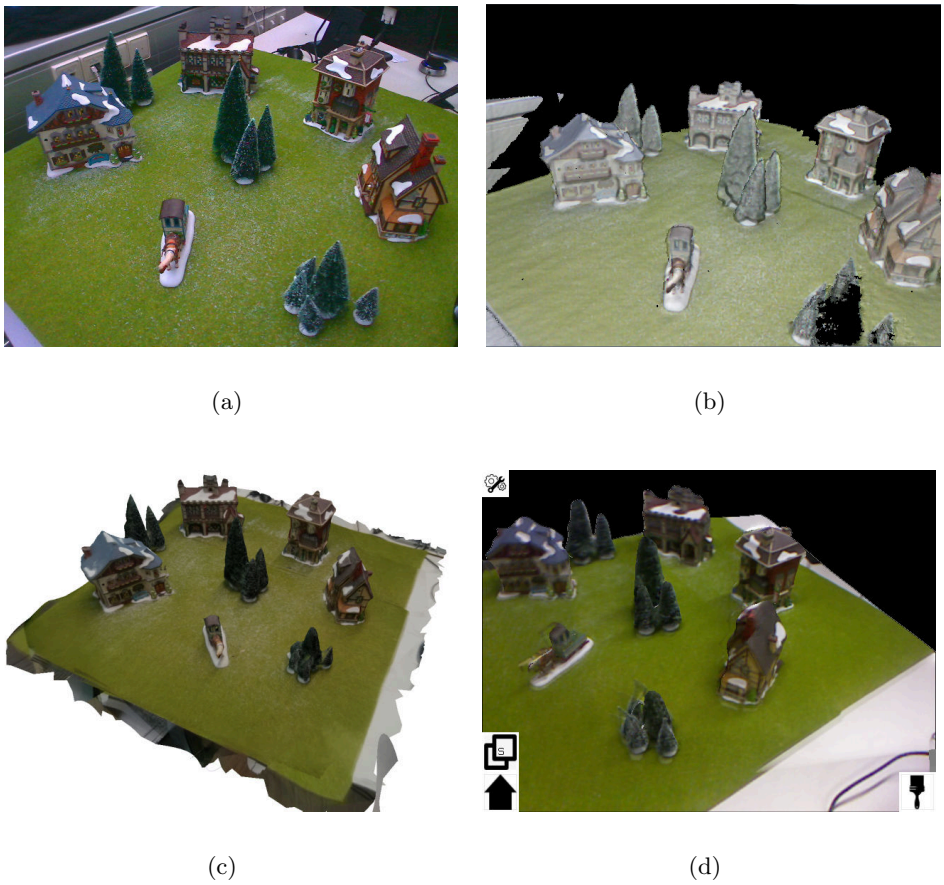
Scene Capturing. Virtual representations of real world scenes can be captured in real-time using SLAM technology [99]. The combination of SLAM with depth sensors, (e.g., KinectFusion [65]) allows live capturing of detailed models, either as volumes, depth images or polygon meshes. Figure 5.13(b) shows the output of an open source KinectFusion implementation[1] applied to the scene shown in Figure 5.13(a). We generally acquire a textured virtual representation of the real world, so that we can fill holes, if the scene is modified by the view management approach and scene objects are moved out of place.

Models can be constructed with monocular SLAM technology, stereo SLAM, or depth sensors. Alternatively, models can also be reconstructed using an online reconstruction service. While capturing the scene, the data is processed in the cloud and sent back to the user. For example, we created the model in Figure 5.13(c) by using a freely available online reconstruction service[2].

In our prototype system, we create a mesh approximation of the volume generated by an open source KinectFusion implementation. We reduce the number of voxels by using a grid filter that calculates the centroid of voxels within a cube having a side length of $1cm$. Then, we use a poisson mesh algorithm [75] using the filtered points to create the mesh representation. We chose the tree depth used for the poisson reconstruction to find a trade off between reconstruction performance and accuracy of the created geometry. A tree depth of six allows for more frequent mesh updates, but the details of the geometry are smoothed. However, the general shape of the object is approximated well. A tree depth of seven creates a good approximation of the objects at the cost of a lower update frequency. In our system, we allow the user to switch at runtime between these settings. Hence, a user may choose faster updates, when the real world scene is rearranged frequently, or a better mesh approximation with higher visual quality after the scene was arranged.

We create a separate thread for the meshing process to avoid a performance impact on the rendering thread. In our prototype system, a tree depth of seven took around eight seconds to reconstruct, while a depth of six took only around two seconds for a point cloud containing $25k$ points after filtering. This size corresponds to a complete scan of the scene.

[1] `http://pointclouds.org`
[2] `http://www.123dapp.com/catch`



(a) (b)

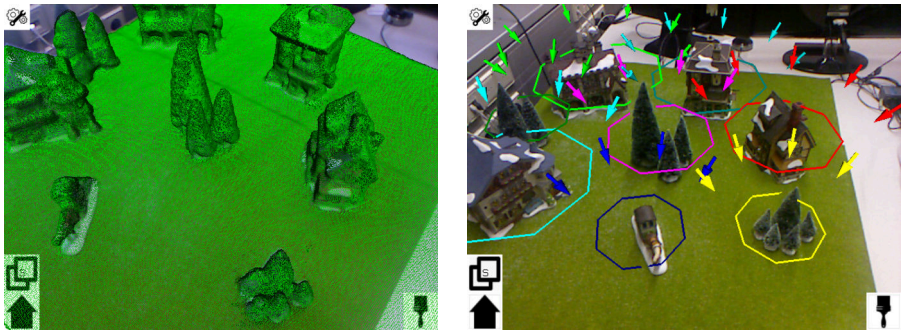
(c) (d)

Figure 5.13: Scene Representation. Our system works independently of the scene capturing technology or rendering representation. The real scene shown in (a) can be reconstructed and rendered using different methods, such as (b) rendering colored voxels of a volumetric representation generated with KinectFusion, (c) rendering a polygonal mesh of a reconstruction or (d) image based rendering of a reconstructed proxy mesh.

We deployed the prototype on a PC running Windows 7, equipped with an Intel i7 CPU quad-core 2.66GHz, 12 GB RAM and an Nvidia 780GTX graphics board.

We use an IBR approach to create a textured reconstruction of the real world. The IBR method provides view dependent updates of the virtual representation (Figure 5.13(d)). This also improves visual fidelity and the visual similarity between a physical view of the scene (AR) and a virtual view of the scene (VR). Note the similarity of the IBR in Figure 5.10(Right) with the real world scene shown in Figure 5.10(Left). We record the images used for the IBR during the capturing process by sampling images at points that are located on a regular grid in 3D space. For each point, we store a number of images

with viewing directions offset by 45°. To be able to react to changes to the scene, we implemented a simple image aging algorithm, which expires stored images after a certain amount of time. Alternatively, images can also expire when the geometry shown in the image changes.



(a) (b)

Figure 5.14: Reconstruction Feedback and Guidance. (a) We overlay the point cloud from the KinectFusion tracker in real-time to provide reconstruction feedback. (b) To support the user during scene capturing, we visualize viewpoints that have not yet been visited by rendering arrows around the objects segmented by the scene analysis.

To support the user in capturing the real world scene, we provide reconstruction feedback to the user by rendering the currently reconstructed point cloud in AR (Figure 5.14(a)). In contrast to the mesh representation, this visualization is updated almost instantaneously. Furthermore, the system can guide the user around objects in the scene to ensure that all sides of an object have been viewed with the camera (Figure 5.14(b)). Arrows around the object guide the user to viewpoints that have not been visited before. Arrows are removed when the viewpoint has been visited. To enable this feature, we use automatic scene segmentation, which is discussed in the next section.

Scene Analysis. To be able to modify certain elements of the scene, we need to acquire not only a virtual representation, but also scene knowledge, e.g., about the single objects contained in the scene. Niederauer et al. [100] show how to generate semantic information from a 3D mesh. Additionally, the semantic information can be derived from others sources such as object recognition from captured video data [25]. Sensors in mobile devices facilitate the analysis by providing information about gravity or compass reading, which allows to identify a ground plane or a specific direction.

We need scene information for controlling the camera, but also for the transition between different AR and VR views. Depending on the user's current task, selecting a particular object can trigger a different viewpoint change. For instance, when selecting the ground plane during authoring, the virtual viewpoint might transition to a top-down view to allow the user to arrange virtual objects on the ground plane. However, during a game session, the camera may transition to a ego-perspective on the ground level.

Figure 5.15: Scene Segmentation. We automatically segment the scene to identify the ground plane defining the world coordinate system and single objects on the ground plane. We use this knowledge to control the viewpoints of transitions. Colors represent the segmented object.

In this thesis, we focus on tabletop scenarios (e.g., Figure 5.13(a)). We identify the plane corresponding to the table and the single objects on the table by segmenting the reconstructed point cloud. The plane is identified using a sample consensus method that fits a plane model. After removing the plane points from the point cloud, we identify point clusters based on Euclidean point distances. Using this simple segmentation, the system is now able to differentiate between the ground plane and single scene objects (Figure 5.15). Additional information can be added to the analysis using domain knowledge. Because our example scene consists of houses, we can identify walls and roofs of the houses, either geometrically or by defining every point lying above a certain height to be a roof.

5.2.4 Context-Aware Transitions

Context-aware transitions make use of the data gathered from the scene analysis to control the viewpoint of the virtual camera. Aside from having knowledge about the objects in the scene, we also use the mesh normals to control the camera orientation. For instance,

the plane normal defines the up vector of the world so that virtual geometry and virtual cameras can be placed with correct up orientation. We also use the normals of the convex hulls of objects to orient cameras towards the geometry. We use the normals of their convex hull to achieve a more homogenous normal distribution.

Transitional Camera Control. The context-aware transitional camera control changes from the AR view to an automatically calculated virtual viewpoint. The location of this viewpoint is determined by taking the semantic knowledge of the scene and the current task into account. To trigger the transition to the calculated viewpoint, a user can tap on any element of the scene. After reaching the viewpoint, the user can navigate the virtual scene with the provided camera manipulator. To provide feedback about the possible target viewpoints, we display icons representing the calculated viewpoints for these locations. These visual icons are activated by dragging the finger over the scene using a hover or swipe gesture.

The results for our test scene are shown in Figure 5.16. For the viewpoint calculation, we assume an authoring task using the reconstructed scene. Hence, the semantics are interpreted to reflect that task. When the user taps on the ground, the system switches to a map like top-down view and a panning camera manipulator. When the user taps on the top of an object, we provide a top-down view, but this time closer to the object. In both cases, the normal of the ground plane is used to orient the camera to look from top down. When the user clicks on the surface of an object, we automatically provide a close-up viewpoint, which uses the normal of the ground plane as up vector and is oriented towards the geometry using the normal of its convex hull. In the close-up view, we switch to an orbit manipulator to rotate around the object. Once in VR, the user is free to choose other viewpoints.

Transitional Interaction. A context-aware transitional interaction can be used to complement interactions that were started in the AR view. The user can indicate that an interaction that was started in AR should be continued in VR by pressing on the same location on the screen, without releasing it. The system then seamlessly switches to a viewpoint that is most appropriate for the current task. Hence, the interaction of the user is not interrupted by performing a gesture to switch to VR.

In Figure 5.17, the user starts drawing a path in AR and cannot continue drawing, because the view is blocked by scene geometry. By stopping and holding the interaction at the end of the path, the system recognizes the user's intention to continue drawing and switches to a virtual viewpoint. The viewpoint is top-down, because it is best suited for

the task and scene semantics. The user can continue drawing in VR and pan the camera over the scene to extend the path in areas not reachable from the AR view.

