Peter Mohr-Ziak

# Retargeting Instructions to Augmented Reality

**DOCTORAL THESIS**

to achieve the university degree of
Doktor der technischen Wissenschaften

submitted to

**Graz University of Technology**

Supervisor

Ass. Prof. Dr. Denis Kalkofen
Institute for Computer Graphics and Vision

Prof. Dr. Dieter Schmalstieg
Institute for Computer Graphics and Vision

Graz, Austria, April 2020

Die Neugier steht immer an erster Stelle eines Problems, das gelöst werden will.

*Galileo Galilei / Ildefonso*

# Abstract

Augmented reality (AR) has been demonstrated to be an effective way of presenting many types of tutorials and user guide handbooks. However, creating 3D content for AR is usually costly and requires specially trained technical authors. The research in this thesis aims to accelerate the authoring process of AR instructions by providing interactive authoring techniques for retargeting conventional, two-dimensional content into three-dimensional AR tutorials. Unlike previous work, we do not simply overlay images or video but synthesize 3D-registered motion from the 2D input. Since the information in the resulting AR tutorial is registered to 3D objects, the user can freely change the viewpoint without degrading the experience. Our approaches can be applied to many styles of video tutorials. In this work, we concentrate on assembly and disassembly tutorials, body motion, and tutorials which show tools with surface contact, e.g. painting instructions.

In addition to offline authoring, we also show an approach for instant AR instruction authoring in a remote assistance use case. Spontaneous provisioning of remote assistance requires an easy, fast, and robust approach for capturing and sharing of unprepared environments. In this thesis, we make a case for utilizing interactive light fields for remote assistance. We demonstrate the advantages of object representation using light fields over conventional geometric reconstruction. Moreover, we introduce an interaction method for quickly annotating light fields in 3D space without requiring surface geometry to anchor annotations. We present results from a user study demonstrating the effectiveness of our interaction techniques, and we provide feedback on the usability of our overall system.

AR instruction systems require dedicated interaction methods to fully develop their potential. Therefore, we present novel interaction methods for AR on handheld devices, as well as for head-mounted displays. Handheld Augmented Reality commonly implements some variant of magic lens rendering, which turns only a fraction of the user's real environment into AR while the rest of the environment remains unaffected. Since handheld AR devices are commonly equipped with video see-through capabilities, AR magic lens applications often suffer from spatial distortions, because the AR environment is presented from the perspective of the camera of the mobile device. In this thesis, we present a resource-efficient method for user perspective rendering by applying

lightweight optical flow tracking and an estimation of the user's motion before head tracking is started. For HMDs we present TrackCap, a novel approach for 3D tracking of input devices, which turns a conventional smartphone into a precise 6DOF input device for an HMD user. The device can be conveniently operated both inside and outside the HMD field of view, while it provides additional 2D input and output capabilities. Evaluations show that TrackCap competes favorably against common input devices for mobile HMDs.

# Kurzfassung

Augmented Reality (AR) hat sich als effektive Methode zur Präsentation vieler Arten von Tutorials und Handbüchern erwiesen. Das Erstellen von 3D-Inhalten für AR ist jedoch normalerweise kostspielig und erfordert speziell geschulte technische Autor*innen. Die Forschung in dieser Arbeit zielt darauf ab, die Erstellung von AR-Anleitungen zu beschleunigen, indem interaktive Autorentools bereitgestellt werden, um konventionelle zweidimensionale Inhalte in dreidimensionale AR-Tutorials umzuwandeln. Im Gegensatz zu früheren Arbeiten überlagern wir nicht einfach Bilder oder Videos, sondern synthetisieren 3D-registrierte Bewegungen aus den 2D Eingangsdaten. Da die Informationen in der resultierenden AR-Anwendung auf 3D-Objekte registriert sind, können die Nutzer*innen den Standpunkt frei wählen, ohne die Visualisierung zu beeinträchtigen. Unsere Ansätze können auf viele Arten von Video-Tutorials angewendet werden. In dieser Arbeit konzentrieren wir uns auf Montage- und Demontage-Tutorials, Körperbewegungen und Tutorials, die Werkzeuge mit Oberflächenkontakt zeigen, wie zum Beispiel Malanleitungen.

Neben der Offline-Erstellung zeigen wir auch einen Ansatz für die Erstellung von AR-Anweisungen in einem Anwendungsszenario für Remoteunterstützung. Die spontane Bereitstellung von Remoteunterstützung erfordert einen einfachen, schnellen und robusten Ansatz für die Erfassung und Augmentierung unvorbereiteter Umgebungen. In dieser Arbeit demonstrieren wir die Vorteile der Objektdarstellung mit Lichtfeldern gegenüber der herkömmlichen, geometrischen Rekonstruktion. Darüber hinaus führen wir eine Interaktionsmethode zum schnellen Annotieren von Lichtfeldern im 3D-Raum ein, ohne dass eine 3D Oberflächengeometrie zum Verankern von Annotationen erforderlich ist. Wir präsentieren Ergebnisse einer Nutzer*innenstudie, die die Wirksamkeit unserer Interaktionstechniken demonstriert, und geben Feedback zur Benutzungsfreundlichkeit unseres Gesamtsystems.

AR Anleitungen erfordern spezielle Interaktionsmethoden, um ihr Potenzial voll auszuschöpfen. Daher präsentieren wir neuartige Interaktionsmethoden für AR auf Handheld-Geräten sowie für Head-Mounted-Displays. Handheld Augmented Reality implementiert normalerweise eine Variante des Renderings von so genannten „Magic Lenses“, bei der nur ein Bruchteil der realen Umgebung der Nutzer*innen in AR umgewandelt wird,

während der Rest der Umgebung davon nicht betroffen ist. Da mobile AR-Geräte üblicherweise mit einer Video-See-Through Funktion ausgestattet sind, leiden AR-Magic-Lens-Anwendungen häufig unter räumlichen Verzerrungen, da die AR-Umgebung aus der Perspektive der Kamera des mobilen Geräts dargestellt wird. In dieser Arbeit stellen wir eine ressourceneffiziente Methode für das Rendern von Inhalten aus der Benutzerperspektive vor, bei der wir eine effiziente Variante des Optical Flows verwenden, um die Bewegung der Nutzer*innen zu schätzen. Für HMDs präsentieren wir TrackCap, einen neuartigen Ansatz zum 3D-Tracking von Eingabegeräten, der ein herkömmliches Smartphone in ein präzises 6DOF-Eingabegerät für HMD-Benutzer*innen verwandelt. Das Gerät kann bequem sowohl innerhalb als auch außerhalb des HMD-Sichtfelds bedient werden und bietet zusätzliche 2D-Eingabe- und Ausgabefunktionen. Auswertungen zeigen, dass TrackCap durchaus mit gängigen Eingabegeräten für mobile HMDs verglichen werden kann.

## Affidavit

*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 indicated all material which has been quoted either literally or by content from the sources used.*

*The text document uploaded to TUGRAZonline is identical to the present 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 would like to thank all the great people who I worked with for their support. First and foremost I would like to thank my advisors Ass. Prof. Denis Kalkofen and Prof. Dieter Schmalstieg for their great support and guidance over the years of research and during the writing of this thesis. I also want to mention my colleagues Markus Tatzgern, David Mandl, Laurenz Theuerkauf, Shohei Mori, Christoph Ebner and Ana Stanescu with whom I had many interesting discussions and worked on numerous scientific publications. A very special word of thanks again goes to Denis Kalkofen, who was an excellent mentor during my time at the ICG.

My special thanks go to my parents, who supported me during my studies, my wonderful wife, and to my newborn son Viktor, who was peacefully sleeping next to me while I was writing this thesis.

# Contents

# List of Figures

# List of Tables

# CHAPTER 1

# Introduction

Assembly and disassembly procedures are essential elements in many tasks. Popular examples include the assembly of new furniture and the maintenance of household appliances that requires knowledge of both assembly and disassembly procedures: disassembly to reach parts in need of service, and assembly to rebuild them. The necessary assembly and disassembly actions are traditionally communicated using textual descriptions that are often combined with a series of images for illustrating motions in printed manuals [98]. We use handbooks to understand how things work and manuals to learn how to assemble or maintain them.

Most documentation still exists on paper. However, the use of printed manuals arguably introduces a cognitive seam, since users have to infer actions from a sequence of 2D images, which can be a mentally demanding task. Complex motions are difficult to visualize in a series of images, why illustrations are more and more complemented with 3D animations [162]. With the success of video-sharing platforms, the production and distribution of homemade video tutorials are rapidly increasing, why also a large body of video tutorials is available for nearly every aspect of life. Similar to 3D animations, videos allow the demonstration of complex actions required to solve a certain task why video tutorials became a powerful tool for communicating motions.

However, precisely following the instructions can be very challenging. The separation of task locations between the real environment and images or the video screen requires complex hand-eye coordination [22]. Users have to match objects in the video with corresponding objects in their real environment. They must infer 3D motion paths, speed and velocity only from 2D video cues. In addition, object appearances in the video may differ from the real world, making it difficult to identify matching landmarks. The problem is exacerbated by the fact that the user's viewpoint often deviates from the one in the video.

Augmented Reality (AR) overcomes this seam by presenting the documentation directly registered to the object in the user's real environment [115]. It has been shown that this can reduce the cognitive load [56]. However, **authoring**, **visualizing** and **interacting** with AR documentation requires careful design choices. The remainder of this section discusses the major problems of AR documentation and how this thesis addresses them.

## 1.1 Authoring Augmented Reality Documentation

Authoring augmented reality documentation is often a complex and time-consuming process. It requires skills with 3D modeling and animation tools and additional expertise with AR requirements such as registration and tracking. Consequently, few AR documentations exist today.

### 1.1.1 Authoring from Existing Image Documentations

Meanwhile, a large amount of traditional documentation exists on paper (or in two-dimensional digital form), but remains mostly unusable for AR applications. To close this gap, we propose a system capable of automatically transferring traditional printed documentation to AR. We demonstrate our approach on several graphical elements commonly found in traditional documentations. Specifically, we present the transfer of annotations labeling parts of the object, arrows indicating motions, explosion diagrams revealing an object's internal structure, and structural diagrams conveying the assembly or disassembly, translation and rotation of parts. Together, these illustrative elements cover the most frequent documentation styles. The conventions kept by illustrators allow

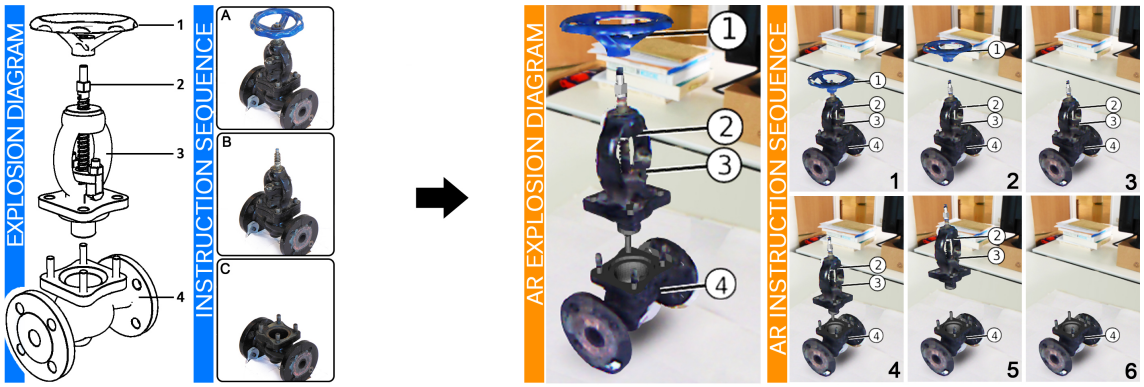



**Figure 1.1:** Augmented Reality documentation from 2D input images. (a) The input documentation consists of an annotated explosion diagram and a sequence of images presenting disassembly instructions. (b) From analyzing the 2D explosion diagram, our system is able to generate a 3D explosion diagram presented in AR. Moreover, our system generates 3D annotations in AR based on the input 2D documentation. (c) In addition, our system is able to analyze image sequences in order to create 3D animations from it, allowing the presentation of animated 3D documentation in AR. Here we show six key-frames from the resulting AR animation.

to automatically interpret the images with only minimal help from the user. Thus, it becomes feasible to produce AR applications from existing image-based documentation at only a fraction of the effort it would take with conventional 3D modeling and animation software. Figure 1.1 shows an example of an annotated explosion diagram, which we successfully retargeted to AR.

### 1.1.2 Authoring from Existing Video Tutorials

To furthermore make use of existing video documentation, this thesis presents a novel system capable of retargeting homemade video tutorials to AR. We concentrate on a class of tutorials showing tools operating on object surfaces. This kind of tutorial describes actions that alter the surface of an object. Common examples are painting, calligraphy, soldering or isolating circuits, make-up, and decorating, e.g., a teapot as shown in Figure 1.2.

Unlike previous work [48, 118], we do not simply overlay the video on the real object. Overlaid videos are efficient to produce, but not as effective as 3D tutorials, if the action is view-dependent or if the object in the video slightly differs from the one available to the user. In addition, a video overlay may clutter and occlude the real object, especially in surface manipulation tutorials, and animation in the video may distract the user while following the tutorial [162]. Therefore, our system extracts 3D motions from the video and registers the results to the 3D objects in the user's environment. This enables the user to freely change the viewpoint without degrading the quality of the AR experience. Moreover, this allows the presentation of instructions using effective illustrative visualizations, such as dynamic glyphs [110]. Illustrative visualizations are able to convey important information in an effective way with minimal clutter and, if necessary, without showing an animation. They also enable quick previews of arbitrary aggregate actions.

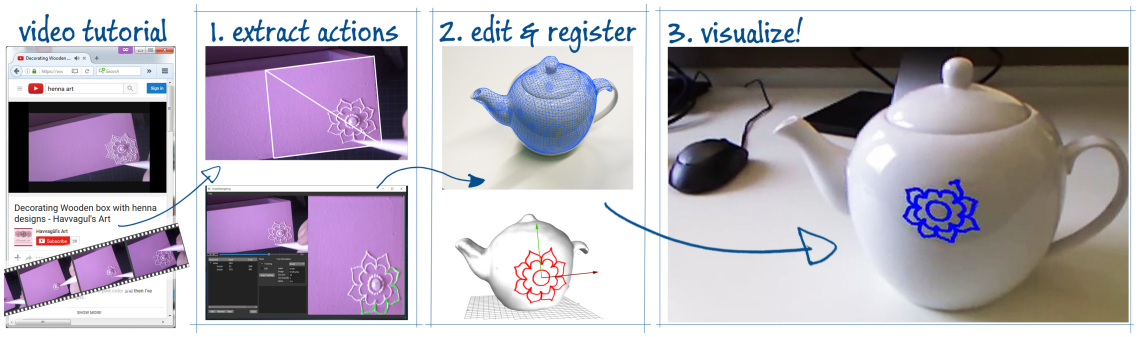



**Figure 1.2:** Retargeting a 'henna decoration' video tutorial to a teapot decoration scenario in the user's workspace. (left) The user extracts relevant motion from the video, (middle) scales it and aligns the result to a 3D scan of a teapot in the current workspace. With our editing tools, the user can quickly alter the original tutorial to meet their requirements. In this example, the original video tutorial shows a decoration consisting of dots, which requires a special henna pen. The user chooses to connect the dots into lines that can be drawn with a ceramic pen on the teapot. The user also scales down the entire ornament to better fit the desired aesthetics. (right) Using augmented reality, the user validates the result directly on the real teapot.

Furthermore, extracting 3D motions allows the editing of the tutorial's temporal structure. This may involve changing or adding actions or mixing multiple video tutorials into a novel one. For example, the tutorial in Figure 1.2 demonstrates drawing thick dots. However, a very specific tool is required to create thick dots, which the user might not have. Therefore, our system allows the connection of these dots into continuous lines representing the overall shape of the decoration. The result can be reproduced with a conventional drawing pen commonly used to decorate ceramics.

### 1.1.3 Remote Authoring of AR Instructions

One particularly important application that uses AR documentation emerges from the combination of Augmented Reality and and *Augmented Virtuality* (AV) is *remote assistance*, where an expert (remote user) helps a worker (local user) in operating or repairing a physical object on location. AV provides the remote user with a live representation of the local user's physical environment in addition to tools for exploring and annotating the shared environment with visual instructions [44, 65, 111]. The local user's AR display overlays the physical environment with the visual instructions that were generated by the remote user.

Implementing a remote assistance application faces two key challenges. First, the remote user requires a virtual representation of the local user's environment, which must be provided on the fly and allows for identifying all details necessary to complete the task. Second, both local user and remote user require intuitive interaction techniques for exploring and annotating the shared environment. Therefore, exploration and annotation must be performed in a 3D space to register the information correctly in the local user's environment.

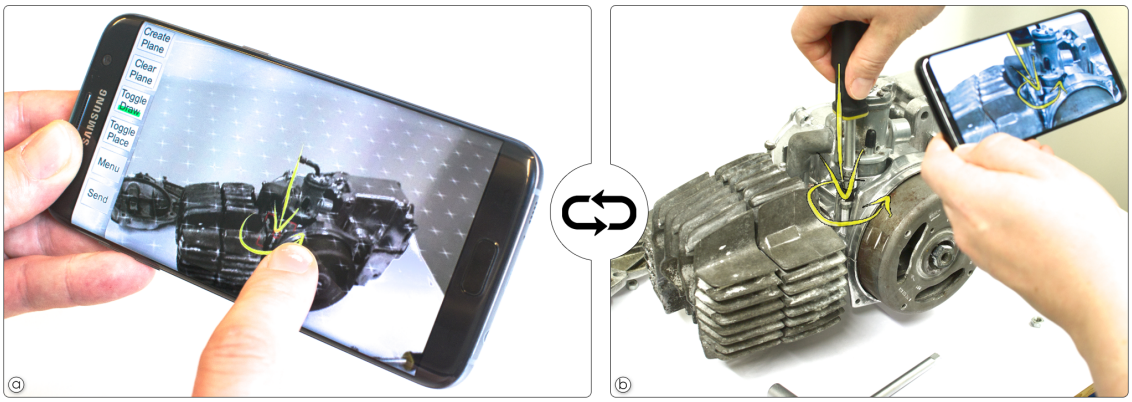



**Figure 1.3:** Live Authoring of AR Instructions. Two parties using our system in a tele-collaboration session. (a) The remote user generates visual instructions on a high-quality light field representation, which has been captured and shared by the local user. Our system supports guided capturing of the light field using off-the-shelf mobile devices. Subsequently, it enables annotating the representation using simple gestures on a mobile touch screen. (b) The local user follows the visual instructions in Augmented Reality.

We are particularly interested in mobile scenarios, as those are free of spatial constraints that encumber spontaneous use on stationary hardware. However, existing approaches relying on mobile devices for remote assistance are often restricted to 2D representations [180], provide 3D representations of limited visual quality [44] or rely on additional stationary equipment [114]. Moreover, we experienced that, in addition to the limited visual quality, existing approaches struggle to create proper virtual representations of featureless, transparent or shiny objects.

To address these challenges, we propose a new approach to remote assistance, which does not require a geometric model, but, instead, purely relies on an image-based representation in the form of an *unstructured light field* [36], i.e., a database of images registered in 3D space, which represent a sampling of the light rays emitted from the local user's workspace. Light fields offer many advantages over previous approaches. For example, no depth sensor is required, and reconstruction is not adversely affected by textureless, shiny or transparent surfaces. This robustness is an essential advantage of light fields over traditional reconstruction approaches, for instance, when considering industrial environments with lots of metallic surfaces. This enables our approach to work in many more environments compared to existing AR remote assistance systems. Figure 1.3 shows an example of this on a metallic engine, which would be challenging for traditional reconstruction methods commonly used in AR [45]. While light fields offer high visual quality, they also face challenges complicating their use in remote assistance applications. Creating light fields can be time-consuming, which is critical for remote assistance applications. Furthermore, a naive light field implementation results in a large number of images, which easily exceeds what can be transmitted, stored or rendered on mobile devices.

Finally, light fields lack explicit 3D geometry, making them difficult to interact with or modify [69]. Common tasks, such as placing graphical annotations on object surfaces captured as light fields, are not trivial without depth or surface information. The *Mixed Reality Light Fields* presented in this thesis address these issues. In particular, we provide a practical approach for utilizing unstructured light fields in AR on mobile devices. To demonstrate capturing, processing and annotation of light fields, we chose a challenging application, namely, remote assistance. In this application, we use light fields as a robust, high-fidelity representation of challenging scenes containing transparent, thin and shiny objects. Using Mixed Reality Light Fields, we do not only support a novel form of instant exploration of reconstructed objects, but we also support collaboration in the shared space through a novel interface for the navigation and annotation of remote scenes.

## 1.2 Visualizing Augmented Reality Instructions

A major part of an AR documentation is providing instructions. To visually communicate instructions we need to generate graphical elements based on known 3D motion. However, these graphical elements need to be comprehensible and they need to fit into the real environment so that the user can easily follow the indicated motion. This requires carefully designing the graphical elements and their integration into the user's workspace.


**Figure 1.4:** Comparing points of view for effective visual guidance. (a) We present motion instructions using the AR mirror visualization technique. We use a split-screen setup on the AR mirror providing a top-down and front view. (b) We provide visual instructions for key poses and motions between these poses. In this example, we show body instructions using a simple stick-figure for key poses. (c) 3D motions are visualized by the corresponding 3D path using a set of 3D arrows. (d) We used a virtual setup to provide the user with a large field of view in all conditions.

In order to provide an effective design of the AR visualization, this thesis contributes two user experiments. The first experiment is investigating dynamic glyphs [110] to be used for communicating motion paths in AR. The second experiment is comparing first person to an AR mirror point of view visualizations for communicating body motions. Figure 1.4 provides an illustration of this experiment.

## 1.3 Interacting with Augmented Reality Documentations

The technique for interacting with AR documentations highly depends on the AR display used for visualizing the augmentation. While a variety of different AR displays exist, this thesis focuses on widely available mobile handheld devices, and optical see-through displays, which allow for natural hand interactions.

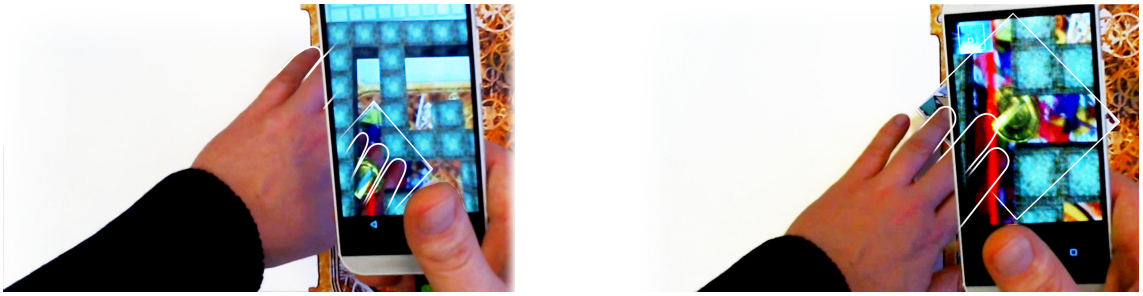
### 1.3.1 Interaction with AR Documentations on Mobile Devices

Handheld AR typically employs smartphone- or tablet-sized screen formats. The commonly applied Magic Lens metaphor, turns only a fraction of the user's real field-of-view into an augmented scene, while the rest of the environment remains unaffected. Furthermore, AR magic lens applications on video see-through displays often suffer from spatial distortions, because the AR environment is presented from the perspective of the camera of the device. The camera is usually located in a corner on the back of the device and captures the scene with a camera-dependent field of view. This is commonly defined as device-perspective rendering (DPR). The mismatch between the camera's and the user's point and field of view results in mismatching visualizations inside and outside of the magic lens (and is also called dual-view problem [167]). The visualization inside

the magic lens depends on the camera parameters while the user's perception outside is based on his or her natural vision (Figure1.6(a)). This misalignment is especially confusing when virtual content inside the magic lens has to be visually connected to real objects outside the magic lens, and during interactions with the AR environment through the magic lens, i.e., when the user sees its own hand inside and outside the lens.

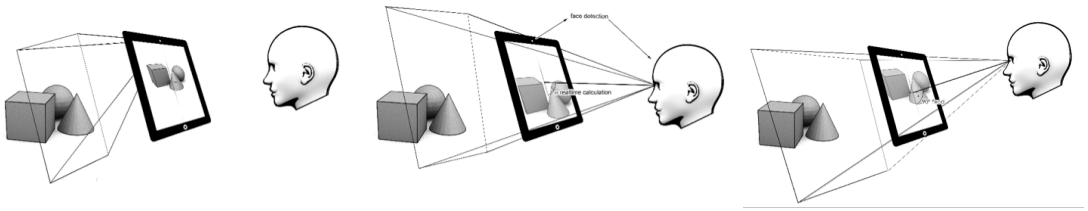
This is demonstrated in Figure 1.5 (a), showing a user interacting with an AR game through the magic lens of the handheld device. In the game, the user has to rotate a small physical marker in order to align a virtual mirror with a laser beam to control reflections of the beam. The interaction requires grasping the physical marker and subsequently rotating it. While the rotations may be performed entirely inside the AR magic lens, manipulating the marker involves moving the hand from the outside into the view inside the magic lens. When rendering the AR scene from the camera's point of view (Figure 1.5(a)) the visual mismatch between parts of the hand, which are outside, and other parts of the hand which are inside, make precisely grasping the marker difficult. User perspective rendering (UPR) has been proposed to overcome this problem [Baričević et al.]. It aligns the AR view inside the magic lens to the view of the user outside of the magic lens, to create the illusion of looking through a transparent glass frame (Figure1.6(b)). In fact, it has been shown that users, who were never exposed to handheld AR before, expect UPR as the default view [167]. Current approaches targeting typically implement UPR by computing the user's head position in each frame before they align the AR view based respectively. Figure 1.5(b) shows the AR view with UPR. Notice that the fingers inside the magic lens visually connect to the hand outside. This potentially makes selection tasks easier [Baričević et al.].

Implementations of UPR have often been explored using head tracking systems in laboratory setups. They either include stationary external camera tracking [135] or additional hardware setups such as depth sensors [Baričević et al.]. On mobile devices, UPR has been implemented using 3D face tracking based on the video feed of the front camera of modern mobile devices [141]. While



(a) Device Perspective Rendering (b) User Perspective Rendering

**Figure 1.5:** Hand Interaction in Device and User Perspective Augmented Reality. (a) Device perspective rendering directly augments the video stream of the handheld device. Objects outside and inside the augmentations appear disconnected. Notice the hand inside the AR device. (b) User perspective rendering estimates the user's head pose in order to adapt the AR rendering as seen from the head position. Therefore, objects outside and inside the AR display visually connect. Notice the fingers visually connect to the hand of the user.



(a) Device Perspective Rendering (b) User Perspective Rendering (c) Approximated User Perspective Rendering

**Figure 1.6:** Traditional approaches to magic lens rendering for handheld AR. (a) Device perspective rendering provides augmentations from the point of view of the camera. (b) User perspective rendering uses 3D head tracking to provide augmentation from the user's point of view. (c) Fixed point of view user perspective rendering does not require 3D head tracking. Instead, it assumes a static spatial relationship between the user's head and the display surface.

this approach works in theory, implementations suffer from the computational demands of the additional head tracking. Based on our experience, this results in overall low performance of the application, which can also be traced down to the devices' entering thermal throttling mode to prevent overheating. As a result, the usability of UPR applications can be substantially be reduced in real-world settings. A computationally less demanding alternative for mobile devices has been proposed by Pucihar et al. [166]. Instead of tracking the user's head pose the authors manually measure the distance of the head to the device once at the beginning of the application, and they assume the user looking perpendicular through the center of the device over the entirety of the application (Figure1.6(c)). This approach is called the *Fixed Point of View* user perspective rendering (FUPR). It avoids the computational effort required to continuously track the user's head pose.

However, FUPR fails to generate user perspective graphics for large interaction spaces. For example, Figure 3(a) shows a maintenance scenario, which requires touching switches and buttons at the top and the bottom of a large electric cabinet. In such a scenario, the spatial relationship between the user's head position and the handheld device frequently changes, which in turn will eventually render FUPR ineffective.

In this thesis, we combine the resource-friendly approach of FUPR with continuously effective UPR. We achieve this by adding a lightweight Kanade-Lucas-Tomasi (KLT) tracker [70, 96] to the head-tracking pipeline of traditional UPR systems. We use KLT-tracking to estimate head motion in image space, which we use to subsequently decide whether the parameters of FUPR need to be refreshed. Since simple thresholding of user motion will eventually introduce a certain amount of error, we present the idea of dual thresholding which incorporates temporal and spatial thresholding in order to derive a more precise 3D head pose when head motion stops. By automatically updating the parameters based on a 3D head tracker, we furthermore do not require any manual initialization. Since our approach incorporates fewer updates of the head pose, it is also more stable and more robust in environments where visual tracking is difficult.


(a) (b)

**Figure 1.7:** Augmented Reality Documentation on a Handheld Device. In a typical usage scenario, the user moves the handheld device from one position (a) to another (b) to view different AR instructions. The resulting transition of the user's head pose, in relation to the device, requires updating the viewing frustum.

### 1.3.2 Enabling Smartphones for 3D Interaction on Mobile HMDs

In non-instrumented environments, such an entry-level HMD offers only very limited natural interaction, as tracking of hands or handheld devices for a spontaneous and direct interaction is not supported. Instead, interaction typically relies on gamepads or orientation-only controllers [58, 128] that do not support direct spatial references to the real world. Re-using HMD tracking for gaze input [157] leads to a Midas touch problem and performance deficiencies at larger distances [34]. Magnetic tracking, as used by the MagicLeap ML1, suffers from magnetic interference problems. Finally, outside-in tracking, such as the VIVE Lighthouse system, requires a prepared environment with a stationary infrastructure [140].

Hand tracking has the potential to support natural interaction and it is conceptually straightforward to implement on an HMD using inside-out looking sensors. However, any practical implementation of this concept faces several challenges. First, the integration of hand tracking adds to the technical complexity of the HMD and increases its hardware cost, computational load, and power consumption. Second, the user would have to keep the hands in the field of view of the

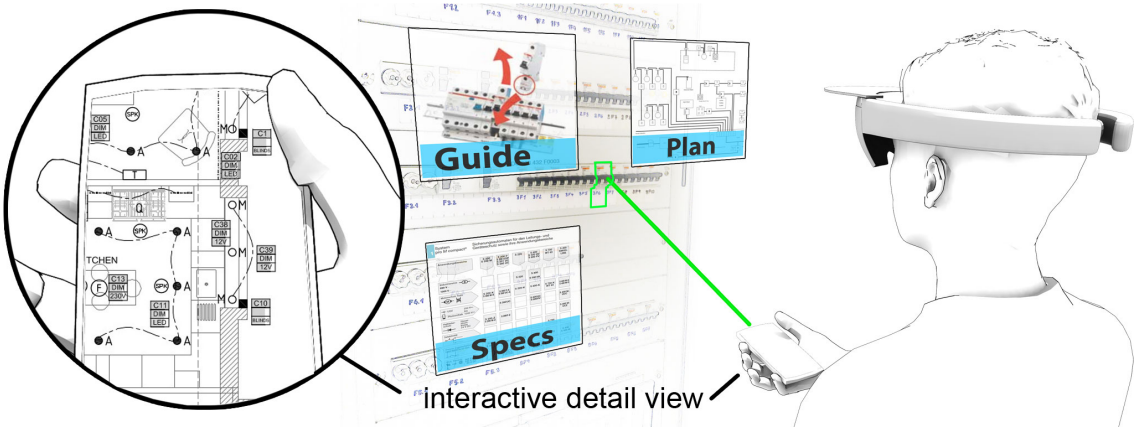



**Figure 1.8:** Using Smartphones for 3D Interaction on Mobile Head-Mounted Displays. The high input and output fidelity of the smartphone can also be used to display detail of the selected objects and to enable high precision interactions with them.

HMD sensors (see the black outline on the HMD in Figure 8.1). This may be natural for pointing gestures, but keeping one's hands permanently raised to chest level quickly leads to fatigue. Third, fine-grained manipulation in free space is difficult to perform and lacks the passive force feedback provided by a touchscreen or other surface.

In this thesis, we propose a different approach where we use a conventional smartphone as a 6DOF tracked handheld companion device for a mobile HMD. We call our approach *TrackCap*, as the phone tracks a "cap"-like structure which is mounted on an HMD. The inside-out tracking is turned around by using the smartphone's camera (rather than the HMD camera) to determine the relative pose from HMD to the device. This approach avoids all the pitfalls listed above. First, the technical requirements of the HMD remain unchanged, while an existing smartphone can be re-purposed as an inexpensive 6DOF input device. Second, the tracking only depends on a line of sight from the smartphone to the HMD and not vice versa. In other words, the smartphone can be operated outside of the forward-looking area of the HMD, for example, at hip level, as long as its camera is facing towards the HMD which we will show is a feasible assumption in most cases. Third, the smartphone can be translated and rotated with high precision, and the passive haptic feedback of the touchscreen allows for even more precise input when required. Finally, TrackCap benefits from the independence of HMD and smartphone, which enables retrofitting an existing HMD with an affordable 6DOF input device and allows to freely mix and match devices and interaction styles.

## 1.4 Publications and Contribution Statement

This thesis is based on multiple publications. In the following, all contributing publications are listed in chronological order.

- Peter Mohr, Bernhard Kerbl, Michael Donoser, Dieter Schmalstieg, and Denis Kalkofen. 2015. *Retargeting Technical Documentation to Augmented Reality.* In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15). Association for Computing Machinery, New York, NY, USA, 3337-3346. DOI:https://doi.org/10.1145/2702123.2702490

  **Abstract:** We present a system which automatically transfers printed technical documentation, such as handbooks, to three-dimensional Augmented Reality. Our system identifies the most frequent forms of instructions found in printed documentation, such as image sequences, explosion diagrams, textual annotations and arrows indicating motion. The analysis of the printed documentation works automatically, with minimal user input. The system only requires the documentation itself and a CAD model or 3D scan of the object described in the documentation. The output is a fully interactive Augmented Reality application, presenting the information from the printed documentation in 3D, registered to the real object.

  **Author's Contribution:** *The author was the main contributor to the system design and implementation of the object registration, label extraction, visualization and animation generation as well as the interactive AR components.*

- Peter Mohr, David Mandl, Markus Tatzgern, Eduardo Veas, Dieter Schmalstieg, and Denis Kalkofen. 2017. *Retargeting Video Tutorials Showing Tools With Surface Contact to Augmented Reality.* In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). Association for Computing Machinery, New York, NY, USA, 6547-6558. DOI:https://doi.org/10.1145/3025453.3025688

  **Abstract:** A video tutorial effectively conveys complex motions, but may be hard to follow precisely because of its restriction to a predetermined viewpoint. Augmented reality (AR) tutorials have been demonstrated to be more effective. We bring the advantages of both together by interactively retargeting conventional, two-dimensional videos into three-dimensional AR tutorials. Unlike previous work, we do not simply overlay video, but synthesize 3D-registered motion from the video. Since the information in the resulting AR tutorial is registered to 3D objects, the user can freely change the viewpoint without degrading the experience. This approach applies to many styles of video tutorials. In this work, we concentrate on a class of tutorials which alter the surface of an object.

  **Author's Contribution:** *The author was the main contributor to the system design and implementation of the object registration, visualization and parts of the user interface. The author designed and performed the user studies, the co-authors contributed to the concept and performed parts of the user studies.*

- Peter Mohr, Markus Tatzgern, Jens Grubert, Dieter Schmalstieg and Denis Kalkofen. 2017. *Adaptive User-Perspective Rendering for Handheld Augmented Reality.* In Proceedings of the 2017 IEEE Symposium on 3D User Interfaces (3DUI), Los Angeles, CA, 2017, 176-181. DOI:https://doi.org/10.1109/3DUI.2017.7893336

  **Abstract:** Handheld Augmented Reality commonly implements some variant of magic lens rendering, which turns only a fraction of the user's real environment into AR while the rest of the environment remains unaffected. Since handheld AR devices are commonly equipped with video see-through capabilities, AR magic lens applications often suffer from spatial distortions, because the AR environment is presented from the perspective of the camera of the mobile device. Recent approaches counteract this distortion based on estimations of the user's head position, rendering the scene from the user's perspective. To this end, approaches usually apply face-tracking algorithms on the front camera of the mobile device. However, this demands high computational resources and therefore commonly affects the performance of the application beyond the already high computational load of AR applications. In this paper, we present a method to reduce the computational demands for user perspective rendering by applying lightweight optical flow tracking and an estimation of the user's motion before head tracking is started. We demonstrate the suitability of our approach for computationally limited mobile devices and we compare it to device perspective rendering, to head tracked user perspective rendering, as well as to fixed point of view user perspective rendering.

  **Author's Contribution:** *The author was the main contributor to the system design and implementation. The author co-designed and performed the user studies, the co-authors contributed to the study concept and the evaluation of the results.*

- Peter Mohr, Markus Tatzgern, Tobias Langlotz, Andreas Lang, Dieter Schmalstieg, and Denis Kalkofen. 2019. *TrackCap: Enabling Smartphones for 3D Interaction on Mobile Head-Mounted Displays.* In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19). Association for Computing Machinery, New York, NY, USA, Paper 585, 1-11. DOI:https://doi.org/10.1145/3290605.3300815

  **Abstract:** The latest generation of consumer market Head-mounted displays (HMD) now include self-contained inside-out tracking of head motions, which makes them suitable for mobile applications. However, 3D tracking of input devices is either not included at all or requires to keep the device in sight, so that it can be observed from a sensor mounted on the HMD. Both approaches make natural interactions cumbersome in mobile applications. TrackCap, a novel approach for 3D tracking of input devices, turns a conventional smartphone into a precise 6DOF input device for an HMD user. The device can be conveniently operated both inside and outside the HMD's field of view, while it provides additional 2D input and output capabilities.

**Author's Contribution:** *The author was the main contributor to the system design and implementation. The author co-designed and performed the user studies, the co-authors contributed to the study concept and the evaluation of the results.*

- Peter Mohr, Shohei Mori, Tobias Langlotz, Bruce Thomas, Dieter Schmalstieg, and Denis Kalkofen. 2020. *Mixed Reality Light Fields for Interactive Remote Assistance.* In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20), Association for Computing Machinery, New York, NY, USA, to appear. `http://dx.doi.org/10.1145/3313831.3376289`

  **Abstract:** Remote assistance represents an important use case for mixed reality. With the rise of handheld and wearable devices, remote assistance has become practical in the wild. However, spontaneous provisioning of remote assistance requires an easy, fast and robust approach for capturing and sharing of unprepared environments. In this work, we make a case for utilizing interactive light fields for remote assistance. We demonstrate the advantages of object representation using light fields over conventional geometric reconstruction. Moreover, we introduce an interaction method for quickly annotating light fields in 3D space without requiring surface geometry to anchor annotations. We present results from a user study demonstrating the effectiveness of our interaction techniques, and we provide feedback on the usability of our overall system.

  **Author's Contribution:** *The author was the main contributor to the system design and implementation. Shohei Mori contributed the light field renderer. The author co-designed and performed the user studies, the co-authors contributed to the study concept and the evaluation of the results.*

## 1.5 Organization

This thesis presents visualization, authoring and interaction techniques for AR instructions. It is structured as follows:

**Chapter 2** presents background and related work. This includes illustrative visualization, cognitive psychology and augmented reality.

**Chapter 3** presents an authoring approach which re-uses existing printed documentation to generate interactive AR documentation with minimal user input.

**Chapter 4** presents methods for retargeting tutorial videos to AR. Several types of the most common video content are analyzed and approaches for data extraction are presented. This includes rigid object motion, motion with surface contact, deformable objects and human body motion.

**Chapter 5** presents an online authoring system for creating ad-hoc augmented reality instructions in arbitrary environments. The chapter introduces a fast light field capturing method as well as an authoring interface that supports light fields. The chapter closes with a user study evaluating the modules of the system an the discussion thereof.

**Chapter 6** discusses several visualization aspects of augmented reality instructions. This includes rendering techniques suited for AR, the indication of optimal viewpoints in 3D, user guidance and tool guidance. The section also includes several user studies evaluating tutorial systems for motion and tool guidance.

**Chapter 7** presents interaction techniques for AR documentation using smartphones. A novel, resource-efficient method for adaptive user perspective rendering on mobile devices is presented and evaluated in a user study.

**Chapter 8** introduces an interaction system for head-mounted displays. It introduces a method for utilizing an off-the-shelf smartphone as a 6DoF tracked controller for HMDs. The chapter also provides a performance analysis as well as an extensive user study evaluating the benefit of the controller.

**Chapter 9** discusses the developed AR documentation systems and summarizes the main contributions and answers to the main research questions. It closes this thesis with directions for future work.

# CHAPTER 2

# Related Work

This chapter will put our contributions to 3D authoring systems, Augmented Reality visualization and interaction in context with existing approaches. In addition, we will address the limitations of current approaches in relation to our work and point out our improvements in the areas of authoring, visualization and interaction. The chapter is thematically divided into three main parts: Augmented Reality tutorial systems and remote assistance systems, authoring AR tutorial content, and visualization and interaction aspects for instructions in Augmented Reality.

## 2.1 Mixed Reality Tutorial Systems

Mixed Reality tutorial systems have been discussed as early as the systems introduced by Caudell et al. [27] and Feiner et al. [41], for the purpose of maintenance and assembly. Reiners et al. [127] introduce an augmented reality system that guides the assembly of a door lock in a car door but admit that the tracking capabilities at that time introduce difficulties for novice users. Zauner et al. [182] deal with furniture assembly, using fiducial markers and RGB cameras for tracking. Their work introduces both an authoring and a guidance module, but also mention the limitations of the used tracking technologies.

### 2.1.1 Assembly Tutorial Systems

The spatial nature of assembly instructions makes them be a suitable application for AR. A user trying to comprehend where and how to place missing components can benefit not only from the locality of AR, but also from the freedom that the virtual content offers in terms of visualization techniques. Moreover, advances in 3D pose estimation and tracking [122, 126, 160, 177] open new possibilities for scene understanding, and therefore, enable the design of more sophisticated and efficient AR assembly tutorials. These AR systems, in comparison to traditional media, have

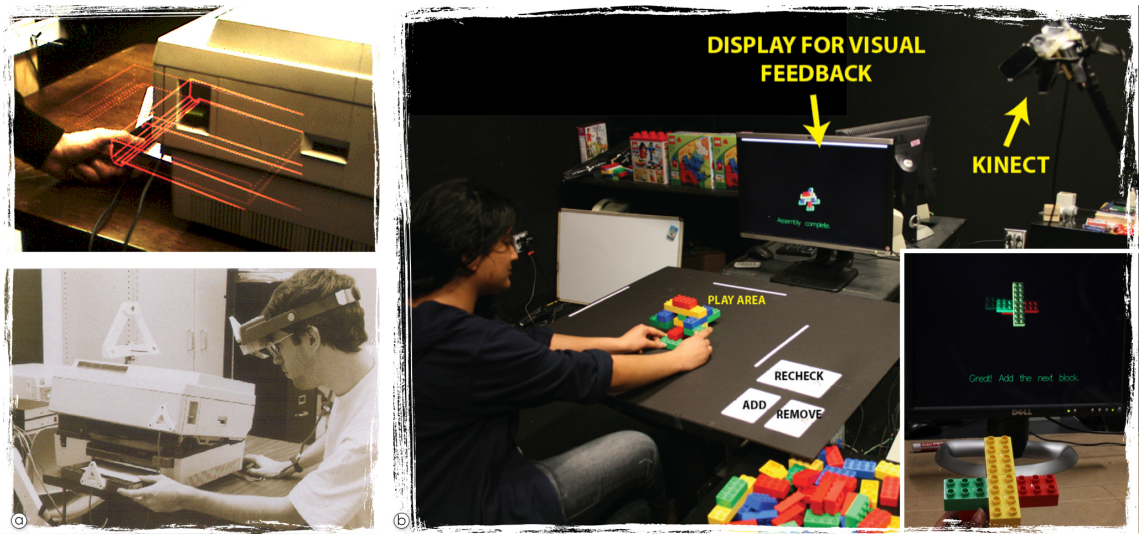



**Figure 2.1:** Early Augmented Reality Tutorial Systems. (a) The KARMA (Knowledge-based Augmented Reality for Maintenance Assistance) [41] system is a testbed system for exploring the automatic generation of maintenance and assembly tasks. Head tracking was achieved using magnetic and ultrasonic sensors. Figure adapted from [41]. (b) DuploTrack system [50]. The instruction animations are shown in real time on a screen in front of a user. The system supports an authoring as well as a guidance mode. It uses a first-generation Kinect depth sensor to detect the Lego Duplo assemblies. Figure adapted from [50].

been shown to relieve mental workload and improve task performance [154]. Henderson et al. [56] argue that AR documentation helps users to localize tasks more quickly, as well as perform fewer head movements. This is one of the main advantages of augmented reality systems, as the instructions are already registered to the real world and the user does not have to switch mentally and physically between a 2D written handbook and the actual scene.

More recent approaches include the one proposed by Wu et al. [178]. The authors introduce an AR instruction system with markerless tracking from an RGBD input, assuming that the 3D models of the objects and the disassembly graph are known. The instructions are then displayed on a computer screen using line and circular arrows. The lines and arrows are registered in the live videos of the manipulated object from an overhead perspective. This is a more indirect form of augmented reality, as the user still has to switch between the display and the real object.

