Context-Aware Adaptive User Interfaces for Mixed Reality

João Marcelo Evangelista Belo

PhD Dissertation

Department of Computer Science
Aarhus University
Denmark

Context-Aware Adaptive User Interfaces for Mixed Reality

A Dissertation
Presented to the Faculty of Natural Sciences
of Aarhus University
in Partial Fulfillment of the Requirements
for the PhD Degree

by
João Marcelo Evangelista Belo
March 31, 2023

Abstract

Mixed Reality (MR) is a promising interaction paradigm with a long history. Back in the 60s, Ivan Sutherland developed the Sword of Damocles, considered the first head-mounted display. Since then, visions of the potential of MR to enable super-human cognition, memory, and sensing have been presented. However, looking at where we stand today, we are still far from the vision of truly compelling MR experiences. There are several obstacles to overcome in terms of hardware, and interactions and interfaces are still relatively underexplored compared to established interaction paradigms, such as the personal computer and the smartphone. A major challenge for designing MR adaptive user interfaces (UIs) is that, in contrast to UIs that live in virtual environments, they co-exist in the user's environment and are highly context-dependent. As the user's context changes over time, such as their environment and posture, MR UIs must adapt to these changes.

This dissertation presents a comprehensive approach for creating context-aware adaptive MR UIs to address this challenge. The journey starts with an exploration of context collection and understanding, where I contribute methods for context collection that combine multiple sensors and a pipeline to simplify existing context collection procedures. Because access to context is not always sufficient to make appropriate adaptation decisions, algorithms and models can play an important role in processing context for informing UI adaptations. On this front, this dissertation contributes a model for ergonomics of the upper limbs, which is used for adapting MR UIs at runtime. Additionally, I present a multi-objective optimization approach for MR adaptations with customizable behavior. To further facilitate the development process of Adaptive UIs, this dissertation proposes a framework to provide a separation of concerns for adaptive UIs, instantiated through a toolkit. This toolkit belongs to a range of tools I propose in this dissertation to empower creators in developing context-aware adaptive UIs, which I hope will inspire others to advance the landscape of tools and methods available in the field.

Resumé

Mixed Reality (MR) er et lovende paradigme for interaktion med en lang historie. Tilbage i 60'erne udviklede Ivan Sutherland *Sword of Damocles*, som tit betragtes som den første hovedmonterede skærm. Siden da er der blevet fremlagt flere visioner om MR's potentiale til at muliggøre overmenneskelig kognition, hukommelse og sansning. Desværre er vi stadig langt fra visionen om virkelig indlevende MR-oplevelser. Der er ikke kun adskillige hindringer, der skal overvindes med hensyn til hardware, men interaktioner og grænseflader er stadig relativt underudforsket sammenlignet med etablerede interaktionsparadigmer som f.eks. pc'er og smartphones. En stor udfordring i forbindelse med udformningen af MR-adaptive *user interfaces* (UI) er, at de i modsætning til UI der lever i virtuelle miljøer, eksisterer i brugerens omgivelser og er derfor meget kontekstafhængige. Da brugerens kontekst ændrer sig over tid, f.eks. i form af omgivelser og kropsholdning, skal MR-UIs tilpasse sig disse ændringer.

For at løse denne udfordring præsenterer denne afhandling en omfattende tilgang til at skabe kontekstbevidste adaptive MR-UIs. Afhandlingen starter med en udforskning af kontekstindsamling, hvor jeg bidrager med metoder til kontekstindsamling, der kombinerer flere sensorer og en *pipeline* for at forenkle eksisterende kontekstindsamlingsprocedurer. Da adgang til kontekst ikke altid er tilstrækkelig til at træffe passende tilpasningsbeslutninger, kan algoritmer og modeller spille en vigtig rolle i forbindelse med behandling af kontekst for at informere om UI-adaptationer. På denne front bidrager denne afhandling med en model for ergonomi af de øvre lemmer, som bruges til løbende at tilpasse MR-UIs. Derudover præsenteres en multiobjektiv optimeringstilgang til MR-adaptioner med tilpasselig adfærd. For yderligere at lette udviklingsprocessen af adaptive UIs foreslår denne afhandling en metode til at kombinere flere adaptive UIs igennem et *toolkit*. Dette *toolkit* hører til en række værktøjer, som jeg foreslår i denne afhandling for at give skaberne mulighed for at udvikle kontekstbevidste adaptive UIs, som jeg håber vil inspirere andre til at fremme landskabet af værktøjer og metoder, der er tilgængelige på området.

Acknowledgments

I would like to start by thanking my supervisor, Kaj Grønbæk. Kaj convinced and motivated me to start my PhD, and was always supportive from the very beginning. I think Kaj is probably the busiest person I know, but somehow he always found the time to meet, discuss research, or read manuscripts. He backed my craziest ideas, helped me acquire all types of hardware, and supported endeavors such as summer schools, conferences, and an internship in the US when I had already gone on a stay abroad in Finland. Thank you, Kaj, for believing in me and making all these amazing years and experiences a possibility.