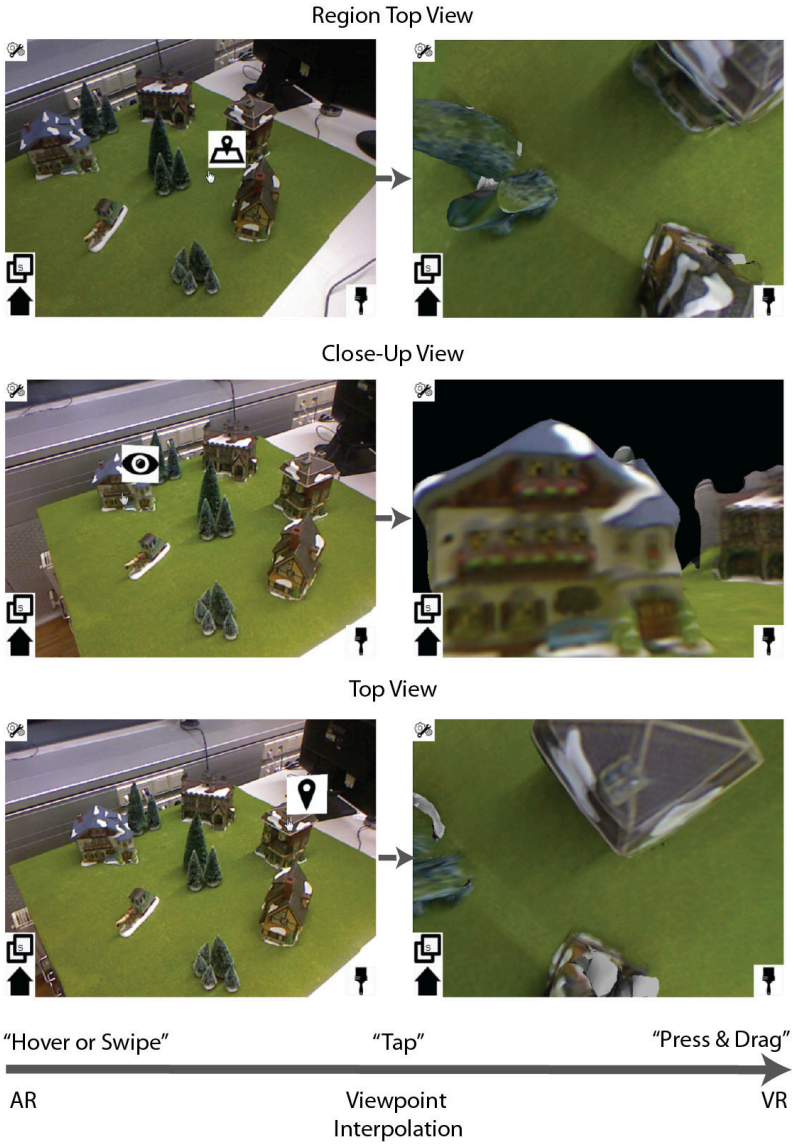


Figure 5.16: Context-Aware Transitional Camera Control. Our system chooses different navigation modes and viewpoints in function of an area selected by a user. (Left) Visual icons define which semantic modes are available. By tapping on a visual icon, semantic navigation modes are triggered, such as (Top) top-down regional view, (Middle) front view of an object or (Bottom) top view of the object.

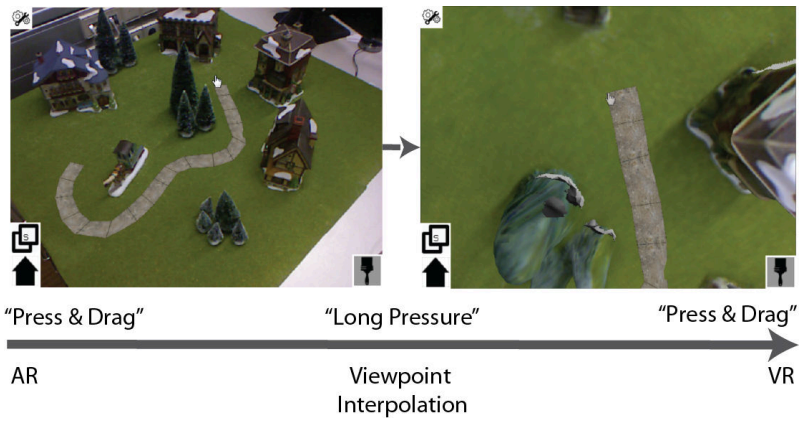


Figure 5.17: Context-Aware Transitional Interaction. Authoring and manipulating 3D content requires constant camera manoeuvring for accomplishing the task. (Left) The user in the AR mode cannot continue the drawing task behind the trees because of limited visibility. (Right) Our technique allows the user to switch to a more adequate viewpoint without stopping the interaction.

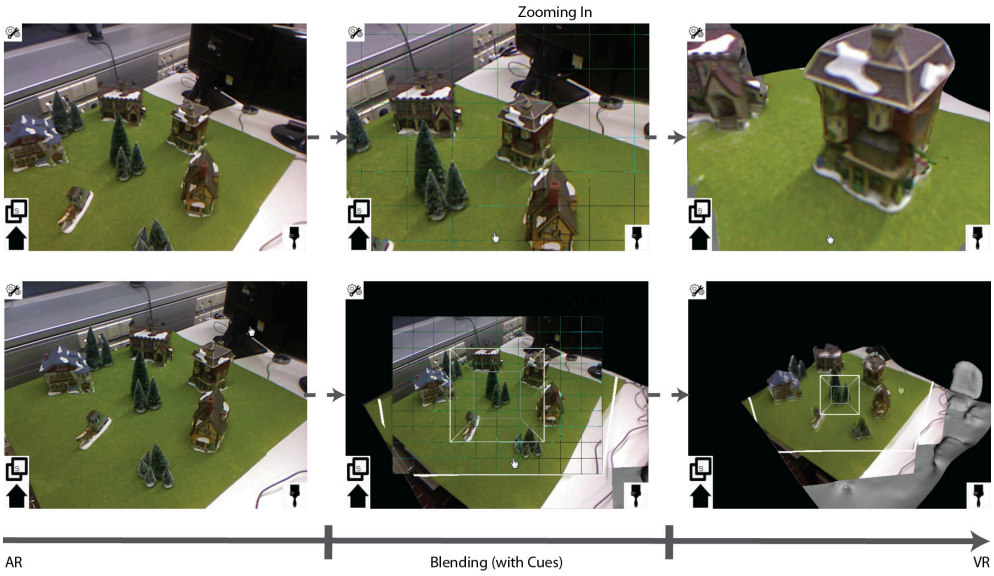
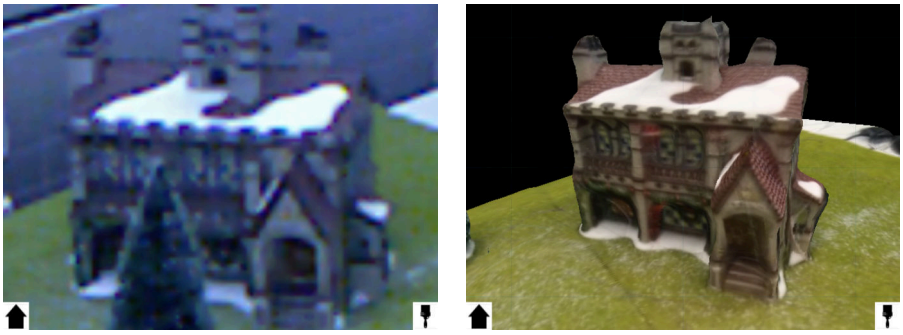


Figure 5.18: Transitional Zooming. (Top) To achieve a close-up view of a real model, a user can directly zoom in an AR view. (Bottom) A user can also zoom out of the AR view to get an overview of the scene. When zooming, the view gradually fades to the VR view when a certain distance threshold is reached. We use a virtual grid as a visual feedback to notify the user that the switch from AR to VR is imminent.

5.2.5 Intermediate Transitions

Intermediate transitions switch to a VR view, but are still loosely connected to AR. From an intermediate transition, a user can always use a context-aware transition to continue navigating in VR.



(a) (b)

Figure 5.19: Scene Zooming Methods. (a) A pure digital zoom causes artifacts in the video image. (b) We zoom into a virtual representation, which provides close-up views with higher resolution.

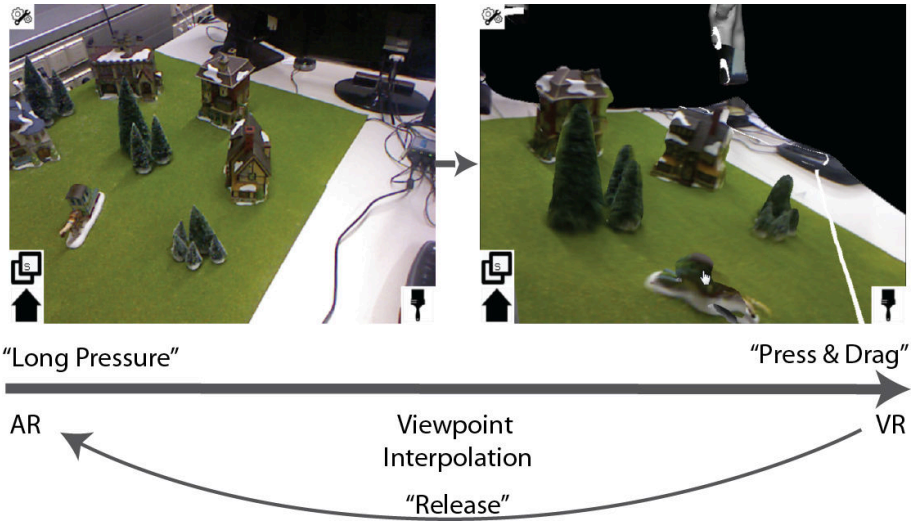


Figure 5.20: Spring Loaded Navigation. A user may want to switch to areas that are not visible from the current AR view. We provide a spring loaded navigation technique that allows a user to rotate around an object directly from the AR view. When the user stops interacting the technique switches back to AR.

Zooming. When showing AR applications on mobile phones, we noticed that users

regularly tried to zoom into the AR view to look at details of the real world scene or the augmentation. A naive implementation might digitally zoom the video image and adjust the augmentations accordingly. However, at higher zoom levels, the quality of the video image degrades. To avoid this decrease in quality, we developed a transitional zooming technique that gradually replaces the video with the virtual representation, as the user zooms in on the model. The virtual geometry is able to provide more detailed close-up views than simple digital zooming (Figure 5.19). The virtual geometry also allows users to zoom out of the AR view to get an overview of the scene from a larger distance. While being in a zoomed view, the system continues tracking as in the AR view. We use the tracked camera pose to directly control the viewpoint of the virtual camera so that the user can naturally explore the scene by moving the mobile device in the real world reference frame.

The zooming is situated in the continuum between the AR view and the VR view. When zooming, we gradually blend to the other view mode when a certain threshold is reached. We indicate this threshold by adding a fading virtual grid before starting to blend to the other view mode. When virtual objects are closer than the blending threshold, we switch to the virtual model before this threshold to avoid zooming through this object. Zooming can be controlled with a swipe gesture, or on a touch screen with a pinch gesture. The steps of the technique are shown in Figure 5.18.

Spring-Loaded Navigation. The context-aware transitional techniques allow a user to quickly navigate to virtual viewpoints of those parts of the scene that are visible from the current viewpoint. To investigate the occluded areas, the user either must change the physical AR viewpoint or manipulate the virtual camera after switching to a VR viewpoint. To avoid these detours, we introduce a spring-loaded navigation technique that enables users to quickly change to a virtual viewpoint of the occluded areas (Figure 5.20).

The spring loaded navigation allows the user to investigate invisible areas by rotating the virtual model, while still being in the AR view. To activate the spring loaded navigation, the user taps and holds a location on the screen, which triggers a transition to the VR view. The user can then drag the view to initiate a rotation around the pressed region. To be able to rotate around an object, we allow users to interrupt the dragging to reposition the input. When the user does not interact with the screen for a certain amount of time, the viewpoint transitions back to the current AR view.

5.3 Multi-perspective Rendering

The previously presented transitional techniques expand the egocentric viewpoint of the user by providing a virtual copy of the real world object. In this Section, we present the alternative method of multi-perspective renderings that also expand the viewpoint. In constrast to transtitional interfaces, multi-perspective renderings typically do not allow for free viewpoint exploration, but integrate the additional viewpoints into the current view of the user.