A similar system by Wang et al. [168] aims to provide real-time user feedback during an object assembly procedure but without special capturing hardware. Instead, they are using only common RGB cameras as input to their system. To ensure tracking and detection are robust enough, they propose a probabilistic model to compute the most likely assembly configuration given the detected state. The visualization consists of showing the assembly graph with the selected next configuration on an external screen. Alternatively, a registered animation of a movement of the next possible added component is also shown. Similar to the previous discussed work, the instructions are visualized in an indirect way.

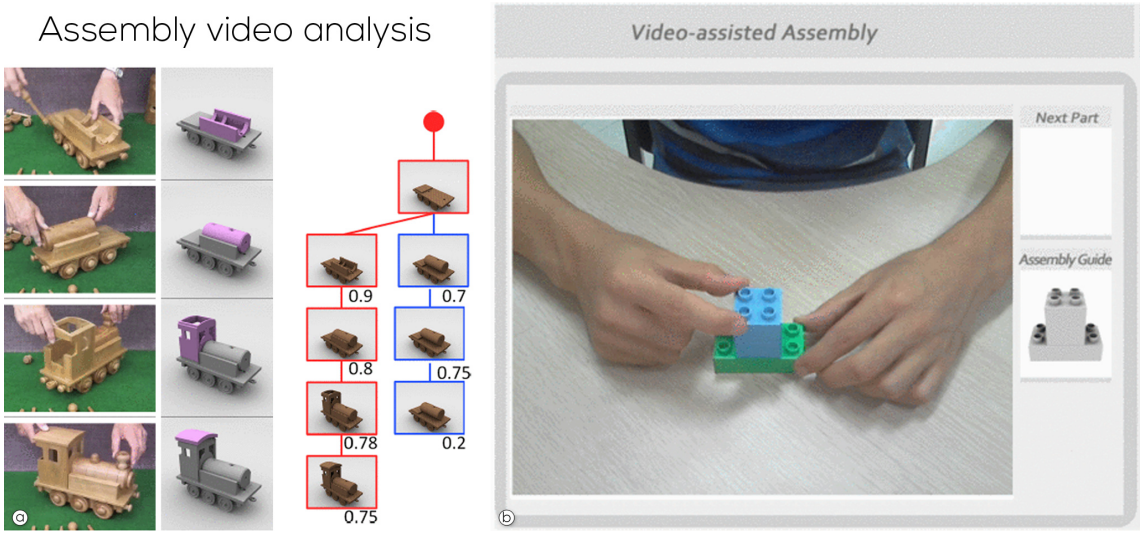



**Figure 2.2:** Active Assembly Guidance system by Wang et al. [168]. (a) The analysis of a YouTube assembly tutorial video yields a valid assembly sequence (red). (b) The system provides a basic user assistance visualization, which shows the current state, errors and next possible parts on a monitor. Figure adapted from [168].

The DuploTrack approach [50] introduces a live setup providing an assembly tutorial for Duplo blocks configurations. The instruction animations are shown in real-time on a screen in front of a user by orthogonal movements of missing pieces. The system supports an authoring as well as a guidance mode, but is limited by assumptions about the structure of the Duplo components, as for example the orthogonality of the assembly steps' direction.

All three aforementioned approaches demonstrate their capabilities on toy-like objects, which are usually chosen because of their availability and simplicity. Furthermore, they are quite easy to track, even with only RGB input.

Some of the already mentioned systems rely on markers, which need to be placed in the scene or on the manipulable objects, to be able to track the part positions relative to the camera(s). While this approach is most robust and simple, it is often unpractical or impossible to place additional markers. The solution is the so-called markerless tracking. Here, usually, the 3D geometry of trackable parts is required beforehand. Alvarez et al [7] also propose an approach for markerless tracking of object parts for an assembly tutorial with RGB input. Assuming the availability of a 3D model of the target object, they implement their own disassembly planning module. The focus of their paper is on a method for real-time 3D markerless tracking and not so much on the instructions or AR visualization themselves.

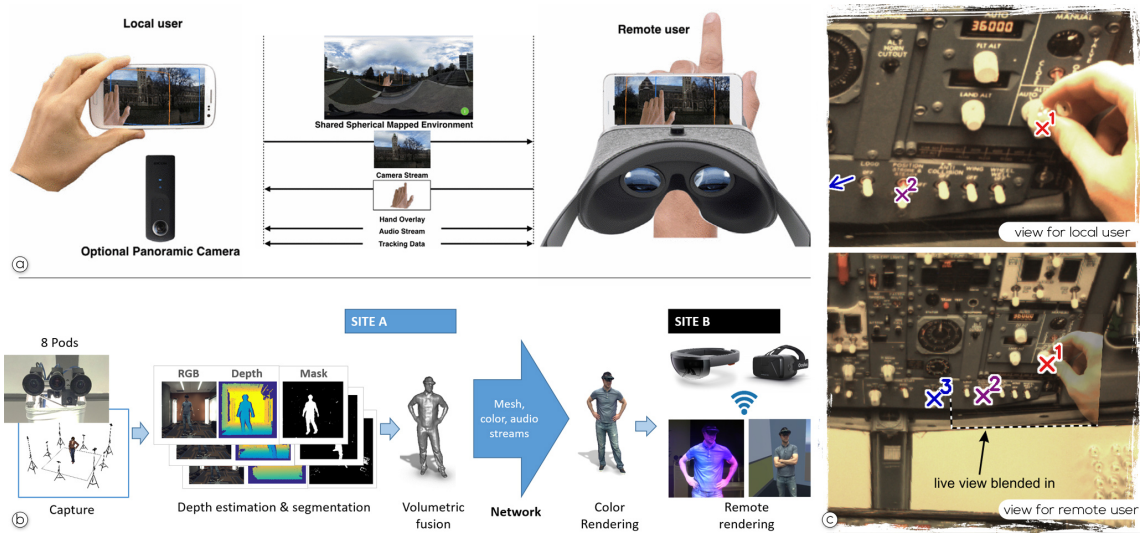



**Figure 2.3:** Augmented Reality remote collaboration systems. (a) Handheld remote assistance using panoramic views by Young et al. [180]. The local user sends their camera stream to the remote user, who in turn can stream segmented hand poses back, which are shown in the local user's view. Figure adapted from [180]. (b) Holoportation system by Microsoft [114]. The system allows real-time 3D reconstructions of an entire space using multiple depth cameras. The reconstruction can be transmitted in real-time, enabling the users to interact with each other using AR glasses. Figure adapted from [114]. (c) World-stabilized annotations in a remote collaboration scenario by [43]. The remote user can place markers in their view, which are transmitted to the local user (top). The remote user also sees a registered live camera stream of the local user's view. Figure adapted from [43].

VR training setups also have the power to improve the rehab process for patients [79]. Furthermore, it can monitor and supervise the user's motion, reducing the amount of supervision by medical personnel needed by a single patient.[171] also showed that a similar system could be used to evaluate and train professional personnel automatically. A Kinect driven system was used to track professional nurses' skeleton while performing a sheet pulling motion. This could be used to give better and more consistent training to nurses to reduce the load and strength needed to perform the sheet pull task.

### 2.1.2 Remote Assistance Systems

In the following, we look into related work in the area of remote assistance systems, with a specific focus on model representations and interaction in Mixed Reality remote assistance and interactive light field processing. Mixed Reality remote assistance has been successfully demonstrated using hand gestures performed by the remote user and spatially registration to the local user's environment [5, 64, 180]. Previous approaches rely on dedicated sensing hardware, such as a Microsoft Kinect [66, 145] or Leap Motion sensor [76].

Visual remote instructions have also been implemented by adding interactive annotations to the shared representation of the local user's environment. Drawing into the live video stream is a simple way to point the attention of the remote user to important objects and places [106]. However, such 2D overlays can only work from a static point of view [2].

Consequently, other research has explored the use of annotations registered in 3D [40]. Early systems use marker tracking to identify planes in the remote environment, where the remote user can place AR annotations [43]. Later work considered various forms of online 3D reconstruction to place AR annotations with respect to the 3D structure [44, 45, 112]. All these annotation techniques are intimately tied to the characteristics and geometric quality of the shared environment reconstruction.

Arguably the simplest form of sharing an environment is by transmitting a live camera stream captured from a single point of view [155], which today is the standard approach for video chat applications, such as Skype. These approaches commonly use static cameras and do not offer the remote conversation partner an independent point of view into the environment [151]. Since a static viewpoint limits the feeling of presence [106, 180], telepresence research using mobile devices has focused on view control. For example, on-the-fly panorama stitching allows remote users to freely rotate their view in an otherwise static environment [43, 106, 180]. Other work has considered robotic camera control. For example, Kratz et al. [82] introduced a robotic arm for letting the remote user control the position and orientation of the remote camera.

Obviously, a full 3D representation of the environment overcomes most of the issues concerning view independence. A common shortcut is to expect that the environment is scanned before the actual collaboration begins. Since this defeats the goal of spontaneous remote assistance in the field, we limit the following discussion to approaches that generate reconstructions spontaneously when required.

Kasahara et al. [76] presented an approach that creates a sparse 3D model on the fly by rendering spatially aligned keyframes from a simultaneous localization and mapping (SLAM) system. Sparse SLAM maps have also been converted into textured polygonal meshes [44, 145], yet, of general low visual quality. With advancements in depth cameras, casual scanning [39, 129] is now much more feasible than even a few years ago. However, real-time scanning with high geometric and photometric fidelity still requires better sensors and more computational power than typically available on a mobile device. Moreover, the quality of geometric reconstruction is often severely degraded for texture-less, shiny, thin or transparent objects even when high-quality scanning systems are employed, which is a major gap addressed later in this work.

## 2.2 Authoring Mixed Reality Tutorials

Documentations can exhibit a large variety of graphical elements, but they usually follow established conventions. In the course of research for this work, we identified the most frequent elements – in the following called diagrams – appearing in books [98], online databases of popular products[1][2] and scientific publications [4, 55, 88].

A number of different strategies have been proposed to author Augmented Reality manuals. The two most common methods define behavior either directly by using commercially available 3D animation and modeling software, such as Maya [3] or 3DStudio MAX[4], or by scripting all actions and transitions between actions using specialized script languages [24, 85]. In both approaches, the user must define the manual from scratch. Generating computer-based tutorials traditionally involves the creation of dynamic glyphs, i. e., graphical elements, to present the path and direction of motions [110].

In AR, arrows are commonly used for this purpose. For instance, Barakonyi et al. [12] use animated arrows to convey actions in an assembly application. However, manual animation and registration is very time and cost-intensive, research has aimed at automating the authoring process. The automatic generation of AR instructions goes back to the pioneering work on KARMA [41], which uses rules to derive graphical representations, as proposed by Seligman and Feiner [142]. However, KARMA requires a manually created knowledge database to derive instructions from.

Script languages, drawing from similar inspiration, have been proposed by multiple authors, such as Butz [24] or Ledermann et al. [85]. However, authoring by scripting always requires both a formal knowledge about the procedure and programming skills.

Therefore, systems that require less user input have drawn attention. Agrawala et al. [4], Li et al. [88], Kalkofen et al. [74] and Kerbl et al. [77] showed the automatic generation of disassembly instructions for rigid objects. Mohr et al. [104] demonstrated the automatic generation of 3D animations from images depicting an assembly sequence. All these systems derive a sequence of removing parts from a CAD model with straight motions. Unfortunately, these methods are unsuitable for content that involves complex motions or deformable objects, such as for sports or dancing.

The authoring of more complex actions has been proposed by capturing the necessary steps. For example, Grabler et. al demonstrate the generation of photo manipulation tutorials [49] from recorded actions in a photo-manipulation tool, and Chi et. al mixed 2D images and video material to generate tutorials [29]. The works in this thesis follow the idea of recording and replaying actions and lift it to 3D AR environments.

[1] http://service.lego.com/en-us/buildinginstructions

[2] http://www.ikea.com/ms/en_US/customer_service/assembly_instructions.html

[3] www.autodesk.com/maya

[4] http://www.autodesk.com/products/3ds-max/overview

AR tutorials that involve complex motions typically rely on some sort of specialized 3D motion capture. Examples include assembling furniture [182], assembling Lego toy sets [50] or gestural commands [172]. While 3D motion capture is the obvious way of providing input, these systems handle rather simple motions. For complex motions, the process can be tedious and requires expensive capturing hardware (e. g., Vicon or Optitrack). More recent approaches make use of deep learning methods to estimate human skeletal poses from single or multiple 2D camera inputs, such as OpenPose [26]. The limitations of these approaches manifest themselves usually when confronted with atypical poses or occlusions. Most of them also do not work on partially visible human shapes.

The very recent work from Chi et al. [30] captures complex motions from which the system generates illustrative step-by-step diagrams. While the generated drawings are optimized for effective communication, the system presents the results in 2D, rather than registered in 3D AR. Similarly, AR instruction systems often make use of the idea of a 2D mirror to show the instructions. Examples include AR instructions for physiotherapy [156] and dance [8] applications. While this concept has been demonstrated to be effective, it is limited to body-centric instructions.

An alternative approach is to build AR manuals from recorded videos. Several works [35, 48, 84, 118] forgo the use of 3D capturing and, instead, overlay recorded 2D video directly in the AR display. This allows to author complicated manuals without additional knowledge about any programming language or animation tool. Using 2D video has the advantage that the richness of the original demonstration is preserved without requiring any spatial or semantic interpretation. However, an AR experience playing back registered 2D video does not allow a truly free choice of viewpoint, and the video occupies substantial screen space. Furthermore, if the video is viewed from a different point of view than the one it was recorded from, it has to be warped to the new point of view, which is prone to rendering artifacts. These limitations severely restrict the practical value of this approach. In contrast, the system of Damen et al. [35] provides the video tutorial using a heads-up display which allows clearly seeing all objects. However, no direct augmentation is given why the user has to mentally match landmarks between real objects and video data.

One of the approaches discussed in this thesis is inspired by the work of Li et al. [89], who generate an animated 2D explosion diagram from a single image. Their system requires the user to manually define all parts and all animations in the diagram. However, this is clearly not practical for more complex structures or multiple images in a sequence.

Recently, Shao et al. [143] presented a system that derives simple animations from interpreting 2D sketches. This reduces the effort required to generate animations between two images. However, this approach only works for simple structures. Since the user has to generate proxy geometry for every single part of the object, this system can handle only objects which consist of a small number of parts. Furthermore, the user has to re-create the 3D model for every image. Therefore, this approach does not scale easily to image sequences and large product databases.

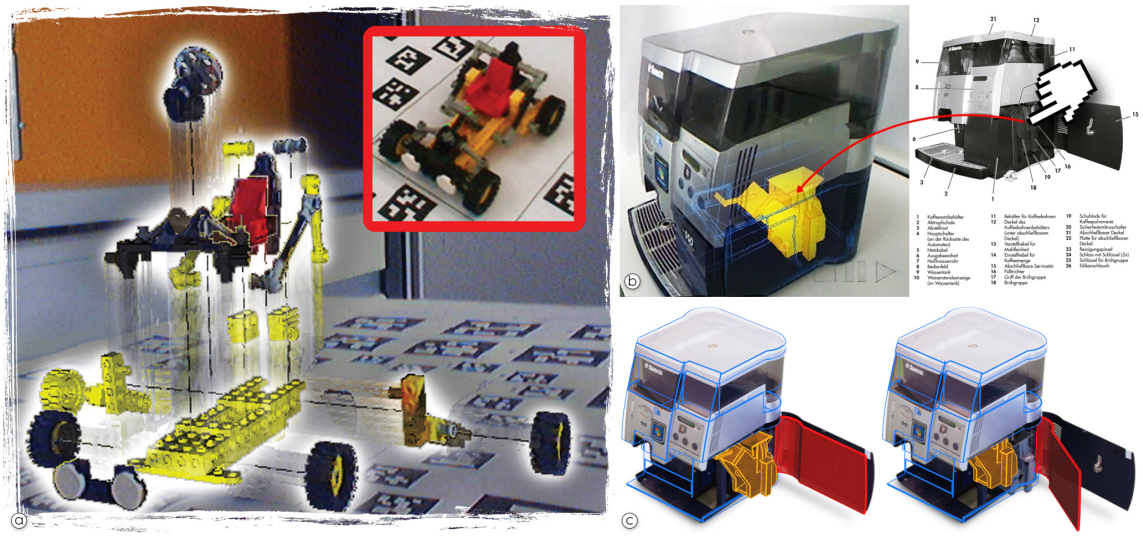


**Figure 2.4:** Authoring instructions for mixed reality. (a) Explosion Diagram in Augmented Reality. Kalkofen et al. [74] displace real-world information using a phantom rendering method. The connecting lines help the user to understand spatial object placement as well as a hierarchical grouping. Figure adapted from [74]. (b) Augmented reality instructions automatically generated from 2D printed manuals. (c) The rotation of the coffee machine's door and the removal of the brewing unit are automatically generated to match an illustration in the printed manual [104].

Bergig et al. [16] present a system that is able to automatically generate 3D reconstructions of simple 2D sketches. Even though this system does not require any interaction to derive 3D geometry, it is limited by the capabilities of the sketch reconstruction approach. Their approach can handle only a few simple shapes. In contrast, our work aims at the transfer of 2D documentation of complex structures, which may even be presented in multiple different configurations.

In some cases, it is required to extract an animated 3D representation of the scene from the video data. High-quality approaches exist [32]. However, they require expensive capturing hardware in specially prepared environments. Scene modeling for interactive AR is commonly done by the registration of phantom models, i. e., a 3D model that represents an object in the scene [73]. Depending on the required accuracy, 3D models may either be carefully prepared offline or interactively generated at run-time [28, 152, 164, 183].

In this thesis, we present an approach to author instructions including human motion, which is also related to computer animation research, which extracts 2D motion from video data and uses it for illustration [33, 78]. However, in contrast to these works, we will interpret all motion in 3D. The generation of 3D motion capture data from video sequences has been a topic of research in computer vision [103]. Most purely vision-based approaches provide solutions for specifically learned motions [3], such as walking or running. Few approaches, such as the work by Wei and Chai [169], are able to capture 3D skeletal motion from monocular video data. However, the human motion data is returned without any relation to the real world. Our work extends the one

of Wei by a new interaction technique to resolve the ambiguity of human poses by supporting arbitrary motion, and by registering the resulting data to 3D objects within the user's 3D environment. The system described in this thesis is an immersive learning environment in VR which is, as concluded by He et al. [54] favorable to a traditional 2D experience.

## 2.3 Visualization and Interaction

### Animation

Tutorials often make use of animated or video instructions that allow users to follow instructions while working on a task. These animations are usually segmented into distinct steps so that users can work along at their own pace [48, 50].

The benefits of using animated instructions can be explained by the cognitive load theory (CLT) that considers the influence of instructional design and the cognitive architecture on information processing [150]. CLT states that information processing depends on the intrinsic nature of the material that must be comprehended (intrinsic cognitive load) and the presentation of the material itself (extraneous cognitive load). Both intrinsic and extraneous cognitive load strain the limited working memory [101] of observers, which can prevent the understanding of instructions.

Intrinsic cognitive load cannot be reduced. Therefore, one goal of instruction design is to reduce extraneous cognitive load by providing appropriate presentation formats for the information material. A meta-review of Höffler and Leutner [62] revealed that instructions using animations and videos outperformed the use of static images. The effect was stronger for instructions related to the acquisition of procedural-motor skills such as assembly tasks.

Findings of Ayers et al. [9] and Wong et al. [176] enforce this further. Their results indicated that animations are especially suited for instructions that require observers to follow human movement. While Höffler and Leutner [62] refer to the benefits of using continuous animations, animations have been found to be more effective and reduce extraneous cognitive load when presented in logical segments [17, 147], and when the viewpoints of the instruction and the viewer are aligned [42].

The design of effective assembly instructions has been investigated by Heiser et al. [55]. Their results have been successfully applied to create algorithms to automatically create step-by-step instructions [4]. Heiser et al. [55] identified action diagrams depicting step-by-step assembly instructions as preferred representation. An action diagram shows the attachment of a single major part to an assembly, including the required smaller parts, such as fasteners. Furthermore, occlusions should be avoided, which may require viewpoint changes in the instructional visualization. Results of Heiser et al. [55] also indicated, that for the initial orientation of users, a realistic depiction of the assembly instructions would be beneficial. This is in line with the findings of Höffler and Leutner [62] that more realistic instructional animations were more effective than CAD-style animations.

Another problem with showing instructions in an immersive VR environment is the requirement of a hands-free control. This requires the instruction to be synchronized temporally and spatially with the user. [6] Alexis et al. align the movements of a dancer temporally and spatially in three dimensions using a quaternionic approach to identify resemblances by comparing the motion signal shape. This only works if the user is required to have the same pace as the instructor, as it is in dancing.

### Rendering and perspective

Salamin et al. [134] show in an augmented reality experiment that a third-person perspective is preferable for displacement and interactions with moving objects to most users. The advantage of the first-person perspective mainly lies within the faster adaption time and object manipulation close to the user. User perspective rendering (UPR) approaches have been primarily investigated for video see-through handheld Augmented Reality systems such as smartphones or tablets.

Baricevic et al. evaluated the effects of display size on UPR and found that, using a simulation, a tablet-sized display allows for significantly faster performance of a selection task compared to a handheld display and that UPR outperformed DPR for a selection task[Baričević et al.]. They also prototyped a UPR system with geometric reconstruction using a Kinect for the reconstruction of the physical surrounding and a Wiimote for head tracking. In follow up works, they proposed to replace an active depth sensor by gradient-domain image-based rendering method combined with semi-dense stereo matching [13, 14]. Other authors also employ depth-sensors for scene reconstruction in UPR [163].

Tomioka et al. and Hill et al. proposed an UPR implementation through transforming the back-facing camera image using homography [57, 161]. Samini et al. investigated UPR when using an external outside-in tracking system for spatial registration and proposed a geometric correction scheme for introduced registration errors [135].

Pucihar et al. investigated fixed point of view UPR (FUPR) versus DPR in a target acquisition task with and without scene continuity across device boundaries [166]. Specifically, they assume that the user's face is in a fixed and predetermined position while interacting with the system They found that most users who never experienced handheld AR before actually expected UPR as the default mode of presentation. The study also indicated that UPR outperformed DPR in terms of accuracy, task completion time, subjective workload and preference. They later extended their investigations to specifically study the use of surrounding visual context in a map navigation task [167] and to sketching applications [125]. Pucihar et al. also proposed a specific variation of UPR, called contact-view, which allows pseudo transparent rendering of documents when a smartphone lies directly on the document [123, 124], achieving similar effects compared to proprietary solutions using transparent displays [59, 60].


**Figure 2.5:** Light field capture guidance and rendering. (a) Active user guidance for light field capture using smartphones [18]. The visualizations in the bottom illustration show the capturing path over time. (b) Unstructured light fields [36]. The illustration shows the poses of all captured images for this single light field using a common handheld camera. (c) The resulting rendering of the light field using the images captured in (b). Figures adapted from [18, 36].

Grubert et al. employed UPR on mobile devices by combining head tracking using the built-in front-facing RGB camera for head tracking and natural feature-based tracking of the AR device using back-facing RGB camera [141]. Recently, Samini et al. compared UPR and DPR for a find-and-select and a 3D object manipulation task [136]. While they found DPR to outperform UPR in terms of task completion time for the find-and-select task, both approaches where on par for an object manipulation task, and UPR was preferred by users.

The approach discussed in this thesis is specifically targeting mobile devices which offer limited computational resources. Therefore, we have positioned our system between fully dynamic, but resource-intensive UPR approaches relying on constant face tracking and the fixed point of view UPR approach (FUPR) of Pucihar et al. which is only applicable in constrained application scenarios, such as looking straight onto a plane parallel to the device.

### Light fields: rendering and interaction

One key idea of our work is to use an unstructured light field of the remote environment instead of a textured surface model to overcome the constraints of existing approaches. A light field is a collection of light rays passing through space [47, 87]. Rendering a light field does not require any geometrical approximation of the remote environment and supports a large variety of objects and material properties. Light fields directly capture photometric appearance, enabling the reproduction of highly detailed geometry and complex materials.

Light fields require densely spaced images. Therefore, light field capturing has traditionally used special setups such as camera arrays [173], microlens arrays [109] or focal stacks [117]. Since the required hardware is often not available, single-camera acquisition in combination with user guidance for light field capturing has been proposed as an alternative [18, 36]. Recent approaches determine sampling requirements in real-time and provide visual feedback to guide the user to discrete positions required for capturing a dense light field [99]. Our approach is inspired by these methods but does not aim to capture a complete light field. Instead, to best preserve bandwidth, we collect just enough information to enable the remote user to annotate the representation in 3D.

One specific challenge of light fields is the lack of 3D surface information, which affects the ability to interact with light fields using traditional editing tools. Therefore, Jarabo et al. [69] investigated WIMP interfaces for editing light fields using multi-view techniques and manual adjustment of the focus plane of the light field. Both techniques allow overcoming the lack of geometry. However, the former is time-consuming, while the latter introduces the need for continuous adjustment of the focal plane, which distracts from the actual editing task. More importantly, their work also integrates 3D that can be reconstructed from light fields although in a computationally expensive approach. Instead, our work aims for mobile devices tracked in 3D instead of 2D WIMP interfaces. In addition, we cannot rely on depth knowledge, because it would be too expensive and time-consuming to recover, or, even worse, might not possible at all because of the challenging material characteristics. Instead, our approach works in the wild and on mobile devices by using 2D image information together with automatic adjustment of the focus plane.

In summary, light fields have not been utilized in mixed reality and remote assistance as their challenges (capture and interaction) have so far out-weight their advantages (visual quality). In this work, we introduce a system and user interface showing how to overcome these challenges.

### Interaction

Support for spontaneous object selection and manipulation are fundamental requirements for most 3D interactions. Grasping virtual objects with bare hands or instrumented gloves is arguably the most natural selection technique [20]. Glove devices may include tracking of finger motion, while a 6DOF wrist tracker provides the position and orientation of the wrist in the scene [63]. However, simple gloves suffer from real-virtual interpenetration issues. The user can penetrate a virtual object, thus, breaking the immersive experience. Haptic feedback devices [19] can increase the realism and detail of virtual grasping techniques [68, 153]. Unfortunately, current mobile systems cannot integrate such devices, as they are usually heavy, stationary and computationally intensive. Recent commercial depth sensors provide an opportunity for inexpensive hand and finger tracking [158]. In combination with mobile projection systems, they can even provide passive haptic feedback while interacting with real surfaces [174]. Several systems turn real surfaces into interactive displays for mobile interactions [75, 102] including human skin [51, 138]. However, interacting with real surfaces requires appropriating such surfaces first. Consequently, the interaction is not generally applicable for mobile users.

**Figure 2.6:** Interaction with AR content. (a) Grasping virtual objects with instrumented gloves. Figure adapted from [19]. (b) ManoMotion VR glove with multiple bend sensors for each finger. The rotation of the wrist is tracked with a gyro / accelerometer combination.

The most widely used alternative to virtual hand input is raycasting. The user controls a ray, and selection is determined by intersecting with the closest object in the scene [165]. Raycasting offers two distinct advantages: It enables operation over longer distances and it is flexible in how the ray origin and direction are specified. For example, raycasting can rely on full 6DOF control of the origin and its direction. However, implementations of raycasting with orientation-only sensors [58] based on inertial measurement units [128], suffer from drift and require frequent re-calibration, which interrupts the interaction.

To provide 6DOF input, gaze-based techniques and handheld controllers have been considered, which, until recently, required stationary tracking systems. With robust outside-in tracking of mobile HMD types, gaze-based techniques can be applied to selection [71]. However, studies indicate that gaze-based selection techniques are slower than traditional hand-based methods [34] and seem to limit the user's ability to recall the environment [157]. Also in particular for phone-driven HMDs such as Google's Daydream, the future integration of gaze tracking is unlikely.

Finally, Microsoft's Mixed Reality headsets provide special controllers that can be tracked using the camera that is mounted on the HMD[5]. While these systems provide precise 6DOF tracking, they require a tethered connection to an external desktop or notebook computer. An even more severe restriction is that field of view of the cameras mounted on the HMD restricts the range in which interaction can be performed.

[5]https://docs.microsoft.com/en-us/windows/mixed-reality

Very recently, also magnetic tracking solutions have been applied to mobile systems as well[6]. However, magnetic systems commonly require a calibration that is specific to a single environment [80] and needed to reduce electromagnetic interference to acceptable levels. This makes magnetic systems difficult to use in mobile applications that are required to function in unknown environments.

Vision-based tracking has been explored before to enhance the interaction capabilities of controllers, e.g., the VideoMouse to extend the degrees of freedom of a standard mouse [61] or to interact with public displays using a phone [130]. However, these approaches still relied on stationary hardware demonstrated in a constrained environment and were often constrained such as to only measure tilt in certain ranges.

[6] https://www.magicleap.com

# CHAPTER 3

## Authoring Augmented Reality Documentation from Images

Our system generates interactive AR presentations from 2D documentation, given as a collection of images, and a 3D CAD model of the target object. The CAD model must be structured such that individually movable parts of the target object can be distinguished. If no CAD model exists, we use an RGB-D sensor (Microsoft Kinect) to obtain a 3D scan which can only be used for annotation transfer. However, oftentimes a 3D CAD model will already be available from the manufacturer of the target object, and the RGB-D sensor is only required for registering the CAD model with the real-world target object.

In this section, we introduce the major types of diagrams that appear in common documentations, and we outline our approach to analyze each of them. After outlining our approach we provide details in the remaining sections of this chapter. The result of our analysis is a 3D scene description including 3D animations representing instructions. Our results can be presented on a desktop PC using a Virtual Reality (VR) viewer or in the user's real-world environment through an Augmented Reality display.

**Annotated diagrams.** A typical illustration is an annotated diagram, which allows identifying parts of an object by external labels, connected with leader lines to the referred parts (Figure 3.1a). The labels are often cross-referenced with more extensive textual descriptions.

The first step for interpreting annotations (and any other diagram considered in this thesis) is to determine camera parameters that were used to create the image, with respect to the coordinate systems used in the corresponding CAD model. Rendering the CAD model with the obtained parameters allows us to determine which part of the target object is covered by a given pixel in the image. Around the target object, we detect leader lines, identify the part of the target object from which the leader line originates and decode the text labels attached to the leader line.

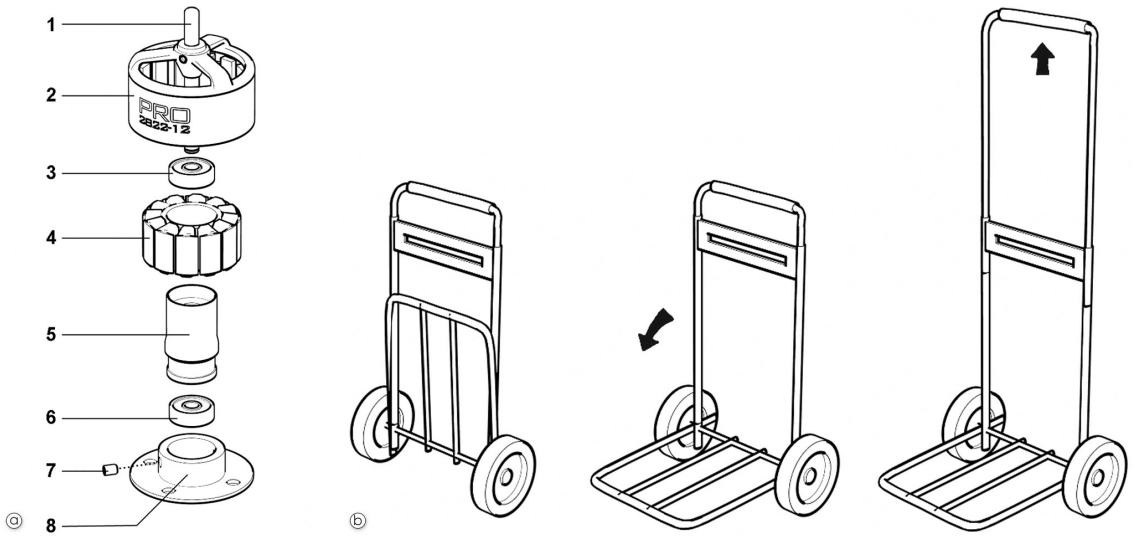



**Figure 3.1:** Diagrams used in traditional 2D documentation. (a) Explosion diagrams are commonly used to present the structure of an object. In addition, annotations identify parts. (b) Sequences of images are often used to represent an action. The images show the object in key poses. Occasionally, arrows are presented to demonstrate the necessary transformations of parts to perform the action. Images adapted from [98].

**Action diagrams.** Action diagrams use auxiliary diagrammatic elements to present instructions within a single image. A common form uses arrows to encode the transformations which have to be applied to a part to perform the presented action (Figure 3.1b).

Like annotations, arrows are complementary graphical elements that cannot be derived from a CAD model. However, their special shape allows detecting them in the image and interpreting the intended motion and direction. The intention of the arrow can be interpreted by comparing its pointing direction to valid displacement directions of parts nearby.

**Explosion diagrams.** Another popular form of technical reference is the explosion diagram, which reveals the internal structure of an object in a single image. Explosions reposition each part of the object along one or more explosion axes. The position of parts is determined based on the structure of the assembly, i. e., parts are arranged on each explosion axis according to the order in which they can be removed (Figure 3.1a). The resulting diagram avoids occlusions and simultaneously encodes blocking relationships between the individual parts.

Detecting which parts are shown in which displaced position is an essential problem considered in this thesis. To retarget explosion diagrams, we use disassembly planning [175] to determine the order in which parts may be removed and generate valid displacements for candidate parts. The candidate parts are then rendered with these displacements, and the resulting image is compared to the input image using robust image matching techniques.

**Structural diagrams** Complex instructions are most commonly represented by image sequences, where each image represents one step of the procedure [55]. The most basic form is the structural diagram, where each consecutive image adds, removes or reconfigures one or multiple parts. Since each image portrays the object at a single point in time only, the procedure has to be interpreted by comparing one image to the next and identifying their differences. Note that structural diagrams change the structure of the object from one image to next, while explosion diagrams offset parts to present the structure of an object.

To retarget structural diagrams, we must thus determine which parts are added, removed, translated or rotated in each consecutive image. By using motion planning, we obtain a set of candidate configurations of the CAD model, which we can use to search for the one depicted in the diagram. To reduce the search space, we can use robust image differencing to determine the area of change. For each image pair, we use the area of change to discard motions occurring outside of these regions.

**Combined Diagrams** Documentation often combines multiple types of diagrams. For example, labels may be used in an image sequence to name the relevant parts of the presented instruction (Figure 3.1a). Also, partially exploded objects or sequences of action diagrams (Figure 3.1b) are commonly used in image sequences to present both an action and the object configuration after applying it to the object. Our system is able to analyze and retarget all these combinations.

## 3.1 Retargeting Annotated diagrams

We transfer 2D image data to 3D space by projecting a 3D model of the object of interest to 2D image space. If this rendering of the object fits its input image, we can relate image elements to the 3D structure. We segment all labels and their corresponding leader lines in image space. Subsequently, we generate 3D annotations relative to the 3D model by using the relations we found between image elements and 3D structures. This process is illustrated in Figure 3.2.

### 3.1.1 Estimating camera parameters

For the subsequent analysis steps, the 3D model needs to be rendered using the same camera parameters which were used to create the original input image. Since the original camera parameters are usually unknown, we need to approximate the intrinsic and extrinsic parameters. In principle, these parameters can be found automatically using the method described in [184], however, in practice, the automatic extraction of reliable 2D-to-3D point correspondences from a single, highly stylized 2D image is not robust in many cases. Thus, we opted for an interactive definition of point correspondences and field of view by providing a basic user interface (UI). The UI features a side-by-side arrangement of a 2D image viewer and a 3D object viewer, wherein the user can select four or more points both in the input image and on the freely rotatable 3D model

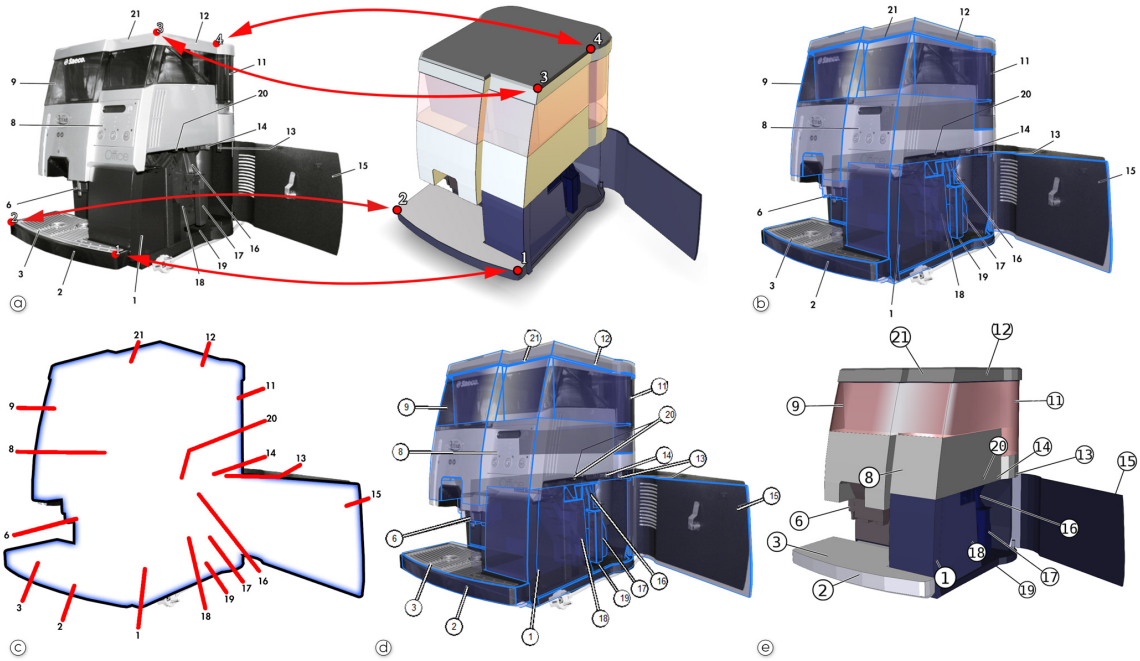


**Figure 3.2:** Retargeting annotations. (a) To estimate the camera pose, the user interactively defines corresponding points between the 3D model and the input image. (b) With the estimated camera pose, the rendering of the 3D model fits the 2D input image. This is illustrated by adding the rendering as semi-transparent blue layer on top of the input image. (c) Searching for MSER with an eccentricity close to 1 allows identifying leader lines. The mask defined by the rendering enables to detect their starting and ending points. (d) Running an OCR tool around the endpoint decodes text labels (e), which allows adding 3D annotations to the 3D model.

(Figure 3.2a). The camera pose is then determined automatically using the POSIT algorithm [37], which solves the 2D-3D point correspondences. This procedure needs to be done only once for a set of instruction images that use the same camera parameters, which covers the majority of examples we have analyzed.

### 3.1.2 Annotation detection

By computing maximally stable extremal regions (MSER) [97], we search the 2D documentation for leader lines and labels. MSER provides an efficient means for segmenting connected components in an image. For every connected component, we calculate its eccentricity. If the eccentricity value is close to 1, we assume that the region contains a leader line. We then determine the endpoints of the line by fitting an ellipse to the region. Rendering the CAD model using the camera parameters estimated in the previous step yields a mask for looking up whether a pixel is occupied by the model. The pixel coincident with the endpoint inside the mask is used to look up the annotated part and defines the 3D anchor point, where the leader line is attached to the 3D object. The region around the other endpoint is automatically scanned with a standard optical character recognition (OCR) tool to decode a text label.

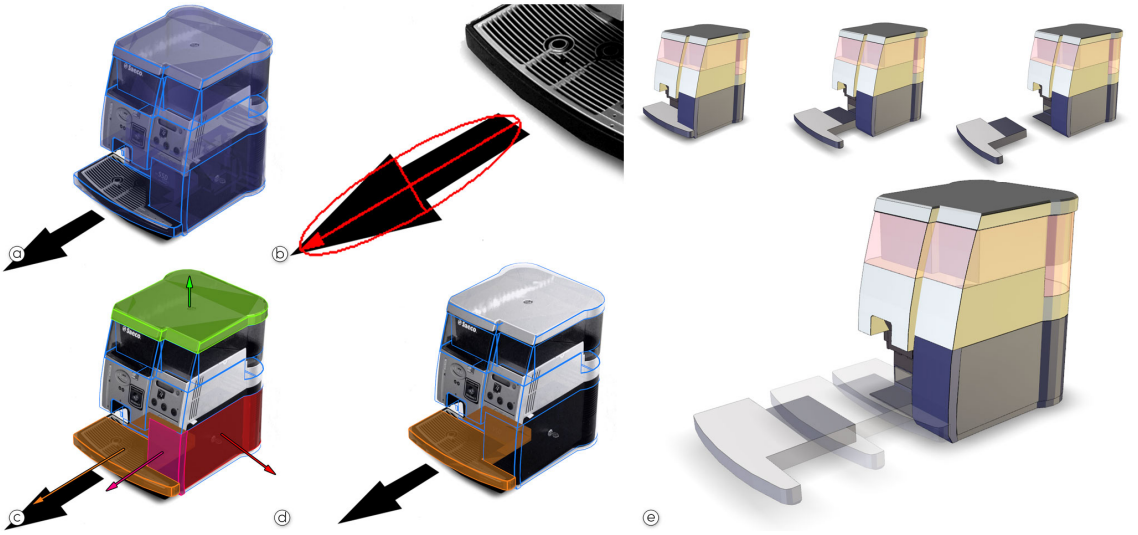


**Figure 3.3:** Retargeting action diagrams. (a) The input image with a semi-transparent overlay of the 3D CAD model. (b) We detect the arrow and its direction in 2D image space. (c) We analyze the 3D CAD model to find all removable parts and their directions. (d) By comparing the assembly directions to the direction of the arrow, we reduce the set of possible part motions. The part which is closest to the shaft of the arrow is selected. (e) The animation is defined by the path which was derived during motion planning.

## 3.2 Retargeting Action diagrams

Action diagrams encode instructions within a single image, mostly by using arrows. Therefore, to retarget action diagrams, we need to identify and interpret arrows and the parts of the target object they are referring to. In most of the cases, the arrows are superimposed on a photograph of the object using a regular 2D illustration software, thus we cannot estimate a 3D position for the arrow similar to the approach described in Section 3.1.1. Our method combines 2D image processing to find direction and 2D position of the arrows and 3D motion planning to find possible reference objects as well as 3D trajectories.

We start by detecting and analyzing each arrow in 2D space (Figure 3.3b). Subsequently, we use motion planning to identify the set of geometrically feasible motions of each part (Figure 3.3c). We then identify a set of parts the arrow might refer to by selecting those which can be moved similarly to the motion encoded in the arrow. To present the action in 3D, we animate the selected part using the corresponding movement, as suggested by the motion planner (Figure 3.3e).

### 3.2.1 Motion planning

We assume that input images show only geometrically feasible configurations, which have been generated by moving parts without penetrating other parts of the object. To find such valid motions for each part in the CAD model, we need to test whether or not a motion causes collisions.

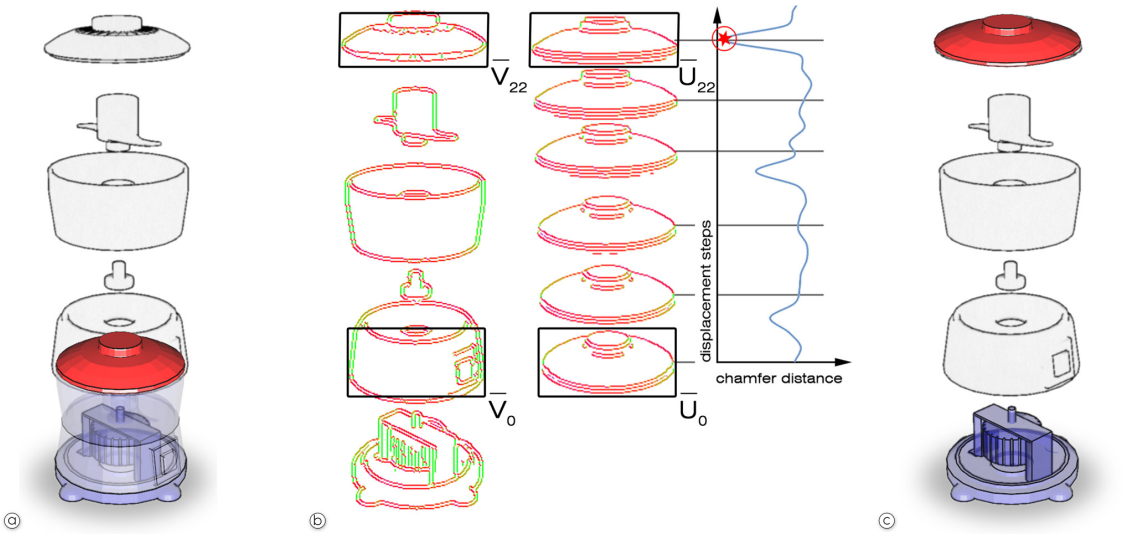



**Figure 3.4:** Retargeting explosion diagrams. (a) The 2D input image and the object rendered semi-transparently using the estimated camera parameters. The motion planner generates candidate places for the lid (red part) first. (b) Our matching compares renderings of the lid within its bounding rectangle $\bar{U}$ at geometrically possible locations $\bar{V}$. The orientation of the local gradient, as used during image matching, has been color-coded from red (horizontal) to green (vertical). The chamfer distance between $\bar{V}_i$ and $\bar{U}_i$ has been plotted in the line chart. The red star in the line chart highlights the minimal chamfer distance $\bar{U}_{22} \rightarrow \bar{V}_{22}$. (c) The lid is placed corresponding to the analysis.