Then, I would like to thank my co-authors. Andreas Fender and Tiare Feuchtner, for helping me get started with everything HCI-related and for introducing me to MR. Thanks, Anna Maria Feit, for believing in me when I had so little to show - discussing research and working with you will always be an honor. I would also like to thank Jon Wissing, Mathias Lystbæk, Ken Pfeuffer, Peter Kán, and Antti Oulasvirta for all the discussions, coding, and writing. Finally, would like to thank and mention how much I appreciate discussing new ideas and collaborating on new projects with Christoph Johns. All of you helped make this dissertation a possibility, and I hope we will have the chance to work together again in the future!

Next, I thank Antti Oulasvirta again for hosting me in the User Interfaces research group at Aalto University. It was a unique opportunity to learn and meet a group full of talented people. Thank you, Yi-Chi Liao, Joongi Shin, Chieh-Ling Shih, Lena Hegemann, Aini Putkonen, Danqing Shi, Aurélien Nioche, Michael Hedderich, and all the others who made this a great experience.

Then, a big thanks to Kash Todi for suggesting the application for the internship at Reality Labs and for being a great manager, colleague, and friend. It was an amazing experience exceeding all my expectations - so many brilliant researchers working together in the same institution! A big thanks to all the other interns who helped me have a great time these five months in the US: Yi-Chi Liao, Alex Ogren, Xuhai "Orson" Xu, Anandghan Waghmare, Feiyu Lu, Matthias De Lange, Rishi Hazra, Naghmeh Zamani, David Bethge, Rishabh Patni, and Xun Qian. Shout out to all the other awesome people on the team!

I could not have finished this PhD without the support from the Ubi group. Thank you all for the casual chats in the kitchen, lunches together, events, and visits to the Friday bar. Special thanks to Andreas Fender, Tiare Feuchtner, Marius Hogräfer, Jens Emil Grønbæk, Wenkai Han, Mathias Lystbæk, Andreas Mathisen, and Troels

Rasmussen for everything throughout these years. Would also like to mention how nice it is to discuss XR topics with Ken Pfeuffer and his PhD students Uta, Mathias, Pavel, and Christoph - really hope you folks keep it going! I'd also like to thank my other friends in Aarhus, always up for a board game - Kevin, Hart, Per, Peter, Morten, Thalles, Bart, Riccardo, and Finn.

Last, I want to thank my family for their unconditional support over the years. My mother, Natércia, who is always asking when I will go back to Portugal. My father, Paulo, who always wants to share the latest projects he is working on. My siblings, Alice and Luís, who are always here to help when something is necessary. I will always love you. Finally, I want to thank my partner Aïna Linn Georges. You made this journey possible - thank you so much for being here for everything. I love you!

João Marcelo Evangelista Belo,
Aarhus, March 31, 2023.

Publication list

This dissertation is comprised of the following publications and manuscripts:

Paper A [17] João Marcelo Evangelista Belo, Andreas Fender, Tiare Feuchtner, and Kaj Grønbæk.
Digital assistance for quality assurance: Augmenting workspaces using deep learning for tracking near-symmetrical objects.
In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces*.
Video: https://www.youtube.com/watch?v=WwD95sD_WM0
This paper won a best application paper award

Paper B [21] João Marcelo Evangelista Belo, Jon Wissing, Tiare Feuchtner, Kaj Grønbæk.
CADTrack: Instructions and Support for Orientation Disambiguation of Near-Symmetrical Objects.
In submission.
Video: https://www.youtube.com/watch?v=edRBfI5JGaM

Paper C [73] João Marcelo Evangelista Belo, Anna Maria Feit, Tiare Feuchtner, Kaj Grønbæk.
XRgonomics: Facilitating the Creation of Ergonomic 3D Interfaces.
In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*.
Video: https://www.youtube.com/watch?v=7-0uJoHRVJE
Code: https://github.com/joaobelo92/xrgonomics
This paper won a best paper award

Paper D [19] João Marcelo Evangelista Belo, Mathias N. Lystbæk, Anna Maria Feit, Ken Pfeuffer, Peter Kán, Antti Oulasvirta, Kaj Grønbæk.
AUIT - the adaptive user interfaces toolkit for designing XR applications.
In *The 35th Annual ACM Symposium on User Interface Software and Technology*
Video: https://www.youtube.com/watch?v=42ikMT6Ij6w
Code: https://github.com/joaobelo92/auit

Additionally, I published or have the following contributions in submission. How-

ever, these are not included in this dissertation:

Paper E [279] Xuhai Xu, Mengjie Yu, Tanya R. Jonker, Kashyap Todi, Feiyu Lu, Xun Qian, João Marcelo Evangelista Belo, Tianyi Wang, Michelle Li, Aran Mun, Te-Yen Wu, Junxiao Shen, Ting Zhang, Narine Kokhlikyan, Fulton Wang, Paul Sorenson, Sophie Kahyun Kim, Hrvoje Benko.
XAIR: A Framework of Explainable AI in Augmented Reality.
In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*

Paper F [20] João Marcelo Evangelista Belo, Felix Izarra, Xuhai Xu, Tanya R. Jonker, Kashyap Todi.
Designing and Optimizing Adaptive Shortcuts for Extended Reality.
In submission

I have also published the following non-archival contributions:

Paper G [18] João Marcelo Evangelista Belo, Tiare Feuchtner, Chiwoong Hwang, Rasmus Lunding, Mathias Lystbæk, Ken Pfeuffer, Troels Rasmussen.
Challenges of XR Transitional Interfaces in Industry 4.0
ISS '21 Workshop: Transitional Interfaces in Mixed and Cross-Reality: A new frontier?