We present a method to create and add renderings of secondary viewpoints to an object. We demonstrate that the method can reveal otherwise occluded parts of the object or parts that are small in scale, thus providing an overview (see Section 5.3.1). Furthermore, we presented embedded views that expand the viewpoint of the user by allowing views around the corner of buildings (see Section 5.3.2).

5.3.1 Secondary Viewpoints of Objects

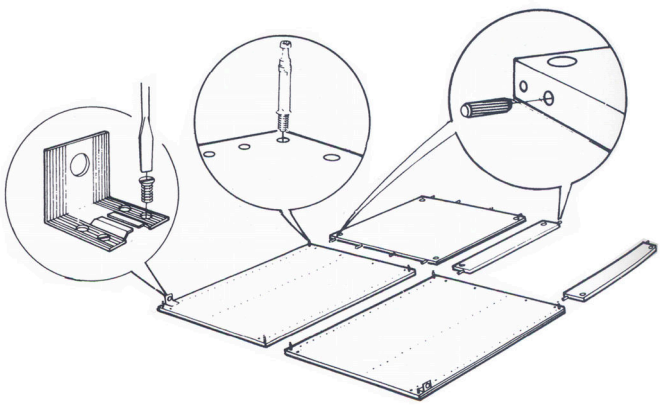


Figure 5.21: Handmade secondary views. Small parts do not explode in the main view onto the explosion. Instead, they have been exploded in an additional presentation shown as label (Image adapted from [94]. © IKEA.)

Illustrators often make use of secondary views to reveal more information in a drawing. In Figure 5.21, the illustrator decided to add secondary views to zoom instructions of the manual. Using this method, a viewer not only gets an overview of the assembly and the location of the assembly instructions, but also gets detailed instructions that are visualized in the secondary views. Inspired by hand-drawn illustrations, we create secondary views that we use to increase the overview of compact explosion diagrams that

have been discussed in Section 3.1.4.

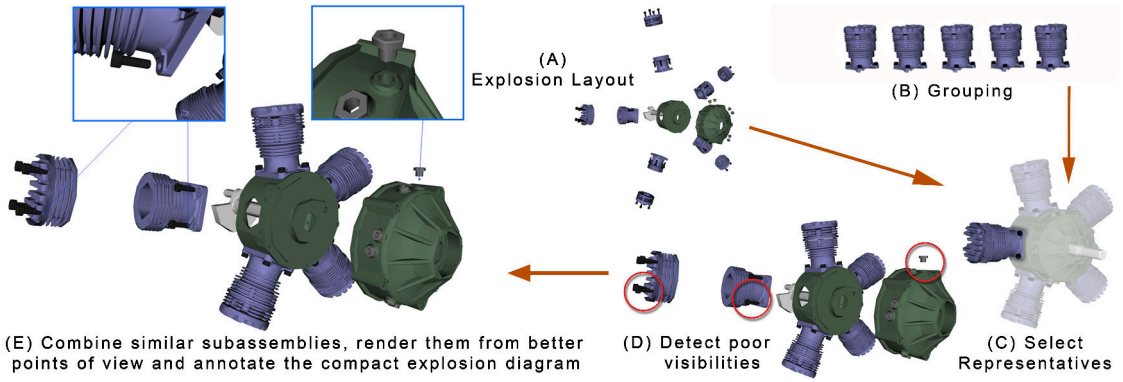


Figure 5.22: Secondary views and compact explosion diagrams. We combine the creation of secondary views with compact explosion diagrams to increase the overview of the exploded parts. Using secondary views, we can visualize occluded parts or items that are small in scale. To reduce the number of annotated viewpoints, we reuse annotations for repeating items such as the black screws.

Even though the optimization process of compact explosion diagrams selects the best combination of representatives, some of the subassemblies may still be presented in a very small scale or highly occluded. We compensate for these problems by rendering poor explosions of subassemblies from a more suitable point of view. The renderings from secondary points of view allow to zoom small parts as well as to resolve occlusions, which appear from the main point of view.

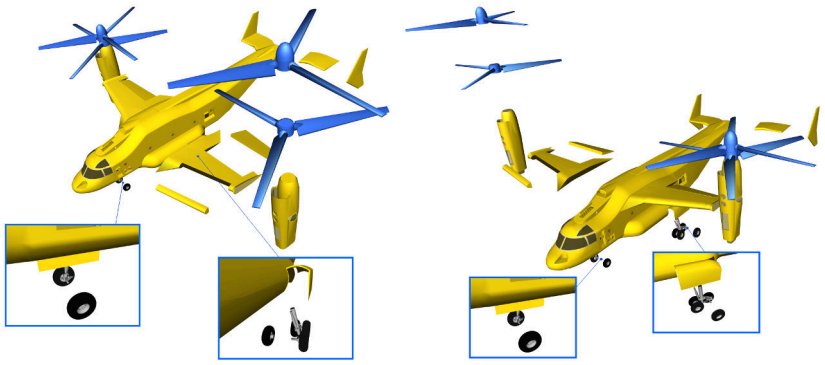
In the following, we describe the integration of the creation process of secondary views into the optimization of compact explosion diagram.

Detecting Poor Explosions of Subassemblies. Our system is able to determine poorly presented parts of the explosion diagram by analyzing the final combination of representatives. The system evaluates each quality parameter of a representative individually and creates renderings from a secondary point of view, if one of them falls below an adjustable threshold. Since the footprint of the unexploded elements can be neglected for a rendering from a secondary point of view, we scale down the impact of this parameter by lowering its threshold to the minimum. However, even though the detection of poor explosions on the final rendering allows us to increase the effectiveness of the compact explosion diagram, we select poor elements of the representation independent of representatives. In consequence, we do not generate an optimal presentation with respect to the visibility of representatives.

Our system detects poorly presented parts during the selection of representatives and integrates the identification of candidates for a secondary rendering into the overall layout optimization process. In each iteration of the algorithm, which evaluates a new combination of representatives, our system analyzes the visibility and the projected size of the explosion of every single subassembly. If any of the evaluated parameters falls below an adjustable threshold, we exclude it from the quality calculation of the current combination of representatives. This strategy results in a quality value, which represents only the relevant parts of the explosion diagram, but not those which will be presented from a more suitable point of view in a later stage in the rendering pipeline.

By integrating the selection of poorly visibly explosions of subassemblies into the combination of representatives, we exclude poorly presented subassemblies from the layout evaluation. Consequently, the final combination will be better for the representatives which are not presented from a secondary point of view. Another advantage of this approach is that it allows us to control the number of secondary points of view and, thus, to avoid clutter due to an excessive number of insets. However, the downside of this approach is that the visibility of the already poorly presented subassemblies may become even worse (Figure 5.23(a)). Mentally relating secondary points of view for such cases may become very difficult, especially if the subassembly is completely occluded in the compact explosion diagram from the main point of view. Consequently, the system is also able to analyze already optimized layouts for poorly represented parts (Figure 5.23(b)). Even though the combination of representative subassemblies may not be perfect, if the visibility of all parts of the assembly is taken into account, the resulting presentation will increase the capability of mentally linking the exploded view and the additional renderings. Therefore, multi-perspective renderings will be supported best if poorly presented parts are detected after the optimization of representatives is finished.

Viewpoint Restriction and Linking of Multi-Perspectives. In order to present the renderings from secondary viewpoints as close as possible to their location in the compact explosion diagram, we place them as annotations into the main explosion diagram. However, by spatially separating the presentations from different points of view, we require the user to put some effort into mentally linking the content of our renderings. To assist the user in this task, the layout of subassemblies shown in additional views is only allowed to change if it is completely occluded in the main view onto the explosion. Otherwise, the layout visible in the secondary view will differ from the one in the main presentation, which makes it difficult to mentally relate these structures to one another.

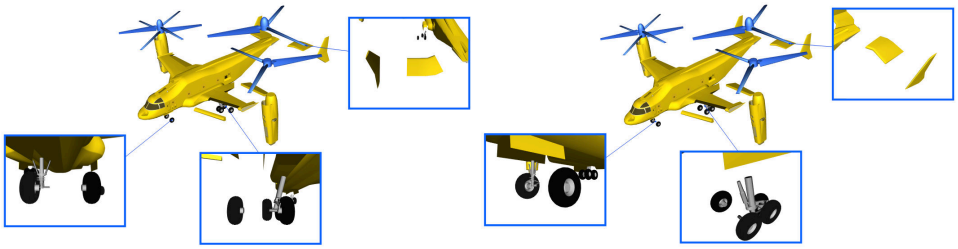


(a) (b)

Figure 5.23: Multi-perspective rendering as a post-process versus integrated multi-perspective rendering. (a) Poorly presented explosions may become even worse, if they have been removed from the layout optimization. Notice the occluded wheels and the resulting lack of context to mentally link the annotation to the main representation. (b) By optimizing the combination of all representatives before rendering from secondary points of view, chances are better to provide enough contextual information to mentally link the content in the secondary with the one in the main rendering.

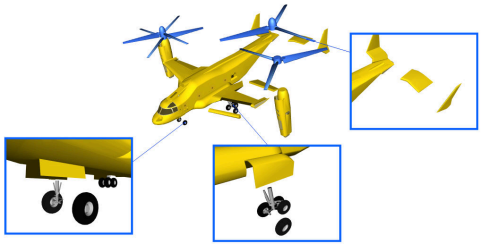
In addition, we restrict the offset between the secondary viewpoint and the main viewpoint to an adjustable threshold. Otherwise, the difference between the secondary viewpoint and the main viewpoint may lead to presentations, which are difficult to read. In figure 5.24(a), the secondary point of view is offset by more than 90 degree to the main point of view. In case of the rear landing gears, the difference reaches nearly 180 degrees. The mental linking between the views may become difficult, if the points of view have been offset too far. Therefore, we restrict secondary viewpoint to vary only within a certain range to the main viewpoint (Figure 5.24(b)).

To compute a secondary point of view, we do not only include the subassembly itself, but also consider its contextual information. Otherwise, our rendering may not show any information besides the subassembly, which may also influence the ability to relate the renderings to one another (Figure 5.24(c)). Additional parts can be forced into the view by adding weight to the measure describing the visibility of the rest. However, since these additional parts may increase visual clutter, we introduce a new parameter to the optimization which controls the amount of presented contextual elements. Only those parts are considered to be contextual information, which are in direct contact with the representative subassembly. We measure the amount of contextual information by using



(a)

(b)



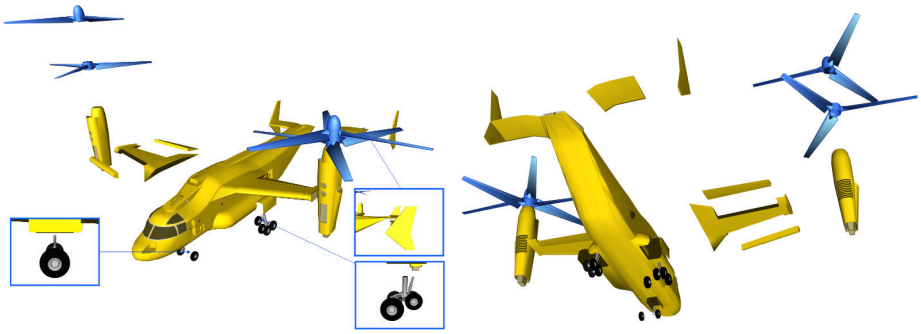
(c)

Figure 5.24: Linking multi-perspective renderings to their original location. (a) The views depicted in the annotations do not correspond with the view on the layout. This is irritating, and mentally linking of annotations and the corresponding subassemblies may be difficult. (b) The secondary view is restricted to stay close to the main point of view. The annotations are less confusing, but lack contextual information. (c) Adding contextual information further supports mentally linking the content of annotations to their counterparts in the main view.