Using Minkowski differences of polyhedra [94], we detect for a given direction of motion how far each part can be displaced. By default, we test for the principal axes in local and global coordinates of the part.

### 3.2.2 Arrow interpretation

In order to identify an arrow, we begin by computing MSER regions outside the target object. Among all candidates with connected components, we select the ones with exactly two concavities along the boundary, and we determine the main axis by ellipse fitting (Figure 3.3b). The tip of the arrow is the extremal point on this axis closer to the concavities. To identify candidate actions, we search for parts that could be removed in the direction indicated by the arrow. Therefore, we project the motion vectors that have been computed during motion planning to image space using the camera parameters we have estimated before (Figure 3.3c). We compare the projected directions to the direction of the arrow. We select part motions as candidates if the angle between their projected motion vector and the arrow vector is below a given threshold. If there is more than one candidate, the part which is overlapping with or which is closest to the shaft of the arrow is chosen (Figure 3.3d).

## 3.3 Retargeting Explosion diagrams

In order to retarget explosion diagrams, we need to find a sequence of unblocked motions of parts such that the resulting 3D scene setup visually matches the input 2D explosion diagram. The problem of detecting such sequences has been considered in assembly planning [4, 46, 175]. Thus, we incorporate our motion planning system into a sequencing approach similar to Srinivasan et al. [148]. This generates a set of feasible candidate explosions in 3D space. We find the candidate that matches the input image best by comparing it with the rendered images of each candidate setup (Figure 3.4).

By using only motions that we identified as geometrically feasible, we generate candidate displacements of each part in discrete locations. The CAD model is rendered to a texture with its parts displaced according to feasible motions. The resulting image $U$ is compared to the input image $V$ using a variation of directional chamfer matching (DCM), as given in Equation 3.1 [92]. To emphasize the relevant edge features of the CAD model for DCM, we assign a random color to each part and add contour lines to the scene before rendering. We then apply a Canny edge detector [25] and store the normalized vector of the locally detected 2D gradient in $\hat{U}(\mathbf{u})$ and $\hat{V}(\mathbf{v})$, respectively. Our notation employs the zero norm $||\cdot||_0$ to count the non-zero pixel values in an image. With a small constant $\varepsilon$, we obtain a continuous distance function $d_{DCM}$ for comparing images $U$ and $V$:

$$d_{DCM}(U,V) = \frac{1}{||\hat{U}||_0} \sum_{\mathbf{u_i} \in U} min_{\mathbf{v_j} \in V} \frac{|\mathbf{v_j} - \mathbf{u_i}|}{|(\hat{U}(\mathbf{u_i}) \cdot \hat{V}(\mathbf{v_j}))| + \varepsilon} \tag{3.1}$$

Once we find the correct sequence, we are able to generate animations by consecutively applying the motions that were identified by our candidate evaluation algorithm. The example shown in Figure 3.5 shows output keyframes of a generated 2D animation for collapsing the input explosion diagram. Since we use the rendered 3D model of the object, all occlusions are calculated correctly without further user intervention.

### 3.3.1 Reducing the size of the test area

Calculating the DCM over the entire image for each candidate explosion is computationally demanding. However, if a part is visible in the explosion diagram, we search specifically for this part at valid candidate locations. We generate a new image template $\bar{U} \subset U$ for each candidate motion by projecting the displaced 3D part into 2D image space and calculating its bounding rectangle. The search space in $V$ is restricted to the matching bounding coordinates $\bar{V} \subset V$. We refer to the procedure of finding the place of the template $\bar{U}$ in $V$ as local template fitting (see Figure 3.4(b)).

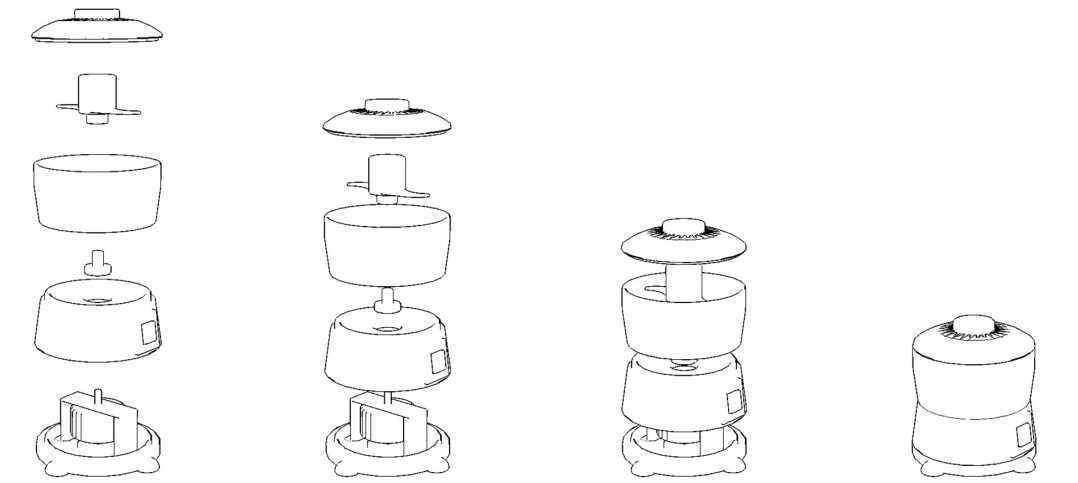


**Figure 3.5:** Retargeting explosion diagrams. To replicate the entire 2D explosion diagram, we search for the optimal placement of each part. We generate animations by consecutively applying the motions that were identified by our candidate evaluation algorithm. This example shows output keyframes of a generated 2D animation for collapsing the input explosion diagram. Note that occlusions are correctly resolved with the 3D model.

### 3.3.2 Reducing the number of tests

While restricting the image matching to the bounding rectangles of displaced parts helps to increase system performance, the number of possible combinations of part displacement easily becomes very large. Even with the restriction to geometrically feasible motions, the search space grows quickly with the number of parts and candidate displacements. Thus, we accelerate the system with a greedy search approach. Instead of generating all possible configurations in advance, we combine candidate generation and evaluation to progressively find the best fit for a given input image. In each iteration, each movable part is offset according to its geometrically feasible candidate motions without modifying the others.

The displacement that produced the smallest local distance value according to $d_{DCM}(\bar{U},\bar{V})$ is applied to the corresponding part. The search continues as long as at least one tested part displacement yields a distance that is significantly smaller than the currently stored minimum. If multiple choices are available, the motion that introduces the largest improvement is chosen and applied. If no such motion exists, the algorithm terminates. Note that exploding a part may unblock others, which results in additional feasible motions of the formerly blocked parts. Thus, the blocking information of the remaining parts needs to be continuously updated.

## 3.4 Retargeting Structural diagrams

Structural diagrams appear in 2D documentations as sequences of at least two images. Every two consecutive input images differ with regard to changes in the configuration that are applied to one or multiple parts. Given a pair of input images, we can restrict candidate motions to those which fall in the image region where visual changes are observed. This allows us to reject candidate motion based on where parts have been moved, rather than on their matching score with the target image. Early rejection of candidate motion based on the region of change is essential to increase system performance. Therefore, we must ensure the reliable identification of these regions.

### 3.4.1 Difference images

In order to evaluate the relevance of a candidate motion with regard to the visible region of change, we need to compute the difference image $\texttt{diff}(U_1,U_2)$ of the rendering of the object before and after applying the candidate motion, and the difference between the two consecutive input images, $\texttt{diff}(V_1,V_2)$. Computing the percentage of overlap of the two regions (Equation 3.2) allows us to discard motions that are not related to the observed differences in the input images, i.e., where the coverage falls below a given threshold.

$$coverage = \frac{||\texttt{diff}(V_1,V_2) \cap \texttt{diff}(U_1,U_2)||_0}{||\texttt{diff}(V_1,V_2)||_0} \tag{3.2}$$

We generate $\texttt{diff}(U_1,U_2)$ by rendering all parts in the object except for the current candidate to the depth buffer in the first pass. In a second pass, we enable the color buffer and render the part before and after applying candidate motion. The resulting pixels in the color buffer correspond to the region of change $\texttt{diff}(U_1,U_2)$ (Figure 3.6b). The method used for obtaining $\texttt{diff}(V_1,V_2)$ differs based on the type of input images. Difference images from input data with clear, distinct colors, such as LEGO assembly manuals, can be directly computed by subtraction and thresholding. However, a reasonable amount of assembly instructions are simple black-and-white closed-line drawings, as known from do-it-yourself furniture [1]. For such line drawings, we apply a silhouette check to identify the background area in both images and reject motions that exhibit low foreground coverage. A flood-fill from the four image corners (which are assumed to be empty) creates enhanced versions of the input images, where background and foreground pixels have a unique color. By conjoining the foreground regions of $V_1$ and $V_2$, we receive a generous enclosure for the area of visible change. Change detection in photographic material is most challenging. Simple image differencing is defeated by lighting, reflection, shadows and surface texture. For robust determination of the region of change, we employ SIFT flow [91], which densely relates pixels from the source image to the most similar pixel in the destination image.

[1] www.ikea.com

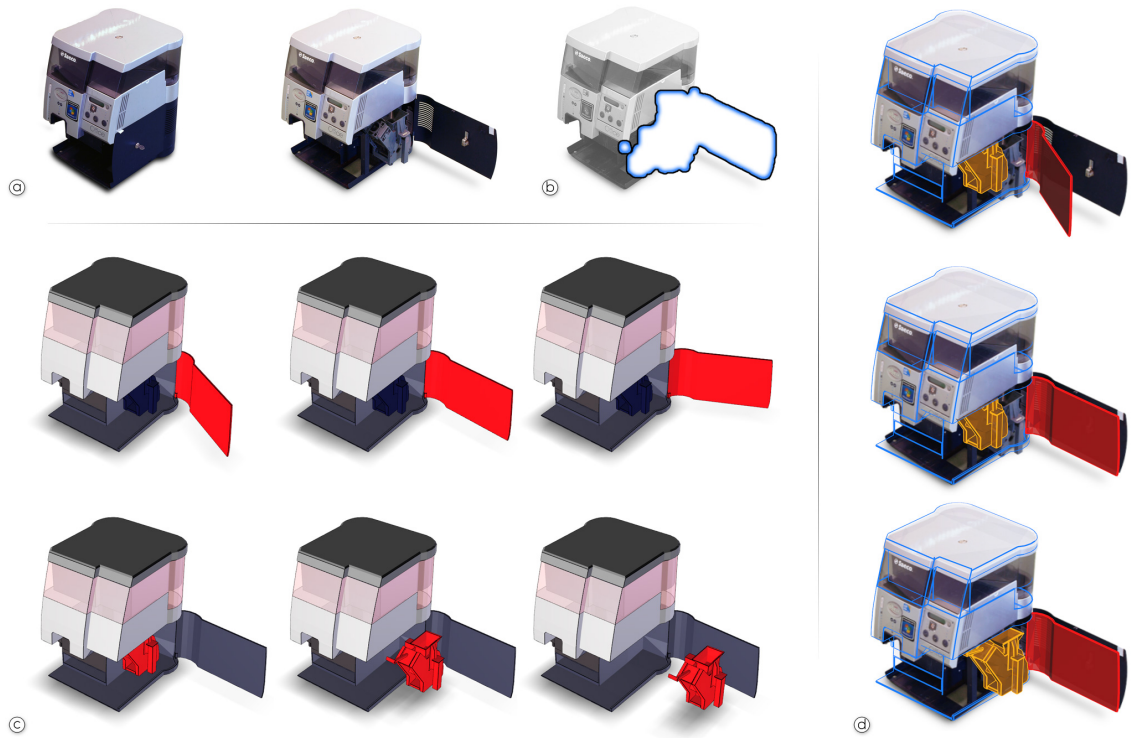


**Figure 3.6:** Retargeting structural diagrams. (a) Input images. (b) We generate a difference image, which reveals changes between the images. Our system creates a set candidate configurations by moving and rotating all parts of the machine which are identified by the difference image. (d) From all candidate configurations, the one which fits the target image best is selected. The motion which had to be applied to transform the object in the selected configuration is used to create the 3D animation. In this case, a rotation is generated to open the service door, and the brewing unit is removed.

### 3.4.2 Global bi-directional chamfer distance matching

Local template fitting works well for detecting detached and isolated parts, as they appear in explosion diagrams. However, this approach performs poorly when trying to verify subtle structural changes, which include reconfiguring, removing or attaching parts. The possibility to compare between images in a sequence obviates the necessity to depict each action individually. Involved parts may be partially obscured or simply disappear from one image to the next, thus making local fitting inapplicable. Consequently, a more thorough analysis of the entire image domain is required. For a full analysis and comparison of two images $U$ and $V$, we compute the DCM over the entire image area in both directions $U \rightarrow V$ and $V \rightarrow U$. We refer to this global image matching of $U$ and $V$ as the bi-directional chamfer distance (BCM):

$$d_{BCM}(U,V) = d_{DCM}(U,V) + d_{DCM}(V,U) \tag{3.3}$$

Finding a candidate rendering $U$ that minimizes $d_{BCM}(U,V)$ ensures a strong visual similarity between $U$ and target image $V$. For a scene with $n$ parts and up to $c$ feasible configurations per part, evaluating all possible setups requires $\mathcal{O}(c^n)$ comparisons. Since this quickly becomes infeasible even for moderately complex objects, we use the greedy algorithm mentioned before to progressively select the best structural improvement based on individual part motions. However, greedy global matching may at early stages favor part configurations that approximately resemble multiple unmatched parts in the target image. To suppress this effect, we put an additional emphasis on local consistency to penalize discrepancies in the sub-image regions $\bar{U}$ and $\bar{V}$ enclosing the projection of the currently tested part. Thus, the distance function uses a weighted bidirectional chamfer matching (WBC):

$$d_{WBC}(U,V) = \underbrace{d_{BCM}(U,V)}_{\text{global matching}} + \underbrace{\lambda\, d_{DCM}(\bar{U},\bar{V})}_{\text{local fitting}} \tag{3.4}$$

The additional weighting of local consistency for $\bar{U}$ and $\bar{V}$ is controlled by the parameter $\lambda$. In practice, we found that choosing $\lambda \approx 0.25$ yields satisfactory results for all examined sequences.

### 3.4.3 Candidate generation

Similar to explosion diagrams, we generate candidate configurations by iteratively moving and testing parts based on geometrically feasible motions. In addition, we allow moving parts outside the visible area in order to handle instructions which require to append or to remove parts. The instructions shown in Figure 3.6 present multiple consecutive actions. To access the brewing unit from the coffee machine, the hatch has to be opened before the unit can be removed. Since we iteratively change candidate configurations, our system succeeds in correctly recreating the depicted motions. This example demonstrates that our algorithm is robust to hidden or incomplete input data. Although the brewing unit is completely obscured by the closed hatch, its removal was successfully detected by our system (Figure 3.6).

Since structural diagrams may also encode actions requiring to rotate a part, we additionally generate candidates that involve part rotations. Candidate rotation axes are generated by combining principal axes with the origin of either the local coordinate system or the center of gravity of a part. The testing of rotations starts with the part in its default configuration, and proceeds by increasing the angle of rotation in discreet steps. Testing rotations around an axis terminates after reaching 360° or if a collision with another part occurs. For example, to analyze the input images in Figure 3.6, our system generates and evaluates candidate configurations by rotating the service door of the coffee machine around its hinge.

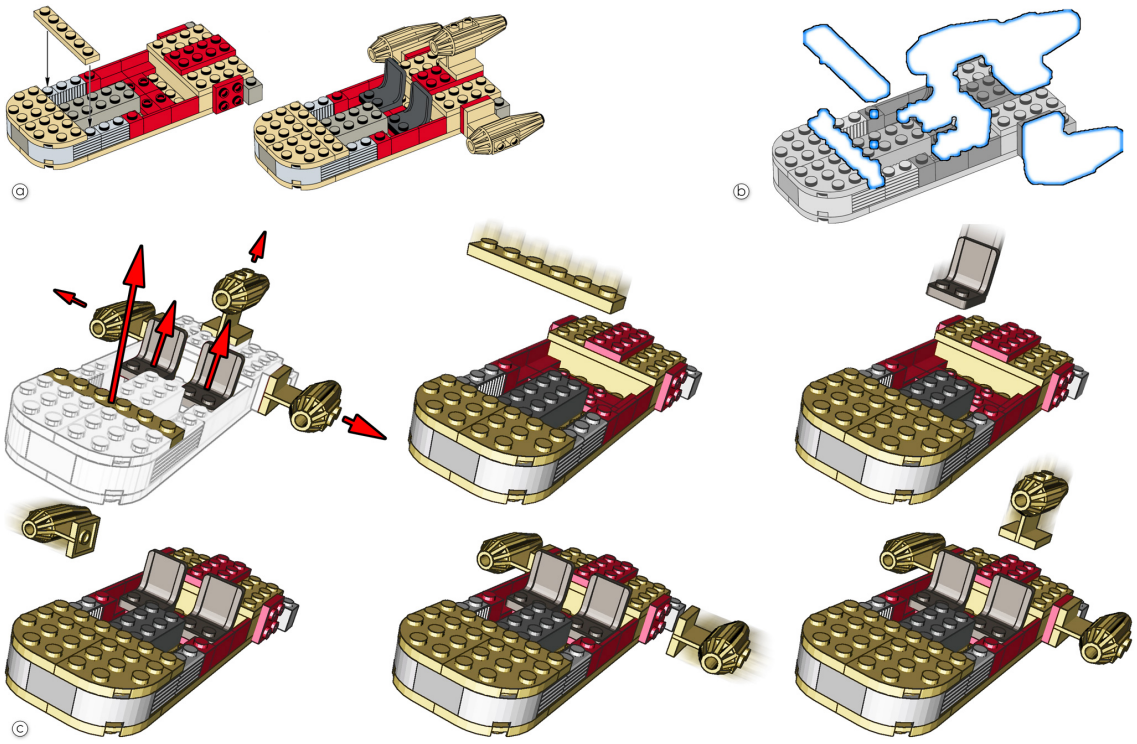


**Figure 3.7:** Retargeting combined diagrams. (a) Two input images illustrate multiple assembly steps. (b) We derive a difference image from the inputs, which indicates where changes occurred. (c) Our analysis yields a set of motions and corresponding parts, in this example indicated by red arrows. The key frames show the animation generated from the computed motions. ©2014 The LEGO Group, used with permission.

## 3.5 Retargeting combined diagrams

Traditional 2D documentation often consists of diagrams that combine aspects of annotated, structural, explosion and action diagrams. A typical example is to use annotations in any of the other diagrams (Figure 3.1a). Another very common combination is to show one or multiple parts in displaced positions (explosion diagram), before showing their final position on the object in the next image of a sequence (structural diagram).

Displaced parts may further be highlighted with diagrammatic elements such as arrows or leader lines. Consider the example in Figure 3.7a, which shows two images in a sequence of a LEGO assembly manual. Instructions to add bricks are given by showing the brick and leader line connecting the brick to the intended location on the object (as used in action diagrams). In addition, some bricks just appear in the second image.

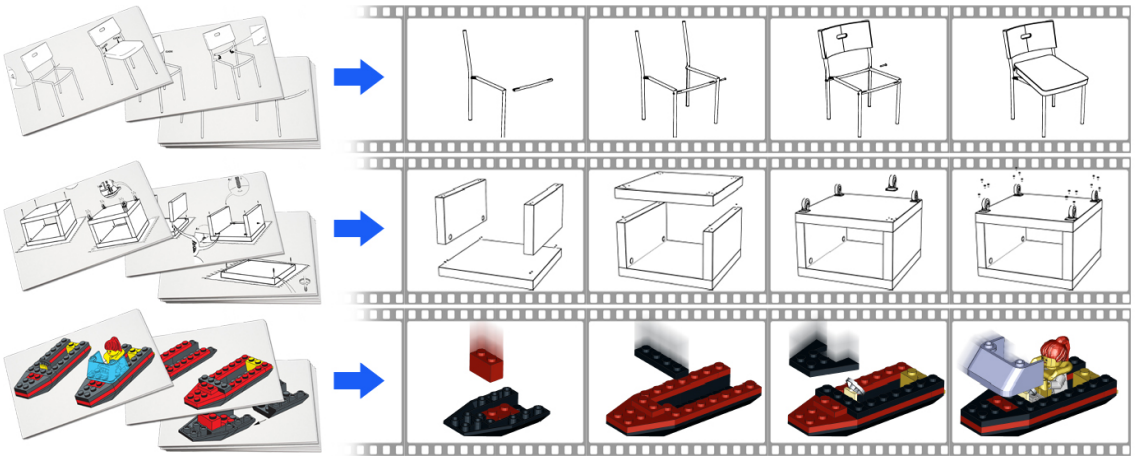


**Figure 3.8:** Various examples. ©IKEA Chair Herman and Table Lack, used with permission. LEGO Boat, ©2014 The LEGO Group, used with permission.

The system we present is able to handle all kinds of combinations. Interpreting the mixed sequence displayed in Figure 3.7a can be achieved by employing the approach that is used for analyzing structural diagrams. Since WBC considers both global and local features, it is applicable to partial explosions as well. The additional visual information that is conveyed by arrows in sequences can be processed by the established methods for arrow diagrams and may be used to complement candidate selection.

However, we found that for all assessed sequences, our algorithm for retargeting structural diagrams was able to reproduce the portrayed instructions without interpreting the arrows. Moreover, since we approximate any displayed object configurations by accumulating several part motions, our system is able to retarget image pairs that convey multiple instructions at once. This includes manipulations of multiple parts between two images as well as multiple consecutive actions. For example, Figure 3.7a shows two images taken from a LEGO instruction which adds six bricks from one image to the next. Since our system correctly identifies the configurations in both images it is able to generate the correct motion to animation.

Our approach for generating 3D annotations is independent of other elements present in the diagram. Therefore, we can handle annotations, which appear in any of the other diagrams. Only the current configuration of the parts depicted in the image is required to render a mask that matches the input image. Since DCM is robust against image noise and smaller artifacts, additional graphical elements do not influence the candidate matching. Figure 1.1 shows an example of an annotated explosion diagram, which we successfully analyzed before retargeting to AR.

| Testcase | #Parts | #Images | Runtime |
|---|---|---|---|
| Valve (Fig. 1.1) | 4 | 3 | 8s |
| Coffee Machine (Fig. 3.2) | 6 | 4 | 9s |
| Mixer (Fig. 3.5) | 6 | *1 | 45s |
| LEGO Boat (Fig. 3.8) | 18 | 8 | 1m 27s |
| LEGO Landspeeder (Fig. 3.7) | 39 | 14 | 4m 10s |
| IKEA Chair Herman (Fig. 3.8) | 10 | 8 | 5m 58s |
| IKEA Table Lack (Fig. 3.8) | 42 | 5 | 16m 29s |
| LEGO Tower | 42 | 7 | 2m 21s |
| LEGO House | 88 | 13 | 10m 25s |

* Explosion diagram, single image only

**Table 3.1:** Assessed test cases. We list the number of parts involved, as well as the length of the input documentation material and runtimes for retargeting.

## 3.6 Performance Image Retargeting

To evaluate the performance of our system, we choose a set of objects, which have been illustrated in different styles. The results are listed in Table 3.1. The table shows the number of parts the object consist of (#Parts), the number of images in the sequence (#Images) and the time used to retarget the input data to a 3D diagram (Runtime). The runtimes were recorded on a personal computer with an i7-4771 CPU @ 3.50 Ghz, 16 GB RAM and an NVidia GeForce GTX 780 Ti with 3 GB graphics memory. Input diagrams and resulting animations for test cases that are not shown elsewhere are depicted in Figure 3.8 and in the supplementary video.

Although the LEGO Landspeeder and the IKEA Table Lack have a similar part count, the manual of the latter takes significantly longer to retarget. This results from the different illustration styles used and the number of parts involved in each step of the manuals. Photos and renderings allow us to apply image differencing to obtain tight areas of change for early rejection of unrelated candidate motions. However, this is not possible for line drawings. Furthermore, the manual of the IKEA Table depicts a larger number of instructions in each individual diagram. This is possible because many parts of the object have no sequential dependency (e.g., wheels and screws can be added in parallel). Consequently, a high number of movable parts have to be considered at each point in time. The performance of our approach thus depends strongly on the employed illustrating style and the overall structure of the examined object.

## 3.7 Limitations of Diagram Retargeting

Our current implementation has several limitations. The pose estimation is the basis for the data extraction in our approach. Therefore, an erroneous estimation will propagate through the entire pipeline. If the error exceeds a certain threshold, the system will fail to associate image regions with 3D objects. In addition, invisible elements in an image cannot be extracted with our current implementation. For example, if several screws have to be removed from an assembly within a single step often only a few representatives are shown in an image, while others remain occluded. Since our system can only detect visible elements it will fail to retarget such cases. A similar problem occurs when parts are small and those barely visible in the image. We provide visual examples for the described limitations in the accompanying video material.

## 3.8 Conclusion

We built and evaluated a system that is able to generate interactive 3D documentation from 2D image data. The resulting 3D scene can be presented on a regular desktop or tablet computer or can be displayed directly within the user's real-world environment. Thus, our system enables reuse of existing 2D documentation material to create 3D documentations. Therefore, we believe that our approach will become a key enabling technology to move from traditional 2D to modern interactive 3D documentation.

The central idea for the transfer of 2D documentation to 3D is to replicate the 2D image by rendering a 3D model of the object of interest. This relates 2D image elements to 3D objects. We have demonstrated how this relation can be used to interact in 2D image space in order to control the corresponding 3D visualization. Thus, we believe that applications of our system will not substitute 2D documentation entirely. Instead, they will combine 2D and 3D data to provide the user with best of both presentation and interaction spaces.

While our system is able to handle images present in existing 2D documentation, it is not limited to documentation material. Our system can generate 3D animations from any image or photo sequence which shows an object in different configurations. Therefore, our system furthermore provides a new tool to control 3D animations by images.

# CHAPTER 4

# Authoring Augmented Reality Documentation from Videos

Videos provide a rich information source, why we have designed a system to let a user quickly extract the relevant information from a source video, and compose an AR tutorial that operates in the environment of the user. Note that the author of the video may not know in advance about the exact objects in the end user's environment and their configuration. For example, a tutorial video may show one coffee pot, while the target scenario may have a differently shaped pot and three cups. Our system is designed to flexibly deal with such ambiguities while requiring only the minimal input from the author. Overall, we author AR application from videos in two successive steps:

**Motion extraction** (Figure 4.1, left). The author loads the input videos, selects an object type and provides enough geometric annotations to track the object in the input video.

**Motion editing** (Figure 4.1, right). The extracted motions are attached to any object or object type in the AR environment. The motions are segmented, and multiple sources can be combined into new tutorial sequences.

## 4.1 Motion extraction

Tracking the 3D motion of objects from a monocular video is a standard task in AR, assuming that a *tracking model* of the object is available. For objects with enough surface texture, a popular approach is to match SIFT features [93] extracted from the live video with a feature point cloud representing the tracking model. The 3D positions associated with matched features are forwarded to pose estimation using a variant of the Perspective-n-Points (PnP) algorithm, such as the one of Lepetit et al. [86]. This pose estimation requires the internal camera parameters to be calibrated in a preparation step. After successful initialization, pose estimation in subsequent frames can be made faster and more robust by incremental tracking [96].

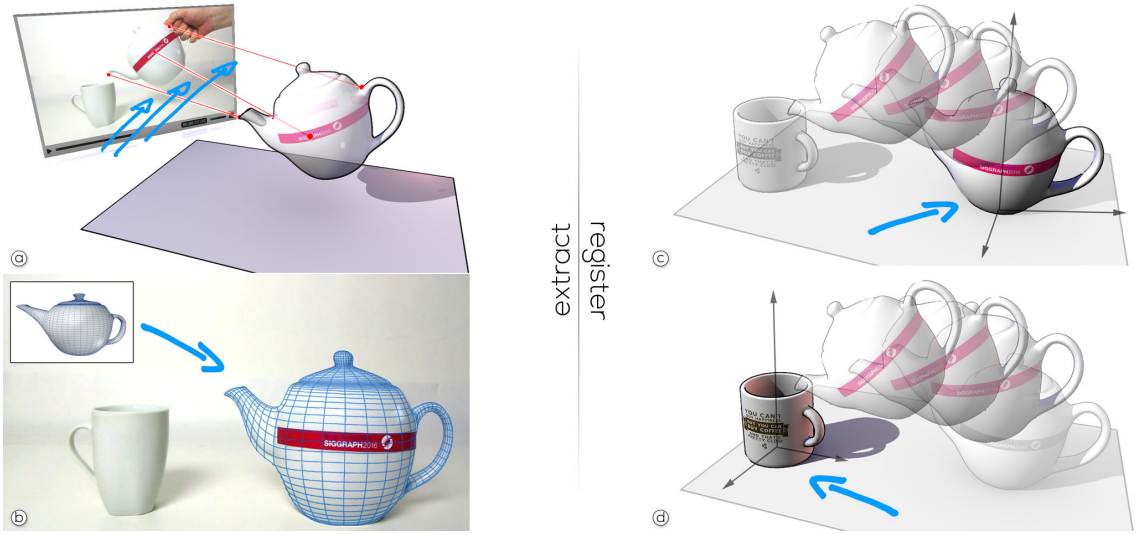



**Figure 4.1:** (a) 3D object motion is extracted by tracking known model features in the video and relating them to corresponding points on a 3D model. (b) The 3D model is created at runtime by deforming a similar 3D model. (c) The motion is retargeted in the AR scene by registering it to an object like the teapot. (d) The motion is, instead, registered to the cup. Motion instructions are overlaid as soon as a real cup is detected.

Extracting the 3D motion from unknown videos is more difficult. Not only is no tracking model available, but the internal camera parameters are unavailable, as well. We cannot assume that the input video is suitable for recovering a tracking model using structure from motion [53], since objects may lack texture features, visibility may be limited due to restricted camera motion, and occlusions, such as from the tutor's hands, may be significant. Instead of using structure from motion, we provide interactive tools that make it easy to specify a tracking model with simple annotations as part of the authoring process. By selecting among several model types, the tutorial author specifies not only the tracking model but also the semantics of the motion, which is beneficial in the following steps.

### 4.1.1 Extracting rigid object motion

Rigid objects, such as tools, workpieces, or furniture, are well suited for monocular tracking. The simplest cases are (roughly) planar objects, such as table surfaces or electronic circuit boards. We let the user specify the corners of a rectangular area with known dimensions in the first video frame, and incrementally track distinctive features inside the rectangular area throughout the video sequence. The location of the distinctive features in the plane can be estimated directly during the extraction from the image. A minimum of four points is sufficient to estimate a pose from a homography.

Non-planar rigid objects require building a feature point cloud as a tracking model. Since we cannot use structure from motion for most source videos, we must give the user an alternative method to specify the feature geometry. Our method utilizes the fact that a user wishing to replicate a tutorial in AR must have access to the same types of objects as used in the video. While actual shape and appearance may differ between the object in the video and the one available to the user, the overall topological and geometric characteristics will be similar.

We can use the object available to the user as a template for building the tracking model for the object in the video. First, the user must create a 3D reconstruction of the object. This is relatively straight forward using consumer depth sensors [108] or even mobile phone cameras [113]. Alternatively, the user may search for a sufficiently similar 3D model in Internet databases, such as Sketchup 3D Warehouse. The template model must be deformed to match the object in the source video.

The user controls the deformation based on the silhouette of the target object in the video. After selecting a video keyframe, the user roughly aligns the template model with the target object. Vertices on the template's silhouette are moved to corresponding points on the target object's silhouette. The target's silhouette is selected using GrabCut [131], initialized by the footprint of the template placed by the user. If necessary, the user can fill in missing silhouette edges by drawing over the occluding elements.

The user's task is to coarsely deform the template to make it similar to the target. If the template has roughly the same shape as the target, it can be automatically deformed to precisely match the target. For this purpose, we let the user specify simple free-form deformations using skin-and-bone controllers. The user places bones in the template object and corresponding bones in the keyframes showing the target object, and the template is transformed interactively using linear skin blending. After the skin deformation, we search for the nearest target silhouette point for each template vertex and let the vertex snap to this location. Finally, the local rigidity energy of the mesh is minimized [146]. An example deformation is shown in Figure 4.2;

### 4.1.2 Extracting motion with surface contact

Video tutorials often show close-up views of tools during surface manipulation and use a variety of different tools. Annotating such videos to extract the motion would be tedious. Fortunately, in many situations, only the position of the tool-tip relative to the workpiece is required. We extract such actions by tracking the tool-tip in image space by relying on appearance only. The workpiece on which the tool is applied is tracked in 3D, and the 2D trajectory of the tool-tip is converted to a 3D trajectory by perspective projection from the known camera position (see the illustration in Figure 4.3).

Even for known surfaces, tracking only the tool-tip is difficult from monocular video without exact knowledge of the tool's appearance in the video. Consequently, conventional approaches using feature detection do not yield robust results. We overcome this problem by applying a tracking-learning-detection approach [107], which identifies robust features by updating a learned

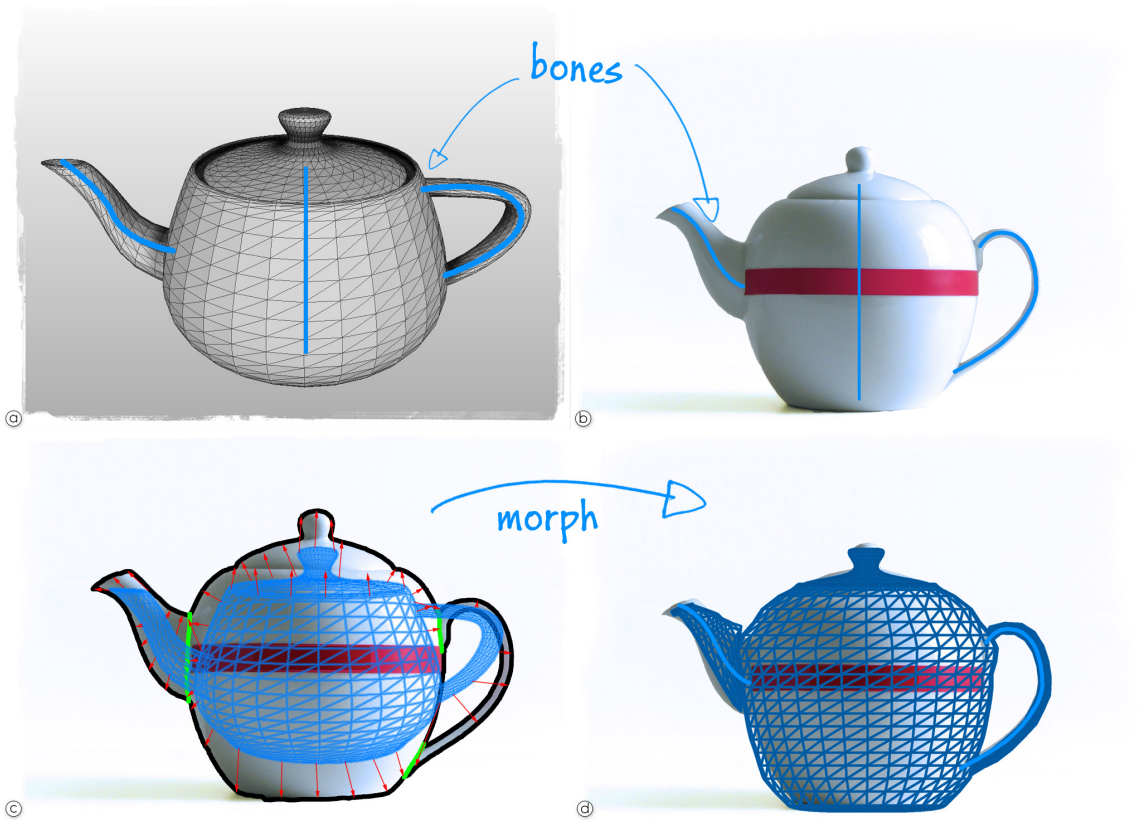



**Figure 4.2:** To extract object motion, we need to provide the system with a 3D model of the target object. (a) We model the object by deforming a template mesh, which we rig in 3D. (b) The user interactively specifies the 2D projections of bones. (c) This allows transforming the template to coarse resemble the target and to subsequently relate vertices on the template's silhouette to the target's silhouette. (d) The deformed template can be used for camera pose detection of the target.

classifier while tracking an initial user selection. In order to track a tool, the user identifies the tool-tip by selecting a rectangular patch in the first video frame (marked red in Figure 4.3). This area is used to initialize the classifier, which is improved in subsequent frames. As propose by Kalal et al. [72], we use an ensemble classifier for detection and P-N learning to update the tracking model.

We project the position of the tool-tip to the workpiece's surface by mapping the tool-tip's trajectory to a texture atlas generated by unwrapping the 3D mesh representing the workpiece. The texture coordinates in the atlas can be looked up directly in image space from a G-buffer which contains rendered texture coordinates. Given the size of each triangle in object space, we calculate the affine transformation $\mathscr{M}_a$ between each model triangle $T_v$ and the corresponding triangle in texture space $T_u$, where $[x_i^v, y_i^v]^\mathbf{T} = \mathscr{M}_a \cdot [x_i^u, y_i^u, 1]^\mathbf{T}, i \in \{0,1,2\}$.

In order to extract motion with surface contact, we first extract the 3D motion of the surface by tracking the corresponding 3D object. Subsequently, we extract the motion of the tip of the tool relative to the surface.

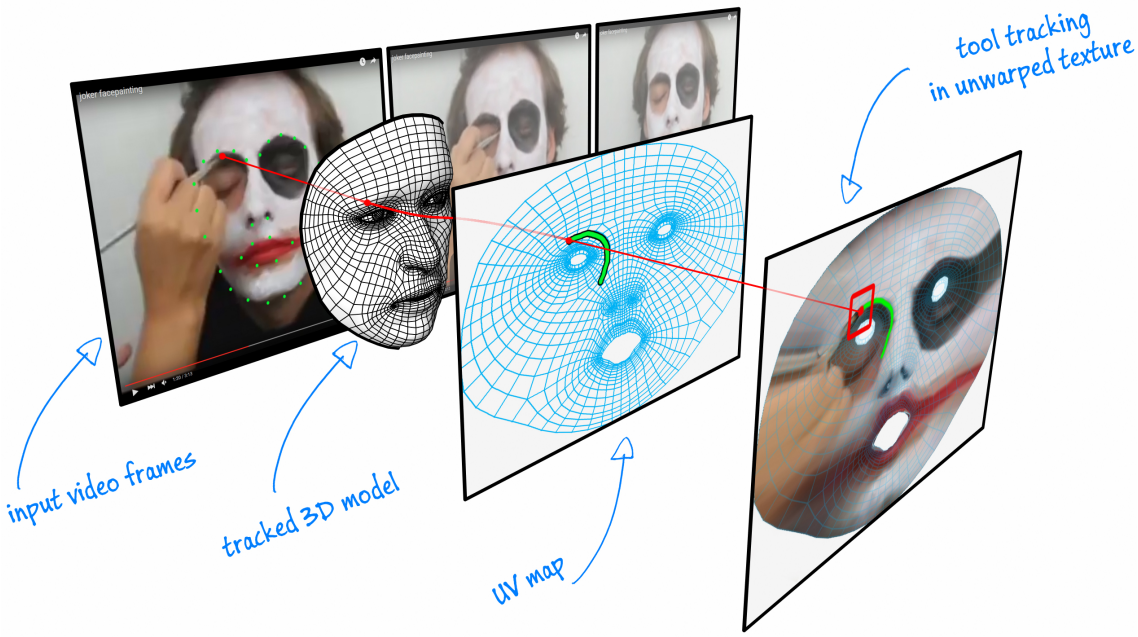



**Figure 4.3:** Extracting motion with surface contact. We convert the 2D trajectory in the input video to a 3D trajectory by back-projecting the video data to a 3D model of the object of interest, in this case, a face. From the 3D model, we furthermore create a texture atlas which we use to track and record the path of the tool-tip.

Tracking the 3D motion of known objects from monocular videos is a standard task in AR, assuming that a 3D model of the object and intrinsic camera parameters are available. For objects with enough surface texture, a popular approach is to match SIFT features [93] extracted from the live video with a 3D feature point cloud representing the object. The 3D positions associated with matched features are forwarded to pose estimation using a variant of the Perspective-n-Points (PnP) algorithm, such as the one of Lepetit et al. [86]. After successfully computing the pose of an object in the first frame, pose estimation in subsequent frames can be made faster by incremental tracking [96].

Extracting the 3D motion from arbitrary videos from, e.g., online sources is more difficult, since the 3D model and the intrinsic camera parameters are unavailable. Therefore, we must interactively create a 3D model and adjust internal camera parameters as part of the authoring process.

Our system extracts 3D motion of three different types of 3D objects. In particular, we provide tools to extract 3D motion of piecewise planar objects, rigid objects, and deformable objects with a known shape model, such as human faces. For each of them, we provide an optimized set of tools to create the necessary 3D model with minimal user input.

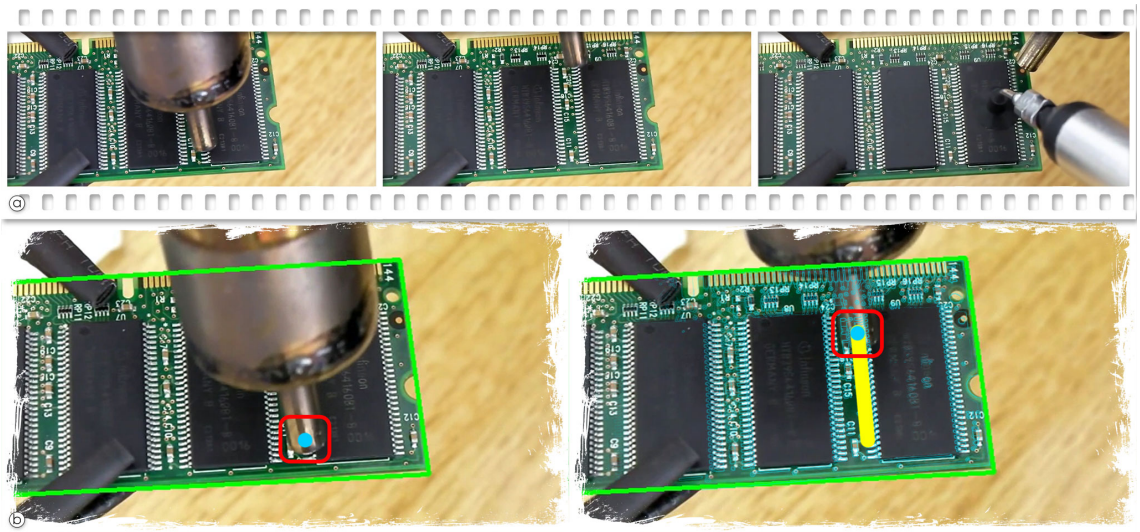


**Figure 4.4:** Planar objects. (a) Example frames from the input video showing a hot air soldering tutorial. (b) Tracking results. The green outline shows the user selection of the planar object. The selection is automatically rectified and the contained image features are used to calculate a homography for each frame. The red marking shows the selected tool-tip, which is tracked relative to the planar object throughout the video, resulting in a tool path (shown in yellow).

### 4.1.3 Planar objects

Piecewise planar objects, such as planes or boxes, can be conveniently specified by interactively drawing on top of the first video frame. We let the user specify the corners of a rectangular area and its dimensions, as shown in Figure 4.4(b) Note that a minimum of four points is sufficient to estimate a pose from a homography. However, to produce more stable results, we incrementally track all distinctive features inside the rectangular area throughout the video sequence. The locations of the distinctive features in the plane are estimated directly during the extraction from the image.

### 4.1.4 Non-planar objects

Non-planar 3D objects require a more complex 3D point cloud than a tracking model. Unfortunately, for most online videos, we cannot expect to successfully perform structure from motion to obtain 3D geometry: The internal camera parameters are unavailable and may even change when optical zoom is used. The objects in the video may lack texture features. Visibility may be limited due to restricted camera motion. Occlusions, such as from the author's hands, may be significant.

Instead, our method utilizes the fact that a user wishing to replicate a tutorial in AR must have access to the same class of objects as the ones used in the video. While actual shape and appearance may differ between the object in the video and the one available to the user, the overall topological and geometric characteristics will be similar. Therefore, we let the user provide a physically available object as a template for the tracking model.

Creating a 3D reconstruction of an existing physical object is relatively straight forward using consumer depth sensors [108] or even mobile phone cameras [113]. If no physical template is available, the user can instead search for a template model in an online database, such as Sketchup 3D Warehouse[1]. If the template model slightly differs from the tracking model, we incrementally deform based on the approach of Kraevoy et al. [81].