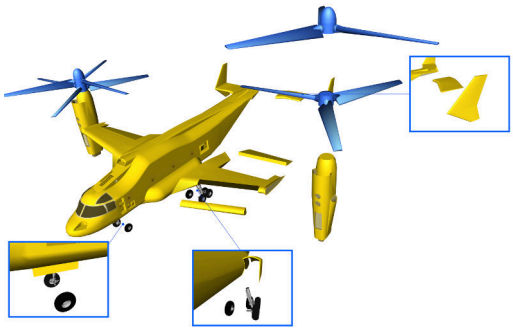
the size of its 2D projection, which we force to be within a certain distance to an optimal value.

Equation 5.1 describes the contextual quality measure, which is based on the distance from the optimal amount of contextual information. The difference between the threshold value $contextTh$ and the normalized amount of pixel showing contextual elements ($contextPixel$), ideally, must be close to zero. We chose a threshold value of approximately 0.33, which scores points of view highest, if a third of the corresponding rendering is covered by contextual information. Such a presentation conforms with the rule of thirds, which is, according to Gooch et al. [48], the best known rule of layout.

$$contextQuality = (1 - |contextTh - contextPixel|) \tag{5.1}$$



(a) (b)



(c)

Figure 5.25: Viewpoint optimization. (a) Ineffective representative explosions due to arbitrary viewpoint selection. (b) Even though the quality parameters of all representatives have been taken into account, an inappropriate view on the explosion diagram may be chosen. (c) Front facing viewpoints have been weighted higher to prefer renderings of frontal views.

To ensure an unobstructed view onto the explosion of the subassembly from a secondary point of view, we have to put a higher emphasis on the visibility as well as the direction of the explosion. Otherwise, close objects may occlude parts of the representative, or representatives explode into to the viewing direction, making the secondary point of view less valuable.

Compact explosion diagrams which consist of a large number of small subassemblies may lead to a cluttered presentation of an equally large number of annotations. To make efficient use of the available space, we reduce the number of annotations by combining similar ones into a single annotation (Figure 5.22). However, even if we combine certain subassemblies within a single secondary presentation, the amount of annotations is still

unpredictable. Therefore, we assign importance values based on the visibility of annotated parts. This allows our system to select the most important annotations until the available screen space is filled.

Viewpoint Optimization. The system described so far optimizes a manually selected viewpoint. To further automate the generation of compact explosion diagrams, we optimize its main viewpoint as well. To render from a proper point of view, we first compute the optimal layout for different viewpoints, before we select the one with the highest score. Similar to previous approaches, we select a set of candidate viewpoints by sampling the bounding sphere of the object of interest [48, 120]. The orientations are derived for each candidate viewpoint by pointing the camera to the center of the bounding sphere. An adjustable threshold determines the number of equidistant sample points on the sphere.

Good viewpoints maximize the quality of combined representative explosions, just as bad viewpoints may result in incomprehensible presentations, even after optimizing the layout according to the set quality parameters. For example, the chosen secondary point of view onto the wheels of the airplane in Figure 5.25(a) does not show the explosion direction. Furthermore, all elements except the exploded wheel are occluded. However, the layout is optimal for this viewpoint.

The quality of viewpoints is evaluated using the parameters presented in section 3.1.4.4. By performing the optimization of representatives for all viewpoints on the bounding sphere and selecting the viewpoint with the highest score, the system selects the best viewpoint combined with the best representative explosion. Notice the different representations in the annotations of the front wheels in Figure 5.25(a) and 5.25(c).

However, even though this algorithm allows us to represent the explosions from an optimal point of view, with respect to the quality parameter of its explosions, the object itself may not be sufficiently represented from this viewpoint. For example, Figure 5.25(b) shows the view with the highest score on the exploded representatives. However, the view from the bottom does not provide a good view on the airplane itself. Such defective viewpoints were studied by Blanz et al. [18]. They found out that users select viewpoints which maintain the natural up-orientation of the object, while simultaneously avoiding occlusions. In addition, rather low diagonal views were often preferred, showing objects from familiar positions which contain as much information as possible. Based on this data, we allow users to influence the allowed viewpoint selection by weighting a certain range of viewpoints higher, than others. Using this restriction, we select the viewpoint with the highest quality value, while simultaneously clearly presenting the object of interest (Figure

5.25(c)).

5.3.2 Embedded Views in Real World Environments.

A common application of AR is to support users in navigation tasks by overlaying navigation aids on the real world. Compared to a traditional map view, the user does not need to mentally switch anymore between an exocentric map view and a personal view of the real world, because the desired route is indicated directly in the user's current video-based view. Additionally, this approach provides more context and is easier to use than GPS navigation systems, since the real view of the world is presented to the user instead of a virtually simplistic representation used in GPS systems.

Even though the presentation of navigation aids is improved when overlaying them on the current view of the user, the inherent egocentric viewpoint of mobile AR enforces a strong limitation on the field of view of the user and the range of viewpoint locations. The user has limited range of information, due to the limited content that can be presented in the AR egocentric viewpoint. Compared to a map overview, users can only see route changes up to the next visible turn, because occlusions block the view on the navigation aids. Combining the AR navigation aid with a map overview, shown on a part of the screen, allows users to investigate the route lying further ahead, but at the cost of occupying some of the already limited screen-space of mobile devices. Furthermore, adding the map as a second spatial representation of the world reintroduces the cognitive effort of switching between different views.

We present embedded virtual views, which provide additional, spatially registered and virtual views on upcoming changes of the route. As presented in Figure 5.27(b), the embedded view naturally extends the egocentric viewpoint of the user and shows the otherwise occluded navigation aid in the real world context (Figure 5.27(a)). The technique is intended to support pedestrians using their mobile handheld device for navigation.

Typically, users are navigating turn-based and do not walk with the device always extended in front of them. They orient themselves at key locations, like crossings, using the provided navigation aid, and walk to the next key location. We believe that the presented technique can reduce the time required for orientation and make users more confident during navigation, because the look-ahead view allows them to see more than only the next route change and also provides visual context for the navigation aid.

The example images were created in a table-top, real world test bed consisting of paper models and their virtual counterparts. A virtual navigation aid shows a path through the

model (Figure 5.26). The images were recorded from a handheld camera moving through the scene.

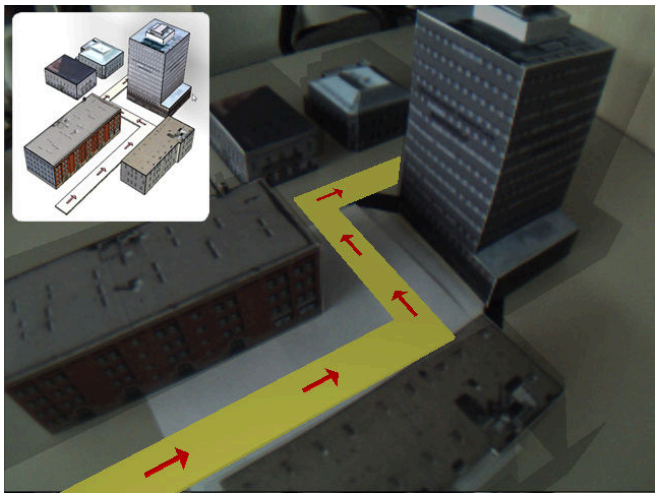
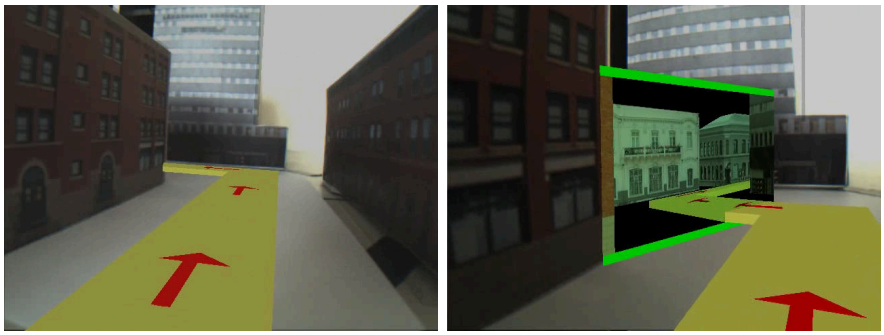


Figure 5.26: The test bed used for testing the different embedded views. The models are made from paper. The small image shows the corresponding virtual geometry.



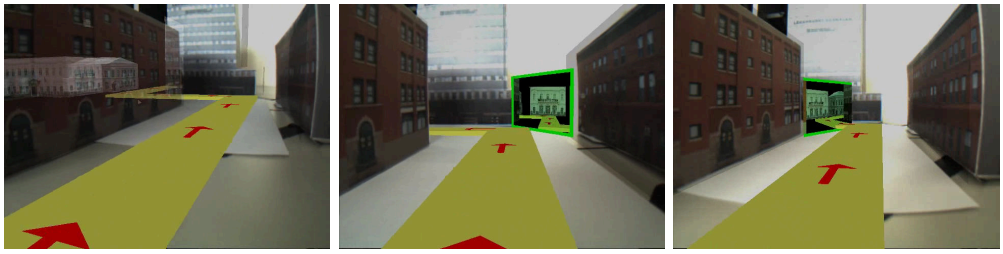
(a) (b)

Figure 5.27: An embedded virtual view allows users to follow a guided route, without leaving the egocentric viewpoint. (a) The navigation aid turns around the corner and the user's view is blocked by the building. (b) By extending the egocentric viewpoint with an additional embedded virtual view, the user is still able to perceive the route indicated by the navigation aid. Note that the additional view embeds the navigation aid correctly in the environment.

An embedded virtual view is used to reveal an AR navigation aid, which is occluded by real world structures, such as buildings in urban environments, lanes of houses in residential area or hills in rural areas. For this purpose, the view is spatially registered

with the world and placed at the location where the navigation aid is occluded by real world structures. The navigation aid naturally passes into this view and provides the user with a view on otherwise occluded upcoming route changes (Figure 5.27(b)).

An embedded virtual view should seamlessly integrate with the surrounding environment. The integration is already indicated in the example image, where the left and right borders are left open so that the content of the embedded view merges with the respective scene elements, such as the red building on the left. In Figure 5.27(b), the embedded view is projected onto a rectangular portal surface. However, bent portals such as shown in Figure 5.29 are also possible, which increase the field of view of the embedded image and thus reveal more of the occluded scene.



(a) (b) (c)

Figure 5.28: Comparison of egocentric occlusion management techniques for revealing a navigation aid. (a) Ghosting, a typical x-ray vision technique, makes the occluding building transparent. The route augmentation and the context it is embedded into are revealed at the cost of screen-space. (b) A mirror also reveals route and context, and saves screen-space. However, mentally linking the mirror image and the world may be hard. (c) An integrated image reveals the occluded information and its context, and also uses less screen-space than the ghosting technique. Compared to the mirror, the mental linking between the view and the real world may be easier.

Additional Virtual Views. Aside from the described embedded views, our system also supports two simple mirror techniques, which we implemented for comparison. The first one places a ***mirror*** at the location, where the navigation aid is occluded. The approach is similar to common traffic mirrors, which allow car drivers to see otherwise invisible parts of the street (Figure 5.28(b)). We decided to increase the size of the mirror compared to a traffic mirror, so that users can already recognize structures from afar. The image shown in the mirror is view dependent, and thus changes with the location of the user. To always show the same view around the corner we created a variation of the real mirror, which

always provides the same view around the corner. Hence, in this ***fake mirror*** the view stays static, even if the user changes location. The fake mirror is similar to the video mirror presented by Au et al [4].

Comparison of Virtual Views. In the following, we compare an embedded view with the fake mirror view and with ghosting, which is a traditional occlusion management technique. Aside from the view-dependency, real mirrors have the same properties as fake mirrors and, therefore, are not part of the comparison. Consider Figure 5.27(a), where an artificial navigation aid indicates a path for the user. The path turns around the corner and is occluded by a building. Using ghosting (Figure 5.28(a)), the occluder is removed and the route is revealed. To enhance the integration of the navigation aid in the environment, additional buildings along the path are also shown. This approach removes a large portion of the occluding building and requires half of the screen-space.