### 4.1.5 Deformable objects

If the shape is known a priori, such as the human face, a deformable model can be tracked by determining an update to the deformation parameters in each frame. This kind of tracking is commonly based on specially trained models. Our system implements facial model deformation based on a constrained local neural field [10]. Since the face tracker operates in image space, we map the 2D facial landmarks to a 3D model of a human face (see Figure 4.3).

### 4.1.6 Motion Capture

To extract the path of a tool that alters the workpiece (i. e., the 3D surface), we extract the trajectory of its tip. The user has to select the starting and ending frames of an action in order to input information about surface contact. In-between the selected frames we assume the tool has contact to the surface. To extract its motion, we track the position of the tool relative to the workpiece, and we track the workpiece in 3D space as described before. This enables us to extract 2D trajectory in image space which we subsequently convert to a 3D trajectory by using perspective projection from the tracked camera's point in space onto the 3D surface (Figure 4.3). Note that we have derived the 3D pose of the object relative to the camera for every frame before. This allows the derivation of the camera pose as the inverse of the 3D object pose transformation.

Since the tool's tip is often very small and usually located in front of a changing background, tracking the tip from monocular video can be difficult without the exact knowledge of the tool's appearance. Therefore, conventional feature detection approaches do not yield satisfying results. We overcome this problem by applying a tracking-learning-detection approach [107], which identifies robust features by updating a learned classifier while tracking an initial user selection. In order to track a tool, the user identifies its tip by selecting a rectangular patch in the first video frame (marked red in Figure 4.3(right)). This area is used to initialize the classifier, which is improved in subsequent frames. As proposed by Kalal et al. [72], we use an ensemble classifier for detection and P-N learning to update the tracking model.

After extracting the path of the tool in image space, we project its position to the surface of the workpiece. We have implemented this by recording the tool trajectory in a 2D texture atlas, generated by unwrapping the 3D mesh of the workpiece. The texture coordinates allow us to directly map the tool trajectory from image space to the surface.

[1] www.warehouse.sketchup.com

| unit:mm | pencil | felt-tip | marker | brush |
|---|---|---|---|---|
| tip-size (lxw) | 2x0.5 | 4x2 | 7x5 | 24x6 |
| stroke-width | 0.5 | 1.2 | 2.2 | 10-14 |
| error (avg) | 0.17 | 0.23 | 0.37 | 4.47 |

**Table 4.1:** Tool tracking accuracy measurements.

### 4.1.7 Accuracy Analysis

The accuracy of the tool tracker depends on the stroke width, tip size and speed. While the tracker follows the tip it might not follow the actual center of the stroke. The author can directly adjust this offset during/after extraction. Table 4.1 shows tracking accuracy measurements of a variety of drawing tools. The watercolor brush has the largest error due to its large and deformable tip, while all other tools produced an average error of less than 0.5 mm.

### 4.1.8 Extracting articulated human motion

We retarget human motion by extracting skeletal poses from the source video using an incremental skeleton tracker. In the first frame, the joints of a skeleton template are aligned with the human in the video. In subsequent frames, joint positions are tracked using dense SIFT [90]. If the automatic tracking fails due to occlusion or excessive motion blur, the user can interactively adjust erroneous joints. Figure 4.5(a) shows an example of tracked joint positions. The green stars are placed by the user in the first frame of the video and tracked incrementally.

**Disambiguation** After tracking the projection of joints in image space, we need to derive the 3D skeletal pose. Even with incremental minimization of reprojection error, the skeleton has sufficient degrees of freedom so that the problem may become under-determined using just the image space constraints. For example, the illustration in Figure 4.5(b) shows two alternative upper leg poses which result in the same 2D projection. Fully automatic approaches to this problem exist for special motions, such as walking or running. However, our source videos show arbitrary human motion, which makes it necessary to let the user disambiguate poses for certain keyframes, followed by pose interpolation.

Wei and Chai [169] propose to let the user manipulate a stick figure for disambiguation. However, we found that this approach is too complicated for users not trained in computer animation. Instead, we let the user select the correct pose of a keyframe from a set of renderings showing possible solutions to the ambiguity. If the solution set to choose from gets too large to be useful, we let the user demonstrate the correct pose using a Microsoft Kinect sensor. We estimate the user's skeleton pose in real-time and use it to disambiguate the pose in the keyframe. Figure 4.5(c)) shows an example keyframe, the user demonstrating the 3D pose to the system and the resulting pose applied to an avatar.

**Keyframe selection** The entire sequence is searched for local maxima of the projected length of each bone connecting two joints. Local maxima represent poses where the motion may change its direction (see Figure 4.5(b) for an example illustration). Consequently, for each joint, only one pose between two maxima may need disambiguation. In order to reduce the work for the user, we greedily choose those frames as keyframes for which the largest possible number of ambiguities is resolved simultaneously.

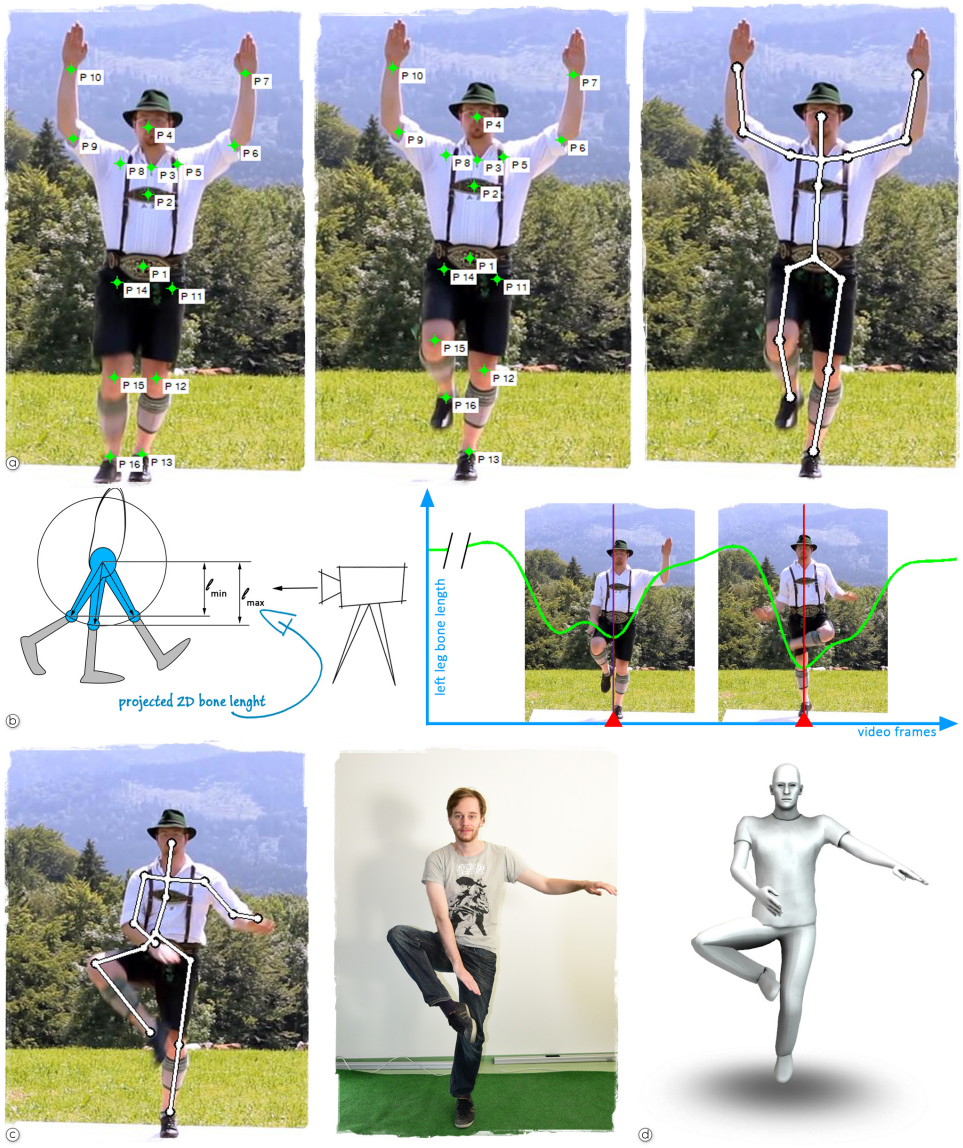



**Figure 4.5:** Generating 3D body poses from monocular 2D video. (a) We track interactively selected joint positions in consecutive frames to compute an animated skeleton in 2D image space. (b) Since the projection of a bone can result from two 3D poses, we interactively provide the system with the one seen in the picture. However, we only need to provide the system with a single solution between two maxima of the length of a bone in 2D. (c) We provide the system with the correct 3D poses by capturing the user's skeleton in 3D, while he is demonstrating 3D key poses. (d) Feedback is presented by posing an avatar with the selected 3D skeleton.

**Figure 4.6:** Combined motion types in a knife skills tutorial. (a) example frames from the input video. The knife should be moved in a fluent elliptical motion. (b) Morphing of a stock knife template to the actual knife used in the video. (c) Illustration of the combined tracking results for the knife and also the hand skeleton.

### 4.1.9 Combined cases

In some tutorials, both rigid objects and articulated human motion appear simultaneously. For example, **knife skills** tutorials, as shown in Figure 4.6, include 3D object motions and finger poses. The knife was modeled using the deformation method described in section 4.1.1. The chopping board was placed by the planar object fitting method. The hand pose was tracked after interactively aligning a hand skeleton. Note that, in this example, the fingers are not moving. However, the 3D position of the hand needs to be tracked. Since the knife has very few usable texture features, we had to re-initialize feature correspondences during extraction. In total, it took approximately 4-5 minutes to retarget 15 seconds of source video material, including scene modeling and re-initialization of the tracker when features were lost. Note that the knife provides only a few good features to track, why most time was spent on re-initializing the tracker in this example. Both the extracted 3D motion of the knife and the extracted 3D hand pose, are registered to the chopping plate, where we later visualize the knife motion using a ghosting animation.

## 4.2 Motion editing

### 4.2.1 Temporal segmentation

Before composing a tutorial, we segment the extracted motions into smaller chunks to allow convenient navigation inspired by the work of Pongnumkul et al. [120]. A segmented tutorial with semantic annotations (depicted actions and causal changes to objects in the environment) allows to easily edit specific segments or compose multiple sources into one tutorial.

The initial segmentation is done automatically in a way that is similar to scene detection in video production. Camera zoom is detected by matching features from the tracking model to each video frame and applying a threshold on the change of the geometric relation between those features. Cut detection relies on the simple heuristic that tracking failures correspond to cuts in the video material. In addition, turning points, where the dominant direction of a trajectory is inverted, are computed as candidates for segmentation (see Figure 6.5 for details).

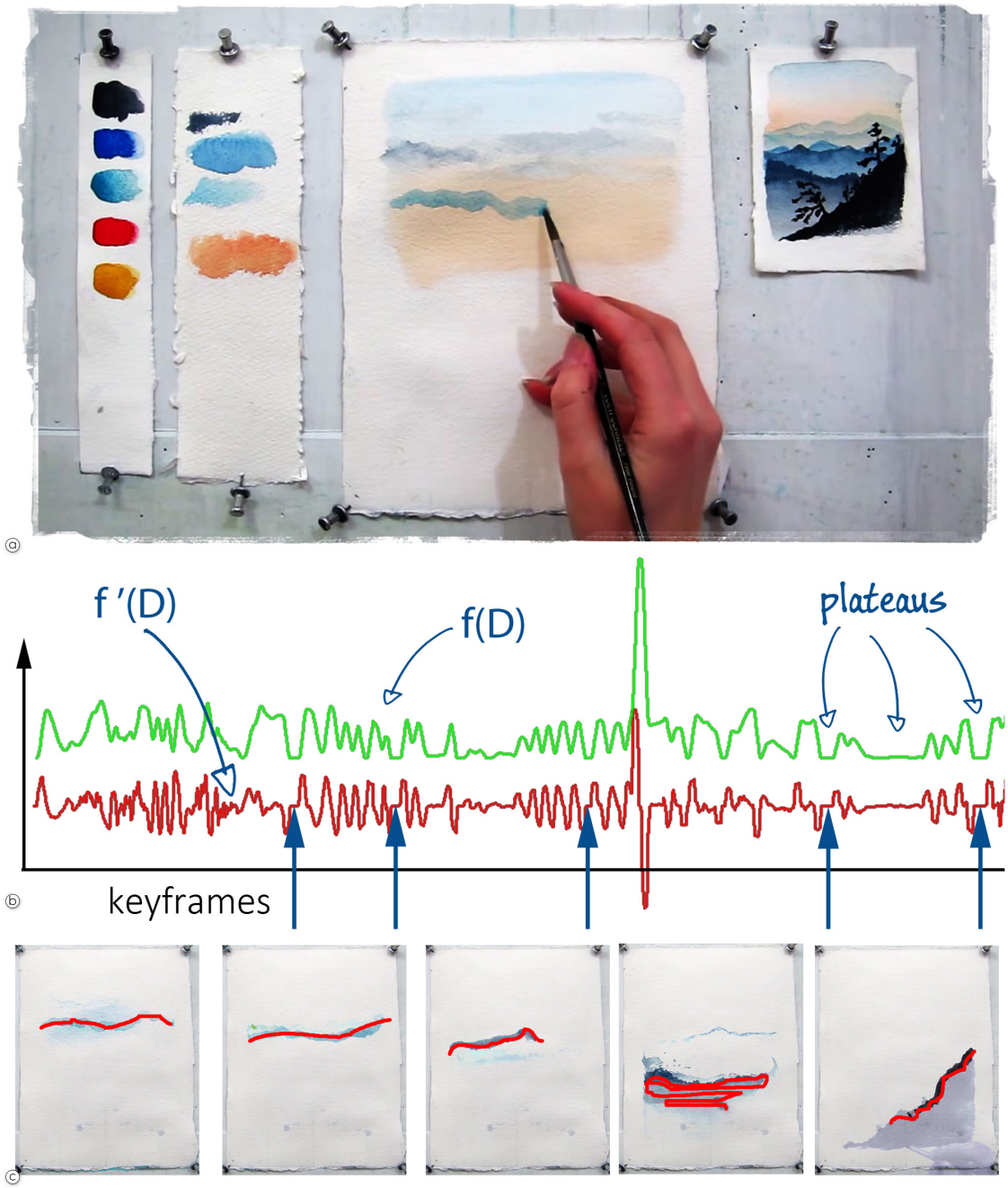



**Figure 4.7:** Temporal segmentation of motion with surface contact. (a) We segment the tutorial into a set of actions (b) by analyzing the sum of absolute differences $f(D)$). We detect starting and ending frames of segments where its derivative is 0. (c) This results in a set of actions, which we use to compute a set of image layers in order to further edit the tutorial.

Furthermore, we segment tutorials which include motions with surface contact (see section 4.1.2) based on the changes to the surface. We define a segment by an action in the input video, e.g., a brushstroke painted on a canvas.

To detect these actions, we compute the sum of absolute differences from the first frame to each subsequent frame of those pixels in the source video that correspond to the contact surface. A derivative of 0 over a short sequence of frames (2-4 frames at 30Hz) indicates that no changes are applied to the surface, indicating a segment boundary.

In order to compose segments of surface manipulations, we use a video matte to add the changes between two keyframes. Since image differencing is insufficient for semi-transparent additions, we calculate the foreground color by intersecting vectors between foreground and background colors in RGB space [31]. The alpha value is calculated by $\alpha = |B_1 - F|/|B_0 - F|$, where $B_0$ is the color in the start frame of the segment, $B_1$ the color in the end frame, and $F$ the estimated foreground color. If insufficient color distribution in the background is detected, we resort to chroma keying to estimate the foreground color instead. In the example in Figure 4.8, we derive the watercolor added during an action.

With the segmentation and the estimated alpha masks, we can generate a layered representation of the source video. A few selected layers are shown in Figure 4.7(c). The red line overlays indicate the extracted movements of the tool-tip for each segment. The layered representation can be manipulated like in any image editing program. We can re-arrange the layers (and their paths) and also combine different video tutorials. Figure 4.8(d) shows an exemplary combination of two painting videos.

Painting tutorials, as shown in Figure 4.8, include motion with surface contact on a canvas. To retarget this kind of tutorial, we model the canvas as a planar object. Subsequently, we extract actions and layers using the method described in section 4.2.1. We use Adobe Photoshop as an interface for editing, where we load the extracted layers during extraction automatically. Adobe Photoshop provides operations like translation, rotation or scale, allowing to modify the input tutorials. We can add layers from multiple tutorials to combine several source tutorials into a new one. Creating the mixture shown in Figure 4.8 took approximately 2-3 minutes. While layer extraction was done in a few seconds, most of the time was spent on rearranging (translating and scaling) layers into a new composition.

### 4.2.2 Registration

Motions are always registered relative to objects. Both the source and the destination of a motion trajectory can be attached to separate objects. Attaching only the source of the motion trajectory prompts the user to pick up a certain source object and use it. This behavior can be used to prompt a response when a particular item (or item type) enters the field of view.

Attaching only the destination of the motion trajectory will draw the user's attention towards the destination object, for example, the location where a tool should be applied. This behavior is desirable to guide the user, if only the destination object is likely to be visible, or if multiple source objects could be used.

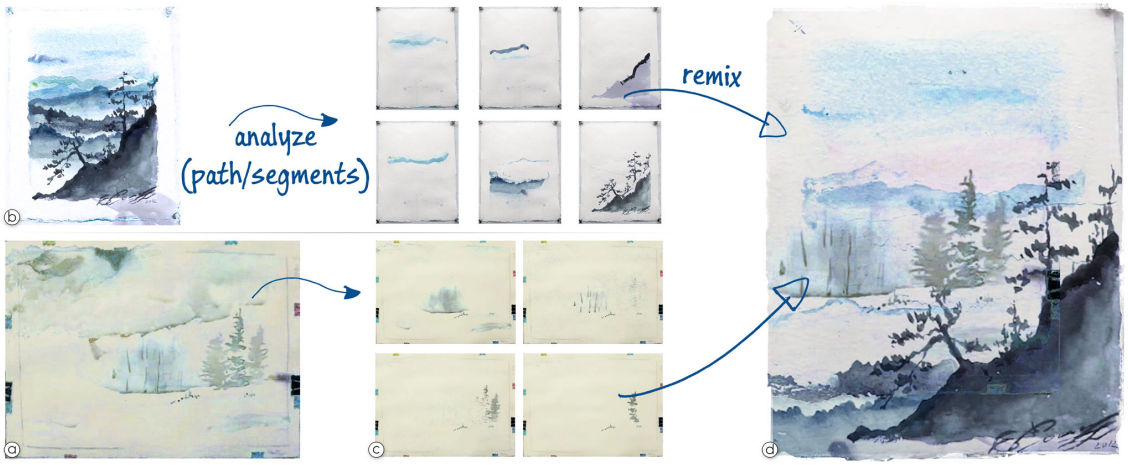



**Figure 4.8:** For mixing several painting tutorials, we split the source videos (a, b) into a set of actions (c). Each action represents a single layer in the painting. (d) For each layer, we compute an alpha mask, that allows us to compose layers into a new tutorial.

Finally, attaching both source and destination of the motion trajectory results in an unambiguous instruction to move the source object towards the destination object. In this case, the trajectory must be warped to fit the actual objects' poses, since these may differ from configuration extracted from the video. We simply scale positions along the trajectory linearly to the desired interval, while orientations are scaled in quaternion representation. This only makes sense if the configuration in the video and in AR are similar enough so that the characteristics of the motion are preserved. Such situations most likely occur in highly restricted environments, such as the assembly of electronic equipment.

In order to register the extracted motions to the AR environment, we must track the source or destination object (or both). If the AR system uses a monocular video camera, we can use the same tracking algorithms used previously for the motion extraction, re-using the tracking models. However, while the motion extraction works offline and affords manual intervention, the final AR display must work in real-time and be capable of automatic initialization. In an environment with poor feature distribution, monocular tracking may give poor results. Therefore, we prefer using a depth sensor, such as the Microsoft Kinect, for the AR tracking at runtime.

### 4.2.3 Evaluation

We have tested our authoring system on a number of different video tutorials in a preliminary user study with the goal to gain insights on usability and timing. We also asked the expert users for their opinion and possible improvements to the software.

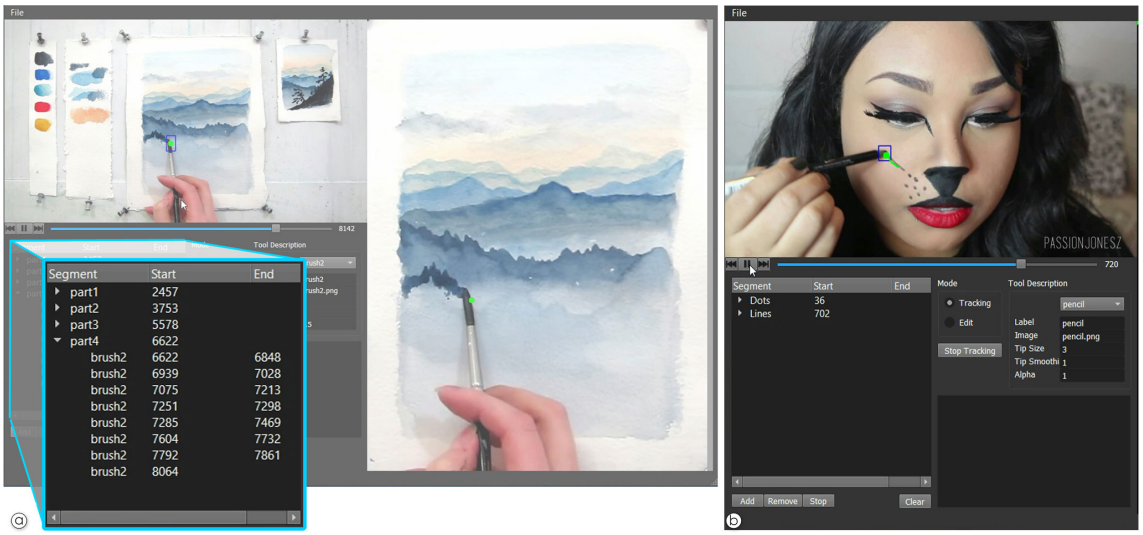



**Figure 4.9:** The video retargeting user interface. (a) Detail view showing the retargeting process of a watercolor painting tutorial. (b) The system automatically detects faces in tutorial videos and uses the deformable face model.

We collected feedback on the authoring step for samples from a facial make-up and a painting scenario in an expert evaluation. Facial make-up tutorials include motions with surface contact on a 3D human face model. Figure 4.9(b) illustrates the video tutorial retargeting process. The retargeting system automatically finds the face of the tutor. The author has to initialize the tool tracker in the first frame of every segment of the video and stop tracking in the last frame of a segment. In a few cases, the tracker had to be re-initialized after tracking failures.

The painting tutorial includes motion with surface contact on a canvas. Figure 4.9(a) illustrates the painting tutorial. We model the canvas as a planar object, and further extract actions and layers. We use Adobe Photoshop as an interface for editing, where we load the extracted layers automatically. Photoshop provides operations like translation, rotation or scale, allowing to modify the input tutorials. We can add layers from multiple tutorials to combine several source tutorials into a new one.

Four expert users, experienced in using image and video editing software, participated in the evaluation. Before starting, the participants familiarized themselves with the content of the videos, the task in the tutorial and the authoring tool. To collect feedback on the tool tracking, we asked the experts to extract the motion in two ways: first, by redrawing the line manually on the surface texture representation that shows the final result of an instruction; second, by using the tool tracker to extract the path of the tools automatically.

To comfortably extract start and end frames of instructions we allowed the modification of the input video frame rate by a scale factor. Based on our experience, we set the scale factor to 0.5, resulting in a time-scaled video. While users appreciated scaling the speed of the video playback, they also asked for interactive control of the scale factor.

For the facial make-up sample, participants required approximately *four minutes for 50 seconds* of time-scaled video. For the painting tutorial, participants required around *eight minutes for three minutes* of time-scaled video.

The tool tracker was met with positive responses. Aside from occasional tracking failures, participants were comfortable using this tool. When directly compared to redrawing the instructions manually, participants considered the tracking was faster for extracting the instruction. Two participants even stated explicitly that it is more accurate. With respect to the tracking, one participant remarked that our results are precise enough to rather rely on the extracted original drawing than on a line she is redrawing manually.

Participants generally suggested more advanced tools, such as shortcuts to fill areas in paintings. The current authoring software relies solely on extracting paths. However, future work will investigate the use of collections of tools that allow the efficient extraction for actions in areas. A fill instruction could be specified in one frame and automatically extended to instruction steps over multiple frames.

## 4.3 Conclusion

This section of the thesis demonstrates how the vast resource of online video tutorials can be retargeted into three-dimensional AR tutorials. Instead of presenting simple video overlay in AR, we synthesize 3D motions from the video data and register the results to the 3D objects around the user and to the user's body, where appropriate. Moreover, our system dynamically generates 3D glyphs to represent the important information in a visually compact and easily comprehensible way. Our resulting AR tutorials allow the user to freely change the viewpoint while watching the instructions. This enables the user to examine elements of the tutorial from a more effective point of view compared to the one used in the original video.

We have demonstrated several example applications for retargeting 2D video tutorials. However, not all videos can be successfully retargeted. For example, if the input video is very dark or noisy, our silhouette segmentation requires excessive user input. Advanced video processing methods, such as super-resolution techniques, may help in such cases.

Moreover, we have only scratched the surface of the potential of letting a user control the point of view of a tutorial. Our initial results also indicate that the benefit depends on the type of motion. Therefore, we plan more formal evaluations on how to guide a user to an optimal viewpoint for different classes of 3D movements.

# CHAPTER 5

# Remote Authoring of Augmented Reality Documentation

One particularly powerful application of AR documentation is *remote assistance*, where an expert (remote user) helps a worker (local user) in operating or repairing a physical object on location. The author, i.e. the expert, uses a live representation of the remote user's physical environment in addition to tools for exploring and annotating the shared environment with visual instructions [44, 65, 111]. Subsequently, the remote user has an AR display to view the visual instructions, which were generated by the expert user, within its physical environment.

Implementing such a remote assistance application faces two key challenges. First, the remote user requires a virtual representation of the local user's environment, which must be provided on the fly and allows for identifying all details necessary to complete the task. Second, both the local user and the remote user require intuitive interaction techniques for exploring and annotating the shared environment. Therefore, exploration and annotation must be performed in 3D space to register the information correctly in the local user's environment.

In this work, we are particularly interested in mobile scenarios, as those are free of spatial constraints that encumber spontaneous use on stationary hardware. However, existing approaches relying on mobile devices for remote assistance are often restricted to 2D representations [180], provide 3D representations of limited visual quality [44] or rely on additional stationary equipment [114]. Moreover, we experienced that, in addition to the limited visual quality, existing approaches struggle to create proper virtual representations of featureless, transparent or shiny objects.

To address these challenges, we propose a new approach to remote assistance, which does not require a geometric model, but, instead, purely relies on an image-based representation in the form of an *unstructured light field* [36], i.e., a database of images registered in 3D space, which represent a sampling of the light rays emitted from the local user's workspace.

As we will show, light fields offer many advantages over previous approaches. For example, no depth sensor is required, and reconstruction is not adversely affected by texture-less, shiny or transparent surfaces. This robustness is an essential advantage of light fields over traditional reconstruction approaches, for instance, when considering industrial environments with lots of metallic surfaces. This enables our approach to work in many more environments compared to existing MR remote assistance systems. Figure 1.3 shows an example of this on a metallic engine, which would be challenging for traditional reconstruction methods commonly used in MR [45]. While light fields offer high visual quality, they also face challenges complicating their use in remote assistance applications. Creating light fields can be time-consuming, which is critical for remote assistance applications. Furthermore, a naive light field implementation results in a large number of images, which easily exceeds what can be transmitted, stored or rendered on mobile devices.

Finally, light fields lack explicit 3D geometry, making them difficult to interact with or modify [69]. Common tasks, such as placing graphical annotations on object surfaces captured as light fields, are not trivial without depth or surface information.

The *Mixed Reality Light Fields* presented in this thesis address these issues. In particular, we provide a practical approach for utilizing unstructured light fields in MR on mobile devices. To demonstrate capturing, processing and annotation of light fields, we chose a challenging application, namely, remote assistance. In this application, we use light fields as a robust, high-fidelity representation of challenging scenes containing transparent, thin and shiny objects. Using Mixed Reality Light Fields, we do not only support a novel form of instant exploration of reconstructed objects, but we also support collaboration in the shared space through a novel method for the navigation and annotation of remote scenes.

## 5.1 Overview

We present a complete end-to-end system for remote assistance in MR using light fields. Our system provides an Augmented Virtuality [100] environment for the remote user and an AR environment for the local user [100]. The system records and captures a scene, sends the recording to the remote user, who annotates it with visual instructions and sends the annotations back for visualization in AR (see Figure 5.1 for an illustration of the components of our system).

In contrast to existing approaches, which use geometric reconstruction, our system is based entirely on images. While this trivially allows reproducing otherwise difficult to reconstruct real-world objects, the surrender of a geometric representation makes a new approach for the collaborative workflow necessary. We provide an overview of the workflow components in the remainder of this section.

*Scene Capture.* We start by capturing an image database using the built-in camera of the user's mobile device and sending it over the network. To limit the captured images to a manageable amount, we follow a strategy of overview+detail [144]. Initially, the user interface presents a minimal set of registered images for the remote user to chose from. The choice is relayed to the


**Figure 5.1:** Overview. (a) Scene capture: The local user shares the environment by capturing a local light field. The sampling process is visually guided by a 3D sphere that surrounds the object of interest. The sphere color encodes the current sampling density per subtended angle, allowing to identify those regions of the light field that require more sampling. The target sampling density is automatically specified by the system but may be adjusted by the remote user on demand. (b) Scene exploration: The remote user explores the light field using image-based rendering techniques. (c, d) Scene annotation: Once a suitable viewpoint has been reached, the remote user places a plane in 3D and starts annotating it with drawings sketched on the touchscreen of the mobile device. (e) AR visualization: The visual instructions are sent to the local user and presented within the 3D coordinate system that was used for capturing the light field. Therefore, the visual instructions naturally appear as 3D-registered augmentation in the local user's environment.

local user with the inquiry to scan selected locations with denser sampling (Figure 5.2(a)). The result of this dialogue is a set of local light fields in a common reference system, captured where deemed necessary by the remote user for supporting the subsequent placement of annotations. Figure 5.1(a) illustrates our interface for capturing the dense local light field. Sampling is guided by visualizing a sphere of directions, indicating by color-coding which directions have already been sufficiently sampled.

*Scene Exploration.* Exploring the remote environment serves two purposes. First, it supports optimizing the capturing process by providing the interface for guiding the local user to those locations where more data capturing is needed. Second, based on a captured local light field, it supports validating and refining the 3D placement of the AR annotations. Therefore, our approach for remote exploration supports camera control with six degrees of freedom based on light field rendering in combination with an overview visualization of all available viewpoints (Figure 5.1(b)).

*Scene Annotation.* After exploring the shared environment, the remote user defines one or more support planes within the 3D remote coordinate system. A support plane serves as a canvas for drawing via the touchscreen of the mobile device. Placement of support planes is assisted by an approximated depth and surface normal, computed dynamically via depth from focus. The location where depth is estimated in the light field is interactively indicated by the remote user with a single touch gesture. (Figure 5.1(c)).

*AR Visualization.* Since local user and remote user share the local user's local coordinate system, the 3D annotations created by the remote user can be presented directly in the local user's AR display (see in Figure 5.1(d)).

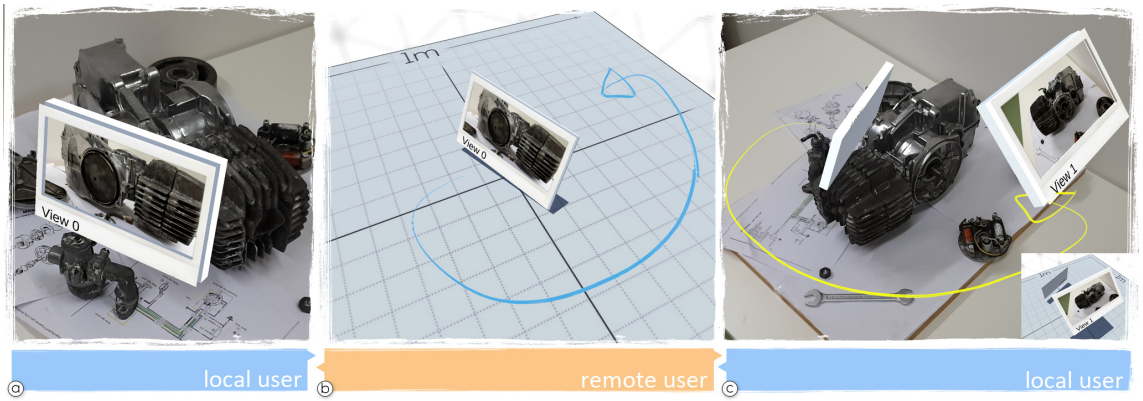



**Figure 5.2:** Spatial user guidance: (a) The local user initiates the session by taking one or more pictures of the scene. The pictures are spatially registered in AR and automatically labeled to simplify communication. Note the label "View 0" in this example. (b) The 3D registered images are immediately sent to the remote user to enable coarse scene exploration. The remote user can then guide the local user to different locations by drawing hints on a world-registered, virtual ground plane. (c) The annotations are sent to the local user and visualized as a registered AR overlay. Once a satisfactory location has been found, the remote user can place a highlight on the correct picture frame. The local user then starts capturing a local light field as outlined in Figure 5.1(a).

## 5.2 Interactive data capturing

Sharing a light field of the entire environment is expensive in terms of time, network and computational resources. Therefore, we capture and send only local light fields. A local light field represents a small section of the environment. In our application, it represents the structure that the remote user is going to annotate. The remote user informs the local user of the locations where local light fields should be acquired, so that the resulting image density is sufficient for maintaining a high visual quality of the environment, thus, allowing a precise anchoring of annotations. For this purpose, spatial guidance is provided to the local user.

### 5.2.1 Spatial user guidance

A remote assistance session starts by asking the local user to capture an overview of the environment. The local user is instructed to acquire a coarsely spaced collection of images by pointing the tracked mobile camera at objects identified as potentially interesting. While capturing the images, our system records the tracked position and orientation of the user's camera using ARCore. Similarly to the work by Sukan et al. using snapshots [149], we use the camera poses to present

the data as 3D registered annotations of the real and the shared virtual environment. In addition, we automatically label the snapshots to allow referring to images by their name (*View 0* in Figure 5.2(a, b)). Note that the image appears in both environments, on the remote user's mobile device (Figure 5.2(b)) as well as within the local user's AR environment (Figure 5.2(a)).

### 5.2.2 Guided light field capture

After the remote user identifies the structure to be annotated, the local user captures local light fields. Capturing is automatically triggered when the mobile device is close to the snapshots marked by the remote users as interesting. The local user only has to move the camera towards these snapshots, which are displayed as overlays in the AR view. We found that capturing a spherical light field [67] is a good fit for our needs since we want to provide the highest detail in the area of interest. The spherical setup ensures that captured rays are oriented towards a single point of interest, the center of the sphere. This provides the highest sampling density for the structure that the remote user is going to annotate.

To restrict the amount of data that must be transmitted over the network, we capture only a small section of the spherical light field. The relevant section is defined by a rectangular window cut-out on the sphere surface. Therefore, the local user first captures four images of the object of interest from the four corner points of the window. We found that a subtended angle of approximately 30° gives enough variation to create high-quality light field renderings. Since the distance of the user to the object of interest can vary, we do not prescribe a minimal subtended angle for a local light field, but, instead, let the local user extend the subtended angle explicitly if deemed necessary. This strategy avoids forcing the user to cover large distances with the camera for far-away objects.

After the system derives the center and the bounds of the local light field, it visualizes the light field coverage as a tessellated sphere. The resolution of the tessellation corresponds to the desired sampling density. The user's task is to move the device around while keeping the object of interest close to the center of the screen. The portion of the sphere currently in the line of sight to the center changes color when an image is captured. Thus, the capturing process becomes a coloring task that supports the user to identify sufficiently sampled and under-sampled areas of the light field. After a local light field is considered complete, its center is re-computed as the closest point to the optical axes of all captured images.

## 5.3 Scene exploration

The remote user explores the scene using a mobile device with a touchscreen, i.e., a tablet or a smartphone. The remote user may navigate by orbiting around the center of the spherical light field and zooming towards it. For fast visual feedback, we blend the closest two views when the virtual camera is in motion. During the exploration, we automatically transition the virtual camera to the closest keyframe. This strategy presents the scene in the highest possible quality whenever the

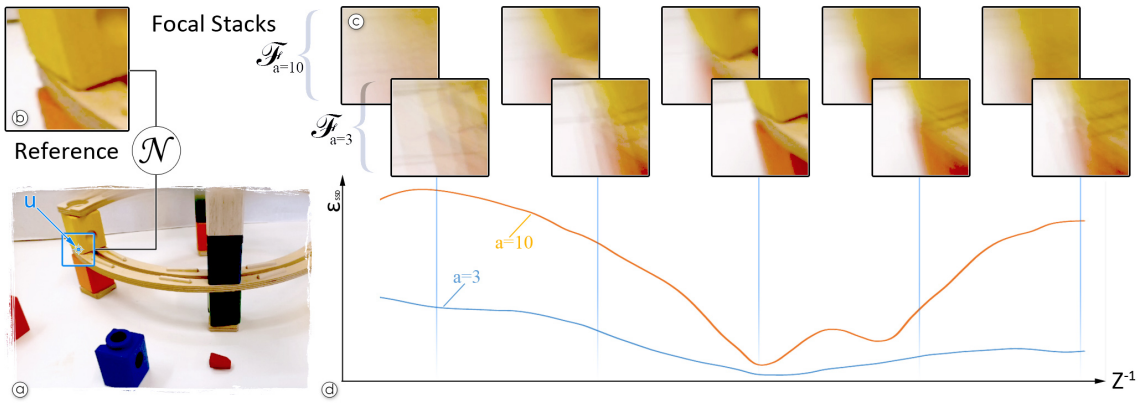



**Figure 5.3:** Auto-focus estimation via synthetic focal stack rendering and interpretation: (a) Camera image for the current view position. The point we want to focus on is denoted as $u$. (b) The search window $\mathcal{N}$ defines the reference image for our focus metric $\varepsilon$. (c) A synthetic focal stack is generated, proving test images at regular intervals along $Z^{-1}$. In this example, the aperture size $a = 3$ and window size $N = 15$ has been used. (d) The minimum of our focus metric $\varepsilon$ denotes the best match in the focal stack, from where the depth of $u$ can be determined.

virtual camera is not in motion. A similar strategy was used by Gauglitz et al. [45]. However, the effect of our version produces significantly better results, as our scene representations consist of densely sampled views instead of just a sparse set of keyframes. For a sufficiently dense sampled light field, transitions between two captured frames are barely noticeable.

## 5.4 Scene annotation

Common tasks during a remote assistance session include the identification of objects (using a circular outline around the object or a cross-hair at its center), the communication of object movements (using an arrow), placement of objects (drawing the outline of the object at the target place), handwriting, and any combination of identification, movement and placement.Our system is designed to support these types of visual instructions by mapping arbitrary 2D drawings to registered 3D AR annotations. Via a support plane, the remote user can draw 2D strokes on the touchscreen. The drawings are presented as an AR overlay to the local user. The remote user can create any number of planes and can draw any number of instructions on each of them. When finished, the remote user releases an annotation to the local user.

After receiving the light field, the remote user navigates the virtual camera to a suitable viewpoint for drawing the annotations. Upon a single tap on the object of interest, the system automatically determines the corresponding depth of the selected area, where it places the support plane as a canvas for the remote user's freehand drawing, e.g. outlining an object to guide the local user's attention. The drawing plane is initially oriented parallel to the camera's image plane, but can be adjusted subsequently, if necessary. We first describe our approach to automatically place the drawing plane, before we outline the interface for refining the initial placement.

### 5.4.1 Automatic canvas placement

To draw annotations in a light field, we automatically place a drawing plane at the depth of a user-selected structure. We estimate this depth from evaluating a synthetic focal stack $\mathscr{F}$, which we generate by rendering the light field at different focal planes. Subsequently, we search the focal stack $\mathscr{F}$ for the slice $f$ that gives the sharpest image according to the metric $\varepsilon$,

$$f \leftarrow \min_{I_f \in \mathscr{F}} \varepsilon(I_{\mathrm{KF}},\, I_f), \tag{5.1}$$

where $I_{\mathrm{KF}}$ is the image in the light field at the current viewpoint, and $\varepsilon$ defines how well the focal slice $I_f$ matches $I_{\mathrm{KF}}$,

$$\varepsilon(I_{\mathrm{KF}}, I_f) = \sum_{\mathbf{u} \in \mathscr{N}} (I_{\mathrm{KF}}(\mathbf{u}) - I_f(\mathbf{u}))^2, \tag{5.2}$$

where $\mathscr{N}$ is a $N \times N$ window centered at $\mathbf{u}$, which is the point of the user selection. Note that, compared to conventional auto-focus photography, our approach benefits from the fact that an all-in-focus image ($I_{\mathrm{KF}}$) is available as a ground truth. Therefore, we can directly compare the sharpness, rather relying on statistical measures within an image patch [83, 116].

The accuracy of our depth estimation approach depends on the quality of each image in the synthetic focal stack. A good focal stack for a subsequent auto-focus analysis provides a sharp image only at the distance of the selected structure. We achieve this by following the approach for unstructured light field rendering proposed by Davis et al. [36], which we modified with a novel non-linear blending scheme.

In the unstructured light field approach, the geometric proxy of the triangulated viewpoints results in the piece-wise linear interpolation of the three closest viewpoints, which form overlapping rings of triangles. Each ring consists of a shared vertex and surrounding vertices $\mathbf{v}_0$ and $\mathbf{v}_i$ $(2 \leq i)$ with the highest and the lowest weights, respectively. Within each triangle forming a ring, the weights are linearly interpolated from the shared center to the outer vertices.

Note that Davis et al. [36] used cubic interpolation across two rings to ensure smoothness with vertex-wise linear blending. We use faster piece-wise linear interpolation, but re-map the weights $w_L$ to achieve pixel-wise non-linear weighting $w_{NL}$,

$$w_{NL} = sin(w_L\, \pi/2). \tag{5.3}$$

This pixel-wise non-linear interpolation achieves a natural Bokeh effect when combined with synthetic aperture. Given a user-defined aperture size $a$ $(\geq 1)$, a synthetic aperture is simulated by shifting the surrounding vertices $\mathbf{v}_i$ in each ring from the shared center vertex $\mathbf{v}_0$ to a new position $\mathbf{v}'_i$ at the rim of the aperture on the focal plane at distance $f$,

$$\mathbf{v}'_i = f\,(\max(a,1)\,\mathbf{d} + P(\mathbf{v}_0))\,, \tag{5.4}$$

where $\mathbf{d} = P(\mathbf{v}_i) - P(\mathbf{v}_0)$, and $P([X,Y,Z]^{\mathrm{T}})$ projects a 3D point to a depth-normalized plane as $[X/Z, Y/Z, 1]^{\mathrm{T}}$.

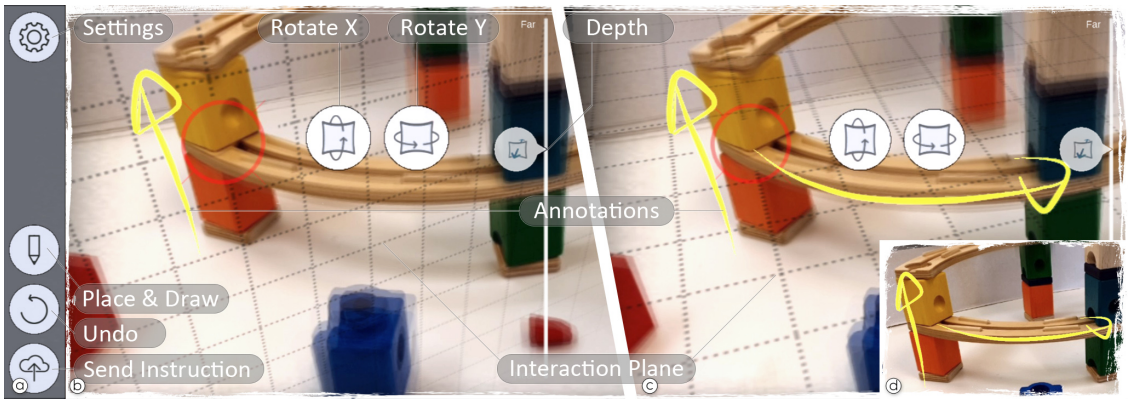



**Figure 5.4:** Interface of the remote expert user. (a) We provide a simple set of three buttons to initialize canvas placement and drawing, to undo the last action and to send visual instructions. (b) The remote user is able to refine an initial placement. The red circle indicates the user's focus selection. By pressing and sliding from one of the two buttons in the center of the screen, the user can rotate the canvas. (c) The rotated canvas after refinement; note the yellow arrow. (d) After sending the instructions, the local user's application shows the instruction as AR annotation.

After projecting all rings and summing up all projected pixel colors, the resulting colors are normalized with respect to the sum of the weights. The resulting quality of the depth of field depends on the number of images that we blend for each triangle of the geometry proxy, determined by the parameter $a$ in Equation 5.4. The larger $a$, the smaller the depth of field becomes. However, using more images causes higher computational costs for pixel blending. Changing aperture furthermore requires to adapt the window size $N$.