An artificial mirror (Figure 5.28(b)) uses less screen-space and allows the user to see around the occluder, but users may not be able to mentally link the revealed content and the occluded data, because the view is separated from the scene. Furthermore, the mirror image is view-dependent and changes with the position of the user. Exchanging the mirror against the fake mirror technique would remove the view-dependence, but still requires effort to mentally link the content with the real world. Using an embedded virtual view, the user can look around the occluding building (Figure 5.28(c)). Like the mirrors, the view resolves the occlusion and, at the same time, saves screen-space. The integrated image is smaller than the structures revealed with the ghosting technique, while still showing the navigation aid and buildings along the path. However, we believe that the mental linking of the shown content with the real world is easier than with virtual mirrors.

Implementation. We use CUDA ray tracing to create the virtual views in real-time. Ray tracing allowed us to quickly prototype and test the different approaches. The virtual mirrors were created by simple reflecting the rays at the mirror surfaces. The embedded view is achieved by bending the viewing rays of the camera at a portal placed at the corner of the building, similar to the technique presented in Cui et al. [30].

Rays are emitted from the camera center located at the location of the user. When they intersect the portal geometry, they are redirected using the surface normal at the intersection, and re-emitted through the portal into the virtual representation of the world.

5.4 Conclusion and Future Work

We presented transitional interfaces and multi-perspective renderings that allow users to see more than what is visible from a single AR view. Note that, while we always discussed these approaches separately, transitional interfaces and multi-perspective renderings are not mutually exclusive. They can easily be combined into one interface. For instance, consider the 2DSEP interface of the OCE techniques (Figure 5.2(d)). In the 2DSEP interface, a ring of images surrounds the virtual copy of the real world building that the user wants to explore. This ring is a multi-perspective rendering that provides an overview of the building. Participants that used the interface during the evaluation could employ this ring to quickly navigate to viewpoints containing the target element.

For real world buildings, such a ring of images can be created using internet photo collections. In fact, our method to create compact visualizations of photo-collections can be used to select appropriate images from different viewpoints of real world buildings (see Section 3.1.3). For a tabletop environments, a textured copy of a real world scene can be created using an online reconstruction system, such as the one presented in Section 5.2.3. The reconstruction can be used as input to render the additional viewpoints for the image ring visualization. This system can also be used to render the secondary views we discussed at the example of compact explosion diagrams.

Aside from using images to provide an overview of objects, we also presented a prototype system for creating embedded views for navigation, which allow users to look around occluding structures by locally bending the viewing rays of the camera. We demonstrated its usage as navigation aid to allow users to peak on upcoming changes of the route.

The approach currently requires spatial knowledge data provided by a virtual model of the occluded scene. Although Google Earth[3] and Bing Maps[4] contain collections of 3D models of certain urban areas, in general, such dense 3D models are not yet available. Therefore, we want to explore other input sources, such as geo-located photos, which can be drawn from online photo collections. Another source of images are databases such as Google Streetview[5] or Bing Streetside[6] that cover the street level of almost all major cities.

The current embedded views do not provide an efficient visual registration between video camera and virtual content. Hence, the views do not merge seamlessly with the

[3] https://earth.google.com
[4] http://www.bing.com/maps/
[5] https://www.google.com/maps/views/streetview/
[6] http://www.microsoft.com/maps/streetside.aspx

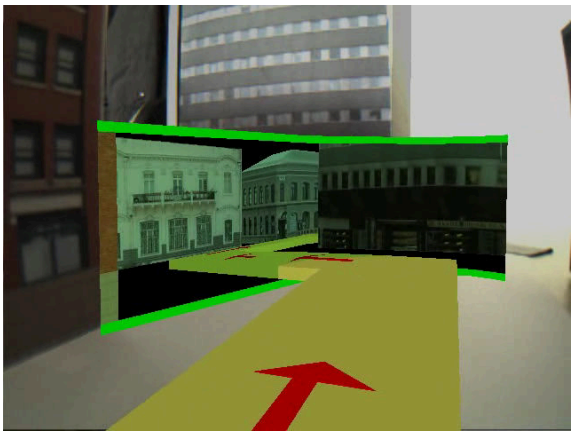


Figure 5.29: The embedded view is not limited to rectangular surfaces. This view uses a bent shape to provide an increased field of view on the occluded regions.

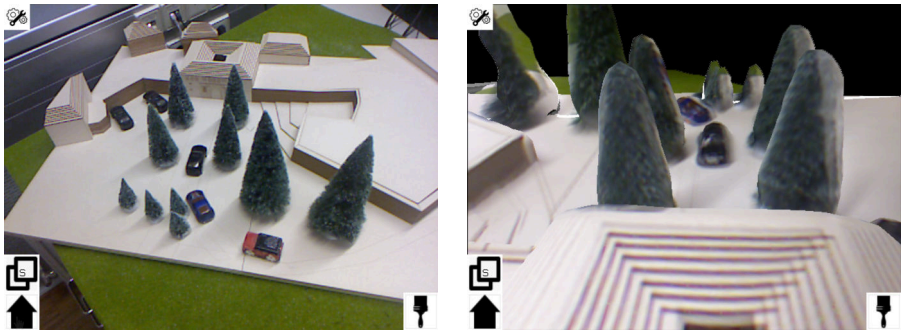
scene. We want to investigate how we can provide seamless visual registration between the video image and the virtual content using more advanced rendering techniques and image analysis algorithms. The integration of the image is already indicated in Figures 5.27(b), 5.28(c) and 5.29. Note that the embedded view shows part of the red building in the left region.

Similar to the ring of images of the OCE techniques, the embedded views can also be used as starting point for a transitional interface. For instance, they allow users to select a POIs around the corner, or to engage into a Google Streetview experience. Multi-perspective renderings can also be used to access objects that require zooming, because they are either too close or too distant from the user's position. Our current OCE designs only considered the ideal object distance to the real world object. In such situations, it may be feasible to use multi-perspective renderings to zoom distant objects [58], or to provide a map view that is integrated into the real world [35].

Combinations of multi-perspective rendering and transitional interfaces may require specialized transitions to connect the views. In our evaluations, we use OCE interfaces that provide a straightforward seamless transition to zoom a virtual copy of a real world object. However, we should also consider using smart transitions that use semantic information to propose strategically chosen VR viewpoints, which are relevant for the current task of the user.

Smart transitions also enable a wide variety of new application scenarios in AR, such as more immersive AR games. A user can switch to an egocentric view of the game, or follow virtual characters that otherwise would disappear behind geometry. Furthermore,

new game designs can make use of a combination of AR and VR views and the transitions between them.



(a) (b)

Figure 5.30: Example Urban Planning. (a) In an urban planning application users rearrange the scene to identify a good space design. (b) Using our system, users can achieve egocentric viewpoints of the scene and visit views that would otherwise be unreachable.

Other application areas are architecture and urban planning. For instance, in a tabletop scenario, users can rely on tangible objects to perform architectural planning. Using transitional interfaces combined with smart transitions, they can switch to a virtual view showing the planned areas from an egocentric virtual viewpoint. This allows users to verify the placement of buildings and detect undesired occlusions from certain viewpoints. As shown in Figure 5.30(a), users might rearrange the trees in the architectural model of a public space to determine a good placement for enhancing the appeal of the space. Using our techniques, they can verify the result from an egocentric perspective. They can also move to viewpoints otherwise blocked by the real environment. In Figure 5.30(b), the user switched to the top of the roof looking down the alley from the direction opposite to the AR view, a viewpoint that would be blocked by the table. In future work, we will evaluate this type of techniques in the context of gaming and urban planning.

The experiences and feedback gathered from our evaluations of transitional OCE techniques make us confident that such interfaces are feasible and practical in real world conditions. Future work will investigate the interfaces in real world conditions in more depth.

Chapter 6

Conclusion

Contents

This thesis presented research performed in the context of situated visualization. We developed filter and view management techniques that handle data overload for different data types and investigated the aspect of temporal coherence. Our filter approach avoids data overload by either removing redundant items, or by building a data hierarchy by aggregating items into clusters and selecting items from the hierarchy. Furthermore, we presented interaction and visualization techniques that compensate for the inherent egocentric viewpoint of users. This last chapter reflects on the presented research and also tries to give a general outlook on situated visualization. Note that, while we performed our research in the context of handheld AR, the presented methods can also be applied to other AR platforms, such as an HMD.

In the following, a summary of the presented work is provided, followed by final remarks about lessons learned and future research directions.

6.1 Summary

Combining Filtering and View Management. The combination of filtering and view management for AR applications allows developers to present overview visualization of input data to the user, without losing information. Compact visualizations achieve this goal for different data types by using an optimization algorithm that removes redundant information from the visualization (see Section 3.1). First, redundancies are detected

and grouped, then an optimization algorithm selects one representative item to represent each group. The presentation of compact visualizations can be refined further by utilizing hierarchical structures in the input data (see Section 3.2.2). Such hierarchical structures basically allow the algorithm to remove redundancies to make the presentation even more compact.

If no redundant data is available, clustering algorithms can be used to create groups of similar items (see Section 3.2.1). Again, a representative item must be chosen or created by averaging over all items of a group. However, in contrast to compact visualizations, showing only one representative item would remove the information about other similar items of the group. Therefore, users must interact with groups to explore the details of the data. At the same time, the screen should not become too cluttered when the user continuously explores the shown groups. We presented an adaptive information density approach, which creates a hierarchical representation of the input data by grouping its items and then selects groups from different levels of the hierarchy, based on the currently available screen space and the potentially created clutter.

Temporal Coherence. AR applications are highly interactive. Even though users maybe cannot interact with the data itself, view management algorithms often update the layout of the presented information, when the user changes the AR viewpoint. Therefore, we investigated several methods to achieve temporally coherent layouts during viewpoints changes.

We investigated two different approaches. First, the filtering step of an application can be influenced to achieve better temporal coherence of the view management algorithm. Second, we improved temporal coherency by explicitly reducing the degrees of freedom of the used view management algorithm.

We demonstrated the first approach within our compact visualization framework, which optimizes the selection of items with respect to temporal coherency (see Section 4.1). Additionally, we applied the same paradigm to the presented adaptive information density algorithm. The algorithm avoids presenting an excessive amount of items on the screen by selecting items from a hierarchical data structure, based on user preferences and the available screen space. This reduction of the number of items reduces the amount of conflicts between the presented items and can improve temporal coherency.

We developed the hedgehog labeling algorithm as part of the second approach (see Section 4.2). Instead of simulating annotations in 2D screen space, hedgehog labeling treats annotations as 3D objects. We improved the temporal coherence by reducing the degrees

of freedom of the annotation, so that it moves along a single line in 3D space, or a plane placed in 3D space. Furthermore, treating annotations as 3D objects allowed us to stop the movement of annotations after the creation of the initial layout. We demonstrated a similar freezing of 2D annotations, when discussing temporal coherency for compact visualizations of annotations. However, freezing annotations described as 3D objects proved to be simpler and more effective, which we could show in a quantitative user study.

In this study, we compared the performance of different view management approaches (see Section 4.3). Participants performed a task, in which they had to locate the same labels multiple times. Our plane-based 3D hedgehog label approach outperformed the other view management systems. The strongest difference between the hedgehog system and these other approaches was that the labels, once the layout was calculated, were frozen in place in the 3D space relative to the object. The other approaches aligned the labels with the screen, thereby lacking this fixated spatial 3D representation of labels. Similar to hedgehog labeling, one approach preserved the order of labels in screen-space. However, hedgehog labeling also outperformed this approach. This strongly indicates that layout constraints do not have to be constantly enforced and that the spatial registration of labels in 3D relative to the object improves the ability of users to keep track of label locations and to interact with the labels.

Extending the Ego-centric Viewpoint. AR is inherently limited to a single ego-centric viewpoint. In order to expand this ego-centric viewpoint, we presented Object-Centric Exploration (OCE) techniques that allow users to investigate a virtual copy of a real world object, instead of physically moving around an object (see Section 5.1). OCE techniques are transitional interfaces that seamlessly switch from the AR view to the virtual view of an object. To improve the transition to VR, we proposed smart transitions that take semantic information of the scene into account when placing the virtual camera and selecting the mode of operation of the camera (see Section 5.2). For instance, based on the current task of the user and the scene structure, the system may decide to place the virtual camera to achieve a close-up view of an object. When the user operates the virtual camera, it moves around the object.

We also explored multi-perspective renderings as means to compensate for the ego-centric viewpoint of AR (see Section 5.3). A multi-perspective rendering shows additional viewpoint of the scene by integrating these viewpoints into the current view of a user. We demonstrated this approach as part of the OCE techniques, where a circular arrangement of images around an object provides an overview of this object. In addition, we included

secondary viewpoints into compact visualizations to show viewpoints of objects, which would otherwise be too small or occluded by other objects. Furthermore, we presented a navigational application that allows users to peek around corners to follow the proposed route.

6.2 Final Remarks

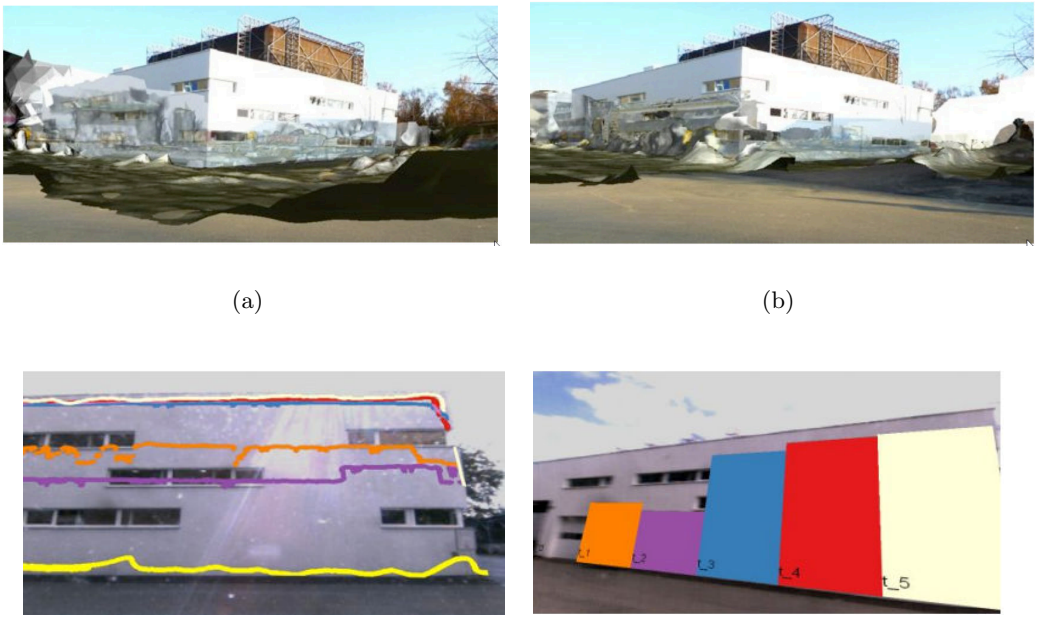
Combine methods. Although, the techniques were developed separately over the course of the research effort, they complement each other. For instance, we demonstrated that compact visualizations can be supported by additional multi-perspective renderings such as secondary views. This allows users to also get an overview of occluded data. Instead of moving the AR viewpoint to reveal the occluded data, a transitional interface can provide smart transitions to zoom the data. A tourist who investigates different views of a large real world building would benefit greatly from such a combination. Hence, the combination of the presented techniques can create a potentially very powerful exploration tool for situated visualization.

Rethink known approaches. Care must be taken when designing situated visualizations for AR. Many conventions of visualization were developed for static images or for a desktop information space and do not take the dynamics of AR into account. For instance, we initially used a standard view management technique that worked in 2D image space for creating layouts of external labels. However, this caused issues with temporal coherence. In addition, the registration of the labels relative to the object was not stable, because the labels were floating in image space. To remedy these issues, we developed a constrained 3D view management approach that sticks label to the 3D object. This allows us to handle the dynamic and frequent viewpoint switches of AR applications. Hence, while information visualization provides many solutions for issues that also occur in AR, they have to be carefully considered when applied to an AR context.

Always register in 3D. Considering our experience with 3D view management, it might be best to always register augmentations of situated visualization directly in 3D space. Consequently, the underlying algorithms and visualizations must be developed to work in such a 3D space. Registering a visualization in 3D also allows visualization designers and users to use perspective distortions to convey information about the spatial arrangement of the data. For instance, we make use of perspective distortions, when we

freeze the planes in our 3D view management approach. Changing the viewpoint creates a parallax effect that conveys information about the arrangement of annotations.

In addition, registering labels in 3D space allows designers and developers to streamline their interaction methods. Interaction methods that work with 3D objects can directly be applied to 3D labels, without a separate solution, which works with 2D representations. Aside from simple manipulations such as translation or scaling, also more complex interactions can be facilitated. For instance, the transitional interfaces presented in this thesis can not only be applied to a virtual copy of a real world object, but can also easily include any associated 3D labels with the same operation. Annotations in 3D space are also beneficial for collaborative tasks. Using 3D labels, users can not only share the location of the annotated object in 3D, but also the location of the corresponding 3D annotation as well.



(a) (b)

(c) (d)

Figure 6.1: Using abstraction to visualize information. (a) and (b) show two reconstruction of a construction site at different points in time. Presenting only the 3D data does not easily allow a viewer to draw conclusions about the progress. In (c) and (d) the 3D geometry is abstracted to 2D diagrams. The abstraction reduces the complexity of the visualization and also reduces clutter (Images taken from Zollmann et al. [153]).

Explore additional visualizations. In this thesis, we used different types of visualizations such as explosion diagrams and textual and pictorial annotations. While annotations are a straightforward visualization for a Point-of-Interest (POI), more complex data such as measurements of physical properties or statistical data may require different visualizations. In handmade illustrations and, also, in information visualization, complex data is often visualized using abstractions. Abstractions can simplify the presentation to easily understandable concepts and reduce the amount of clutter [37].

Although AR can benefit from such abstractions, they have hardly been explored. In AR, White et al. [146] use differently sized spatially registered spheres to visualize air pollution measurements. More recently, Zollmann et al. [153] visualized the progress of a building construction site over time by abstracting the 3D geometry of the building (Figure 6.1). Note that, although the diagrams correspond to standard techniques for visualizing time-varying data, the created visualization is adapted to the needs of AR and registered to the corresponding real world object.

What is a good visualization? Even though the research on computer-generated visualizations exists now for more than two decades, it is still not clear what comprises an effective visualization. Throughout the literature, a main motivation is the avoidance of clutter [37, 38]. However, there are no standardized measures or procedures to assess the effectiveness of visualizations [152]. This issue is even more complex in AR, where the dynamics of the real world influence the effectiveness of the visualization. For instance, the impact of a visualization can constantly change, because text annotations are not readable anymore, when placed over the wrong background [82]. It is also not clear which areas are most suitable for placing the annotations in the field of view of the user. Formal evaluations of this aspect only were performed recently [101].

Standardize evaluation. Evaluating AR visualizations in real world environments is very challenging. For instance, due to the dynamics of outdoor environments, it is almost impossible to run controlled experiments. Therefore, we performed a major part of the evaluation of OCE techniques in a controlled laboratory environment. AR visualization and user interface research would greatly benefit from standardized methods for simulating AR test environments. This has become an active area of research in the last years [11, 80, 81].

Additional open topics. The research area of situated visualization covers many aspects (see Section 1.4). While we provided solutions for some of the discussed challenges, we did not investigate others. For instance, we did not investigate the area of visual coherence. Aside from rendering correct geometric occlusions, we did not use any other coherent rendering methods, such as global illumination estimations [54]. Furthermore, we did not consider the handling of registration errors in our visualizations [28]. We are convinced that successful situated visualization that also works under different AR conditions must combine solutions from all these areas.

Appendix A

Acronyms

List of Acronyms

AR	Augmented Reality
DOI	Degree of Interest
GPS	Global Positioning System
HMD	Head-Mounted Display
IBR	Image-based Rendering
LOD	Level-of-Detail
MR	Mixed Reality
OCE	Object-Centric Exploration
OOI	object-of-interest
PDA	Personal Digital Assistant
POI	Point-of-Interest
POIs	Points-of-Interest
SLAM	Simultaneous Localization and Mapping
VR	Virtual Reality
WIM	World-in-Miniature

Bibliography

[1] Agrawala, M., Phan, D., Heiser, J., Haymaker, J., Klingner, J., Hanrahan, P., and Tversky, B. (2003). Designing effective step-by-step assembly instructions. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2003*, 22(3):828–837.

[2] Ahn, J. and Freeman, H. (1983). A program for automatic name placement. In *AUTOCARTO 6*, pages 444–455.

[3] Ali, K., Hartmann, K., and Strothotte, T. (2005). Label Layout for Interactive 3D Illustrations. *JOURNAL OF THE WSCG*, 13:2005.

[4] Au, C. E., Ng, V., and Clark, J. J. (2011). MirrorMap: Augmenting 2D Mobile Maps with Virtual Mirrors. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 255–264.

[5] Avery, B., Sandor, C., and Thomas, B. H. (2009). Improving Spatial Perception for Augmented Reality X-Ray Vision. In *Virtual Reality (VR) Conference*, pages 79–82. IEEE.

[6] Azuma, R. (1997). A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6:355–385.

[7] Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and MacIntyre, B. (2001). Recent advances in augmented reality. *Computer Graphics and Applications, IEEE*, 21(6):34 –47.

[8] Azuma, R. and Furmanski, C. (2003). Evaluating label placement for augmented reality view management. *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 66–75.

[9] Baldauf, M., Frohlich, P., Masuch, K., and Grechenig, T. (2011). Comparing viewing and filtering techniques for mobile urban exploration. *Location Based Services*, 5(1):38–57.

[10] Bane, R. and Hollerer, T. (2004). Interactive Tools for Virtual X-Ray Vision in Mobile Augmented Reality. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 231–239. IEEE.

[11] Baricevic, D., Cha, L., Turk, M., Höllerer, T., and Bowman, D. A. (2012). Hand-held AR Magic Lenses with User-Perspective Rendering. In *International Symposium on Mixed and Augmented Reality*, pages 197–206.

[12] Bell, B., Feiner, S., and Höllerer, T. (2001). View management for virtual and augmented reality. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 101–110, Orlando, Florida. ACM.

[13] Bell, B., Höllerer, T., and Feiner, S. (2002). An annotated situation-awareness aid for augmented reality. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, page 213, Paris, France.

[14] Bichlmeier, C., Heining, S. M., Rustaee, M., and Navab, N. (2007). Laparoscopic Virtual Mirror for Understanding Vessel Structure Evaluation Study by Twelve Surgeons. In *International Symposium on Mixed and Augmented Reality*, pages 125–128, Charlotte, North Carolina, USA. IEEE Computer Society.

[15] Bier, E. A., Stone, M. C., Pier, K., Buxton, W., and DeRose, T. D. (1993). Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 73–80.

[16] Billinghurst, M., Kato, H., and Poupyrev, I. (2001). The MagicBook - Moving Seamlessly between Reality and Virtuality. *IEEE Computer Graphics and Applications*, 21(1):6–9.

[17] Bimber, O. and Raskar, R. (2005). Spatial Augmented Reality: Merging Real and Virtual Worlds.

[18] Blanz, V., Tarr, M. J., and Bülthoff, H. H. (1999). What object attributes determine canonical views? *Perception*, 28(5):575 – 600.

[19] Bowman, D. A., Koller, D., and Hodges, L. F. (1997). Travel in immersive virtual environments: an evaluation of viewpoint motion control techniques. In *Proceedings of the 1997 Virtual Reality Annual International Symposium (VRAIS '97)*, pages 45–52. IEEE.

[20] Breen, D. E., Whitaker, R. T., Rose, E., and Tuceryan, M. (1996). Interactive Occlusion and Automatic Object Placement for Augmented Reality. *Computer Graphics Forum*, 15(3):11–22.

[21] Bruckner, S. and Gröiller, M. E. (2006). Exploded views for volume data. *IEEE transactions on visualization and computer graphics*, 12(5):1077–84.