A minimum of three images must be blended on a single triangle proxy. This configuration ($a$=1) results in the sharpest possible image. There is no upper bound on $a$, but blending across a large portion of the proxy mesh slows down the light field rendering. We empirically found that $a = 3$ in a $15 \times 15$ window represents a good trade-off between performance and quality on a Samsung Galaxy S9. We used this setting in the user study.

### 5.4.2 Canvas refinement

We support adjusting the canvas interactively to align its rotation and translation when needed. Therefore, we allow adjusting yaw and pitch rotations using the two modifiers shown in the center of Figure 5.4(b). By pressing and dragging one of the modifier buttons, the user can rotate the drawing plane. The modifiers act as clutches, making all modifications incremental. Larger displacement can be aggregated by repeating smaller motions. In addition to the rotation modifiers, the user can fine-tune the position of the annotation plane by dragging the slider on the far right of the interface back and forth (see Figure 5.4(b) and (c)). Note that we do not support editing roll rotations as we are only interested in adjusting the placement of the plane.

We also support editing the strokes which the user draws on the canvas. Thus, we allow redrawing instructions by providing a delete option. Unneeded or wrong annotations can be dismissed, either by the remote user before transmission or, later, by the local user. The option to allow the local user to dismiss annotations enables to remove already performed instructions.

## 5.5 Evaluation

We performed a series of evaluations on the performance and usability of our system. The evaluations focus on our approach for placing annotations in the light field data, on the effectiveness of the resulting AR annotations, and on the interface for capturing the light field data.

### 5.5.1 Experiment 1: Light field annotation

We tested the capability of our interface for annotation placement in light field data by comparing it to a multi-view approach [69, 119]. Our interface combines the light field renderer for exploring the scene, the auto-focus-based canvas placement, and the manual refinement for further adjustments. This interface is denoted as *AF*. We compared it to a multi-view approach *Multi-View*, which is an alternative 3D interaction method requiring no depth information. *Multi-View* relies on manual interactions to place a point in a 3D environment. In *Multi-View*, a ray is projected along a vector from the center of projection of the camera to the screen point indicated by the user in a single image. The user can observe the 3D line from a different angle by interactively changing the viewpoint. To adjust the depth, the user slides the target point along the ray.

**Task.** We designed a task for placing annotations in 3D environments. We prepared four light fields in different scenes (Figure 5.5). In each annotation method, participants were required to place points at five given positions per light field.

**Apparatus.** We used a Samsung Galaxy S9 smartphone, both for recording light fields (using ARCore for 3D tracking) and for touch interaction. We collected four light fields. The smallest contains 110, and the largest, 186 images at a resolution of 800 $\times$ 400 pixels. In each light field, we manually placed five target points to be annotated by participants.

**Design.** We designed a repeated-measures, within-subject study. We define an independent variable "system" with two conditions: *AF* and *Multi-View*. We measured task completion time (TCT), i.e., the time between starting and finishing a 3D annotation placement, distance error, i.e., the point-to-plane distance between the prepared point and the placed drawing plane, subjective workload, using the raw NASA TLX [52], usability, via the single ease question (SEQ) [139], and overall performance.

**Procedure.** After filling out a consent form and demographics questionnaire, users were introduced to the first condition. The starting order of conditions was counterbalanced using a Latin Square. Participants were standing and used their dominant hand for interacting on the touchscreen. Participants familiarized themselves with the system by performing as many test placements as they liked, then they performed the task by placing the annotations in one scene per time. Participants were instructed to be fast and accurate. Between the scenes, participants were forced

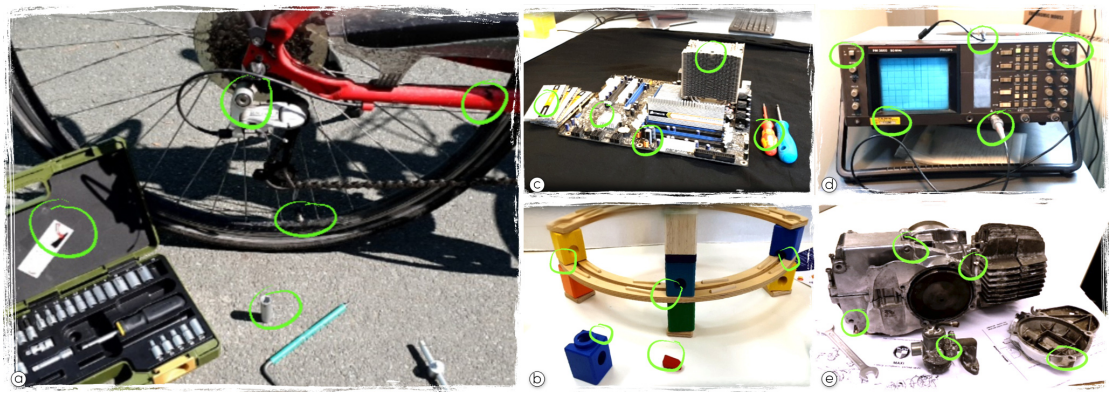


**Figure 5.5:** Evaluation scenes. (a) The light field used for training, (b-e) four light fields for measuring user performance. Each scene has been prepared with five different target points (marked with a green circle).

to rest for 10-20 seconds to recover from possible fatigue caused by holding and interacting on the touchscreen of the mobile device. Upon completion of the condition, users filled in the SEQ and NASA TLX questionnaires and continued with the remaining system. After completing the final task, the user filled out the preference questionnaire.

**Hypotheses.** We expected that *AF* would outperform *Multi-View* in terms of (**H1**) speed (TCT) and (**H2**) error rate, as *AF* provides depth automatically.

**Pilot.** We performed a pilot study with the described setup. Six participants (1 female, $\bar{X} = 27.3$ ($SD = 2.2$) years old) volunteered. Performance analysis revealed no significant differences in time (*AF*: $\bar{X} = 17.3$, $SD = 9.7$; *Multi-View*: $\bar{X} = 16.1$, $SD = 7.1$; $p = 0.79$), error (*AF*: $\bar{X} = 2.3$, $SD = 2.7$; *Multi-View*: $\bar{X} = 2.7$, $SD = 4.1$; $p = 0.85$), TLX (*AF*: $\bar{X} = 31.9$, $SD = 15.8$; *Multi-View*: $\bar{X} = 24.3$, $SD = 9.1$; $p = 0.43$), and SEQ (*AF*: $\bar{X} = 4$, $SD = 1.3$; *Multi-View*: $\bar{X} = 3.8$, $SD = 0.7$; $p = 0.89$). Four out of the six users preferred *Multi-View*.

Participants commented on missing visual feedback ("There is no visual feedback after placing the canvas in the auto-focus mode, which made me wonder whether the system worked."). Since our module for exploring the remote scene aims at providing an all-in-focus image, no visual feedback about the selected focal stack is provided. Participants mentioned a lack of confidence when no visual feedback was provided. Without the visual feedback, they had to use camera rotation to rely on perspective cues for validation or the slider for changing the placement. This caused similar additional interactions in the *AF* condition compared to the *Multi-View* condition, while in the *Multi-View* condition, the ray visualization helped to follow the adjustment. Participants commented that this was the main reason for preferring *Multi-View*.

Participants also commented on the slider precision for manual adjustment in the auto-focus condition ("The sensitivity of the slider is too high. The focus is either too far or too close.").

**Revision.** The interface was modified in the following way:

- We visualized the focal slice at the selected depth in the auto-focus interface, until canvas placement was finalized and confirmed by pressing a button. This change added visual feedback about the performance of the auto-focus.

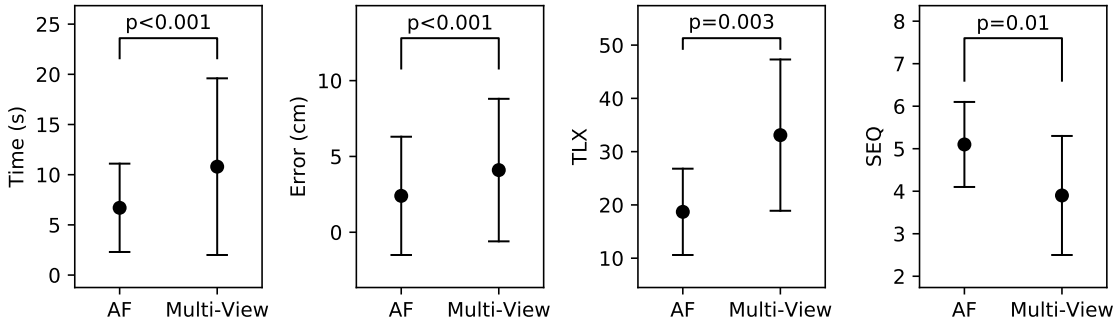



**Figure 5.6:** Results from experiment 1.

**Figure 5.7:** Following AR instructions. We tested the effectiveness of our system despite the added registration error in two step-by-step instruction tasks. (a) Two steps of a calibration procedure. (b) A participant following the instructions. (c) A computer maintenance procedure used in the second task.

- We reduced the sensitivity of the slider for manually adjusting the depth of the canvas.

We recruited 20 participants (3 female, $\bar{X} = 29.3$ ($SD = 4.1$) years). The setup and the procedure were identical to the pilot.

**Results.** Wilcoxon signed-rank tests revealed significant differences between *AF* and *Multi-View* for time (*AF*: $\bar{X} = 6.7$, $SD = 4.4$; *Multi-View*: $\bar{X} = 10.8$, $SD = 8.8$; $p < 0.001$), error (*AF*: $\bar{X} = 2.4$, $SD = 3.9$; *Multi-View*: $\bar{X} = 4.1$, $SD = 4.7$; $p < 0.001$), TLX (*AF*: $\bar{X} = 18.7$, $SD = 8.1$; *Multi-View*: $\bar{X} = 33.1$, $SD = 14.2$; $p = 0.003$), and SEQ (*AF*: $\bar{X} = 5.1$, $SD = 1.0$; *Multi-View*: $\bar{X} = 3.9$, $SD = 1.4$; $p = 0.01$) (Figure 5.6). Finally, sixteen out of the twenty users preferred *AF*.

**Discussion.** Overall, the results of the revised study greatly favor using *AF* over one relying on *Multi-View*. Even some participants did not always press the confirmation button immediately, *AF* was significantly faster. Also, *AF* was significantly easier to use (TLX and SEQ), which led to a significantly reduced error and was overall preferred by the majority of the users. These results are in general agreement with prior work on editing light fields using desktop interfaces, which showed that focus-based approaches are faster than multi-view approaches [69].

However, we should emphasize again that our interface is different in several ways from prior work: Existing focus-based approaches were fully manual, while ours is automatic. Moreover, participants used a 2D WIMP interface, while our participants conducted the task on a mobile device tracked in 3D space. These differences may also explain the discrepancies in error rates. In the study by Jarabo et al. [69], multi-view interfaces had the same error as focus-based interfaces, while our automatic focus led to a significant reduction of the error.

### 5.5.2 Experiment 2: Following annotations

We tested the effectiveness of AR annotations generated with the *AF* interface. Since the auto-focus approach possibly introduces a small registration error, causing an offset between the real object and the visual annotations in AR, we were especially interested in the user's performance in case of such erroneous registration.

**Task.** We designed a task that requires following step-by-step instructions using the AR annotations. We prepared two real-world use cases with AR annotations. The annotations guide the user through the calibration of an oscilloscope (Figure 5.7(a, b)) and the maintenance of a computer (Figure 5.7(c)). Both tasks were unknown to all participants.

**Apparatus.** We used a Samsung Galaxy S9 smartphone running our AR interface. We used ARCore for 3D tracking and for image-based pose initialization. The image-based pose initialization allows us to measure the ground truth positions of all AR annotations in 3D space. To include a registration error, we added the mean error with a randomized offset relative to the standard deviation, which we both derived from experiment 1. Note that we could have used user-generated annotations for this task. However, since we were interested in the user performance in the presence of erroneous registration data, we wanted to make sure that a plausible error exists in the registration of the AR annotations.

**Design.** After completing both step-by-step instruction tasks, we asked participants to fill in a system usability scale (SUS) questionnaire. In addition, we asked the users to provide verbal feedback on the effectiveness of the visual instructions.

**Procedure.** After completing a consent form and demographics questionnaire, users were introduced to AR guidance system in a small training session, which included two instructions. Sixteen (16) participants (2 female, $\bar{X} = 28.1$ ($SD = 3.1$) years) volunteered.

**Results.** We measured an average SUS value of 91.56 with a standard deviation of $SD = 7.28$. Verbal comments were very positive, including statements like 'It feels very responsive and useful. I really want to use that application.', 'It was fun to use. Especially in the oscilloscope task, I learned something useful'. The only negative comment we received was addressing the lack of user-perspective AR rendering on smartphones: One user noticed the mismatching perspective.

**Discussion.** The SUS score of the AR interface is higher than the average of 70 and, based on the analysis of Bangor et al. [11], can be translated into the rating "excellent." The verbal comments demonstrate that users were able to focus on the actual task and were not distracted by the erroneous registration. The problem of device perspective rendering in an instruction task can be overcome by an implementation of user-perspective rendering [105].

### 5.5.3 Experiment 3: Guided light field capturing

We tested the usability of our interfaces for spatial user guidance and for light field capture. We were interested in the effectiveness of the overview visualization in the remote user's interface and on the usability of the interface for placing visual instructions on the local user's ground plane. Furthermore, we were interested in the effectiveness of the resulting AR guidance visualization and on the usability of the interface for capturing the light field images using the tessellated sphere visualization, as described before.

**Task.** We designed a data capturing task, in which a local and remote user capture arbitrary local light fields. An experimenter, who was familiar with the system, acted as the local user, while test subjects were asked to assume the role of the remote user. We showed them two images taken from different points in the local user's environment, and we asked the remote users to guide the local user to the object in the picture.

**Apparatus.** We used a Samsung Galaxy S9 smartphone for both, the remote user and the local user. The applications were connected through a Wifi hotspot. In addition, we used an extra mobile phone and a Bluetooth headset for verbal communication.

**Design.** After capturing the target light fields, we asked the participants to switch roles. Before switching, we asked the participant to fill in a SUS questionnaire, and we collected verbal feedback. The scenes showed random objects.

**Procedure.** After completing a consent form and demographics questionnaire, the participant was introduced to the interface. We recruited ten (10) participants for the experiment (2 female, $\bar{X} = 29.2 (SD = 3.7)$ years).

**Results.** We measured an average SUS value of 81.5 with a standard deviation $SD = 10.5$ for the expert's spatial guidance interface, and we measured an average SUS value of 80.5 with a standard deviation $SD = 11.7$ for the interface for guided light field capture.

Verbal recordings show a mixture of positive comments on certain features of the interface and suggestions for improvement. User comments include "I liked the simplicity of the sphere indicator and the painting task for capturing, this is easy to use," "I'd like to see the video" [of the local user], "the remote expert needs to get a notification that the instruction was received," "it is difficult to estimate the scale of the instruction" [in the expert's interface], and "the annotations are sometimes visible even when behind objects."

**Discussion.** The SUS score of both interfaces is higher than the average of 70, why, based on the analysis of Bangor et al. [11], both can be translated into the rating "excellent." Although the SUS scores are high, we noted several suggestions for improvement.

*Live video stream.* We noticed that users of the expert interface were asking for the live video stream to get more information about the local environment. We will add low-resolution video streaming. However, to get more information about the local user's position, we will furthermore provide the local user's current position and orientation in the expert user's interface. For better history browsing we will also increase the density of the overview visualization by adding the frames from the live video stream as 3D registered billboard annotations, similar to the current keyframe visualization.

*Performance visualization.* During the introduction of the capture interface, we noticed that defining the extension of the light field required more explanation than we expected. Users were uncertain to estimate the angular distance from the center. We explained that it is not important to precisely find the corner points, and we gave verbal feedback whenever we thought it was necessary, telling users that the corner points were good enough. Therefore, in the next release of our interface, we will add visual feedback, showing a performance indicator based on the initialized extents of the light field. In addition, sharing the sphere visualization with the expert will enable remote adjustment of the light field size.

*Scale visualization.* Users that were using the expert interface commented on the challenge to estimate the scale of the local user's environment. This makes correctly bending arrows difficult. The scale is visualized as a grid on the ground plane (see Figure 5.2(b)). However, we will add additional scale indicators to simplify the spatial understanding of the local user's environment. For example, we will provide the local user with an interface for roughly framing the object of interest with a box. The registered box will be sent to the expert user and visualized in addition to the ground plane and the keyframes.

*Occlusion management.* Users were also commenting on wrong occlusions handling. As we render the keyframes and the visual instructions on top of the AR user's camera feed we cannot resolve occlusions correctly. This problem is inherent to an AR rendering without explicit proxy geometry. However, as real objects will occlude virtual drawings mainly after large viewpoint changes, this problem will mostly occur during spatial user guidance. In the spatial guidance interface, we provide keyframe billboard annotations in addition to drawings which are commonly arrows to indicate a certain direction. To mitigate occlusion errors for these cases, we implemented rendering of front-facing billboards only. This reduces the amount of occluding fragments caused by billboards placed behind the object from the user's current point of view.

## 5.6 Conclusion

In this chapter, we focused on the questions if mixed reality light fields are a good representation for remote assistance scenario and if challenges associated with light fields (in particular, capture and annotation) can be addressed with a carefully designed user interface. We believe both questions can be answered affirmatively. We were able to confirm our expectation that mixed reality light fields support remote assistance well, even on objects that are otherwise difficult or impossible to reconstruct. Adding annotations to light fields that lack explicit surface geometry can be successfully facilitated using automatically computed support planes derived via depth-from-focus. These findings and their embodiment in our telepresence system show that mobile devices are sufficient for capturing light fields in practice.

A remaining technical limitation is that annotations are restricted to 2D support planes. An extension of our work to 3D annotations would require new visualization and interaction techniques that lift the interaction beyond 2D planes. To handle occlusions, a coarse 3D approximation generated from the images could be generated in a background process.

There are many more avenues for future work. Our evaluations concentrate on the interaction to capture and annotate light fields. A comparison to existing telecollaboration frameworks would be inherently difficult, as they are highly diverse, and the outcome of such a comparison would depend on the chosen tasks and application scenarios. Nonetheless, important insight could be gained from such comparisons.

We would also like to evaluate the presence aspect of our telepresence system. While the focus of the work presented in this thesis was primarily on usability and not on the feeling of "being there" (spatial presence) or "being there together" (co-presence), a follow-up study could deliver important insights on how the photorealism afforded by light fields can enhance presence. However, it should be noted here again that the motivation for using light fields was not necessarily only the visual quality they offer but the robustness to material properties that are otherwise hard to capture.

We believe that our work has relevance beyond the current scope of remote assistance. Mixed reality light fields are a versatile extension of the current scope of remote assistance technologies. They lend themselves to use cases where complex geometry and appearance must be comprehended quickly using just a mobile device. For example, in medical education, anatomical models can be explored and discussed, and a lecturer could assist by correcting label placements. As many parts of our everyday world tend to be visually complex, we expect that many more compelling use cases can be identified.

# CHAPTER 6

# Visualization

Augmented reality presentations of instructions can significantly reduce the cognitive load for the user [56]. However, the visualization methods must be carefully chosen for each type of tutorial. To name an example, users following a dance-tutorial must match the speed, velocity and 3D position of their legs and arms to the instruction. This is quite difficult for the learning user, thus the visualization must adapt to the performance of the user, otherwise, it would be overwhelming and irritating. In this chapter, we discuss the challenges of visualizing different kinds of instructions in an AR environment. We start with the most important rendering techniques such as video phantoms, dynamic 3D motion path visualization, tool path visualization on surfaces. Another important topic of this section is the optimal viewpoint selection and how to guide the user there. The chapter is completed by two user studies, where we analyzed the behavior of users in a 3D motion guidance system, as well as users following an AR tool path guidance system.

## 6.1 Rendering techniques

Our analysis of 2D documentations reveals any combination of an annotated 3D model and a set of consecutive 3D motions. Our final goal is to easily relate the information to real-world objects using an AR interface. Therefore, we register the 3D model to its real-world counterpart and track the user's display to make the augmentation visible. Since we present the documentation in a dynamic and often visually complex environment, we also provide interactive visualization tools as part of the AR interface.

### 6.1.1 Registration and tracking

The registration of the 3D model to its physical counterpart uses an RGB-D sensor and PCL-Kinfu, which is an implementation of KinectFusion [108] using the Point Cloud Library (PCL) [133]. Our system registers the 3D model automatically to the point cloud received from the RGB-D sensor using an implementation of Sample Consensus Initial Alignment (SAC-IA) [132]. Camera tracking is realized using PCL-Kinfu and initialized with SAC-IA.

### 6.1.2 Visualization methods for AR

The registered 3D model allows presenting the 2D documentation within the real-world environment of the user. However, in AR, we usually have to cope with a visually complex background and dynamic camera motion. Therefore, AR documentations demand special techniques to provide effective visualizations. To reduce the visual complexity of our AR visualizations, we allow animations to be presented as video phantoms [74]. This resembles a real-time photomontage (Figure 1.1, Figure 6.1(b)) and thus reduces the visual complexity by decreasing the number of virtual objects in AR.

While ghosted views enable effective exploration of hidden objects, they fail to create effective visualizations for visually complex structures. However, since our 2D input graphics provide optimized visualization, we furthermore provide the user with a replicate of the 2D configuration in 3D AR. We implement this by computing the object configuration shown in the optimized 2D image. To provide a smooth animation from the object configuration in reality to the one depicted in 2D, we furthermore derive the configuration of the real-world object in front of the user. Therefore, we run our diagram analysis using a screenshot of the live video and the optimized 2D graphics. Figure 6.1(b) shows the resulting animation from using the images seen in Figure 6.1(a) as input. The selected brewing unit is clearly shown after replicating the object configuration used in the traditional documentation material.

## 6.2 Indicating optimal Viewpoints

Replicating the object configuration from an optimized 2D illustration works best from a point of view which is close to the one used in 2D. During camera estimation, we derive this point in space, and, thus, we can guide the user to these optimized points of view. As demonstrated in Figure 6.1(c) we provide this information to the user by adding an avatar looking at the object to the scene. The avatar indicates the point of view the image was taken from, and thus for which the configuration of the object is optimized.

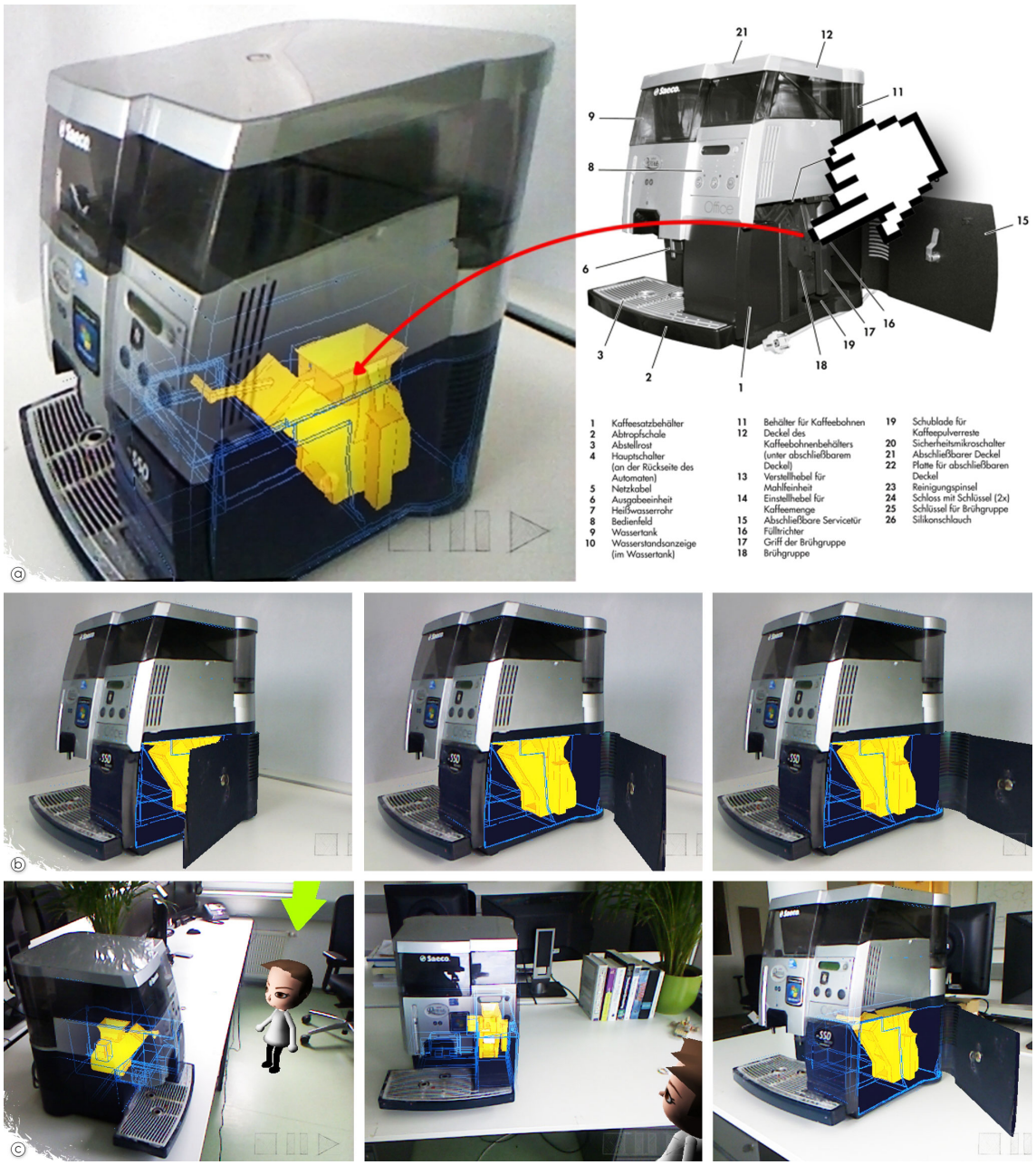



**Figure 6.1:** Interaction. (a) The relation between 2D image elements and 3D structure allows us to manipulate the 3D visualization with 2D interactions. The selection of the brewing unit triggers the highlighting of the corresponding 3D object. If the selected object in the real-world is occluded, we provide the user with a ghosted view x-ray visualization. (b) We allow to replicate an optimized object overview configuration in a 2D image by computing the steps for transforming the real-world object into the one depicted in the 2D documentation (e.g., opening the service door). (c) The optimized object configuration is most effective from the point of view used in the 2D documentation. We extract the point of view from which the 2D image was generated and present it to the user by showing an avatar. This example shows the user navigating to the extracted point of view.

## 6.3 Guidance for body motion

A key feature of our video retargeting system is the ability to visualize extracted 3D poses and motion from an arbitrary and interactively controlled point of view. Our goal is to increase the user's understanding of the expected motion and its position in relation to himself. Therefore, we generate graphical elements that effectively visualize key poses and motions. The visualization is based on a 3D avatar that we derive from a rigged 3D reconstruction of a real person 1.4.

**Pose Visualization** We automatically segment long motion sequences into chunks of shorter and easier to understand movements. This allows us to concentrate on smaller parts of a long and complex motion without cluttering the user's view. We guide the user to the next step in the sequence by visualizing a 3D stick-figure (see Figure 1.4(a) which is registered to the user's body center. Bones are displayed as thin white tubes, and joints are marked using circular glyphs that are oriented towards the image plane. To guide the user to the first pose, we furthermore connect the tracked joint positions to the desired positions using rubber-bands. As soon as the user has reached the desired pose, the circular glyphs are highlighted in green, and the path visualization starts to animate.

**3D Path Visualization** The motion in-between two key poses is visualized using an array of 3D arrows along the path and pointing in the direction of the movement. The stick-figure depiction will move with the velocity that has been extracted from the video. However, the stick-figure depiction will only move a pre-defined amount of distance ahead in time to give the user time to react and reorient himself, if the deviation is too far from the instruction. The arrows will vanish after successfully following the path. Thus, only the remaining part of a 3D motion is presented keeping clutter in more complex motion minimal. To control clutter, we allow controlling the level of detail for leg and arm motion. This results in path visualizations for all joints (e.g. wrist, elbow and shoulder) or only a subset of those (e.g. the wrist). Which subset of paths to visualize can be selected by the user at run-time. As more path visualizations increase the complexity of the instruction, a level of detail visualization allows adjusting the complexity based on the user's confidence.

### 6.3.1 Evaluation

While our motion instruction system can be configured for presenting the extracted 3D tutorial from an arbitrary point of view, one particularly interesting configuration is to present the 3D motion from the user's head position. Therefore, we started exploring the effectiveness of our system by presenting our user registered 3D body instructions from an ego-centric point of view. We attach the camera which renders the 3D motion to the user's head position, and we allow the user to change the viewpoint onto the motion path by naturally looking around. We call this type of presentation egocentric Augmented Reality (*EAR*) visualization as it directly augments 3D instructions to the user. We compared this type of visualization to a more established presentation on a large screen showing the video data by using the extracted avatar in combination with the extracted instruction in front of the user (Figure 6.2(a)). Similar to Tang et al. [156], we use a split-

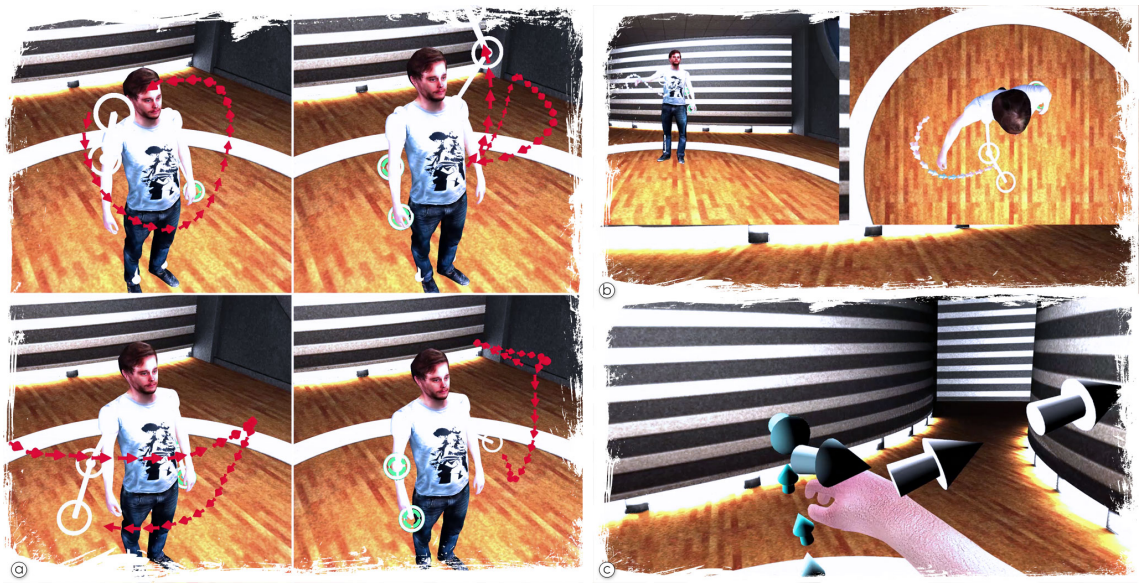


**Figure 6.2:** User evaluation. (a) To compare ego-centric visualizations to an AR mirror setup, we present 3D body motions that are registered to the user's skeleton. For egocentric visualization the 3D point of view is attached to the user's head position. By tracking head motion, the user can control the orientation of the viewpoint by natural head movements. In our experiment, we asked the users to follow the 4 motion paths which are illustrated in this figure. (a) We present motion instructions using the AR mirror visualization technique. To provide the user with the same hardware setup in both conditions, we render a virtual AR mirror in front of the user in a virtual environment. The user is wearing an Oculus Rift display in both setups (see Figure 1). We use a split-screen setup on the AR mirror providing a top-down and front view. (b) (top) (bottom) Mean values and standard deviations of (left) task completion time in seconds and (right) euclidean distance to the target path (in mm).

screen setup to provide the user with two views, a top-down and a front view. This setup represents a common Augmented Reality Mirror, which is why we refer to this condition as (*ARM*). Note, that both viewing conditions can be set up using an Augmented Reality headset such as the Microsoft Hololens. However, we choose to test our system in VR, as current VR headsets provide a much larger view of view which we considered necessary for this application.

**Setup** We use the same setup for EAR and ARM. The head tracker which is integrated into the head-mounted display maps the user's head motion to camera motion in 3D space. We track the user's skeleton with a Microsoft Kinect, using the open-source skeleton tracking framework NiTE. NiTE's skeletal tracking is providing us 15 joints with positional and rotational data which we applied to the rigged avatar. Our implementation runs in over 30 frames per second on a standard desktop PC (Intel Core-i7 CPU, NVidia GTX 680 GPU, Windows 8.1).

**Task** Each participant was asked to follow four motions using EAR and ARM (Figure 6.2). As 3D motion is mostly demonstrated in video tutorials, we are specifically interested in testing the performance of our visualization for 3D movements. Therefore, we explicitly avoid motion within a single plane, such as used in physiotherapy training [156].

**Procedure** We recruited participants on the campus of the university. After filling out an informed consent form and a demographic questionnaire, we explained the visualizations and allowed them to get used to the immersive environment. After they were confident with the setup, we asked them to perform the given task. We counterbalanced the tasks by randomizing the order and the type of motions for each user.

**Results** A total of 12 subjects (9m, 3f) aged 22-32 participated in our experiment. Figure 6.2(bottom) presents the average time taken to complete each task and the average error while following the motions. The measurements show that the 3D presentation (EAR) outperforms the 2D presentation using a split-screen mirror setup (ARM). The data also indicates that the difference between both forms of presentation varies depending on the 3D motion. As this evaluation was meant to provide first feedback from user performance, we did not design an extensive study. Due to the lack of a sufficient amount of data we did not perform any further statistical analysis.

However, next to collecting quantitative indicators we asked all users about comments and personal preference for EAR or ARM. Eleven out of the twelve participants preferred EAR over ARM, while the remaining participant was not in favor of either one. A common comment from the participants was that EAR allows traveling along the path much more self-reliant, while ARM requires looking at one's own body and the mirror concurrently. A few subjects noticed some jitter of the tracked skeleton but did not feel obstructed in solving the task.

### 6.3.2 Discussion

Our instruction visualization system enables the playback of 2D video tutorials with synchronized 3D animations in a Mixed Reality environment. This enables many different forms of presentation of the same scene. For example, instead of presenting simple augmented video in AR, we synthesize 3D motions from the video data and register the resulting visualization to the user's body. Our system dynamically generates interactive 3D visualizations to guide a user to certain key poses and to visualize the motion in-between the poses. The resulting 3D tutorials allow to arbitrarily change the viewpoint while following the instructions. This enables to examine elements of the tutorial from a more effective point of view compared to the one used in the video.

Our system can be applied to a wide variety of 2D video tutorials showing human motion. However, as following the tutorials relies on online tracking of the user's skeleton we are currently limited to moderately fast motion. The availability of more powerful hardware will improve the skeleton tracking at runtime, which will also reduce the jitter that some of the users were noticing.

The preliminary experiment indicates the potential of a user-controlled point of view in 3D space. However, as our system enables many more configurations, including static and dynamic 3D viewpoints, it provides a powerful framework for further research on optimized and interactive viewpoint control for Mixed Reality tutorials. Since our initial results indicate that the benefit of our system depends on the type of motion, our future research on viewpoint optimization will consider different classes of 3D movements.

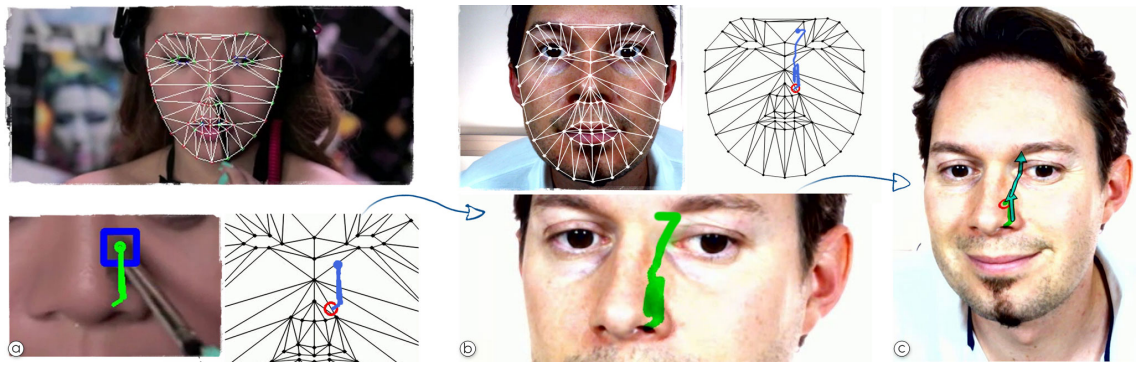


**Figure 6.3:** System overview. (a) We extract object and user motions by tracking known model features in the 2D video. Here, tracked features are used to record the path of the brush and to align a face model in each frame. (b) After validating and possibly editing the extracted motion, we retarget the motion data to real-world 3D objects. This requires registering the same 3D model as used in the extraction stage, in this case, a face model, to the live camera image. By tracking the model in 3D, we are able to relate video data to the real world. In this example, we present the recorded path of the brush directly on the user's face. (c) Since we retarget the extracted motion data in 3D, we can choose an arbitrary point of view. To provide effective visual instructions, we generate dynamic glyphs (here: timed arrows) and we indicate the position of the brush over time using a red circle.

## 6.4 Guidance for tools

To visually communicate instructions we generate graphical elements based on the extracted 3D motion. Following the work of Nienhaus et al. [110] the goal of our design was to introduce minimal visual clutter by providing abstractions of the motion. Therefore, in our initial design, we create dynamic glyphs based on arrows which we combine with an animation of the motion. We present the animation using a red circle which marks the tip of the captured tool. However, we iteratively optimized our design as users were not totally satisfied with its usability. We present the design iterations in the evaluation sections.

In all our visualizations we reduce the complexity of motion paths, as raw motion trajectories are often too cluttered and jittery to be suitable for path visualization (Figure 6.3, middle). Our filter simplifies a path by first using the approach of Douglas et al. [38](Figure 6.5(b)), followed by an additional segmentation of the paths into smaller segments. To segment the path, we search for turning points by comparing the angle between two neighboring line segments to a threshold (the example in Figure 6.5(c) uses a threshold of 90°). Subsequently, we cluster turning points that are placed close to each other by recursive merging based on distance. The resulting paths can be used to abstract the motion using arrows. We create an arrowhead at each turning point and the endpoint (Figure 6.5(c)).

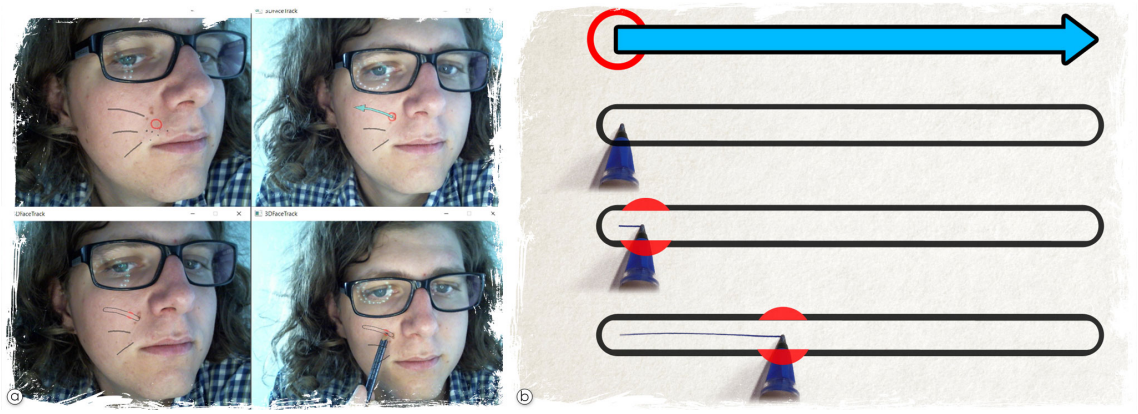


**Figure 6.4:** First revision of AR path visualization. (a) The combination of visualization techniques provides an overview first, before the user can follow the exact motion. (b) At runtime, we use the arrows to provide a preview of the motions. To minimize occlusion, the arrow is replaced by the border of the tool's trajectory. The red dot shows the extracted tool position over time.

### 6.4.1 Evaluating Efficiency of AR Make-Up Tutorial

A second experiment was conducted with the goal to corroborate the precision of activities performed when following the AR tutorials. The intention was to observe the reaction of non-technical users in performing a common task aided by AR tutorials and to compare the results with those obtained when following conventional video instructions.

**Design.** We introduced a structure for making comparisons. The study had two conditions: video (V) and AR. The AR condition showed the instructions step by step. We chose a face-painting task based on a video downloaded from Youtube (Figure 6.6(a)). The tutorial involved two types of precision tasks, namely painting points and painting lines. It had the advantage that it could be divided into two symmetrical parts: left and right side of the face.

The study was organized as a repeated measures design with two independent variables: interface (V, AR) and task (left, right side of face). The task was treated as random variable for counterbalancing the design so that each participant uses a different configuration. The possible configurations were $(AR_{left}, V_{right})$, $(AR_{right}, V_{left})$, $(V_{left}, AR_{right})$, $(V_{right}, AR_{left})$. Participants were randomly assigned to configurations. A make-up AR-Mirror was built for the study, replacing a table-stand make-up mirror with a display (Mimo Magic-Touch, 10.1", 1024x600 pixels) and camera, shown in Figure 6.6(b). We control the AR visualization using a standard PC mouse and a next button, and we control the video using the interface of a common video player with functionality to scroll back and forth.

**Pilot.** We performed a pilot study with the described setup. Three female participants ($\overline{X} =$ 33 years old) were asked to take part in the test. They signed a consent form, accepting that their performance will be video-recorded. Answers to a pre-test questionnaire indicated that one participant relies on make-up videos, whereas the other two had never followed video instructions for make-up before. The session was closed with the participants rating the difficulty of reaching for the controls and a semi-structured interview.

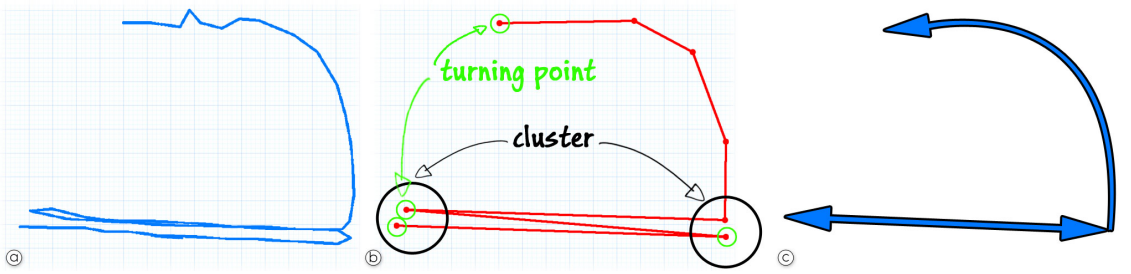



**Figure 6.5:** (a) We generate path illustrations from motion capture data. (b) The extracted path data is analyzed and simplified. In particular, we remove zig-zag overdraw along the trajectory by clustering and detect turning points (marked in green). (c) We generate arrows in-between turning points, the start point and endpoint.

Having to reach for controls was not seen as hindrance either in AR ($M = 4.6$ of 7, higher means easier) or in V ($M = 4$). Participants commented on the lack of preview in AR ("There is no preview. I have no idea what I am trying to achieve because I only see each individual instruction.") and on occlusion issues ("Occlusion. I cannot see my skin. The instruction is getting in the way, and I cannot see if I am painting it correctly or not.").

**Revision and Experiment.** After the pilot, the interface was modified as follows:

- Full preview was added for future steps of the tutorial.
- The visualization was modified to avoid occluding the user's skin. It displays only the outline of the motion trajectory and the motion is shown using an animated circle (Figure 6.4(b)). The arrow is still used to preview a segment's path, but it fades out after the preview phase (Figure 6.4(b)).

Six participants took part in the study (1 female, $\overline{X} = 34.3$ years old sd=4.8). The setup and the procedure were identical to the pilot. However, in contrast to the pilot study, we asked to follow the instructions as accurately as possible. Note that, while other researchers have compared monitor versus AR instructions [56], we were interested in the perceived quality of the instructions generated with our method. However, we also measured error and task completion time to evaluate the performance of our AR visualization system.

**Results and Discussion.** The task completion time was measured using a stopwatch from the point in time where participants announced the start of the drawing. The drawing was considered finished when the last stroke was placed. The error was measured by normalizing the texture atlases containing the strokes of the tutorial make-up and the user-drawn make-up and comparing them using the l2-distance between the image pixels. Wilcoxon signed-rank tests did not reveal any significant differences in time (V: mean=82.2s, sd=29, median=72; AR: mean=102.4s, sd=31.9, median=88) or error (V: mean=0.45, sd=0.04, median=0.47; AR: mean=0.42, sd=0.02, median=0.42).

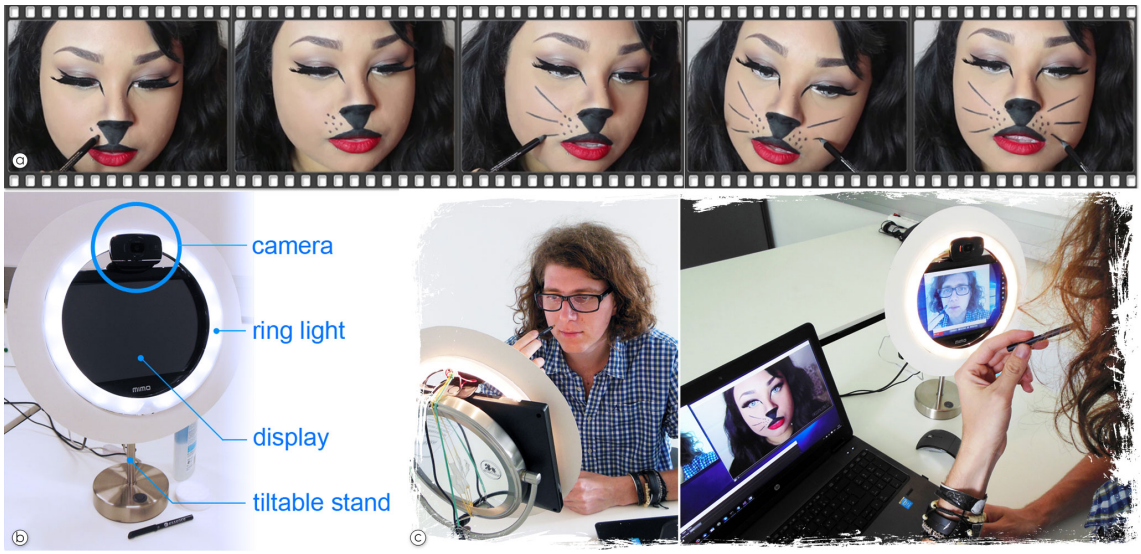



**Figure 6.6:** Experiment setup for a retargeted make-up tutorial. (a) Input video tutorial. (b) We showed the resulting AR tutorial using an AR mirror, which consisted of a camera and a USB display. (c) Participants could use the AR mirror and the video which we placed next to the mirror.

We received overall positive feedback on the AR condition. Comments from the experiment included "I was more confident of being accurate when using AR."; "I felt I was quicker using AR. Being accurate with the video was difficult, because I didn't know where exactly I have to place the dots."; "The mirrored video was difficult to (mentally) invert."; "In AR I didn't have to think, I could concentrate on the drawing.".

All participants unanimously preferred AR over the video when the goal was being as accurate as possible. In a comparative questionnaire, they unanimously expressed feeling more confident and faster with the AR interface. In the video condition, two participants did not pick the correct side of the face, when starting the task. They were instructed to continue on the correct side. This was not an issue in AR, which showed instructions directly on the face of the participants. However, while the participants felt that AR allowed them to follow the motion exactly where they should appear, we noticed that lines drawn in the AR mode tend to include a bit more jitter. We believe that this is because in V people drew continuously while in AR they used a stop and go strategy which allowed them to repetitively validate the result. We see two possible reasons for this behavior. First, our 3D face tracker is not perfectly modeling face deformations. Therefore, the augmented lines were floating a bit over the skin of the user as soon as the face was deformed. This may have distracted the user resulting in a stop and go strategy. Second, our visualization encodes the speed of the motion using a moving dot. This may have distracted participants, because they switched their focus from being accurate in space to being accurate in time and the other way around, thereby causing them to continuously interrupt the motion.

Even though we received positive feedback on our AR interface, it did not outperform video-based tutorials in task completion time or error rate. We believe we detected no differences due to the imperfect real-world modeling of the user's face in AR and the distracting animation which was not necessary for the task.

### 6.4.2 Evaluating Efficiency of AR Kanji Tutorial

After the make-up study, animations were removed from the interface. Instead, the direction of the motion is encoded in the border of the instruction glyph (Figure 6.7(c)). We performed a third experiment to collect quantitative data on the performance of the second revision of our AR visualization system. Since we speculate that the face tracking solution was not accurate enough to allow objective comparison of the quantitative data, we switched to a drawing scenario, in which accurate tracking and a rigid scene (without deformation) could be ensured. Therefore, participants were asked to follow Kanji drawing tutorials on paper using our AR interface and a common video interface.

In the AR condition, participants used an Optical See-Through Head-Mounted Display (HMD), a Microsoft Hololens, to receive instructions augmented on a piece of white paper (Figure 6.7(a) (top left)). The Hololens has no standard mouse interface, therefore we had to change the AR interface so that switching to the next instruction was done using a handheld controller (Figure 6.7(a) (top right)).

**Design.** We designed a repeated measures within-subjects study to compare the performance and user experience of the AR interface to a common video interface. We introduced one independent variable interface with two conditions: AR interface (AR2) and video interface (V2). The task was to follow a tutorial with the goal to draw a single Kanji symbol in a target area as it was shown in the respective interface. The task was repeated ten times for each interface using a different Kanji symbol for each repetition. Correctly following the tutorial involves drawing strokes of the proper size, from the correct direction and in the right order.

We prepared a set of 20 symbols, which were of similar complexity (based on the number of strokes) consisting of 6 to 8 strokes. The 20 symbols were divided into two pools of ten symbols, each pool contained an equal total number of strokes. The tasks and the pools of symbols were counterbalanced to avoid learning effects and bias by the choice of symbols for each pool.

As dependent variables, we measured task completion time and error of each task, subjective workload measured by the NASA TLX [52], usability on the System Usability Scale (SUS) [23] and overall preference.

**Apparatus.** Participants performed the task standing in front of a whiteboard and drawing with an ordinary pen on a $10cm \times 10cm$ piece of paper attached to the board (Figure 6.7).

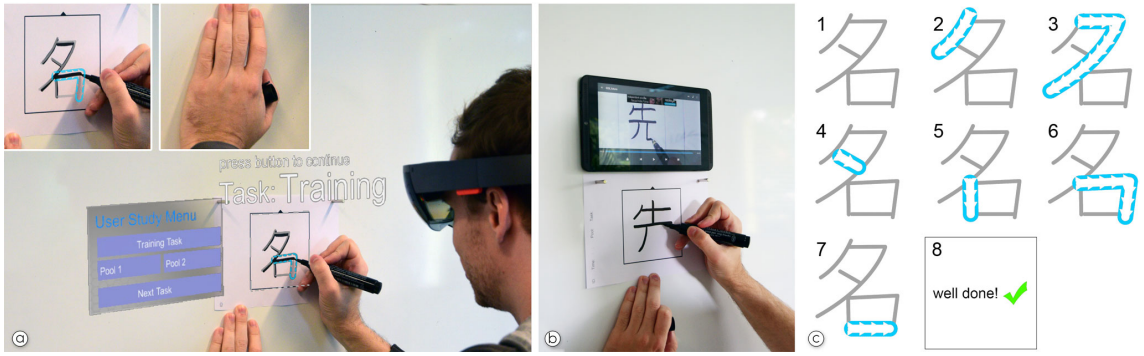



**Figure 6.7:** Retargeted Kanji tutorial and final revision of AR visualization. (a) The AR visualization is presented using an Optical See-Through HMD (Microsoft Hololens) and a handheld clicker that the user is holding in one hand. (b) The video tutorial is shown on a tablet mounted right above the drawing area. This reduced the influence of head motion. (c) Our final glyph design encodes the direction of the stroke on its border using arrowheads. The system presents one glyph at a time next to a full preview of the final drawing. This picture shows the six instructions presented to the user in AR.

In AR2, participants wore a Microsoft Hololens HMD and used the second revision of our visualization (Figure 6.7). In V2, an Nvidia Shield tablet ($17.2cm \times 10.8cm$) showing the instruction video was mounted above the target area (Figure 6.7(b)). Participants were presented the unmodified tutorial video and had to follow the instructions. The MX Player application[1] was used as the video player. The unmodified tutorial videos did not show an overview at the beginning of the tutorial. Participants could browse through the video using common video controls (Figure 6.7(b)).

**Procedure.** After an introduction and filling out a demographic questionnaire, participants performed a training task for the first interface using a training symbol, different for AR2 and V2, that was not part of the tested set of symbols. After participants were comfortable using the interface, the measured tasks started and participants were instructed to be fast and accurate. The symbols from the current pool were shown in random order. After finishing the 10 symbols for one interface, participants filled out the NASA TLX and the SUS. The second condition started thereafter, following the same procedure. After filling the second SUS questionnaire, participants filled out a preference questionnaire and a semi-structured interview was conducted. A session took approximately 45m.

Task completion time was measured using a stopwatch from the point in time where participants announced the start of the drawing. The drawing was considered finished when the last stroke was placed. The error was measured as in the previous study using the l2-distance between the image pixels of the tutorial Kanji and user-drawn Kanji. **Hypotheses.** Due to the presentation of the tutorial using AR and the preceding authoring step to process the tutorial instructions, we expect that when working with AR2 users will be significantly faster and more accurate than when using unmodified video instructions (**H1**). Furthermore, users will prefer AR2, due to its improved usability and intuitive visualization (**H2**).

[1] `goo.gl/xd5rb6`, last accessed September $20^{th}$, 2016

| Cond. | Time (s) | Error | SUS | Nasa TLX |
|---|---|---|---|---|
| AR2 | 20 (3.7), 19.9 | 0.41 (0.03), 0.4 | 89.6 (10.1), 92.5 | 22.4 (11.6), 21.3 |
| V2 | 30.7 (6.6), 30.1 | 0.46 (0.02), 0.46 | 68.1 (20.8), 76.3 | 46.5 (16.1), 45.4 |

**Table 6.1:** Measurements of Kanji study (mean (sd), median).

**Results.** 12 participants (3 female, $\overline{X}$ =31.3 (sd=6.2) years old) volunteered for the study. On a scale from one to five, five meaning best, the mean of self-rated AR experience was 2.7 (sd=1.2), video tutorial experience was 3 (sd=0.6), Kanji experience was 1.3 (sd=0.65) and general drawing ability rated as 2 (sd=1.3). With 12 participants, two interfaces and ten different symbols per interface, there were a total of $12 \times 2 \times 10 = 240$ trials. The data was evaluated using a level of significance of 0.05 and Wilcoxon signed-rank tests.

Table 6.1 shows the mean, standard deviation and median of task completion time, error, NASA TLX and SUS for AR2 and VR2. Figure 6.8 shows the boxplots of the measurements. Wilcoxon signed-rank tests revealed a statistically significant difference in task completion time ($(Z = -3.0594, p < 0.001, r = 0.62)$), error ($Z = -2.9025, p < 0.05, r = 0.59$), NASA TLX ($Z = -3.0594, p < 0.001, r = 0.62$) and SUS ($Z = 2.3552, p < 0.05, r = 0.48$). In all cases AR2 outperformed V2. These results support H1 and H2.

All 12 participants preferred AR2 over V2, when asked to choose one of the two interfaces in the after-study questionnaire.

**Discussion.** Our results support H1 and H2. Our system clearly outperforms traditional video tutorials and is also preferred by the participants. The median SUS value of 92.5 for the AR interface is higher than the average of 70 and, based on the analysis of Bangor et al. [11], can be translated into the adjective "excellent". The traditional video interface has a median SUS value of 76.25 and receives the adjective "good" [11].

Participants were very positive about the AR interface in the after-study interview and clearly preferred this interface, similar to the results of the previous study.

Five participants mentioned the clear benefit of AR in the ability to control the speed in which the instructions were shown. The lack of speed control in V2 was considered as stressful because the video was either too slow or too fast. Two participants remarked that they appreciated the preview of the finished Kanji at the beginning of the instructions, which underlines the usefulness of our authoring system in reformatting video tutorials into a more sophisticated format for learning.

Four participants again noted problems with occlusions in the AR condition. The visualization sometimes occluded the drawn line, especially when they drew out of the bounds. While closing off the indicated drawing area could be regarded as desirable feature, it would be better, if a visualization could identify user-relevant content to avoid occlusion. Seven participants noticed that the focal plane of the Hololens did not match the surface they were working on. This is a long-known problem with this kind of HMD technology. While unpleasant, participants could still easily finish the task.

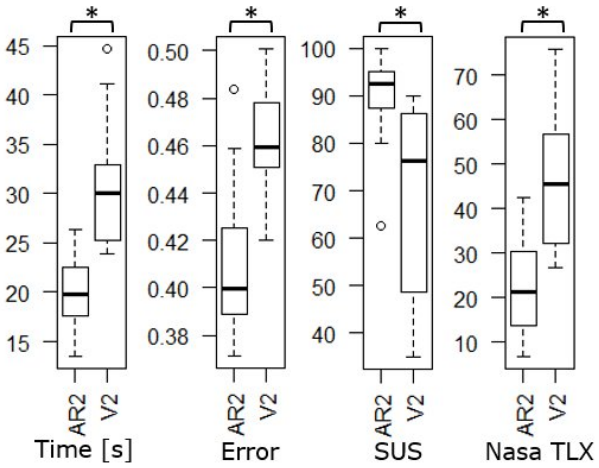



**Figure 6.8:** Kanji study results. Stars indicate significant differences.

In condition V2, participants paused the video frequently to keep up with the instructions. This indicates the value of step-by-step instructions as extracted by our system. While we compared our system to a standard video interface V2, it would be interesting to compare the AR condition with a more advanced video interface that also supports step-by-step instructions or even automatic pause-and-play techniques such as the one presented by Pongnumkul et al. [120]. We speculate that part of the performance difference between AR and V2 comes from the segmentation of the tutorials into steps in the AR condition and that the performance of a more advanced video tutorial will be closer to the AR condition.

Based on a visual comparison of the resulting drawings, we noticed no difference in jitter between drawings generated with interface V2 compared to interface AR2 (see the complementary material for scans of the drawings). Since we changed both, the glyph design and the application (to one which does not require deformable object modeling and tracking), a future experiment will have to investigate the actual impact of each of the two factors to jittery drawings in AR instructions.

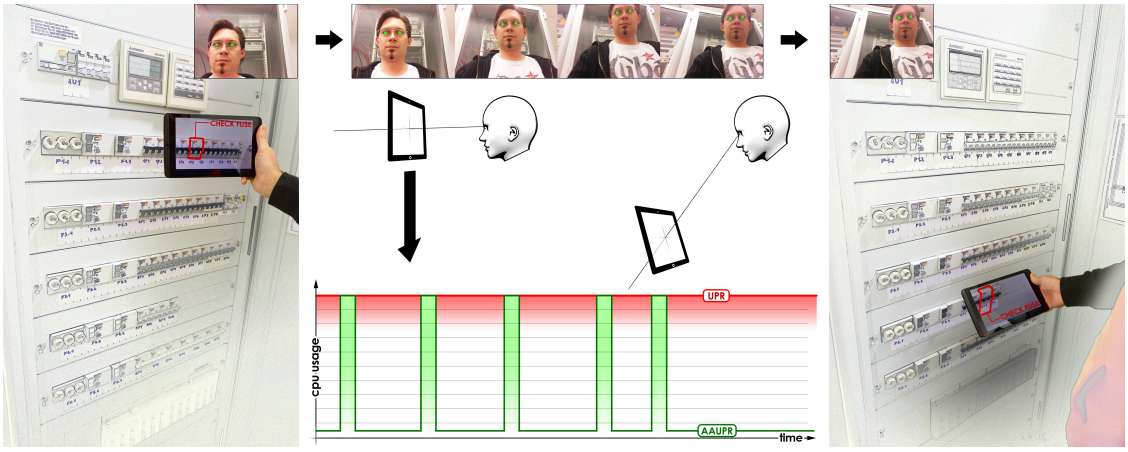
# CHAPTER 7

# Interacting with Augmented Reality Documentation using Smartphones

Augmented reality applications on handheld devices commonly implement some variant of the so-called magic lens rendering, which augments only a fraction of the user's real environment while the rest of the view remains unaffected. Since AR on handheld devices is commonly implemented as video see-through, AR magic lens applications often suffer from spatial distortions. Since the AR content is presented from the perspective of the camera of the mobile device, the user needs to re-map his actions to the real environment, e.g. the user's hand would appear at a different position within the augmented screen and the real world (see Figure 1.5). Recent approaches counteract this distortion based on estimations of the user's head position, rendering the scene from the user's perspective. However, user perspective rendering (UPR) demands high computational resources and therefore commonly affects the performance of the application beyond the already high computational load of AR applications. In this section, we present a method to facilitate AR interaction on handheld devices which also reduces the computational demands for UPR by applying lightweight optical flow tracking.

## 7.1 Overview

A number of research prototypes successfully demonstrate the implementation of user perspective rendering on high-end laboratory setups [Baričević et al.]. In contrast, our system is designed to provide user perspective rendering specifically on computationally limited mobile devices. The main insight used for the design of our system is that updates on the user's head pose are only necessary after a certain amount of motion and that small head motion can be ignored. Since FUPR assumes no head motion at all, it cannot support scenarios that require a large interaction space. Figure 7.1 shows an example application where a maintenance worker is receiving visual




(a) (b) (c)

**Figure 7.1:** System overview. In a typical usage scenario, the user moves the handheld device from one position (a) to another (c) to view different AR instructions. The resulting transition of the user's head pose, in relation to the device, is illustrated in the upper row of (b) showing single images from the front camera of the mobile device. The diagram in (b) shows a symbolic graph of the CPU usage during our approach (AAUPR) compared to UPR. During user motion, the head position is updated depending on the current threshold value. Once the user has moved to the desired point of view, the head pose is refined for this position (last peak in the graph).

instructions on a handheld AR device which include pressing buttons on a large electric cabinet. In such a scenario, the user has to correctly perceive instructions both, in the right top corner as well as in the corner on the bottom left of the cabinet. Such large distances between points in space that need augmentations cannot be handled by FUPR systems because the calibration of the user's spatial relationship to the display cannot hold.

However, based on the user experiment provided by Pucihar et al. [166] we assume FUPR provides effective UPR for small to no changes of the user's head position relative to the device. Therefore, we efficiently compute user perspective graphics by adding a low-cost tracker to estimate user motion before we start expensive head tracking for motion larger than a certain threshold distance. This approach can be seen as adaptive approximated user perspective rendering. In the following, we explain the main components of our system: (1) user perspective rendering on mobile devices, (2) efficient motion estimation, and (3) dual thresholding.

**User Perspective Rendering on Mobile Devices.** Traditional UPR requires estimating the user's 3D head position and the 6 degrees of freedom pose of the mobile device at every frame. Implementations on modern mobile smartphones derive the head pose using a 3D face tracker. The face tracker is usually applied to the video stream of the front camera while AR scene tracking is performed on the video stream of the back camera [141].

For device tracking, we use natural feature tracking. In our current prototype, we estimate the pose of the device using PTC's Vuforia SDK [1]. For 3D head tracking, we use a combination of a 2D deformable FaceTracker [137] and a subsequent 3D model, similar to the approach of Grubert et al [141], which provides real-time deformable face tracking and head pose estimation.

**Motion Estimation.** We derive an estimate of the user's motion in order to handle updates of the 3D head tracker. While 3D head tracking is expensive we aim at a low-cost motion estimation. Therefore, we estimate motion in image space only. Our prototype uses KLT-tracking of few very distinctive features. We use the face tracker which was applied during the last 3D head pose estimation in order to find the points which describe the eyes of the user. Motion estimation is subsequently performed on these two points only. Whenever KLT-tracking fails we start full 3D head tracking to update UPR and to re-initialize the motion estimation.

**Dual Thresholding.** After estimating the user's motion in image-space we apply simple thresholding to decide if an update of the user's 3D head pose is necessary. However, simple thresholding introduces an error relative to the size of the current threshold value. Furthermore, since we apply thresholding in image space the error increases for distant head poses.

In order to reduce the error during the interaction, we combine spatial with temporal thresholding. We assume that the spatial relationship between the AR display and the 3D head pose of the user mostly changes during scene exploration or while moving the display from one point of interaction to the next point. However, during interaction with the scene, the 3D head pose usually stays rather steady. Therefore, we reduce the threshold over time and re-initialize it to its maximal value each time we estimate the 3D head pose using the head tracker. This approach allows us to provide regular updates of the 3D head pose during scene exploration, while also ensuring a precise 3D head pose during interaction (assuming stable head poses during interaction). This approach is outlined in Algorithm 1 and illustrated in Figure 7.1.

**Algorithm 1** Dual Thresholding

1: $E \leftarrow |PosEyeCalc - PosEyeFlow|$
2: $\Delta E \leftarrow |PosEyeFlow_{last} - PosEyeFlow|$
3: **if** $E > \varepsilon$ OR ($\Delta E < \varepsilon * 0.1$ AND $!isPrecise$) **then**
4: $\quad recalculateFacePose$
5: $\quad isPrecise \leftarrow TRUE$
6: **else**
7: $\quad isPrecise \leftarrow FALSE$

$\varepsilon$ refers to the spatial threshold and $E$ to the error in pixels between the current estimated eye positions and the position calculated in the last precise detection step. $\Delta E$ is the difference of $E$ between the current and the last head tracking frame. The threshold $\varepsilon$ determines the trade-off between coarse but fast head pose estimation and precise but expensive tracking during user motion. Our system uses an empirically estimated $\varepsilon$ of 3% of the diagonal of the input image in pixels.

[1] https://www.vuforia.com

| System | Resolution | Frame Time | Tracking Time |
|---|---|---|---|
| UPR | 320x240 | 29.379 | 14.080 |
| AAUPR | 320x240 | 23.890 | 4.602 |
| FUPR | 320x240 | 20.733 | 0.0 |
| UPR | 640x480 | 42.860 | 30.094 |
| AAUPR | 640x480 | 28.706 | 13.363 |
| FUPR | 640x480 | 20.733 | 0.0 |

**Table 7.1:** Performance of different UPR implementations measured in milliseconds.

## 7.2 Performance Analysis

We compared the rendering performance of our system (AAUPR) to full-featured UPR and FUPR (which does not require any head tracking). The performance measures of have been recorded on an HTC-M8 Android smartphone. The numbers in Table 7.1 indicate mean values over 1000 frames for all conditions. *Resolution* refers to the image resolution of the back camera, *Frame Time* refers to the average time spent to render a single frame, and *Tracking Time* refers to the time spent for head tracking in total over 1000 frames. The resolution of the back camera was set to 640x480 pixels in all conditions, and visual tracking was performed in the same environment during all system measurements to provide the same number of visual features in all conditions. The front camera delivered new video frames at a maximum of 15 frames per second (which is the hardware limit of the camera). Please note that the head tracker runs in a separate thread.

## 7.3 User Experiment

**Design.** We designed a repeated measures within-subjects study to compare the performance and user experience of different implementations of user-perspective rendering. Therefore, we introduced an independent variable rendering with four conditions: device-perspective rendering (DPR), user-perspective rendering (UPR), approximated user-perspective rendering (FUPR), adative user-perspective rendering (AAUPR).

The task was a pointing task in which participants had to align the mobile device with a circular target area and touch the target area while looking through the view of the mobile device. The target area was only visible in the device view so that participants were forced to interact with the target area by using the different rendering conditions. Like Pucihar et al. [166] we are interested in the effect of the spatial distortion when looking through the device. Therefore, we do not show the live video during our experiment so that participants do not see their hands during interaction. The target area had a radius of 20mm based on the recommended size of ISO-9241 [1] for button input. The viewpoints of the target areas were placed randomly. For each rendering condition participants performed 40 repetitions. Rendering was counterbalanced using a balanced Latin square.

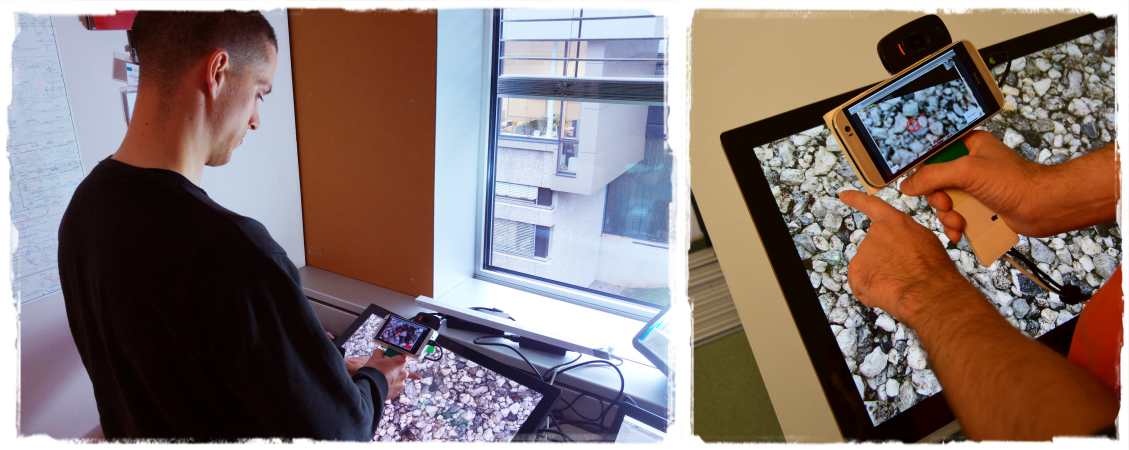


**Figure 7.2:** User Experiment.

As dependent variables, we measured task completion time (TCT) and error of each task, subjective workload was measured by the raw NASA TLX [52], usability using the Single Ease Question (SEQ) [139] and overall preference.

Sixteen participants (3 female, $\overline{X}$ =30.7 (sd=3.5) years old) volunteered for the study. On a scale from one to five, five meaning best, the mean of self-rated AR experience was 3.3 (sd=1.4).

**Apparatus.** Initially, we planned to perform the experiment using our mobile implementation on an Android-based smartphone (HTC-M8). However, due to the computational demands of the head tracking, the phone overheated and throttled the CPU during the pilot test after approximately 5 minutes in full UPR mode. All other conditions did not show this behavior. However, since the phone didn't cool down fast enough, CPU throttling had an impact on all subsequent measurements. Therefore, we did not use a mobile phone in the study setup but settled with a PC setup and a wired connection to a mobile display (Figure 7.2).

The apparatus consists of an installed touch screen and a handheld screen. The installed screen was a Dell S2340T multi-touch monitor of size 23" (506 x 287 mm) and was used for recording the touch input of the user. The handheld touchscreen was a HTC M8 phone (5" screen, 109 x 61 mm) attached to a PC via USB and was used to achieve an AR view implementing the different rendering conditions. The circular target areas of the task were shown as augmentation registered on the installed screen. The augmentation was only visible when viewed through the handheld AR device. The touch screen could be rotated to allow participants to comfortably work with the screen while standing. The head tracking and the tracking of the handheld screen was performed on the PC.

**Procedure.** After an introduction and filling out a demographic questionnaire, measurements were taken to calibrate the systems of the rendering condition. We measured inter-pupillary distance and the distance between the handheld device and the participant's head to set up the fixed viewpoint for FUPR. To determine the distance participants were asked to hold the handheld device centered onto the touch screen at a distance of 15 cm. The distance was only calibrated once. Afterward, participants familiarized themselves with the first rendering condition by performing

| | DPR | UPR | FUPR | AAUPR |
|---|---|---|---|---|
| Time (s) | 1.5 (0.8) | 2 (1.4) | 1.7 (1.1) | 1.8 (1.1) |
| Error (pxl) | 26.8 (14.4) | 17.3 (11.6) | 20.9 (12.5) | 15.9 (10.4) |
| TLX | 52 (16.3) | 38.6 (14.6) | 44.6 (14) | 30.9 (14.4) |
| SEQ | 3.1 (1.6) | 4.75 (1.3) | 3.6 (1.5) | 5.3 (1.7) |
| Preference | 0 | 2 | 3 | 8 |

**Table 7.2:** Study results. Mean and standard deviation of time and error, and SEQ and TLX results. The last row indicates the number of participants preferring the interface. Three participants did not state a clear preference, except for not choosing DPR.

the task until they felt comfortable using the system. Then, the measured tasks started. Participants were instructed to quickly point to the target area by performing one fluid natural hand motion to the area, where the target was expected. Participants were instructed to not move out of their initial position, but to turn their body to reach the target areas farther away from the center. This should simulate the movement of narrow workspaces, where the head position is not always ideal for FUPR.

For each task, participants first had to locate the target area by searching with the AR view. After locating the target area, participants touched the handheld device screen and then, with the same hand, the target area on the screen. The TCT was measured between the touch of the handheld device and the touch screen. Hence, TCT does not include search time for the target but focuses only on the coordination of the hand as expected to be seen through the AR device. The error was recorded as the Euclidean distance between the detected touchpoint and the center of the target area. Participants received a distinct visual and audio confirmation, for either hitting or missing the target area.

Participants performed 40 repetitions per rendering condition. After a rendering condition, participants filled out the NASA TLX and the SEQ. The next rendering condition started thereafter, following the same procedure. After finishing the last rendering condition and filling out NASA TLX and SEQ, participants filled out a preference questionnaire and a semi-structured interview was conducted. A session took approximately 30 minutes.

With 16 participants, four rendering conditions and 40 repetitions per rendering condition, there were a total of $16 \times 4 \times 40 = 2560$ trials.

**Hypotheses.** Due to the nature of the pointing task, we did not expect significant differences in task completion time. However, due to the spatial distortion of the view of the DPR condition, we expected DPR to have a significantly higher error rate than any other condition (**H1**). Due to the optimal compensation of the spatial distortion, UPR will have fewer errors than FUPR (**H2**). We hypothesize that our novel approach taken with AAUPR produces significantly fewer errors than FUPR (**H3**). We also expected that our AAUPR implementation will be non-inferior to UPR (**H4**).

**Results** The data was evaluated using a level of significance of 0.05. The data fulfilled sphericity and normality requirements and, therefore, was analyzed using ANOVA and post-hoc pairwise t-tests with Bonferroni correction. Questionnaire data was analyzed using a non-parametric Friedman test followed by pairwise Wilcoxon signed-rank tests with Bonferroni correction. The reported p-values have been Bonferroni corrected to reflect a significance level of 0.05. The statistical analysis was performed using the software R.

The ANOVA revealed a significant difference in error for the rendering condition ($F(3,45)=12.26$, $p<0.001$). Post-hoc tests revealed significant differences between DPR and UPR ($p<0.001$), DPR and AAUPR ($p<0.001$) and a weak significant difference between FUPR and AAUPR ($p=0.06$).

Friedman tests revealed significant differences in TLX ($\chi^2(3) = 20.01, p < 0.001$) and SEQ ($\chi^2(3) = 20.59, p < 0.001$). Post-hoc tests revealed significant differences for TLX between DPR and AAUPR ($Z=3.11$, $p<0.005$), FUPR and AAUPR ($Z=2.64$,$p<0.05$) and a near significant difference between DPR and UPR ($Z=-2.53$, $p=0.053$). Post-hoc tests revealed significant differences for SEQ between DPR and UPR ($Z=2.63$, $p<0.05$), DPR and AAUPR ($Z=3.09$, $p<0.005$) and FUPR and AAUPR ($Z=3.04$, $p<0.01$).

**Discussion** As hypothesized, the results of DPR were worst in all tested measurements. DPR had a significantly higher error than UPR and AAUPR due to the optimal spatial distortion of the rendered user-perspective view in these conditions. This is also reflected in the SEQ, which was rated significantly lower compared to UPR and AAUPR. In terms of TLX, DPR also had a significantly higher workload than AAUPR and a weak significant difference to UPR. However, we could not find a significant difference between DPR and FUPR and, therefore, could not replicate the findings of Pucihar et al. [166]. One possible explanation could be the nature of the task, that required interactions over a large distance which eventually requires updating the user's head pose relative to the device. Overall, we partially accept H1. One possible explanation could be the nature of the task of Pucihar, that encouraged participants to spend more time on pointing the target area, while our task required participants to spontaneously point to the target area. Therefore, we partially accept H1.

Interestingly, UPR did not perform significantly better than FUPR in any measurement. Therefore, we reject H2. We believe that the lack of performance comes from the implementation of the head tracking. During the experiments, we noticed jitter in the head tracking, that might influenced the pointing accuracy. This also could explain the better performance of AAUPR, which did not suffer from the problem of continuous jitter, because the tracking rate was lower than the one of UPR. Hence, AAUPR performed significantly better than FUPR in terms of error, TLX and SEQ measurement. Thereby, we accept H3. Note that the head tracking could be implemented more stable using head-mounted fiducials. However, we are aiming at a mobile and self contained system, why we have implemented head tracking based on visual face tracking.

In terms of user preference (Table 7.2) three participants did not clearly prefer any user perspective rendering. However, they were clear in indicating that they did not prefer DPR. For the preference results of the other 13 participants, we performed an exact binomial test comparing to chance (0.25) and found a significant difference in preference for the AAUPR interface ($p<0.05$). This is in line with the results of the study, indicating an advantage of AAUPR over FUPR. The significant preference over UPR also underlines the advantage of AAUPR in terms of more stable, discrete tracking updates.

To be able to record user interactions we performed the experiment on a large touch screen. However, the interaction space is often much larger which requires the user to move the AR display much more around (see Figure7.1 for a real-life example). Since our system is designed to compensate user motion, we believe that interactions in spaces larger than the one used in our experiment will lead to similar results or to a favorable bias towards AAUPR.

## 7.4 Conclusion

In this chapter, we have presented a system for user perspective rendering on handheld devices which support large interaction spaces. Our system does not perform 3D head tracking in each frame, instead, we measure motion over time to trigger updates. This reduces the overall computational demands of the system. During our experiment, we furthermore noticed that fewer updates of the user's head pose result in more stable renderings on mobile phones. We believe this is partly the reason why users prefer our approach over continuous user perspective rendering. This being said, one could also potentially improve the stability of monocular head tracking algorithms by temporal or spatial filtering.

The described system for adaptive user perspective rendering is designed for the current generation of mobile devices. However, faster and more reliable head tracking approaches, in combination with more powerful devices will make our adaptive solution for user perspective rendering less important. Therefore, future work needs to further investigate head tracking on mobile devices.

While our system reduces the impact of tracking failure, erroneous head tracking still impacts the performance of our system to some degree. Therefore, future work needs to further investigate 3D head tracking on mobile devices. In addition, the number of necessary updates of the user's head pose needs to be evaluated separately. Additional information available to the system at runtime, such as the state of the application or the current task to perform or the estimated distance between the current pose and an anticipated future pose of the device could also prove beneficial for tracking stability and computational cost.

# CHAPTER 8

# Interacting with Augmented Reality Documentation using Head Mounted Displays

We developed a system called TrackCap to provide mobile applications with a 6DOF, high fidelity input and output device that allows for spontaneous use in unprepared environments. By sensing the handheld input device relative to the HMD our system has only minimal requirements, which consist of these two devices. Both of these are mobile and, in the case of the smartphone, also widely available. This makes our solution ideal for entry-level consumer-oriented devices such as Google's cardboard or daydream but as we will show has also advantages for top-of-the line HDMs such as the Hololens. With 6DOF tracking and an integrated touchscreen, TrackCap provides multiple input channels, so that established virtual cursor, virtual hand [121] and raycasting techniques can be easily integrated into a single system.

## 8.1 System overview

TrackCap provides relative 6DOF tracking from the smartphone to the "cap" on the HMD using inside-out-tracking. Contrary to existing solutions that use inside-out tracking to track a device within the world [170], for world-scale localization, our system uses the high precision 6DOF tracker of the HMD and does not need to perform any non-relative world positioning by itself. The world-space pose of the smartphone is determined by concatenating the relative pose smartphone-HMD and the HMD-to-world measurement. We use a carefully designed fiducial, which is placed on the HMD. The fiducial's size is optimized to be as small as possible, while still being well observable from the handheld input device. Furthermore, we use a standard 6DOF pose estimation based on point correspondences between the current camera frame and the fiducial [86].

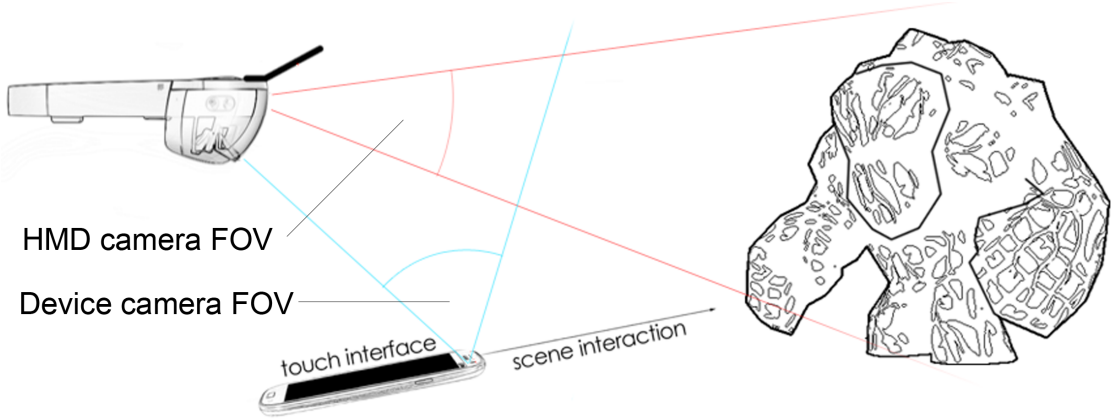



**Figure 8.1:** Interaction space. The HMD's camera is usually forward-facing (indicated in red), limiting the possible tracking space. By using the smartphone's front camera (blue), the device can be operated conveniently at e.g. hip level. The smartphone itself provides an additional high-resolution screen that can be used for non-situated data. 6DOF tracking enables direct scene interaction next to a pick ray metaphor.

Our system is illustrated in Figure 5.1. It consists of a 6DOF pose estimator, which runs on the smartphone, a network module to transmit the estimated pose to the HMD, and a pose combiner which maps the pose of the smartphone into the world coordinate frame.

**Pose estimation.** We implement the pose estimation of the interaction device directly on the smartphone. Our prototype uses planar marker tracking provided by the Vuforia SDK[1] and a fiducial target ("cap") mounted on the HMD. Figure 8.1 illustrates the interaction as seen from the device camera of the HMD. Note that the HMD camera would generally not keep the smartphone in sight.

**Network communication.** Our approach requires transmitting the estimated 6DOF pose as well as user input on the touchscreen from the phone to the HMD. Keeping latency as low as possible is crucial, especially when using an optical see-through HMD. Therefore, we sent the pose in a single UDP packet over WiFi, using a payload of 28 bytes to represent the position and orientation data and 1 byte to transmit the command header. Additional data from button or touch input is sent only on demand. The UDP data payload for such data consists of a 1-byte command header and 0-4 bytes for optional parameters.

**Pose combination.** After receiving the pose data of the smartphone, we concatenate it with the current world pose matrix of the HMD. Since the pose of the smartphone is calculated relative to the camera center, we add an offset to the center of the physical device. Additionally, we take the offset between the origin of the HMD tracking system and the center of the fiducial marker into account. Both are static transformations and need to be defined only once.

[1] www.vuforia.com

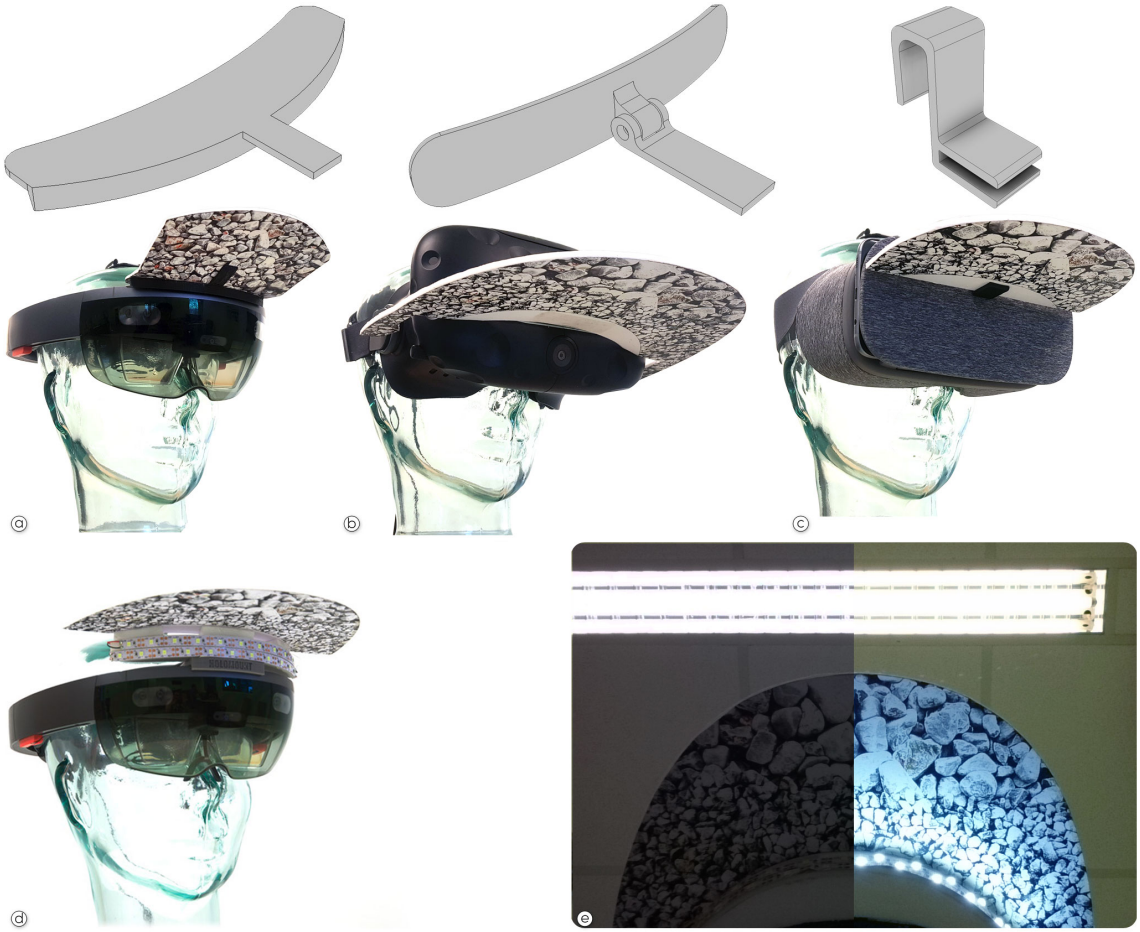


**Figure 8.2:** Designs. We designed a set of 3D printable parts to mount TrackCap to a variety of HMDs (STL files will be made publicly available). (a) Microsoft HoloLens. Note that the cap was designed not to obstruct the view of the scene understanding sensors, so the marker was bent upwards. (b) HTC Vive and (c) Google Daydream designs use a flat marker instead. Note that we use the HTC Vive only for measuring precision of TrackCap, since the HTC Vive already comes with a precise hand tracking system. (d) We provide additional illumination for the marker to compensate for strong back-lighting from the ceiling. (e) The additional light source illuminates the marker (right).

### 8.1.1 Technical analysis

Our system benefits from see-through displays, like the Microsoft HoloLens, since one can see one's hands using the device. However, immersive VR works fine, even on entry-level systems such as Google Cardboard. Figure 8.2(c) shows a Google Daydream headset using ASUS ZenfoneAR smartphone. Since our system runs the computationally expensive tracking of the input device on the smartphone itself, performance on the HMD only depends on the application. For our lightweight test scenes, we were able to achieve high frame rates on all test devices – 90 frames per second (fps) on the HoloLens and 60 fps on the Daydream. Tracking performance on the smartphone was 30 fps, limited only by the camera frame rate.

In addition to framerate, we compared the precision and latency of our tracking solution to the outside-in tracking provided by the HTC Lighthouse system. Therefore, we added a setup using TrackCap with the HTC Vive (Figure 8.2(c)). To compare the Lighthouse tracking performance to TrackCap, we mounted an additional Lighthouse tracker on the smartphone. This rig was calibrated so that the HTC tracker's virtual center point coincided with the center of the smartphone.

To sample the interaction space around the user, we placed a 3D grid of $4 \times 4 \times 3$ reference points within the world coordinate system. The position and orientation data of both tracking systems were collected for a duration of three seconds at each reference point. During the procedure, we also recorded the network latency. The results are shown in Table 8.1. TrackCap delivered a positional error below 10mm, an orientation error of less than 2°, and an average latency of 10 ms.

In theory, the smartphone's front camera FOV could be a limiting factor, since it determines the possible tracking range. In practice, a typical horizontal FOV of a recent phone range from 56.3°(HTC One M8) to 71.4°(ZenPhone AR), and newer hardware tends to have an even larger FOV. In our studies, we did not notice any differences across our test devices in terms of coverage. The tracking range of our solution, as tested using a Samsung Galaxy S8 phone (68.0°FOV), covers a $1 \times 1 \times 0.6m$ volume in front of the HMD. Since the phone will necessarily be held no further than arm's length, the tracking volume proved to be sufficient.

During our tests, we noticed that pointing the smartphone camera point upwards introduces occasional tracking failures caused by strong backlighting, e.g., from overhead lights or the sun, as the dynamic range of the camera is limited and automatic exposure correction makes the marker appear very dim. Figure 8.2(e-left) illustrates the problem. To mitigate the effect of strong backlighting, we installed a USB-powered LED array to illuminate the maker, as shown in Figure 8.2(d). The effect on the camera image can be seen in Figure 8.2(e-right).

| | Position Error | Orientation Error | Latency |
|---|---|---|---|
| Mean | 9.812 mm | 1.849° | 10 ms |
| Std. Dev. | 3.903 mm | 0.291° | 14 ms |

**Table 8.1:** Tracking precision and latency of the TrackCap system

## 8.2 Evaluation

We performed a series of evaluations on the performance of TrackCap versus other mobile, untethered input options. The evaluations focus on object selection and manipulation tasks as fundamental elements of 3D interactions. TrackCap is compared to standard methods for 3D interaction, as available for commercially available untethered systems, such as the Google Daydream.