[22] Burigat, S. and Chittaro, L. (2008). Decluttering of Icons Based on Aggregation in Mobile Maps. In Meng, L., Zipf, A., and Winter, S., editors, *Map-based Mobile Services*, pages 13–32. Springer Berlin Heidelberg, Berlin, Heidelberg.

[23] Card, S. K., Mackinlay, J. D., and Shneiderman, B. (1999). *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

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

[25] Chia, A. Y.-S., Rahardja, S., Rajan, D., and Leung, M. K. (2010). Object recognition by discriminative combinations of line segments and ellipses. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2225–2232. IEEE.

[26] Chigona, W., Sonnet, H., Ritter, F., and Strothotte, T. (2003). Shadows with a message. In *Proceedings of the 3rd international conference on Smart graphics*, pages 91–101. Springer-Verlag.

[27] Cockburn, A., Karlson, A., and Bederson, B. B. (2008). A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 41(1):1–31.

[28] Coelho, E., MacIntyre, B., and Julier, S. (2004). OSGAR: A Scene Graph with Uncertain Transformations. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 6–15. IEEE.

[29] Craft, B. and Cairns, P. (2005). Beyond Guidelines: What Can We Learn from the Visual Information Seeking Mantra? *Ninth International Conference on Information Visualisation (IV'05)*, pages 110–118.

[30] Cui, J., Rosen, P., Popescu, V., and Hoffmann, C. (2010). A Curved Ray Camera for Handling Occlusions through Continuous Multiperspective Visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1235–1242.

[31] Ding, C. (2002). Cluster merging and splitting in hierarchical clustering algorithms. In *IEEE International Conference on Data Mining*, pages 139–146. IEEE Comput. Soc.

[32] Dix, A. and Ellis, G. (2002). by chance: enhancing interaction with large data sets through statistical sampling. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '02, pages 167–176, New York, NY, USA. ACM.

[33] dos Santos, S. and Brodlie, K. (2004). Gaining understanding of multivariate and multidimensional data through visualization. *Computers & Graphics*, 28(3):311–325.

[34] Dueck, G. and Scheuer, T. (1990). Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1):161–175.

[35] Eduardo Veas, Raphael Grasset, Ernst Kruijff, and Dieter Schmalstieg (2012). Extended Overview Techniques for Outdoor Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics*, 18(4):565–572.

[36] Eissele, M., Kreiser, M., and Ertl, T. (2008). Context-controlled flow visualization in augmented reality. In *Proceedings of Graphics Interface 2008*, pages 89–96. Canadian Information Processing Society.

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

[38] Elmqvist, N. and Fekete, J.-D. D. (2010). Hierarchical aggregation for information visualization: overview, techniques, and design guidelines. *IEEE transactions on visualization and computer graphics*, 16(3):439–54.

[39] Feiner, S., MacIntyre, B., Hollerer, T., and Webster, A. (1997). A touring machine: prototyping 3D mobile augmented reality systems for exploring the urban environment. *Digest of Papers. First International Symposium on Wearable Computers*, pages 74–81.

[40] Feiner, S., Macintyre, B., and Seligmann, D. (1993). Knowledge-based augmented reality. *Communications of the ACM*, 36(7):53–62.

[41] Ferreira de Oliveira, M. C. and Levkowitz, H. (2003). From visual data exploration to visual data mining: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):378– 394.

[42] Fink, M., Haunert, J.-H., Schulz, A., Spoerhase, J., and Wolff, A. (2012). Algorithms for Labeling Focus Regions. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2583–2592.

[43] Fournier, A. (1995). Illumination Problems in Computer Augmented Reality. Technical report, Department of Computer Science, University of British Columbia.

[44] Fua, Y.-H., Ward, M. O., and Rundensteiner, E. A. (1999). Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings of the conference on Visualization '99: celebrating ten years*, pages 43–50. IEEE Computer Society Press.

[45] Fuhrmann, A., Löffelmann, H., and Schmalstieg, D. (1997). Collaborative augmented reality: exploring dynamical systems. In *Proceedings. Visualization '97*, pages 459–462,. IEEE.

[46] Furnas, G. W. (1986). Generalized fisheye views. *Special issue: CHI '86 Conference Proceedings*, 17(4):16–23.

[47] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

[48] Gooch, B., Reinhard, E., Moulding, C., and Shirley, P. (2001). Artistic Composition for Image Creation. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 83–88. Springer-Verlag.

[49] Götzelmann, T., Ali, K., Hartmann, K., and Strothotte, T. (2005). Adaptive Labeling for Illustrations. In *Pacific Graphics*. Otto-von-Guericke University of Magdeburg.

[50] Grasset, R., Billinghurst, M., and Dünser, A. (2008). Moving Between Contexts - A User Evaluation of a Transitional Interface. In *International Conference on Artificial Reality and Telexistence (ICAT)*.

[51] Grasset, R., Langlotz, T., Kalkofen, D., Tatzgern, M., and Schmalstieg, D. (2012). Image-driven view management for augmented reality brows ers. In *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR'12)*, pages 177–186. IEEE Computer Society.

[52] Gray, H. (1918). *Anatomy of the Human Body*. Lea&Febiger, 1918.

[53] Gruber, L., Kalkofen, D., and Schmalstieg, D. (2010). Color harmonization for Augmented Reality. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 227–228. IEEE.

[54] Gruber, L., Richter-Trummer, T., and Schmalstieg, D. (2012). Real-time photometric registration from arbitrary geometry. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 119–128. IEEE.

[55] Haber, R. and McNabb, D. (1990). Visualization idioms: A conceptual model for scientific visualization systems. In *Visualization in Scientific Computing*, pages 74–93. IEEE Computer Society Press.

[56] Hartmann, K., Ali, K., and Strothotte, T. (2004). Floating labels: Applying dynamic potential fields for label layout. *IN 4TH INTERNATIONAL SYMPOSIUM ON SMART GRAPHICS*, 3031:101—-113.

[57] Hartmann, K., Götzelmann, T., and Strothotte, T. (2005). Metrics for Functional and Aesthetic Label Layouts. In A. Butz, B. F. and Olivier, P., editors, *Smart Graphics*, pages 115–126. Springer Verlag, Berlin.

[58] Hoang, T. N. and Thomas, B. H. (2010). Augmented Viewport: An action at a distance technique for outdoor AR using distant and zoom lens cameras. In *International Symposium on Wearable Computers (ISWC)*, pages 1–4. IEEE.

[59] Hoenig, F. (2005). Defining computational aesthetics. In *Proceedings of the First Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging*, pages 13–18. Eurographics Association.

[60] Höllerer, T. and Feiner, S. (2004). *Mobile augmented reality*. CRC Press.

[61] Höllerer, T., Feiner, S., and Pavlik, J. (1999). Situated documentaries: embedding multimedia presentations in the real world. In *Digest of Papers. Third International Symposium on Wearable Computers*, pages 79–86. IEEE Comput. Soc.

[62] Holten, D. (2006). Hierarchical edge bundles: visualization of adjacency relations in hierarchical data. *IEEE transactions on visualization and computer graphics*, 12(5):741–8.

[63] Homem de Mello, L. and Sanderson, A. (1990). AND/OR graph representation of assembly plans. *IEEE Transactions on Robotics and Automation*, 6(2):188–199.

[64] Imhof, E. (1975). Positioning Names on Maps. *The American Cartographer*, 2(2):128–144.

[65] Izadi, S., Davison, A., Fitzgibbon, A., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., and Freeman, D. (2011). KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*, pages 559–568, New York, New York, USA. ACM Press.

[66] J. Newman, G. S., Newman, J., and Schall, G. (2006). Wide-Area Tracking Tools for Augmented Reality. *Advances in Pervasive Computing*, 207:3 – 6.

[67] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323.

[68] Jiang, Z., Nimura, Y., Hayashi, Y., Kitasaka, T., Misawa, K., Fujiwara, M., Kajita, Y., Wakabayashi, T., and Mori, K. (2013). Anatomical annotation on vascular structure in volume rendered images. *Computerized medical imaging and graphics : the official journal of the Computerized Medical Imaging Society*, 37(2):131–41.

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

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

[71] Kalkofen, D. (2009). *Illustrative X-Ray Visualization in Augmented Reality Environments*. Phd thesis, Graz University of Technology.

[72] Kalkofen, D., Mendez, E., and Schmalstieg, D. (2009a). Comprehensible visualization for augmented reality. *IEEE transactions on visualization and computer graphics*, 15(2):193–204.

[73] Kalkofen, D., Sandor, C., White, S., and Dieter, S. (2011). Visualization Techniques for Augmented Reality. In *Handbook of Augmented Reality*, pages 65–98.

[74] Kalkofen, D., Tatzgern, M., Schmalstieg, D., Denis Kalkofen, Markus Tatzgern, and Dieter Schmalstieg (2009b). Explosion Diagrams in Augmented Reality. In *IEEE Virtual Reality Conference*, pages 71–78. IEEE.

[75] Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 61–70. Eurographics Association.

[76] Khan, A., Komalo, B., Stam, J., Fitzmaurice, G., and Kurtenbach, G. (2005). HoverCam: interactive 3D navigation for proximal object inspection. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, I3D '05, pages 73–80, Washington, District of Columbia. ACM.

[77] Kiyokawa, K., Takemura, H., and Yokoya, N. (1999). SeamlessDesign: a face-to-face collaborative virtual/augmented environment for rapid prototyping of geometrically constrained 3-D objects. In *Proceedings IEEE International Conference on Multimedia Computing and Systems*, volume 2, pages 447–453. IEEE Comput. Soc.

[78] Klein, G. and Murray, D. W. (2010). Simulating low-cost cameras for augmented reality compositing. *IEEE transactions on visualization and computer graphics*, 16(3):369–80.

[79] Kruijff, E., Swan, J., and Feiner, S. (2010). Perceptual issues in augmented reality revisited. In *International Symposium on Mixed and Augmented Reality*, pages 3 –12.

[80] Lee, C., Bonebrake, S., Bowman, D. A., and Hollerer, T. (2010). The role of latency in the validity of AR simulation. In *2010 IEEE Virtual Reality Conference (VR)*, pages 11–18. IEEE.

[81] Lee, C., Rincon, G. A., Meyer, G., Höllerer, T., and Bowman, D. A. (2013). The effects of visual realism on search tasks in mixed reality simulation. *IEEE transactions on visualization and computer graphics*, 19(4):547–56.

[82] Leykin, A. and Tuceryan, M. (2004). Automatic Determination of Text Readability over Textured Backgrounds for Augmented Reality Systems. VRCAI '04, pages 224–230, Singapore. IEEE Computer Society.

[83] Li, W., Agrawala, M., Curless, B., and Salesin, D. (2008a). Automated Generation of Interactive 3D Exploded View Diagrams. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2008*, 27(3).

[84] Li, W., Agrawala, M., and Salesin, D. (2004). Interactive Image-Based Exploded View Diagrams. In *Grapics Interface*, pages 203–212.

[85] Li, W., Ritter, L., Agrawala, M., Curless, B., and Salesin, D. (2007). Interactive cutaway illustrations of complex 3D models. *ACM Transactions on Graphics*, 26(3):31.

[86] Li, X., Wu, C., Zach, C., Lazebnik, S., and Frahm, J.-M. (2008b). Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs. In *Computer Vision - ECCV 2008*, ECCV '08, pages 427–440, Berlin, Heidelberg. Springer-Verlag.

[87] Maass, S. and Döllner, J. (2006). Dynamic Annotation of Interactive Environments using Object-Integrated Billboards. In *14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 327–334.

[88] Maass, S. and Döllner, J. (2008). Seamless integration of labels into interactive virtual 3D environments using parameterized hulls. In *Proceedings of the Fourth Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging*, Computational Aesthetics'08, pages 33–40, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[89] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. The Regents of the University of California.

[90] Madsen, J. B., Tatzgern, M., Kalkofen, D., Schmalstieg, D., and Madsen, C. B. (2015). Evaluating Adaptive Labeling for Dynamic Handheld Augmented Reality. *submitted*.

[91] Manning, C. D., Raghavan, P., and Schütze, H. (2008). Introduction to Information Retrieval.