Since we are interested in how well TrackCap can support natural interactions, we disabled any supporting visualizations, such as a crosshair or a thin ray, during our experiments. Note that future applications of TrackCap would most likely make use of such supportive visualizations. However, supportive visualizations may require additional calibration effort and may be a confounding factor in experiments. In the interest of brevity, we also decided to steer clear from comparing TrackCap to gaze-based interaction techniques, since previous research indicates that gaze-based techniques are slower than hand-based methods and restrict the user's ability to recall the environment [157].

For all evaluations, the data was evaluated using a significance level of 0.05. The analysis was performed using the statistics software *R*.

### 8.2.1 Experiment 1: Selecting distant objects

We tested the capability of TrackCap for selecting distant objects. To this end, we compared the interaction supported by TrackCap to a hand-held IMU [58] on a picking-by-raycasting task. We expected TrackCap to perform better than IMU due to the inherent drift of the latter.

In a pilot study, we compared both approaches using a standard Fitts's law test environment of shapes arranged in a circle, as defined in ISO-9241 [1]. The shapes were arranged in front of the user, requiring to use only a small part of the interaction space. Consequently, users performed only small motions and held the input device close to their body (Figure 8.3(a)). Thus, the ability of TrackCap to extend the interaction space for larger arm motions was not fully taken advantage of. Based on the pilot study, we modified our task to maximize the usage of motor and interaction space. We changed the task to force the user to move in 3D, which we achieved by distributing the targets in the 3D space around the user (see Figure 8.3(d)).

**Task.** Inspired by the Fitts's law test from the pilot study, we designed another pointing task. A set of 21 blue spheres was arranged at a distance of $2m$ around the participant. The task started by pointing the mobile device at the first highlighted sphere and performing a click. The next highlighted sphere was always be located on the opposite side and required the user to turn. A hit was confirmed visually.

We varied the task difficulty by presenting spheres of different sizes and at different heights around the participants. We presented sizes of 5, 10 and 15 cm. The height of the center of circles was set to the height of the users' head with a random offset in the range $\pm 0.175$ m added.

To measure the effectiveness of TrackCap for intuitive 3D pointing at distant objects and its ability for drift compensation, we did not present the ray visually during the interaction. However, spheres hit by the invisible ray were highlighted by changing the color to gray to provide visual feedback. To compensate for the small FOV of the used HMD, the application guided the participant to the targets by showing green arrows at the border of the view area pointing into the direction of the target.

**Design.** We designed a repeated-measures within-subject study to compare the performance and user experience of drift-compensated TrackCap and interaction using only the device's IMU. We defined an independent variable "system" with two conditions: TrackCap and mobile device only (MBO). In MBO, the orientation of the ray relied only on the internal 3DOF sensor. As

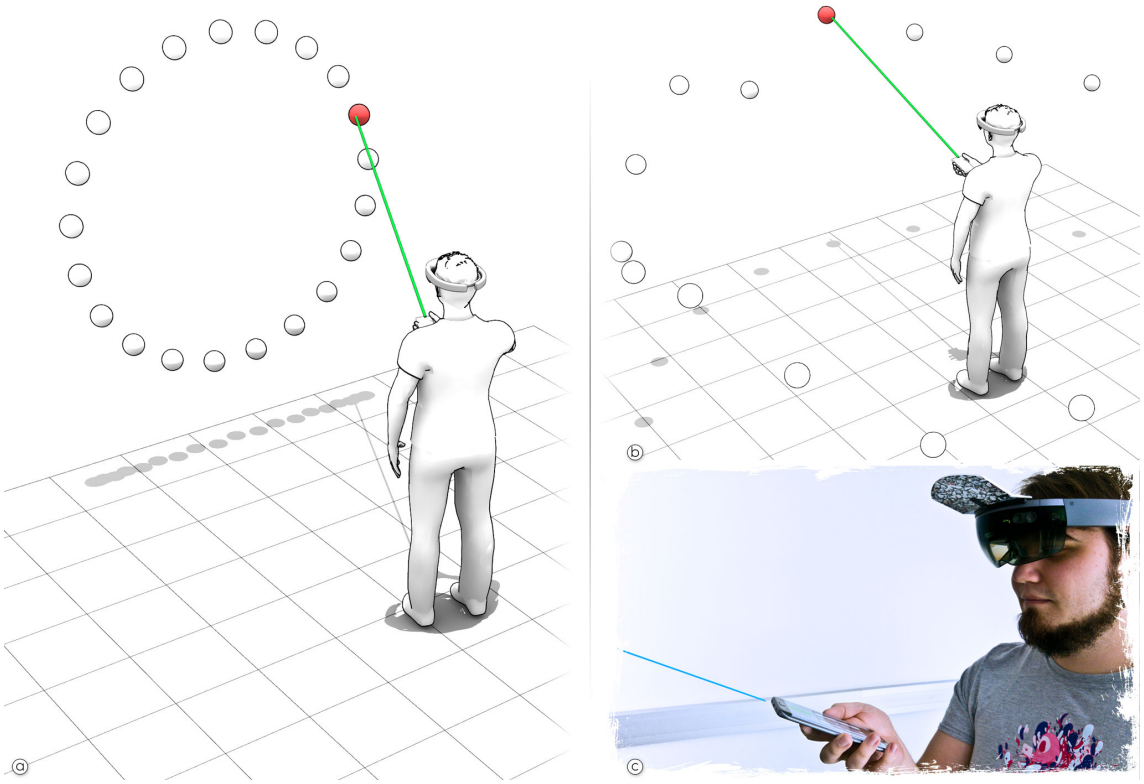


**Figure 8.3:** Study setup to measure performance during the selection of distant objects. (a) Fitts's law test on a virtual plane in front of the user. (b) Variation of the Fitts's law test in 3D. The targets are located in a circle around the user, with varying heights. (c) User with HMD during the task. The user sees a virtual picking ray and a virtual target sphere. The virtual picking ray is shown for demonstration.

dependent variables, we measured task completion time (TCT) determined as the time of clicks between successive sphere targets, and error rate of the task, as the percentage of spheres missed. In addition, we measured subjective workload with the raw NASA TLX [52], usability with the Single Ease Question (SEQ) [139], and overall preference.

Eight participants (1 female, $\overline{X}$ =30.3 (sd=4.2) years old) volunteered for the study. On a scale from one to five, five meaning best, the mean of self-rated AR experience was 3.3 (sd=1.2).

**Apparatus.** The apparatus consisted of an optical see-through HMD (Microsoft HoloLens) and a mobile phone with a touchscreen (Samsung Galaxy S7). The spheres were visible in the HMD view only. The mobile device was used to control the orientation of the ray. Input was confirmed by touch on the mobile phone.

6DOF head tracking was achieved via the HoloLens. In the TrackCap condition, the smartphone was registered in the same coordinate system as the HoloLens, and drift was compensated using TrackCap. In the MBO condition, the orientation of the ray depended only on the hardware sensors of the mobile phone.

**Procedure.** After filling out a consent form and demographics questionnaire, users were introduced to the first condition. The starting order of systems was counterbalanced using a Latin Square setup. The systems were tested by using multiple trial blocks, each block consisting of 21 trials. Each system was used for all three sphere sizes. Sizes were varied between trial blocks and presented in random order. The height of the spheres was randomized within a trial block. Participants were standing throughout the task and used their dominant hand for pointing.

After the participants familiarized themselves with the system by performing two test blocks, the task was performed by repeating one block for the system for each size condition. The participants were instructed to be fast and accurate. Between blocks, participants were forced to rest for 10-20 seconds to recover from fatigue due to the mid-air interaction [181]. Upon completion of the condition, users filled in the SEQ and NASA TLX questionnaire. The procedure was repeated for the remaining system. After completing the task with the last system, the user filled out the preference questionnaire. In total, we collected 2 systems $\times$ 3 sizes $\times$ 21 trials = 126 samples per task and participant, and, 126 $\times$ 8 participants = 1008 samples over all participants.

**Hypotheses.** We expected that TrackCap would successfully enable mobile interaction and compensate for IMU drift. Therefore, TrackCap will perform better than MBO with respect to TCT (**H1**) and error rate (**H2**) in the distant pointing task.

**Results.** The data did not fulfill the normality requirements and, therefore, was analyzed using Wilcoxon signed-rank tests. Wilcoxon signed-rank tests revealed significant differences between TrackCap and MBO for error rate ($Z=2.20$, $p<0.05$), TLX ($Z=-2.52$, $p<0.01$) and SEQ ($Z=2.58$, $p<0.01$) (see Table 8.2 mean and standard deviation values).

**Discussion.** We investigated the ability of TrackCap to provide intuitive and precise 3D pointing interactions, when compared to a standard technique using only an IMU. Our results provide evidence that TrackCap enables more precise interactions with lower perceived task load. Therefore, we accept H2. We believe that the significantly higher error for MBO is caused by a large amount of drift introduced during interaction using the orientation sensors on the smartphone. TrackCap is able to reliably and automatically compensate for this drift and make pointing natural and intuitive.

The data did not reveal any significant difference in TCT. Therefore, we reject H1. To ensure natural and intuitive pointing, we asked the user's to not only be accurate but also fast. We believe this instruction may have influenced their behavior, as they did not take much time to manually compensate for the drift in MBO. This behavior was intended since we are aiming at a system for intuitive pointing in 3D. Participants stated that the drift and the many failures in MBO were frustrating, which influenced the time users spent on trying to hit the targets as the task progressed. These observations are reflected in the significantly higher task load and lower perceived ease of use of MBO. In addition, when asked for preferences, all participants (100%) preferred TrackCap over MBO.

### 8.2.2 Experiment 2: Selecting close proximity objects

The support of TrackCap for 6DOF input allows implementing direct 3D object selection techniques [95]. We show the capability of our approach by comparing direct 6DOF selection (implemented via TrackCap) to 3DOF raycasting that would otherwise be used in such a scenario. We use the approach of Hincapié-Ramos et al. [58] for raycasting.

**Task.** While the first experiment investigated a distant pointing task, this experiment investigated direct selection of targets within a user's reach. The task uses the same setup as in the first experiment, only with the spheres placed closer to the user. Participants again had to alternate selection between opposing spheres in their surroundings. However, in this experiment, the distance of spheres was set to be within arm's length so that participants could reach them comfortably (see Figure 8.4(a)).

**Design.** We designed a repeated-measures within-subject study to measure the performance and user experience of drift-compensated 6DOF TrackCap in close proximity object selection tasks. The 6DOF tracking of TrackCap was used to let the users interact with the smartphone as an extension of their own hand.

Again, we defined an independent variable "system" with two conditions: TrackCap and mobile device only (MBO). As dependent variables, we measured task completion time (TCT) and error rate of the task, i.e., the percentage of spheres missed. In addition, we measured subjective workload measured by the raw NASA TLX [52], usability using the Single Ease Question (SEQ) [139], and overall preference.

Eight participants (all male, $\overline{X}$ =31.1 (sd=3.4) years old) volunteered for the study. On a scale from one to five, five meaning best, the mean of self-rated AR experience was 3.5 (sd=1.2).

**Apparatus.** The same apparatus was used as in the first experiment.

**Procedure.** After filling out a consent form and demographics questionnaire, the height and distance of the circles were set up for each participant so that it could be reached comfortably. The rest of the procedure was the same as in the first experiment.

**Hypotheses.** We expected that direct selection using TrackCap will be more successful than MBO, due to the lack of sensor drift in TrackCap and its ability to track the position of the input device in 3D space. Therefore, TrackCap will perform better than MBO with respect to TCT (**H3**) and error rate (**H4**) in the direct selection task.

**Results.** The data did not fulfill the normality requirements and, therefore, was analyzed using Wilcoxon signed-rank tests. Wilcoxon signed-rank tests revealed significant differences between TrackCap and MBO for error rate ($Z=2.203$, $p<0.05$), TLX ($Z=-2.52$, $p<0.01$) and SEQ ($Z=2.584$, $p<0.01$) (see Table 8.2 for mean and standard deviation values).

**Discussion.** While we did not find significant performance differences regarding TCT, the error rate was again significantly lower for TrackCap than for MBO. Therefore, we accept H4, but reject H3. The lack of difference in TCT may again be explained with the instructions we gave to participants. By asking participants to be precise and fast, we were aiming to measure the performance of both interfaces during intuitive and natural pointing.

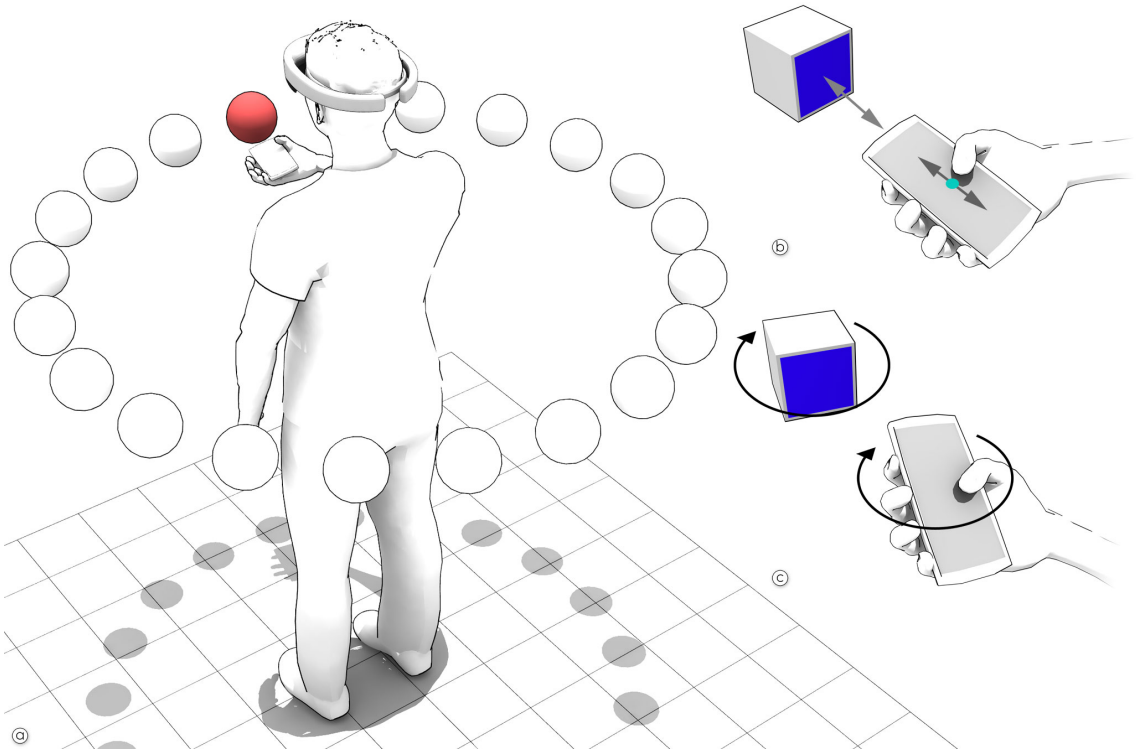



**Figure 8.4:** Direct selection. Using the 6DOF pose generated by TrackCap, the user can select virtual objects by simply touching them with the physical device.

Similar to the first experiment, the users mostly commented on the high frustration with the MBO interface. This is also reflected in the significantly higher task load and lower perceived ease of use of MBO. Both, lower task load and higher perceived ease of use, can also be explained with the instructions we gave to intuitively select objects, which seems better supported with TrackCap.

When asked for preferences, seven participants (87.5%) preferred TrackCap over MBO. For one user, MBO worked very well, since he was able to estimate the current and future drift of the 3DOF device. Since this user was a self-rated expert on mobile devices, we believe his ability to cognitively compensate for drift is the result of his long experience with mobile sensors.

In general, the error rate was lower over all conditions than in the first experiment. This can be explained by the shorter distance between the user and the spheres, which have the same 3D size as in the first experiment. Hence, spheres appeared to be larger in the view of the participants, thereby improving the general accuracy. The larger spheres and, thus, the larger interaction area also seem to have implicitly compensated part of the drift that occurred in MBO.

### 8.2.3 Experiment 3: Object manipulation

While the second experiment investigated the direct *selection* of objects in the vicinity of the user, the third experiment investigates direct *manipulation* of such objects. We show the practical value of our approach by comparing direct 6DOF manipulation via TrackCap to an established 3DOF manipulation technique utilizing raycasting for selection and hand-centered manipulation in combination a fishing-reel technique, as proposed by Bowman et al. [21].

**Task.** The task uses a similar setup as in the second experiment. However, instead of spheres, this task uses cubes with colored sides. Participants had to alternate selection between opposing cubes in their surroundings. However, in this experiment, participants had to drag and drop the selected cube to the opposing side and align it with a corresponding platform in the target location. The orientation of the cube was defined uniquely by the differently colored sides (Figure 8.4(c)).

For this experiment, the rotation of the target was limited, so that the smartphone camera was able to observe the cap during the entire task. In a follow-up experiment (reported below), we extended the task to include full 360°rotational changes.

**Design.** We designed a repeated-measures within-subject study to compare the performance and user experience of drift-compensated 6DOF TrackCap for direct manipulation and common 3DOF mobile device interaction. Holding a button on the touch screen allowed participants to grab objects; releasing the button also released the object. We compared TrackCap to a raycasting manipulation with a fishing reel: After selecting objects by raycasting, objects stick to the ray. Thus, the 3DOF of the interaction device manipulated the orientation of the object. The distance of the object along the ray could be manipulated using a sliding motion on the touch screen of the mobile phone.

As before, we defined an independent variable "system" with two conditions: TrackCap and mobile device only (MBO). As dependent variables, we measured task completion time (TCT) and error, i.e., the precision of the alignment. In addition, we measured subjective workload measured by the raw NASA TLX [52], usability using the Single Ease Question (SEQ) [139], and overall preference.

The participants of the second experiment took part in this experiment.

**Apparatus.** The same apparatus was used as in the first experiment.

**Procedure.** After filling out a consent form and demographics questionnaire, the height and distance of the cubes were set up for each participant, so that it could be reached comfortably. The orientation required to align the cubes was randomized. The rest of the procedure was the same as the one in the first experiment.

**Hypotheses.** We expected that direct manipulation using TrackCap will be more successful than MBO due to the more natural interaction. Therefore, TrackCap will perform better than MBO with respect to TCT (**H5**) and error rate (**H6**) in the direct manipulation task.

**Results.** The data did not fulfill the normality requirements and, therefore, was analyzed using Wilcoxon signed-rank tests. Wilcoxon signed-rank tests revealed significant differences between TrackCap and MBO for TCT ($Z=2.2$, $p<0.05$) and TLX ($Z=-2.52$, $p<0.01$), but not for alignment error or SEQ (see Table 8.2 mean and standard deviation values).

| | Exp. 1 | | Exp. 2 | | Exp. 3 | |
|---|---|---|---|---|---|---|
| | TrackCap | MBO | TrackCap | MBO | TrackCap | MBO |
| TCT (ms) | 9.3 (3.4) | 10.1 (3.5) | 4.9 (1.3) | 6.8 (2.9) | 13.9 (4.5) | 20.1 (7.1) |
| Error (%) | 35.1 (24.4) | 75.6 (16.2) | 10.1 (8.6) | 27.4 (24.1) | 0.05 (0.01) | 0.06 (0.01) |
| TLX | 45.3 (20.9) | 79.7 (12.5) | 27.1 (13.7) | 55.1 (20.4) | 33 (9.2) | 63.8 (16.9) |
| SEQ | 4.8 (1.3) | 1.5 (0.5) | 6.4 (0.7) | 4.0 (1.4) | 5.5 (1.6) | 3.9 (1.6) |
| Preference | 8 | 0 | 7 | 1 | 8 | 0 |

**Table 8.2:** Results of Experiments 1-3. Mean and standard deviation of time and error, SEQ results, and TLX results. Last row indicates the number of participants preferring the interface.

**Discussion.** Participants could interact significantly faster when using direct manipulation supported by TrackCap than when using the fishing reel metaphor of MBO. Therefore, we accept H5. However, we did not find a significant difference in error rate, why we reject H6. There was also no significant difference in perceived ease between TrackCap and MBO. However, the significantly lower TLX for TrackCap indicates that directly interacting with virtual objects using TrackCap is less demanding. Consequently, all participants (100%) preferred TrackCap over MBO.

In contrast to the previous experiments, there was no significant difference in error rate between TrackCap and MBO. We believe that participants could efficiently align the virtual cubes despite the drift of MBO due to the visual feedback provided by the cube itself. Hence, participants could compensate for the drift using the virtual cube as a visual reference.

Apart from the more natural manipulation using TrackCap, a part of the difference in TCT could also be explained by the need to successfully select the virtual object before being able to continue the alignment task. During the selection phase of the task, participants had to select the virtual cube using the invisible raycasting. Participants could not skip this selection step as easily as in the previous experiment but had to take their time to align the drifting raycasting with the virtual object before continuing. Only after the virtual cube was stuck to the invisible ray, participants could visually compensate for the drift. The focus of these experiments was to evaluate the ability of TrackCap to automatically compensate for drift. Future work will additionally investigate the ability of users to be able to compensate for drift by providing visual cues. Visual aids have been shown to have an impact on pointing tasks such as these ones [159].

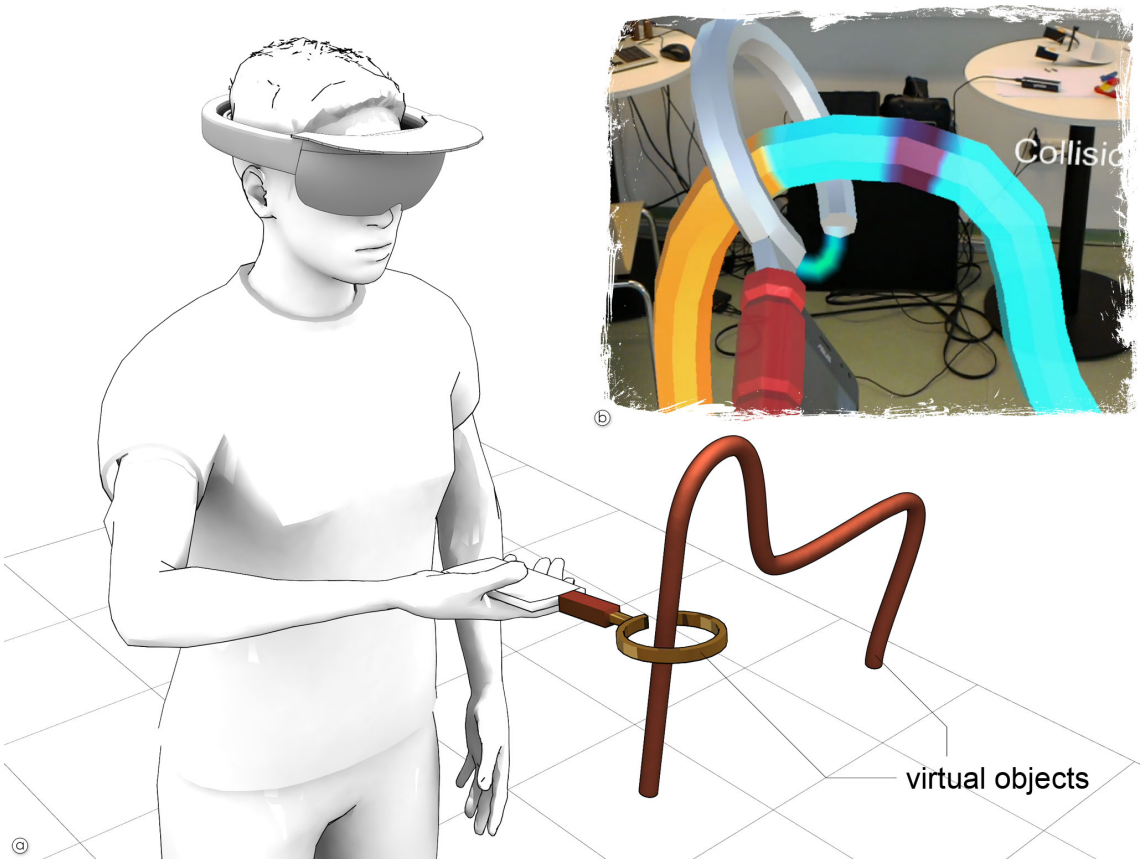



**Figure 8.5:** Complex object manipulation - AR wire game. (a) Illustration of the AR game used to measure the performance of TrackCap in complex object manipulation. (b) Screenshot through the HoloLens as seen by a user. Upper row: training tasks. Lower row: Tasks used during the experiment.

### 8.2.4 Experiment 4: 6DOF interactions using camera switching

Our previous experiment revealed that TrackCap is able to outperform manipulation which consists of 3D positional changes and moderate rotational changes. Since we designed TrackCap to support full 6DOF interactions for mobile devices, we are interested in its performance in tasks that use full 360°rotations, as well as continuous translational movements in 3D space. In order to support such interactions, we have extended our approach using the smartphone's IMU, allowing us to switch between the front and back camera of the mobile device depending on its current orientation.

**Task.** To test the capability of our approach, we have set up a "don't touch the wire" game, which requires full 6DOF interactions (Figure 8.5) involving a large number of rotations paired with continuous translational motion. The user must move a virtual wire loop along a winding pipe in 3D without colliding. Whenever the user hits the pipe, an acoustic signal is played and a particle spray marks the collision location. This game forces the user to move slowly and carefully in 3D.

**Design.** We designed a repeated-measures within-subject study to compare the performance and user experience of our dual-camera TrackCap with a system using the Google Tango API to read additional mobile sensors for 6DOF API. This tracking system was implemented on the ASUS ZenfoneAR smartphone, which is packed with additional hardware, including a fisheye camera, a time-of-flight camera and a custom DSP processor. The hardware of the ZenfoneAR can be compared to the HoloLens and does not compare to current-generation smartphones. However, we were interested in how TrackCap could compare to a powerful alternative.

We define an independent variable "system" with two conditions: the modified TrackCap (CamSwitch) and Project Tango (Tango). As dependent variables we measured task completion time (TCT) and error rate of the task, i.e., the number of hits between the wire loop and the pipe. In addition, we measured subjective workload by the raw NASA TLX [52], usability using the Single Ease Question (SEQ) [139], and overall preference.

Eight participants (1 female, $\overline{X}$ =32.5 (sd=3.9) years old) volunteered for the study. On a scale from one to five, five meaning best, the mean of self-rated AR experience was 3.9 (sd=1.1).

**Apparatus.** The apparatus consisted of an optical see-through HMD (Microsoft HoloLens) and a mobile phone with Project Tango support (Asus ZenfoneAR). We use the same phone in both conditions. The wire and the pipe were visible in the HMD view only. The mobile device was used to control the wire.

**Procedure.** After filling out a consent form and demographics questionnaire, the height of the pipe was roughly set to the height of the participant. Then, users were introduced to the first condition. The starting order of systems was counterbalanced using a Latin Square setup. Participants were standing throughout the task and used their dominant hand for holding the virtual wire loop.

After the participants familiarized themselves with the interaction method by performing two test games, the task was performed. The participants were instructed to be fast and accurate. Upon completion of the condition, users filled in the SEQ and NASA TLX questionnaire. The procedure was repeated for the remaining system. After completing the task with the last system, the user filled out the preference questionnaire.

**Hypotheses.** We expected that CamSwitch will successfully support the entire task. Due to stable 6DOF tracking of the device, we expected to see equivalent results in TCT (**H7**) and Error (**H8**), when compared to Tango.

**Results.** To verify H7 and H8, we planned to perform equivalence tests, which require a larger sample size. However, we aborted the experiment after only eight participants, because the feedback and our observations indicated that CamSwitch suffered from the time-consuming switching between cameras that influenced the interaction performance. Due to the small sample size, we did not perform equivalence tests, but checked for significant differences to explore the differences between CamSwitch and Tango.

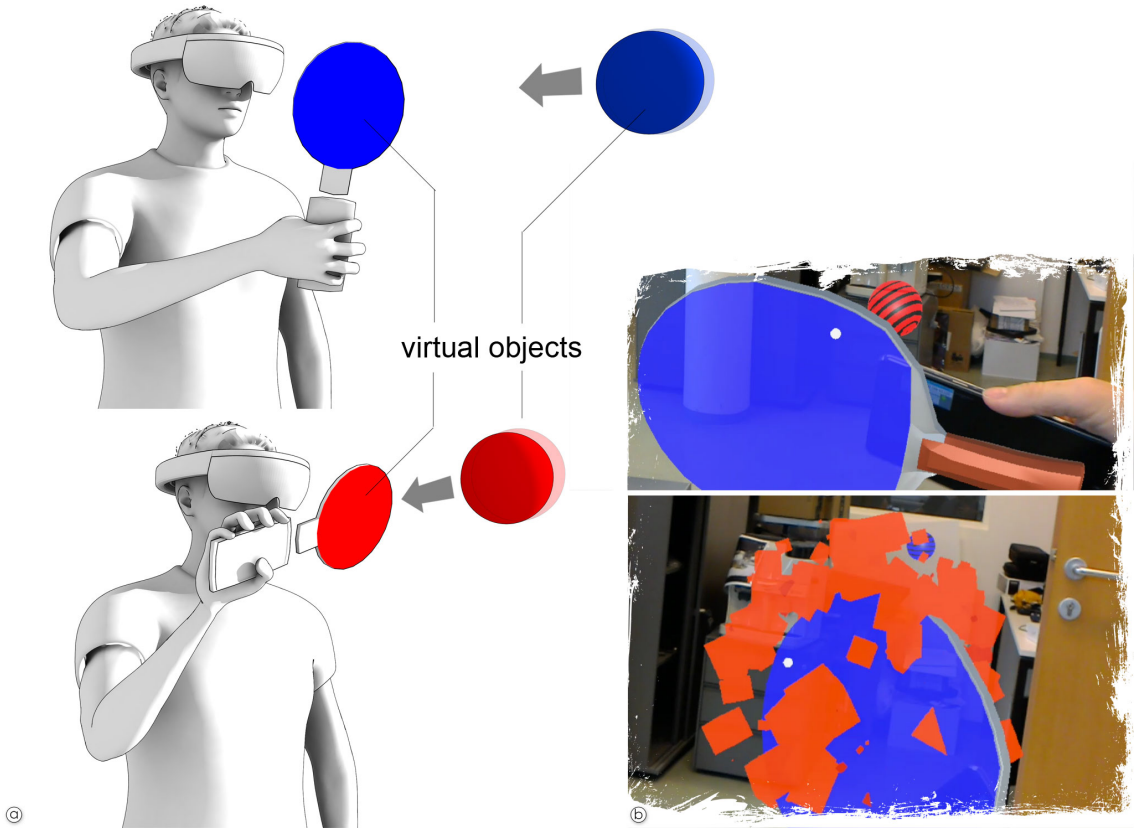



**Figure 8.6:** AR Squash. We implemented a squash game for evaluating the performance of TrackCap in the complementary operation with a model-free tracking solution. (a) Illustration of the interaction. Blue and red balls are thrown towards the user, who has to hit them with the matching side of the virtual paddle (indicated by blue and red colors). (b) Screenshots captured through the HoloLens while playing the game. Balls explode when they are hit.

The data did not fulfill the normality requirements and, therefore, was analyzed using Wilcoxon signed-rank tests. Wilcoxon signed-rank tests revealed no significant differences between CamSwitch and Tango for TCT (CamSwitch 28.4, sd=14.9; Tango 16.3, sd=4.4), error rate (CamSwitch 1.5, sd=2.1; Tango 1.5, sd=1.7 and TLX (CamSwitch 34.5, sd=17.9; Tango 29.1, sd=16.6). However, the test revealed a significant difference for SEQ (CamSwitch 2.1, sd=0.8; Tango 6.3, sd=0.7; $Z=2.56, p<0.01$). The TCT of CamSwitch was close to being significantly worse than Tango ($Z=0.14$, $p=0.055$).

**Discussion.** We could not establish the equivalence of CamSwitch and Tango and, therefore, reject H7 and H8. After only eight participants, we determined, based on user feedback and observations, that CamSwitch suffered from technical issues. Common smartphones require too much time to switch between back and front cameras. This makes an uninterrupted motion in CamSwitch unfeasible. Participants were forced to wait for the completion of the camera switch, which led to frustration during interaction. This is reflected by the 100% preference and the higher perceived ease of use of the Tango device. The waiting time is also reflected in the higher TCT of

CamSwitch, when compared to Tango. However, the small error rate of CamSwitch indicates that TrackCap is suitable for precise 6DOF motion in 3D space. Therefore, TrackCap would likely benefit from better support for dual-camera solutions on smartphones, which can quickly search for the "cap" in both camera streams simultaneously.

### 8.2.5 Experiment 5: 6DOF interaction by complementary operation of TrackCap and model-free tracking

As demonstrated in the previous experiment, a self-contained model-free 6DOF tracking system such as Tango makes a smartphone even more valuable as an input device companion to an HMD. Even though Tango hardware will likely not become available to a mass audience, self-contained 6DOF tracking with somewhat lower performance is becoming available as part of Google's AR-Core or Apple's ARKit. Even though these solutions rely on an opportunistic mapping of the environment and can easily become confused under fast motion, they can support an enhanced version of interaction in the style of TrackCap style.

We were interested in a longer-term technical trajectory, where technologies such as ARCore are widely available, and users would like to use them for fast motion, rather than the slow interaction of our previous experiments. Therefore, we designed a final experiment to assess the benefit of TrackCap to a self-contained model-free 6DOF tracking as well.

**Task.** The task was inspired by tennis ball serving machines. Virtual balls with a diameter of 10 cm were thrown at a constant speed of 1 meter per second at the participant, who had to hit the ball using the mobile device as a tennis racket to make the ball disappear. Once per second, red or blue balls originated from the same location, moving in a random direction within a cone of 30 degrees. Red balls had to be hit with the front of the racket, blue balls with the back (see Figure 8.6).

**Design.** We designed a repeated-measures within-subject study to compare the performance and user experience of the combination of TrackCap and Project Tango, and a system using the 6DOF tracking of Project Tango only. Therefore, we define an independent variable "system" with two conditions: Project Tango and TrackCap (TangoCap) and Tango Only (Tango).

As dependent variables, we measured task completion time (TCT) and success rate of the task, i.e., the number of spheres hit. In addition, we measured subjective workload measured by the raw NASA TLX [52], usability using the Single Ease Question (SEQ) [139] and overall preference.

Eight participants (1 female, $\overline{X}$ =30.5 (sd=3.3) years old) volunteered. On a scale from one to five, five meaning best, the mean of self-rated AR experience was 3.5 (sd=1.4).

**Apparatus.** The apparatus consisted of an optical see-through HMD (Microsoft HoloLens) and a mobile phone with Project Tango support (ASUS Zenfone AR). The spheres were visible in the HMD view only. The mobile device was used to control the tennis racket.

**Procedure.** After filling out a consent form and demographics questionnaire, the height of the ball's origin was set to the height of the participants. Then users were introduced to the first condition. The starting order of systems was counterbalanced using a Latin Square. Participants performed runs of 100 task repetitions, i.e., they had to hit 100 balls in a row. Participants were standing throughout the task and used their dominant hand for holding the virtual racket.

After the participants familiarized themselves with the interaction method by performing two test runs, the task was performed by repeating one run for the system for each size condition. The participants were instructed to be fast and accurate. Upon completion of the condition, users filled in the SEQ and NASA TLX questionnaire. The procedure was repeated for the remaining system. After completing the task with the last system, the user filled out the preference questionnaire.

**Hypotheses.** We expected that TrackCap could successfully support the relocalization of Tango, if tracking was lost. Due to the speed and appearance of balls at fixed time intervals, we did not expect to see differences in TCT (**H8**). However, we expected that TrackCap would supplement the capabilities of Tango and lead to better error rates in this task than when only using Tango tracking only (**H9**).

**Results.** The data was evaluated using a level of significance of 0.05. The data did not fulfill the normality requirements and, therefore, was analyzed using Wilcoxon signed-rank tests. Wilcoxon signed-rank tests revealed significant differences between TangoCap and Tango for success rate (TangoCap 81.4, sd=12.5; Tango 65.4, sd=20.9; $Z=2.52$, $p<0.01$) TLX (TangoCap 31.6, sd=14.5; Tango 45, sd=16.4; $Z=2.52$, $p<0.01$) and SEQ (TangoCap 5.4, sd=1.1; Tango 3.1, sd=0.8; $Z=2.4$, $p<0.05$).

**Discussion.** As expected, relocalization failed after the device lost tracking due to fast motion. This required the user to scan the room for a position known to the Tango device, thereby slowing down the user interaction. However, TrackCap ensured fast and accurate relocalization after tracking failure due to fast motion. This is reflected in the significantly higher rate of balls that users hit successfully. Note that measuring timing difference was not possible due to the balls spawning at constant time intervals. Due to the nature of TrackCap, this method works reliably also in unknown environments. Participants preferred TangoCap (87.5%) and found it more intuitive and easier to use, which is reflected in the lower workload and higher perceived ease of use. Overall TrackCap could successfully expand the usability of the existing 6DOF Tango tracking.

## 8.3 Discussion and conclusion

We have presented TrackCap, a novel system aimed for mobile VR and AR that allows for spontaneous, precise, and natural interactions with 6DOF using consumer-grade smartphones. We have presented evaluation results that indicate that TrackCap improves over current HMD input devices in standard 3D selection and manipulation tasks.

**Evaluation summary.** Our results show that TrackCap allows for precise interaction at a distance and in close proximity. In these scenarios, TrackCap outperforms traditional techniques that rely only on the IMU of the smartphone or input device.

Our fourth experiment showed a limitation of our system. When designing a task that required rotating the phone's camera out of view of the "cap", the 6DOF tracking was lost. To compensate for this issue, we expanded the capability of TrackCap to switch between front and back camera of the smartphone, depending on its orientation to the user. However, we found that most current smartphones cannot switch between front and back camera sufficiently fast. Despite the systematic interruption, TrackCap also allowed for precise 6DOF interaction in this situation. It is also worth mentioning that only a few phones model (e.g., HTC M8) support simultaneous usage of several cameras and do not require a camera switch. However, this is not a fundamental technical limitation; adding simultaneous support for both cameras should be cheaper than adding addition sensors or other hardware components, as in Project Tango.

During fast motion, where the tracking of Project Tango was thrown off, TrackCap successfully supported the Tango's relocalization and enabled more fluid interaction than when using Tango alone. This demonstrates that TrackCap is not only able to make use of older smartphones for interaction, but also extended the usability of current solutions, such as those based on ARCore and ARKit.

**Limitations.** There are several limitations that are worth mentioning. First, we intentionally decided to not include supportive visualizations in the user evaluation. Our system, like most practical applications, supports visual aids such as cross-hairs or virtual laser pointers that might affect the performance in the studies. However, we focused on showing the ability of our system to provide reliable and intuitive interaction using only natural hand-eye coordination and proprioceptive cues.

Similarly, we did not focus on utilizing the screen for complex interaction apart from confirming selection. However, as outlined later there are opportunities there to further improve the results but we intentionally focused on the spatial interaction with the controller (smartphone) for the studies.

Another point for discussion is the limited number of participants for our studies. This limitation is owed to the explorative nature of our experiments. We studied several aspects of our system using five different experiments that took two hours per participant and included a large number of trials and measured samples.

**Implications** We argue that the work has relevance beyond the scope of this thesis. Foremost, we show the lack of input devices that specifically aim for spontaneous and natural interaction with a mobile HMD. This is particularly important for commercial AR/VR solutions that leave the boundaries of scientific environments and find applications in classrooms, workplaces, but also for personal entertainment and recreation.

Our work shows that inside-out tracking can be a feasible option for hand controllers, but has been largely ignored. We believe this finding is significant, as smartphones are ubiquitous and the alternatives require additional hardware or put additional constraints on the user or the environment.

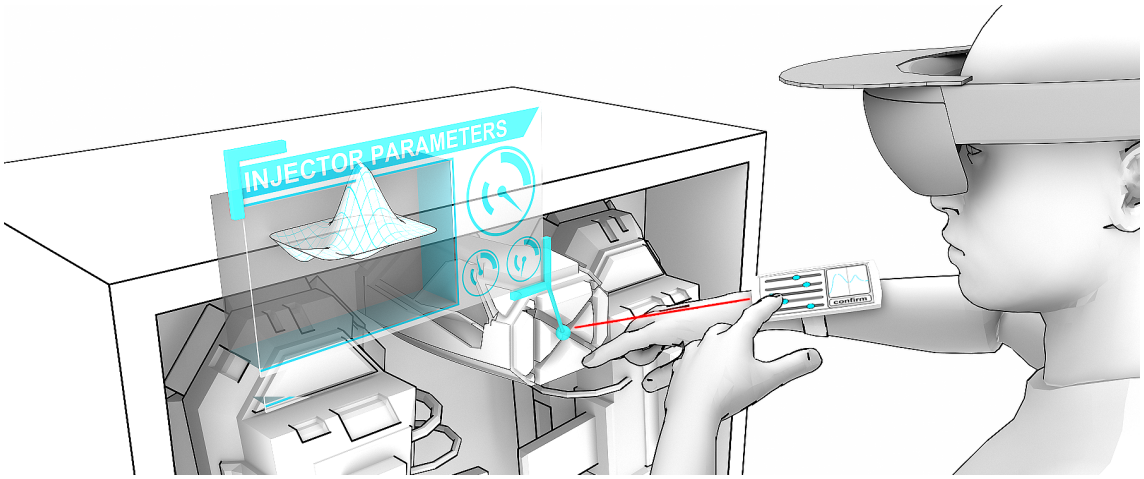



**Figure 8.7:** Application scope. TrackCap can not only be used to select objects, as it also provides a device with high input and output fidelity, it can also be used to display detail of the selected objects and as an interface for manipulating part of that details.

We also argue that adopting phones as controllers is not a compromise, but an opportunity to exploit capabilities so far not offered by most available controllers. While we show the performance of TrackCap in a variety of experiments, we did not yet utilize the full potential offered by the touchscreen or haptic feedback [179]. Figure 8.7 shows an illustration of this concept where TrackCap is used to interact with the digital environment shown in a HMD while offering additional controls and providing haptic feedback. We also did not further explore how personal phones can fuel personalization of the experience or support collaborative activities.

Finally, future commercial solutions could replace the tracker with a 3D model-based tracker or a silhouette-based tracker. The idea here is not to track the features on the TrackCap, but to track the shape or appearance of the HMD itself. One advantage of our existing approach though is that it can be retrofitted to existing HMD designs, but a streamlined version would include the "cap" into the HMD design itself.

# CHAPTER 9

# Conclusion

This thesis aimed to identify effective methods for authoring, visualization and interaction of augmented reality tutorials. Augmented Reality reduces the cognitive load by presenting instructions directly registered to objects in the user's real environment. In our work we have discussed and addressed major problems of AR documentation and answered the main research questions:

Since the creation of AR content is costly, can we utilize existing sources to speed up the authoring process? We developed several methods to retarget printed media, videos and also arbitrary live scenes to AR tutorials, as described in chapters 3, 4 and 5. Based on our findings we can conclude that we can re-use large portions of existing sources with the appropriate methods to extract said data. In some cases, these workflows are not fully automated but require a certain amount of user interaction and supervision.

Which visualization methods are best suited for AR tutorial systems? We provide insights into this research question in chapter 6. We iteratively developed multiple visualization methods, which are suited for different situations. We conducted several user studies and based on the findings as well as various user comments we could improve our methods. We can conclude that the most important factors when it comes to the design of visualizations are simplicity and adaptivity. Too much clutter or information quickly overwhelmed the users, so reducing the visuals to a minimum is often the best option. Especially when it comes to animations, it is important that the users are able to follow the instructions, otherwise, the experience quickly turns into a stressful situation.

How can we improve the interaction possibilities for AR tutorial systems? This thesis presented novel interaction techniques for two different AR presentation devices. First, we analyzed interaction improvements for handheld AR, which uses smartphones or tablets as output devices. We showed in a user study that user perspective rendering (UPR) can greatly improve the experience for AR instructions by correcting the view through the handheld display. Using UPR, the user's hand appears at the expected position behind the AR display, which is especially important

for following AR tutorials. Second, we studied interaction using an optical see-through head-mounted display, such as the Microsoft HoloLens. Unlike most VR systems, it comes without tracked controllers and is completely untethered. Inputs to the system are usually done via hand gestures in mid-air in front of the HMD, a method which lacks haptic feedback and is very tiring for the user. We showed that a smartphone can be used as an inexpensive, fully tracked control device, which also provides a high-resolution display and passive haptic feedback. We conducted a user study, showing that a fully tracked controller aids in many object selection and manipulation tasks. Furthermore, our method does not add computational load on the HMD, because tracking is handled on the smartphone.

The research work done during this thesis is not only theoretical but resulted in a set of working systems, which support authoring tasks, the discussed visualization and interaction methods. Several user studies were conducted using these systems to gather more insights and were used to iteratively improve said systems.

Besides our design recommendations, several directions for future work exist. For example, the authoring components could be extended to support more types of tutorials, such as documentations which depict interactions with soft tissue or deformable objects. Generating candidate configurations of 3D deformable objects presents additional challenges such as a large number of candidate configurations and complex models describing the deformations. Additional research in the area of deep learning could increase the number of object classes our systems can support. In the area of human body motion tracking improvements for tracking other body parts, such as hands, will be another challenge. Furthermore, the investigation of tools to effectively extract and visualize 3D tutorials that require precise motion in time is an interesting topic. This will require the design of visualizations that encode speed, velocity and the direction of tools in 3D. An important part of this work will be the evaluation of these visualizations, which need to convey different attributes without distracting the user. Regarding online authoring, as described in our remote light field approach, it could be extended to support dynamic scenes, which would require different data structures. Our UPR system is designed for the current generation of mobile devices. However, faster and more reliable head tracking approaches, in combination with more powerful devices will make our adaptive solution for user perspective rendering less important. Therefore, future work needs to further investigate head tracking on mobile devices.

In the course of this thesis, we presented novel methods for **authoring** and **visualization** of AR documentation as well as novel **interaction** approaches for AR instruction systems. We have provided important design considerations for AR instruction visualization based on multiple user studies.

# APPENDIX A

## List of Publications

My work at the Institute of Computer Graphics and Vision led to the following peer-reviewed publications. For the sake of completeness of this Thesis, they are listed in chronological order along with the respective abstracts.

### 2015

#### Retargeting Technical Documentation to Augmented Reality

Peter Mohr, Bernhard Kerbl, Michael Donoser, Dieter Schmalstieg, and Denis Kalkofen.
In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15). Association for Computing Machinery, New York, NY, USA, 3337-3346.
DOI:https://doi.org/10.1145/2702123.2702490

**Abstract:** We present a system which automatically transfers printed technical documentation, such as handbooks, to three-dimensional Augmented Reality. Our system identifies the most frequent forms of instructions found in printed documentation, such as image sequences, explosion diagrams, textual annotations and arrows indicating motion. The analysis of the printed documentation works automatically, with minimal user input. The system only requires the documentation itself and a CAD model or 3D scan of the object described in the documentation. The output is a fully interactive Augmented Reality application, presenting the information from the printed documentation in 3D, registered to the real object.

## 2017

### Retargeting Video Tutorials Showing Tools With Surface Contact to Augmented Reality

Peter Mohr, David Mandl, Markus Tatzgern, Eduardo Veas, Dieter Schmalstieg, and Denis Kalkofen.
In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). Association for Computing Machinery, New York, NY, USA, 6547-6558. DOI:https://doi.org/10.1145/3025453.3025688

**Abstract:** A video tutorial effectively conveys complex motions, but may be hard to follow precisely because of its restriction to a predetermined viewpoint. Augmented reality (AR) tutorials have been demonstrated to be more effective. We bring the advantages of both together by interactively retargeting conventional, two-dimensional videos into three-dimensional AR tutorials. Unlike previous work, we do not simply overlay video, but synthesize 3D-registered motion from the video. Since the information in the resulting AR tutorial is registered to 3D objects, the user can freely change the viewpoint without degrading the experience. This approach applies to many styles of video tutorials. In this work, we concentrate on a class of tutorials which alter the surface of an object.

### Adaptive User-Perspective Rendering for Handheld Augmented Reality

Peter Mohr, Markus Tatzgern, Jens Grubert, Dieter Schmalstieg and Denis Kalkofen.
In Proceedings of the 2017 IEEE Symposium on 3D User Interfaces (3DUI), Los Angeles, CA, 2017, 176-181. DOI:https://doi.org/10.1109/3DUI.2017.7893336

**Abstract:** Handheld Augmented Reality commonly implements some variant of magic lens rendering, which turns only a fraction of the user's real environment into AR while the rest of the environment remains unaffected. Since handheld AR devices are commonly equipped with video see-through capabilities, AR magic lens applications often suffer from spatial distortions, because the AR environment is presented from the perspective of the camera of the mobile device. Recent approaches counteract this distortion based on estimations of the user's head position, rendering the scene from the user's perspective. To this end, approaches usually apply face-tracking algorithms on the front camera of the mobile device. However, this demands high computational resources and therefore commonly affects the performance of the application beyond the already high computational load of AR applications. In this paper, we present a method to reduce the computational demands for user perspective rendering by applying lightweight optical flow tracking and an estimation of the user's motion before head tracking is started. We demonstrate the suitability of our approach for computationally limited mobile devices and we compare it to device perspective rendering, to head tracked user perspective rendering, as well as to fixed point of view user perspective rendering.

## 2019

### TrackCap: Enabling Smartphones for 3D Interaction on Mobile Head-Mounted Displays

Peter Mohr, Markus Tatzgern, Tobias Langlotz, Andreas Lang, Dieter Schmalstieg, and Denis Kalkofen.
In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19). Association for Computing Machinery, New York, NY, USA, Paper 585, 1-11. DOI:https://doi.org/10.1145/3290605.3300815

**Abstract:** The latest generation of consumer market Head-mounted displays (HMD) now include self-contained inside-out tracking of head motions, which makes them suitable for mobile applications. However, 3D tracking of input devices is either not included at all or requires to keep the device in sight, so that it can be observed from a sensor mounted on the HMD. Both approaches make natural interactions cumbersome in mobile applications. TrackCap, a novel approach for 3D tracking of input devices, turns a conventional smartphone into a precise 6DOF input device for an HMD user. The device can be conveniently operated both inside and outside the HMD's field of view, while it provides additional 2D input and output capabilities.

## 2020

### Mixed Reality Light Fields for Interactive Remote Assistance

Peter Mohr, Shohei Mori, Tobias Langlotz, Bruce Thomas, Dieter Schmalstieg, and Denis Kalkofen.
In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20), Association for Computing Machinery, New York, NY, USA, to appear. http://dx.doi.org/10.1145/3313831.3376289

**Abstract:** Remote assistance represents an important use case for mixed reality. With the rise of handheld and wearable devices, remote assistance has become practical in the wild. However, spontaneous provisioning of remote assistance requires an easy, fast and robust approach for capturing and sharing of unprepared environments. In this work, we make a case for utilizing interactive light fields for remote assistance. We demonstrate the advantages of object representation using light fields over conventional geometric reconstruction. Moreover, we introduce an interaction method for quickly annotating light fields in 3D space without requiring surface geometry to anchor annotations. We present results from a user study demonstrating the effectiveness of our interaction techniques, and we provide feedback on the usability of our overall system.

# Bibliography

[1] (2011). ISO 9241-420:2011: Ergonomics of human-system interaction – Part 420: Selection of physical input devices, International Standard, International Organization for Standardization. (page 94, 103)

[2] Adcock, M., Ranatunga, D., Smith, R., and Thomas, B. H. (2014). Object-based Touch Manipulation for Remote Guidance of Physical Tasks. In *Proceedings of the 2nd ACM Symposium on Spatial User Interaction*, SUI '14, pages 113–122, New York, NY, USA. ACM. (page 19)

[3] Agarwal, A. and Triggs, B. (2006). Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58. (page 22)

[4] 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*, 22(3):828–837. (page 20, 23, 35)

[5] Alem, L. and Huang, W. (2011). Developing mobile remote collaboration systems for industrial use: Some design challenges. In Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., and Winckler, M., editors, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6949 LNCS, pages 442–445, Berlin, Heidelberg. Springer Berlin Heidelberg. (page 18)

[6] Alexiadis, D. S. and Daras, P. (2014). Quaternionic signal processing techniques for automatic evaluation of dance performances from mocap data. *IEEE Transactions on Multimedia*, 16(5):1391–1406. (page 24)

[7] Alvarez, H., Aguinaga, I., and Borro, D. (2011). Providing guidance for maintenance operations using automatic markerless augmented reality system. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 181–190. IEEE. (page 17)

[8] Anderson, F., Grossman, T., Matejka, J., and Fitzmaurice, G. (2013). YouMove: Enhancing Movement Training with an Augmented Reality Mirror. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 311–320, New York, NY, USA. ACM. (page 21)

[9] Ayres, P., Marcus, N., Chan, C., and Qian, N. (2009). Learning hand manipulative tasks: When instructional animations are superior to equivalent static representations. *Computers in Human Behavior*, 25(2):348–353. (page 23)

[10] Baltrušaitis, T., Robinson, P., and Morency, L. P. (2013). Constrained local neural fields for robust facial landmark detection in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 354–361. (page 51)

[11] Bangor, A., Kortum, P., and Miller, J. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123. (page 72, 73, 89)

[12] Barakonyi, I. and Schmalstieg, D. (2007). Ubiquitous animated agents for augmented reality. In *Proceedings - ISMAR 2006: Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '06, pages 145–154. (page 20)

[13] Baričević, D., Höllerer, T., Sen, P., and Turk, M. (2014). User-perspective augmented reality magic lens from gradients. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology - VRST '14*, pages 87–96. (page 24)

[14] Baričević, D., Höllerer, T., Sen, P., and Turk, M. (2016). User-Perspective AR Magic Lens from Gradient-Based IBR and Semi-Dense Stereo. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1. (page 24)

[Baričević et al.] Baričević, D., Lee, C., Turk, M., Höllerer, T., and Bowman, D. A. A hand-held ar magic lens with user-perspective rendering.pdf. In *Proc. of IEEE ISMAR*, pages 197–206. (page 7, 24, 91)

[16] Bergig, O., Hagbi, N., El-Sana, J., and Billinghurst, M. (2009). In-place 3D sketching for authoring and augmenting mechanical systems. In *Science and Technology Proceedings - IEEE 2009 International Symposium on Mixed and Augmented Reality, ISMAR 2009*, pages 87–94. (page 22)

[17] Biard, N., Cojean, S., and Jamet, E. (2018). Effects of segmentation and pacing on procedural learning by video. *Computers in Human Behavior*, 89:411–417. (page 23)

[18] Birklbauer, C. and Bimber, O. (2015). Active Guidance for Light-field Photography on Smartphones. *Computers and Graphics (Pergamon)*, 53:127–135. (page xix, 25, 26)

[19] Borst, C. W. and Indugula, A. P. (2005). Realistic Virtual Grasping. In *Proc. of IEEE VR*, pages 91–98. (page xix, 26, 27)

[20] Bowman, D. A. (2005). *3D User Interfaces: Theory and Practice*. Usability Series. Pearson Education, Limited. (page 26)

[21] Bowman, D. A. and Hodges, L. F. (1997). An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments. In *Proc. of I3D*, pages 35—-ff. (page 108)

[22] Breedveld, P. (1997). Observation, manipulation, and eye-hand coordination problems in minimally invasive surgery. In *Proceedings of the 16th European Conference on Human Decision Making and Manual Control*, pages 219–231. (page 1)

[23] Brooke, J. (1996). SUS - A quick and dirty usability scale. *Usability Evaluation in Industry*, 189(194):4–7. (page 87)

[24] Butz, A. and Intelligence, A. (1994). BETTY : Planning and Generating Animations for the Visualization of Movements and Spatial Relations. In *Artificial Intelligence*, pages 53–58. (page 20)

[25] Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698. (page 35)

[26] Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., and Sheikh, Y. (2018). OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*. (page 21)

[27] Caudell, T. P. and Mizell, D. W. (1992). Augmented reality: An application of heads-up display technology to manual manufacturing processes. In *Proceedings of the twenty-fifth Hawaii international conference on system sciences*, volume 2, pages 659–669. IEEE. (page 15)

[28] Chen, T., Zhu, Z., Shamir, A., Hu, S.-M., and Cohen-Or, D. (2013). 3-Sweep. *ACM Transactions on Graphics*, 32(6):1–10. (page 22)

[29] Chi, P.-Y. P., Ahn, S., Ren, A., Dontcheva, M., Li, W., and Hartmann, B. (2012). MixT: Automatic Generation of Step-by-Step Mixed Media Tutorials. In *Proceedings of the ACM Symposium on User Interface Software and Technology - UIST'12*, pages 93–102. ACM. (page 20)

[30] Chi, P.-y. P., Vogel, D., Dontcheva, M., Li, W., and Hartmann, B. (2016). Authoring Illustrations of Human Movements by Iterative Physical Demonstration. In *Proceedings of the 29$^{th}$ Annual Symposium on User Interface Software and Technology - UIST '16*, UIST '16, pages 809–820. (page 21)

[31] Chuang, Y. Y., Agarwala, A., Curless, B., Salesin, D. H., and Szeliski, R. (2002). Video matting of complex scenes. In *ACM Transactions on Graphics (TOG)*, volume 21 of *SIGGRAPH '02*, pages 243–248, New York, NY, USA. ACM. (page 56)

[32] Collet, A., Chuang, M., Sweeney, P., Gillett, D., Evseev, D., Calabrese, D., Hoppe, H., Kirk, A., and Sullivan, S. (2015). High-quality streamable free-viewpoint video. *ACM Transactions on Graphics*, 34(4):69:1–69:13. (page 22)

[33] Collomosse, J. P., Rowntree, D., and Hall, P. M. (2003). Cartoon-Style Rendering of Motion from Video. *Proceedings of the 1$^{st}$ International Conference on Vision, Video and Graphics (VVG)*, 67(6):117–124. (page 22)

[34] Cournia, N., Smith, J. D., and Duchowski, A. T. (2003). Gaze- vs. Hand-based Pointing in Virtual Environments. In *Proceedings of CHI Extended Abstracts*, pages 772–773. (page 9, 27)

[35] Damen, D., Haines, O., Leelasawassuk, T., Calway, A., and Mayol-Cuevas, W. (2014). Multi-user egocentric Online System for Unsupervised Assistance on Object Usage. In *ECCV Workshop on Assistive Computer Vision and Robotics (ACVR)*. (page 21)

[36] Davis, A., Levoy, M., and Durand, F. (2012). Unstructured Light Fields. *Computer Graphics Forum*, 31(2):305–314. (page xix, 5, 25, 26, 61, 67)

[37] DeMenthon, D. F. and Davis, L. S. (1992). Model-based object pose in 25 lines of code. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 588 LNCS(1-2):335–343. (page 32)

[38] Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122. (page 83)

[39] Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-Scale Direct Monocular SLAM. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 834–849, Cham. Springer International Publishing. (page 19)

[40] Feiner, S., MacIntyre, B., and Seligmann, D. (1992). Annotating the real world with knowledge-based graphics on a see-through head-mounted display. In *proceedings of Graphics Interface*, volume 92, pages 78–85. (page 19)

[41] Feiner, S., Macintyre, B., and Seligmann, D. (1993). Knowledge-based augmented reality. *Communications of the ACM*, 36(7):53–62. (page xviii, 15, 16, 20)

[42] Garland, T. B. and Sanchez, C. A. (2013). Rotational perspective and learning procedural tasks from dynamic media. *Computers and Education*, 69:31–37. (page 23)

[43] Gauglitz, S., Lee, C., Turk, M., and Höllerer, T. (2012). Integrating the physical environment into mobile remote collaboration. In *Proceedings of the 14$^{th}$ international conference on Human-computer interaction with mobile devices and services*, MobileHCI '12, pages 241–250, New York, NY, USA. ACM. (page xix, 18, 19)

[44] Gauglitz, S., Nuernberger, B., Turk, M., and Höllerer, T. (2014a). In Touch with the Remote World: Remote Collaboration with Augmented Reality Drawings and Virtual Navigation. In *Proceedings of the 20$^{th}$ ACM Symposium on Virtual Reality Software and Technology*, VRST '14. (page 4, 5, 19, 61)

[45] Gauglitz, S., Nuernberger, B., Turk, M., and Höllerer, T. (2014b). World-stabilized Annotations and Virtual Scene Navigation for Remote Collaboration. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 449–459, New York, NY, USA. ACM. (page 5, 19, 62, 66)

[46] Godard, C., Goldwasser, M., and Ramkumar, G. (1995). An Efficient System for Geometric Assembly Sequence Generation and Evaluation. In *International computers in Engineering conference*, pages 699–712. (page 35)

[47] Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F. (1996). The Lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 43–54, New York, NY, USA. ACM. (page 25)

[48] Goto, M., Uematsu, Y., Saito, H., Senda, S., and Iketani, A. (2010). Task support system by displaying instructional video onto AR workspace. In *9th IEEE International Symposium on Mixed and Augmented Reality 2010: Science and Technology, ISMAR 2010 - Proceedings*, pages 83–90. (page 3, 21, 23)

[49] Grabler, F., Agrawala, M., Li, W., Dontcheva, M., and Igarashi, T. (2009). Generating photo manipulation tutorials by demonstration. *ACM Transactions on Graphics*, 28(3):1. (page 20)

[50] Gupta, A., Fox, D., Curless, B., and Cohen, M. (2012). DuploTrack: A Real-time System for Authoring and Guiding Duplo Block Assembly. In *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*, page 389. (page xviii, 16, 17, 21, 23)

[51] Harrison, C., Tan, D., and Morris, D. (2010). Skinput: Appropriating the Body As an Input Surface. In *Proc. of CHI*, pages 453–462. (page 26)

[52] Hart, S. G. and Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Advances in psychology*, 52:139–183. (page 69, 87, 95, 104, 106, 108, 111, 113)

[53] Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition. (page 46)

[54] He, T., Chen, X., Chen, Z., Li, Y., Liu, S., Hou, J., and He, Y. (2017). Immersive and collaborative taichi motion learning in various vr environments. *Proceedings - IEEE Virtual Reality*, 2(c):307–308. (page 23)

[55] Heiser, J., Phan, D., Agrawala, M., Tversky, B., and Hanrahan, P. (2004). Identification and validation of cognitive design principles for automated generation of assembly instructions. In *Proceedings of the working conference on Advanced visual interfaces - AVI '04*, page 311. (page 20, 23, 31)

[56] Henderson, S. and Feiner, S. (2011). Exploring the benefits of augmented reality documentation for maintenance and repair. *IEEE Transactions on Visualization and Computer Graphics*, 17(10):1355–1368. (page 2, 16, 77, 85)

[57] Hill, A., Schiefer, J., Wilson, J., Davidson, B., Gandy, M., and MacIntyre, B. (2011). Virtual transparency: Introducing parallax view into video see-through AR. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011*, pages 239–240. (page 24)

[58] Hincapié-Ramos, J. D., Ozacar, K., Irani, P. P., and Kitamura, Y. (2015). GyroWand: IMU-based Raycasting for Augmented Reality Head-Mounted Displays. In *Proceedings of the 3rd ACM Symposium on Spatial User Interaction*, pages 89–98. (page 9, 27, 103, 106)

[59] Hincapié-Ramos, J. D., Roscher, S., Büschel, W., Kister, U., Dachselt, R., and Irani, P. (2014a). cAR: Contact Augmented Reality with Transparent-Display Mobile Devices. In *Proc. of ACM International Symposium on Pervasive Displays*. (page 24)

[60] Hincapié-Ramos, J. D., Roscher, S., Büschel, W., Kister, U., Dachselt, R., and Irani, P. (2014b). tPad. In *Proceedings of the 2014 conference on Designing interactive systems - DIS '14*, pages 161–170. (page 24)

[61] Hinckley, K., Sinclair, M., Hanson, E., Szeliski, R., and Conway, M. (1999). The VideoMouse: A Camera-based Multi-degree-of-freedom Input Device. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, UIST '99, pages 103–112. (page 28)

[62] Höffler, T. N. and Leutner, D. (2007). Instructional animation versus static pictures: A meta-analysis. *Learning and Instruction*, 17(6):722–738. (page 23)

[63] Holz, D., Ullrich, S., Wolter, M., and Kuhlen, T. (2008). Multi-Contact Grasp Interaction for Virtual Environments. *JVRB - Journal of Virtual Reality and Broadcasting*, 5(2008)(7). (page 26)

[64] Huang, W. and Alem, L. (2013). Gesturing in the Air: Supporting Full Mobility in Remote Collaboration on Physical Tasks. *Journal of Universal Computer Science*, 19(8):1158–1174. (page 18)

[65] Huang, W., Alem, L., and Tecchia, F. (2013). Handsin3d: Augmenting the shared 3d visual space with unmediated hand gestures. In *SIGGRAPH Asia 2013 Emerging Technologies*, SA '13, pages 10:1–10:3, New York, NY, USA. ACM. (page 4, 61)

[66] Huang, W., Alem, L., Tecchia, F., and Duh, H. B.-L. (2018). Augmented 3d hands: a gesture-based mixed reality system for distributed collaboration. *Journal on Multimodal User Interfaces*, 12(2):77–89. (page 18)

[67] Ihm, I., Park, S., and Lee, R. K. (1997). Rendering of spherical light fields. In *Proceedings of the 5th Pacific Conference on Computer Graphics and Applications*, PG '97, pages 59–68. IEEE Computer Society. (page 65)

[68] Jacobs, J. and Froehlich, B. (2011). A soft hand model for physically-based manipulation of virtual objects. In *IEEE Virtual Reality Conference, VR 2011, Singapore, 19-23 March 2011*, pages 11–18. (page 26)

[69] Jarabo, A., Masia, B., Bousseau, A., Pellacini, F., and Gutierrez, D. (2014). How Do People Edit Light Fields? *ACM Transactions on Graphics*, 33(4):146:1—-146:10. (page 5, 26, 62, 69, 71, 72)

[70] Jianbo Shi and Tomasi (1994). Good features to track. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, pages 593–600. (page 8)

[71] Jimenez, J., Gutierrez, D., and Latorre, P. (2008). Gaze-based Interaction for Virtual Environments. *j-jucs*, 14(19):3085–3098. (page 27)

[72] Kalal, Z., Mikolajczyk, K., and Matas, J. (2012). Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422. (page 48, 51)

[73] Kalkofen, D., Sandor, C., White, S., and Schmalstieg, D. (2011). Visualization Techniques for Augmented Reality. In Furht, B., editor, *Handbook of Augmented Reality*, pages 65–98. Springer. (page 22)

[74] Kalkofen, D., Tatzgern, M., and Schmalstieg, D. (2009). Explosion diagrams in augmented reality. In *Proceedings - IEEE Virtual Reality*, pages 71–78. (page xix, 20, 22, 78)

[75] Karitsuka, T. and Sato, K. (2003). A wearable mixed reality with an on-board projector. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 321–322. (page 26)

[76] Kasahara, S. and Rekimoto, J. (2014). JackIn: Integrating First-person View with Out-of-body Vision Generation for Human-human Augmentation. In *Proceedings of the 5$^{th}$ Augmented Human International Conference*, AH '14, pages 46:1—-46:8, New York, NY, USA. ACM. (page 18, 19)

[77] Kerbl, B., Kalkofen, D., Steinberger, M., and Schmalstieg, D. (2015). Interactive Disassembly Planning for Complex Objects. *Computer Graphics Forum*, 34(2):287–297. (page 20)

[78] Kim, B. and Essa, I. (2005). Video-based nonphotorealistic and expressive illustration of motion. In *Proceedings of Computer Graphics International Conference, CGI*, pages 32–35. (page 22)

[79] Kim, J., Lee, M., Kim, Y., Eun, S. D., and Yoon, B. C. (2016). Feasibility of an individually tailored virtual reality program for improving upper motor functions and activities of daily living in chronic stroke survivors: A case series. *European Journal of Integrative Medicine*, 8(5):731–737. (page 18)

[80] Kindratenko, V. V. (2000). A survey of electromagnetic position tracker calibration techniques. *Virtual Reality*, 5(3):169–182. (page 28)

[81] Kraevoy, V., Sheffer, A., and van de Panne, M. (2009). Modeling from contour drawings. In *Sketch Based Interfaces and Modeling*, SBIM '09, pages 37–44, New York, NY, USA. ACM. (page 51)

[82] Kratz, S., Kimber, D., Su, W., Gordon, G., and Severns, D. (2014). Polly: "Being There" Through the Parrot and a Guide. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices & Services*, MobileHCI '14, pages 625–630, New York, NY, USA. ACM. (page 19)

[83] Krotkov, E. and Martin, J. . (1986). Range from focus. In *Proceedings of 1986 IEEE International Conference on Robotics and Automation*, volume 3 of *ICRA '86*, pages 1093–1098. IEEE. (page 67)

[84] Langlotz, T., Zingerle, M., Grasset, R., Kaufmann, H., and Reitmayr, G. (2012). AR Record Replay: Situated Compositing of Video Content in Mobile Augmented Reality. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, OzCHI '12, pages 318–326. (page 21)

[85] Ledermann, F. and Schmalstieg, D. (2005). APRIL: a high-level framework for creating augmented reality presentations. In *IEEE Proceedings. VR 2005. Virtual Reality, 2005.*, pages 187–194. (page 20)

[86] Lepetit, V., F.Moreno-Noguer, and P.Fua (2009). EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal Computer Vision*, 81(2). (page 45, 49, 99)

[87] Levoy, M. and Hanrahan, P. (1996). Light Field Rendering. In *SIGGRAPH*, pages 1–12. (page 25)

[88] Li, W., Agrawala, M., Curless, B., and Salesin, D. (2008). Automated generation of interactive 3D exploded view diagrams. *ACM Transactions on Graphics*, 27(3):1. (page 20)

[89] Li, W., Agrawala, M., and Salesin, D. (2004). Interactive Image-Based Exploded View Diagrams. In *Prof of Graphics Interface*, pages 203–212. (page 21)

[90] Liu, C., Yuen, J., and Torralba, A. (2011). SIFT Flow: Dense Correspondence Across Scenes and Its Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–994. (page 52)

[91] Liu, C., Yuen, J., Torralba, A., Sivic, J., and Freeman, W. T. (2008). SIFT Flow: Dense Correspondence Across Different Scenes. In *Proc. of ECCV*, pages 28–42. (page 37)

[92] Liu, M. Y., Tuzel, O., Veeraraghavan, A., and Chellappa, R. (2010). Fast directional chamfer matching. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1696–1703. (page 35)

[93] Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*, 60(2):91–110. (page 45, 49)

[94] Lozano-Pérez, T. (1983). Spatial Planning: A Configuration Space Approach. *IEEE Transactions on Computers*, C-32(2):108–120. (page 34)

[95] Lubos, P., Bruder, G., and Steinicke, F. (2014). Analysis of direct selection in head-mounted display environments. In *IEEE Symposium on 3D User Interfaces 2014, 3DUI 2014 - Proceedings*, pages 11–18. (page 106)

[96] Lucas, B. D. and Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. In *Robotics*, volume 81 of *IJCAI'81*, pages 674–679. (page 8, 45, 49)

[97] Matas, J., Chum, O., Urban, M., and Pajdla, T. (2004). Robust wide baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22:761–767. (page 32)

[98] Mijksenaar, P. and Westendorp, P. (1999). *Open here: the art of instructional design*. Thames & Hudson. (page 1, 20, 30)

[99] Mildenhall, B., Srinivasan, P. P., Ortiz-Cayon, R., Kalantari, N. K., Ramamoorthi, R., Ng, R., and Kar, A. (2019). Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *ACM Transactions on Graphics*, 38(4):29:1—-29:14. (page 26)

[100] Milgram, P. and Kishino, F. (1994). A Taxonomy of Mixed Reality Visual Displays. *IEICE Transactions on Information Systems*, E77-D(12):1321–1329. (page 62)

[101] Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97. (page 23)

[102] Mistry, P. and Maes, P. (2009). SixthSense: A Wearable Gestural Interface. In *ACM SIGGRAPH ASIA Sketches*, pages 11:1—-11:1. (page 26)

[103] Moeslund, T. B., Hilton, A., and Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3 SPEC. ISS.):90–126. (page 22)

[104] Mohr, P., Kerbl, B., Donoser, M., Schmalstieg, D., and Kalkofen, D. (2015). Retargeting Technical Documentation to Augmented Reality. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, CHI '15, pages 3337–3346, New York, NY, USA. ACM. (page xix, 20, 22)

[105] Mohr, P., Tatzgern, M., Grubert, J., Schmalstieg, D., and Kalkofen, D. (2017). Adaptive user-perspective rendering for handheld augmented reality. In *Proceedings of 2017 IEEE Symposium on 3D User Interfaces*, 3DUI '17. IEEE. (page 72)

[106] Müller, J., Langlotz, T., and Regenbrecht, H. (2016). PanoVC: Pervasive telepresence using mobile phones. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. (page 19)

[107] Nebehay, G. (2012). Robust Object Tracking Based on Tracking-Learning-Detection. Master's thesis, Faculty of Informatics, TU Vienna. (page 47, 51)

[108] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *2011 10$^{th}$ IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011*, ISMAR '11, pages 127–136, Washington, DC, USA. IEEE Computer Society. (page 47, 51, 78)

[109] Ng, R., Levoy, M., Brédif, M., Duval, G., Horowitz, M., and Hanrahan, P. (2005). Light Field Photography with a Hand-Held Plenoptic Camera. Technical report, Stanford Tech Report CTSR 2005-02 Light. (page 26)

[110] Nienhaus, M. and Döllner, J. (2003). Dynamic Glyphs – Depicting Dynamics in Images of 3D Scenes. In *Proceedings of Third International Symposium on Smart Graphics 2003*, SG'03, pages 102–111, Berlin, Heidelberg. Springer-Verlag. (page 3, 6, 20, 83)

[111] Nienhaus, M. and Döllner, J. (2005). Depicting dynamics using principles of visual art and narrations. *IEEE Computer Graphics and Applications*, 25(3):40–51. (page 4, 61)

[112] Nuernberger, B., Lien, K.-C., Höllerer, T., and Turk, M. (2016). Interpreting 2D gesture annotations in 3D augmented reality. In *2016 IEEE Symposium on 3D User Interfaces, 3DUI 2016, Greenville, SC, USA, March 19-20, 2016*, pages 149–158. (page 19)

[113] Ondruska, P., Kohli, P., and Izadi, S. (2015). MobileFusion: Real-Time Volumetric Surface Reconstruction and Dense Tracking on Mobile Phones. In *IEEE Transactions on Visualization and Computer Graphics*, volume 21, pages 1251–1258, Fukuoka, Japan. (page 47, 51)

[114] Orts-Escolano, S., Kim, D., Cai, Q., Rhemann, C., Davidson, P., Chou, P., Fanello, S., Khamis, S., Mennicken, S., Chang, W., Dou, M., Valentin, J., Kowdle, A., Tankovich, V., Pradeep, V., Degtyarev, Y., Loop, C., Wang, S., Kang, S. B., Kohli, P., Lutchyn, Y., Keskin, C., and Izadi, S. (2016). Holoportation: Virtual 3D teleportation in real-time. In *UIST 2016 - Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 741–754, New York, New York, USA. ACM Press. (page xix, 5, 18, 61)

[115] Park, M., Serefoglou, S., Schmidt, L., Radermacher, K., Schlick, C., and Luczak, H. (2008). Hand-Eye Coordination Using a Video See-Through Augmented Reality System. *The Ergonomics Open Journal*, 1(1):46–53. (page 2)

[116] Pech-Pacheco, J. L., Cristóbal, G., Chamorro-Martinez, J., and Fernández-Valdivia, J. (2000). Diatom autofocusing in brightfield microscopy: A comparative study. In *Proceedings of 15th International Conference on Pattern Recognition*, volume 3 of *ICPR '00*, pages 314–317. IEEE. (page 67)

[117] Pérez, F., Pérez, A., Rodríguez, M., and Magdaleno, E. (2016). Lightfield Recovery from Its Focal Stack. *Journal of Mathematical Imaging and Vision*, 56(3):573–590. (page 26)

[118] Petersen, N. and Stricker, D. (2012). Learning task structure from video examples for workflow tracking and authoring. In *ISMAR 2012 - 11th IEEE International Symposium on Mixed and Augmented Reality 2012, Science and Technology Papers*, pages 237–246. (page 3, 21)

[119] Polvi, J., Taketomi, T., Yamamoto, G., Dey, A., Sandor, C., and Kato, H. (2016). Slidar: A 3d positioning method for slam-based handheld augmented reality. *Computers & Graphics*, 55:33–43. (page 69)

[120] Pongnumkul, S., Dontcheva, M., Li, W., Wang, J., Bourdev, L., Avidan, S., and Cohen, M. F. (2011). Pause-and-play: Automatically Linking Screencast Video Tutorials with Applications. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 135–144, New York, NY, USA. ACM. (page 54, 90)

[121] Poupyrev, I., Billinghurst, M., Weghorst, S., and Ichikawa, T. (1996). The Go-go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR. In *Proc. of UIST*, pages 79–80. (page 99)

[122] Prisacariu, V. A. and Reid, I. D. (2012). PWP3D: Real-time segmentation and tracking of 3D objects. *International journal of computer vision*, 98(3):335–354. (page 15)

[123] Pucihar, K. Č. and Coulton, P. (2014a). Contact-view: A magic-lens paradigm designed to solve the dual-view problem. In *ISMAR 2014 - IEEE International Symposium on Mixed and Augmented Reality - Science and Technology 2014, Proceedings*, pages 297–298. (page 24)

[124] Pucihar, K. Č. and Coulton, P. (2014b). Utilizing contact-view as an augmented reality authoring method for printed document annotation. In *ISMAR 2014 - IEEE International Symposium on Mixed and Augmented Reality - Science and Technology 2014, Proceedings*, pages 299–300. (page 24)

[125] Pucihar, K. Č., Grubert, J., and Kljun, M. (2015). Dual Camera Magic Lens for Handheld AR Sketching. In *Human-Computer Interaction*, pages 523–527. Springer. (page 24)

[126] Rad, M. and Lepetit, V. (2017). BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3836. (page 15)

[127] Reiners, D., Stricker, D., Klinker, G., and Müller, S. (1999). Augmented reality for construction tasks: Doorlock assembly. In *Proceedings of the international workshop on Augmented reality: placing artificial objects in real scenes: placing artificial objects in real scenes*, pages 31–46. AK Peters, Ltd. (page 15)

[128] Reitmayr, G., Chiu, C., Kusternig, A., Kusternig, M., and Witzmann, H. (2005). iOrb - Unifying Command and 3D Input for Mobile Augmented Reality. In *Proc. IEEE Virtual Reality Workshop on New Diretions in 3D User Interfaces*. (page 9, 27)

[129] Richter-Trummer, T., Kalkofen, D., Park, J., and Schmalstieg, D. (2016). Instant Mixed Reality Lighting from Casual Scanning. In *2016 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2016, Merida, Yucatan, Mexico, September 19-23, 2016*, pages 27–36. (page 19)

[130] Rohs, M. (2005). Real-World Interaction with Camera Phones. In Murakami, H., Nakashima, H., Tokuda, H., and Yasumura, M., editors, *Ubiquitous Computing Systems*, pages 74–89, Berlin, Heidelberg. Springer Berlin Heidelberg. (page 28)

[131] Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM. (page 47)

[132] Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast Point Feature Histograms (FPFH) for 3D registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE. (page 78)

[133] Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4. (page 78)

[134] Salamin, P., Thalmann, D., and Vexo, F. (2006). The benefits of third-person perspective in virtual and augmented reality? *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, page 27. (page 24)

[135] Samini, A. and Palmerius, K. L. (2014). A perspective geometry approach to user-perspective rendering in hand-held video see-through augmented reality. In *Proceedings of the 20$^{th}$ ACM Symposium on Virtual Reality Software and Technology - VRST '14*, pages 207–208. (page 7, 24)

[136] Samini, A. and Palmerius, K. L. (2016). *A user study on touch interaction for user-perspective rendering in hand-held video see-through augmented reality*, volume 9769, pages 304–317. (page 25)

[137] Saragih, J. M., Lucey, S., and Cohn, J. F. (2009). Face alignment through subspace constrained mean-shifts. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1034–1041. IEEE. (page 93)

[138] Sato, G. Y. and Kosuke (2007). PALMbit: A PALM Interface with Projector-Camera System. *Ubiquitous Computing Adjunct Proceedings*, pages 276–279. (page 26)

[139] Sauro, J. and Dumas, J. S. (2009). Comparison of Three One-question, Post-task Usability Questionnaires. In *Proceedings of CHI*, pages 1599–1608. (page 69, 95, 104, 106, 108, 111, 113)

[140] Schmalstieg, D. and Höllerer, T. (2017). *Augmented reality: Principles and practice*. Addison Wesley Professional. (page 9)

[141] Seichter, H. and Grubert, J. (2014). [ Poster ] Towards User Perspective Augmented Reality for Public Displays. In *Proc of. IEEE ISMAR*, pages 267–268. (page 7, 25, 92, 93)

[142] Seligmann, D. D. and Feiner, S. (1991). Automated Generation of Intent-Based 3D Illustrations Design Rules : Mapping Intent to Stylistic. In *Computer Graphics, Volume 25, Number 4*, volume 25, pages 123–132. (page 20)

[143] Shao, T., Li, W., Zhou, K., Xu, W., Guo, B., and Mitra, N. J. (2013). Interpreting concept sketches. *ACM Transactions on Graphics*, 32(4):1. (page 21)

[144] Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, pages 336—-, Washington, DC, USA. IEEE Computer Society. (page 62)

[145] Sodhi, R. S., Jones, B. R., Forsyth, D., Bailey, B. P., and Maciocci, G. (2013). BeThere: 3D Mobile Collaboration with Spatial Input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 179–188, New York, NY, USA. ACM. (page 18, 19)

[146] Sorkine, O. and Alexa, M. (2007). As-Rigid-As-Possible Surface Modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '07, pages 109–116. (page 47)

[147] Spanjers, I. A. E., Van Gog, T., Wouters, P., and Van Merriënboer, J. J. G. (2012). Explaining the segmentation effect in learning from animations: The role of pausing and temporal cueing. *Computers and Education*, 59(2):274–280. (page 23)

[148] Srinivasan, H. and Gadh, R. (2000). Efficient geometric disassembly of multiple components from an assembly using wave propagation. *Mech. Design*, 122:179. (page 35)

[149] 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 *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '12, pages 217–226. IEEE Computer Society. (page 64)

[150] Sweller, J., Van Merrienboer, J. J. G., and Paas, F. G. W. C. (1998). Cognitive Architecture and Instructional Design. *Educational Psychology Review*, 10(3):251–296. (page 23)

[151] Tait, M. and Billinghurst, M. (2015). The Effect of View Independence in a Collaborative AR System. *Comput. Supported Coop. Work*, 24(6):563–589. (page 19)

[152] Takeuchi, Y. and Perlin, K. (2012). ClayVision: The (Elastic) Image of the City. In *Proc. of CHI*, pages 2411–2420. (page 22)

[153] Talvas, A., Marchal, M., and Lecuyer, A. (2013). The god-finger method for improving 3D interaction with virtual objects through simulation of contact area. In *IEEE Symposium on 3D User Interface 2013, 3DUI 2013 - Proceedings*, pages 111–114. IEEE Computer Society. (page 26)

[154] Tang, A., Owen, C., Biocca, F., and Mou, W. (2003). Comparative effectiveness of augmented reality in object assembly. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 73–80. ACM. (page 16)

[155] Tang, J. C. and Minneman, S. L. (1990). VideoDraw: A Video Interface for Collaborative Drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, pages 313–320, New York, NY, USA. ACM. (page 19)

[156] Tang, R., Yang, X.-D., Bateman, S., Jorge, J., and Tang, A. (2015). Physio@Home. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, pages 4123–4132. (page 21, 80, 81)

[157] Tanriverdi, V. and Jacob, R. J. K. (2000). Interacting with Eye Movements in Virtual Environments. In *Proc. of CHI*, pages 265–272. (page 9, 27, 103)

[158] Taylor, J., Luff, B., Topalian, A., Wood, E., Khamis, S., Kohli, P., Izadi, S., Banks, R., Fitzgibbon, A., Shotton, J., Bordeaux, L., Cashman, T., Corish, B., Keskin, C., Sharp, T., Soto, E., Sweeney, D., and Valentin, J. (2016). Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics*, 35(4):1–12. (page 26)

[159] Teather, R. J. and Stuerzlinger, W. (2014). Visual aids in 3D point selection experiments. In *Proceedings of the 2nd ACM symposium on Spatial user interaction - SUI '14*, pages 127–136. (page 109)

[160] Tjaden, H., Schwanecke, U., and Schomer, E. (2017). Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 124–132. (page 15)

[161] Tomioka, M., Ikeda, S., and Sato, K. (2013). Approximated user-perspective rendering in tablet-based augmented reality. In *2013 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2013*, pages 21–28. (page 24)

[162] Tversky, B., Morrison, J. B., and Betrancourt, M. (2002). Animation: Can it facilitate? *International Journal of Human Computer Studies*, 57(4):247–262. (page 1, 3)

[163] Unuma, Y., Niikura, T., and Komuro, T. (2014). See-through Mobile AR System for Natural 3D Interaction. In *Proceedings of IUI Companion*, pages 17–20. (page 24)

[164] van den Hengel, A., Dick, A., Thormählen, T., Ward, B., and Torr, P. H. S. (2007). VideoTrace: Rapid Interactive Scene Modelling from Video. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA. ACM. (page 22)

[165] Vanacken, L., Grossman, T., and Coninx, K. (2007). Exploring the effects of environment density and target visibility on object selection in 3D virtual environments. In *IEEE Symposium on 3D User Interfaces 2007 - Proceedings, 3DUI 2007*, pages 115–122. (page 27)

[166] Čopič Pucihar, K., Coulton, P., and Alexander, J. (2013). Evaluating dual-view perceptual issues in handheld augmented reality : device vs . user perspective rendering. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction (ICMI)*, pages 381–388. (page 8, 24, 92, 94, 97)

[167] Čopič Pucihar, K., Coulton, P., and Alexander, J. (2014). The use of surrounding visual context in handheld AR. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*, pages 197–206. (page 6, 7, 24)

[168] Wang, B., Wang, G., Sharf, A., Li, Y., Zhong, F., Qin, X., CohenOr, D., and Chen, B. (2018). Active Assembly Guidance with Online Video Parsing. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 459–466. IEEE. (page xviii, 16, 17)

[169] Wei, X. and Chai, J. (2010). VideoMocap: Modeling Physically Realistic Human Motion from Monocular Video Sequences. In *ACM SIGGRAPH 2010*, pages 42:1—-42:10. (page 22, 52)

[170] Welch, G., Bishop, G., Vicci, L., Brumback, S., Keller, K., and Colucci, D. (1999). The HiBall Tracker: High-performance Wide-area Tracking for Virtual and Augmented Environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '99, pages 1—-ff. (page 99)

[171] Wen, W., Qiao, X., Yanai, K., Nakagawa, J., Yasuda, J., Yamashita, A., and Asama, H. (2017). Skill evaluation and education services for bed-care nursing with sliding sheet with regression analysis. In *Serviceology for Smart Service System*, pages 253–259. Springer. (page 18)

[172] White, S., Feng, D., and Feiner, S. (2009). Interaction and presentation techniques for shake menus in tangible augmented reality. In *Science and Technology Proceedings - IEEE 2009 International Symposium on Mixed and Augmented Reality, ISMAR 2009*, pages 39–48. (page 21)

[173] Wilburn, B. S., Smulski, M., Lee, H.-H. K., and Horowitz, M. A. (2001). Light field video camera. In *Media Processors 2002*, volume 4674, pages 29–37. International Society for Optics and Photonics. (page 26)

[174] Wilson, A. D. and Benko, H. (2010). Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology - UIST '10*, page 273. (page 26)

[175] Wilson, R. H. and Latombe, J. C. (1994). Geometric reasoning about mechanical assembly. In *Artificial Intelligence*, volume 71, pages 371–396. (page 30, 35)

[176] Wong, A., Marcus, N., Ayres, P., Smith, L., Cooper, G. A., Paas, F., and Sweller, J. (2009). Instructional animations can be superior to statics when learning human motor skills. *Computers in Human Behavior*, 25(2):339–347. (page 23)

[177] Wu, J., Zhou, B., Russell, R., Kee, V., Wagner, S., Hebert, M., Torralba, A., and Johnson, D. M. (2018). Real-time object pose estimation with pose interpreter networks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6798–6805. IEEE. (page 15)

[178] Wu, L.-C., Lin, I., Tsai, M.-H., and Others (2016). Augmented reality instruction for object assembly based on markerless tracking. In *Proceedings of the 20$^{th}$ ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 95–102. ACM. (page 16)

[179] Yang, X.-D., Mak, E., McCallum, D., Irani, P., Cao, X., and Izadi, S. (2010). LensMouse: Augmenting the Mouse with an Interactive Touch Display. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2431–2440. (page 116)

[180] Young, J., Langlotz, T., Cook, M., Mills, S., and Regenbrecht, H. (2019). Immersive Telepresence and Remote Collaboration using Mobile and Wearable Devices. *IEEE Transactions on Visualization and Computer Graphics*, page 1. (page xix, 5, 18, 19, 61)

[181] Young, T. S., Teather, R. J., and Mackenzie, I. S. (2017). An arm-mounted inertial controller for 6DOF input: Design and evaluation. *2017 IEEE Symposium on 3D User Interfaces, 3DUI 2017 - Proceedings*, pages 26–35. (page 105)

[182] Zauner, J., Haller, M., Brandl, A., and Hartman, W. (2003). Authoring of a mixed reality assembly instructor for hierarchical structures. In *Proceedings - 2$^{nd}$ IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR 2003*, pages 237–246. (page 15, 21)

[183] Zheng, Y., Chen, X., Cheng, M.-M., Zhou, K., Hu, S.-M., and Mitra, N. J. (2012). Interactive images. *ACM Transactions on Graphics*, 31(4):1–11. (page 22)

[184] Zheng, Y., Kuang, Y., Sugimoto, S., and Åstr, K. (2013). Supplementary Materials for Revisiting the P n P Problem : A Fast , General and Optimal Solution. In *ICCV*, number 13, pages 2344–2351. (page 31)