[92] Marks, J. and Shieber, S. (1991). The Computational Complexity of Cartographic Label Placement. Technical report, Center for Research in Computing Technology, Harvard University.

[93] Mendez, E., Kalkofen, D., Schmalstieg, D., and Méndez, E. (2006). Interactive context-driven visualization tools for augmented reality. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '06, pages 209–218, Washington, DC, USA. IEEE Computer Society.

[94] Mijksenaar, P. and Westendorp, P. (1999). *Open Here: The Art of Instructional Design*. Thames & Hudson.

[95] Milgram, P., Takemura, H., Utsumi, A., and Kishino, F. (1995). Augmented reality: A class of displays on the reality-virtuality continuum. In *Telemanipulator and Telepresence Technologies*, volume 2351, pages 282–292.

[96] Mohring, M., Lessig, C., and Bimber, O. (2004). Video See-Through AR on Consumer Cell-Phones. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 252–253. IEEE.

[97] Mulloni, A., Dünser, A., and Schmalstieg, D. (2010). Zooming interfaces for augmented reality browsers. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services - MobileHCI '10*, page 161, New York, New York, USA. ACM Press.

[98] Mulloni, A., Seichter, H., Dünser, A., Baudisch, P., and Schmalstieg, D. (2012). 360Â° panoramic overviews for location-based services. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*, page 2565, New York, New York, USA. ACM Press.

[99] Newcombe, R. and Andrew, J. (2010). Live Dense Reconstruction with a Single Moving Camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1498–1505.

[100] Niederauer, C., Houston, M., Agrawala, M., and Humphreys, G. (2003). Non-Invasive Interactive Visualization of Dynamic Architectural Environments. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 55–58.

[101] Orlosky, J., Kiyokawa, K., and Takemura, H. (2013). Dynamic text management for see-through wearable and heads-up display systems. In *Proceedings of the 2013 international conference on Intelligent user interfaces - IUI '13*, IUI '13, pages 363–370, New York, NY, USA. ACM.

[102] Oulasvirta, A., Estlander, S., and Nurminen, A. (2008). Embodied interaction with a 3D versus 2D mobile map. *Personal and Ubiquitous Computing*, 13(4):303–320.

[103] Pearl, J. (1984). Heuristics: intelligent search strategies for computer problem solving.

[104] Pick, S., Hentschel, B., Tedjo-Palczynski, I., Wolter, M., and Kuhlen, T. (2010). Automated Positioning of Annotations in Immersive Virtual Environments. In *EGVE - JVRC'10 Proceedings of the 16th Eurographics conference on Virtual Environments & Second Joint Virtual Reality*, pages 1–8. The Eurographics Association.

[105] Pierce, J. S., Stearns, B. C., and Pausch, R. (1999). Voodoo dolls: seamless interaction at multiple scales in virtual environments. In *Proceedings of the 1999 symposium*

on Interactive 3D graphics - SI3D '99, pages 141–145, New York, New York, USA. ACM Press.

[106] Raab, A. and Rüger, M. (1996). 3D-Zoom: Interactive Visualisation of Structures and Relations in Complex Graphics. In B. Girod, H. N., editor, *3D Image Analysis and Synthesis*, pages 87–93. infix-Verlag.

[107] Resnick, P. and Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3):56–58.

[108] Rist, T., Krüger, A., Schneider, G., and Zimmermann, D. (1994). AWI: A Workbench for Semi-Automated Illustration Design. In *Proceedings of the workshop on Advanced visual interfaces*, pages 59–68, New York, NY, USA. ACM Press.

[109] Robertson, C. M. and MacIntyre, B. (2007). An Evaluation of Graphical Context as a Means for Ameliorating the Effects of Registration Error. In *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE.

[110] Ropinski, T., Praß ni, J.-S., Roters, J., and Hinrichs, K. H. (2007). Internal Labels as Shape Cues for Medical Illustration. In *Proceedings of the 12th International Fall Workshop on Vision, Modeling, and Visualization (VMV07)*, pages 203–212.

[111] Rosenholtz, R., Li, Y., Mansfield, J., and Jin, Z. (2005). Feature congestion: a measure of display clutter. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 761–770, Portland, Oregon, USA. ACM.

[112] Rosten, E., Reitmayr, G., and Drummond, T. (2005). *Advances in Visual Computing*, volume 3804 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[113] Ruiz, M., Viola, I., Boada, I., Bruckner, S., Feixas, M., and Sbert, M. (2008). Similarity-Based Exploded Views. In *Proceedings of Smart Graphics*, pages 154–165, Berlin, Heidelberg. Springer-Verlag.

[114] Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation*, pages 1–4. IEEE.

[115] Sandor, C., Dey, A., Cunningham, A., Barbier, S., Eck, U., Urquhart, D., Marner, M. R., Jarvis, G., and Rhee, S. (2010). Egocentric space-distorting visualizations for

rapid environment exploration in mobile mixed reality. In *IEEE Virtual Reality Conference (VR)*, pages 47–50. IEEE.

[116] Schulz, M., Reck, F., Bertelheimer, W., and Ertl, T. (1999). Interactive visualization of fluid dynamics simulations in locally refined cartesian grids. In *Proceedings Visualization '99 (Cat. No.99CB37067)*, pages 413–553. IEEE.

[117] Schwinger, W., Grün, C., Pröll, B., Retschitzegger, W., and Schauerhuber, A. (2005). Context-Awareness in Mobile Tourism Guides - A Comprehensive Survey. Technical report, Johannes Kepler University Linz.

[118] Shibata, F., Nakamoto, H., Sasaki, R., Kimura, A., and Tamura, H. (2008). A view management method for mobile mixed reality systems. In *Proceedings of the 14th Eurographics conference on Virtual Environments*, pages 17–24. Eurographics Association.

[119] Shneiderman, B. (1996). The eyes have it: a task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE ...*, pages 336–343. IEEE.

[120] Sokolov, D. and Plemenos, D. (2005). Viewpoint quality and scene understanding. In *Proceedings of the 6th International conference on Virtual Reality, Archaeology and Intelligent Cultural Heritage*, pages 67–73.

[121] Sonnet, H., Carpendale, S., and Strothotte, T. (2004). Integrating Expanding Annotations with a 3D Explosion Probe. In *Advanced Visual Interfaces*, pages 63–70, New York, NY, USA. ACM Press.

[122] Spence, R. (2007). *Information Visualization - Design for Interaction*. Pearson Education Limited, 2nd edition.

[123] Stoakley, R., Conway, M. J., and Pausch, R. (1995). Virtual reality on a WIM. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '95*, pages 265–272, New York, New York, USA. ACM Press.

[124] Sukan, M., Feiner, S., Tversky, B., and Energin, S. (2012). Quick viewpoint switching for manipulating virtual objects in hand-held augmented reality using stored snapshots. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 217–226. IEEE.

[125] Sutherland, I. E. (1968). A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)*, pages 757–764, New York, New York, USA. ACM Press.

[126] Tatzgern, M., Grasset, R., Veas, E., Kalkofen, D., Seichter, H., and Schmalstieg, D. (2013a). Exploring Distant Objects with Augmented Reality. In *Proceedings of the Joint Virtual Reality Conference of EGVE - EuroVR*, pages 49–56. The Eurographics Association.

[127] Tatzgern, M., Grasset, R., Veas, E., Kalkofen, D., Seichter, H., and Schmalstieg, D. (2014a). Exploring Real World Points of Interest: Design and Evaluation of Object-centric Exploration Techniques for Augmented Reality. *Pervasive Mobile Computing: Special Issue on Mobile and Pervasive Applications in Tourism*.

[128] Tatzgern, M., Kalkofen, D., Grasset, R., and Schmalstieg, D. (2011a). Embedded Virtual Views for Augmented Reality Navigation. In *International Symposium on Mixed and Augmented Reality - Workshop on Visualization in Mixed Reality Environments*.

[129] Tatzgern, M., Kalkofen, D., Grasset, R., and Schmalstieg, D. (2014b). Hedgehog Labeling: View Management Techniques for External Labels in 3D Space. In *IEEE Virtual Reality*.

[130] Tatzgern, M., Kalkofen, D., Grasset, R., and Schmalstieg, D. (2014c). Transitional Augmented Reality Navigation for Live Captured Scenes. In *IEEE Virtual Reality*.

[131] Tatzgern, M., Kalkofen, D., and Schmalstieg, D. (2010). Compact explosion diagrams. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, volume 35, pages 17–26, New York, New York, USA. ACM Press.

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

[133] Tatzgern, M., Kalkofen, D., and Schmalstieg, D. (2013b). Dynamic compact visualizations for augmented reality. In *IEEE Virtual Reality (VR)*, pages 3–6. IEEE.

[134] Tatzgern, M., Orso, V., Kalkofen, D., Jacucci, G., and Gamberini, Luciano Schmalstieg, D. (2015). Adaptive Information Density for Augmented Reality Displays. *submitted*.

[135] Thomas, B., Demczuk, V., Piekarski, W., Hepworth, D., and Gunther, B. (1998). A wearable computer system with augmented reality to support terrestrial navigation. In *Digest of Papers. Second International Symposium on Wearable Computers*, pages 168–171. IEEE Comput. Soc.

[136] Trapp, M., Beesk, C., Pasewaldt, S., and Jürgen, D. (2011). Interactive Rendering Techniques for Highlighting in 3D Geovirtual Environments. In *Advances in 3D Geo-Information Sciences*, pages 197–210. Springer Berlin Heidelberg.

[137] Veas, E., Grasset, R., Ferencik, I., Grünewald, T., and Schmalstieg, D. (2012). Mobile augmented reality for environmental monitoring. *Personal and Ubiquitous Computing*, 17(7):1515–1531.

[138] Veas, E., Mulloni, A., Kruijff, E., Regenbrecht, H., and Schmalstieg, D. (2010). Techniques for view transition in multi-camera outdoor environments. In *Proceedings of Graphics Interface*, GI '10, pages 193–200, Toronto, Ont., Canada, Canada. Canadian Information Processing Society.

[139] Virrantaus, K., Markkula, J., Garmash, A., Terziyan, V., Veijalainen, J., Katanosov, A., and Tirri, H. (2001). Developing GIS-supported location-based services. In *Proceedings of the Second International Conference on Web Information Systems Engineering*, volume 2, pages 66–75. IEEE Comput. Soc.

[140] Vranic, D. (2005). DESIRE: a composite 3D-shape descriptor. In *International Conference on Multimedia and Expo (ICME)*, page 4 pp.

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

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

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

[144] Wertheim, A. H., Hooge, I. T. C., Krikke, K., and Johnson, A. (2006). How important is lateral masking in visual search? *Experimental brain research*, 170(3):387–402.

[145] White, S. (2009). *Interaction and Presentation Techniques for Situated Visualization*. PhD thesis, Columbia University.

[146] White, S. and Feiner, S. (2009). SiteLens: situated visualization techniques for urban site visits. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1117–1120, New York, NY, USA. ACM.

[147] Wilson, R. H. (1992). *On Geometric Assembly Planning*. PhD thesis, Stanford University, Stanford, California.

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

[149] Wolfe, J. M. (1998). What Can 1 Million Trials Tell Us About Visual Search? *Psychological Science*, 9(1):33–39.

[150] Woodruff, A., Landay, J., and Stonebraker, M. (1998). Constant density visualizations of non-uniform distributions of data. In *Annual ACM Symposium on User Interface Software and Technology (UIST)*, UIST '98, pages 19–28, New York, NY, USA. ACM.

[151] Yan, X. and Jiawei Han (2002). gSpan: graph-based substructure pattern mining. In *IEEE International Conference on Data Mining*, pages 721–724. IEEE Comput. Soc.

[152] Zhu, Y. (2007). Measuring Effective Data Visualization. In *Advances in Visual Computing*, pages 652–661.

[153] Zollmann, S., Kalkofen, D., Hoppe, C., Kluckner, S., Bischof, H., and Reitmayr, G. (2012). Interactive 4D overview and detail visualization in augmented reality. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 167–176. IEEE.