Paper H [136] Christoph Albert Johns, João Marcelo Evangelista Belo, Ken Pfeuffer, Clemens Nylandsted Klokmose.
Pareto Optimal Layouts for Adaptive Mixed Reality
In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*

Contents

Part I

Overview

Chapter 1

Introduction

Mixed-Reality (MR) is a human-computer interaction paradigm with a long history. Pioneering work on head-mounted displays dates back to as early as 1965 [248]. In 1992, Caudell and Mizell coined the term Augmented Reality (AR) and presented a prototype to demonstrate its benefits in manufacturing [45]. Two years later, Milgram and Kishino defined the concept of MR [181] as a spectrum of immersive environments where AR is an intermediate point. Azuma proposed another popular definition of AR, where three characteristics identify AR experiences: 1) combining the real and virtual world, 2) interactive in real-time, and 3) spatial registration. Nowadays, there is still no universally agreed upon definition of what MR is [244] - in this thesis, I refer to MR as the interaction paradigm where virtual and real environments are seamlessly blended, and AR as a variation leaning towards the physical world in Milgram and Kishino's continuum. As Virtual Reality immerses the user in a virtual environment, it is not considered a form of MR. Therefore, to encompass the whole range of immersive technologies (AR, MR, and VR), I use the umbrella term eXtended Reality (XR).

The inspiring work discussed so far led researchers to explore a variety of compelling cases, such as situated holograms and interaction with virtual content in the real world. However, the vision of pervasive Mixed Reality that people use throughout the day is still distant. While interest and resources poured into the field increased substantially over the last decade, technological advancements concerning foundational technology such as displays, batteries, and computing are necessary to achieve Mixed Reality's true potential [1]. Other crucial requirements often overlooked are the interface and interaction breakthroughs associated with novel display form factors - for Mixed Reality to take over as the next computing era, we must find the ideal interfaces and interactions.

In contrast to established computing paradigms such as the personal computer or the smartphone, Mixed Reality interfaces blend with the user's environment. Dimensions and availability of the screen in personal computers and smartphones stay the same at runtime, making it easier for creators to foresee the conditions in which systems will run when used. However, the same is not true for Mixed Reality applications, which are by definition context-dependent. It is difficult to predict the environment

where the system will run and how it changes over time. Consider an MR application

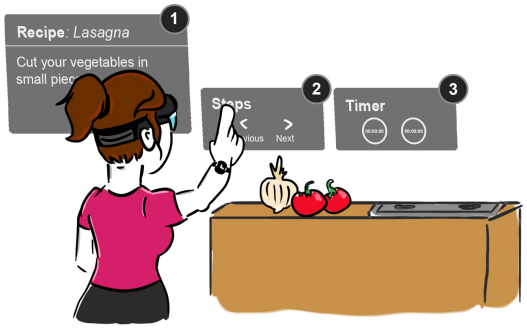


Figure 1.1: MR UIs are context sensitive, and must adapt to a plethora of factors to be usable, such as the use's environment, posture, and action.

to guide users through a recipe - kitchens are different across households, and as the user moves ingredients and appliances, the interface must adapt accordingly. In the most basic interface type, a panel of instructions will not be readable when outside the user's field of view or occluded by other objects (Figure 1.1). Now consider instructions blended in the user's environment, such as a visualization for cutting ingredients in a particular technique or an animation of what ingredients to put into the pot. In this case, the system must understand various contextual information, such as the environment geometry, the objects around it, and the recipe stage the user is on. Furthermore, the system must also be able to assess what adaptations are suitable, how, and when these are applied. This example highlights how context-sensitive MR applications are because they augment the real world, and interfaces in this interaction paradigm must adapt to context during runtime to be usable.

The first evident requirement to enable such context-aware adaptive user interfaces is that systems must be able to acquire and represent context. In the recipe example where virtual instructions blend with the real world, environment understanding is crucial to generate such instructions. The system's usability can improve further using other sources of context - for example, considering aspects such as the user's pose to provide ergonomic interactions or assessing the user's cognitive load to adjust the level of detail of instructions. MR systems must also represent context in a format that creators can easily use. For example, it is necessary to process raw sensor data, such as color or depth images from a camera, into data in higher abstraction levels, such as information about the environment's geometry, what objects are in the real world, or the user's pose. Context acquisition and representation is the first of the three topics investigated in this thesis.

Although context plays a central role in adaptive MR user interfaces, representing some context categories in a high level of abstraction is not always sufficient for a computer to decide on the best design over a myriad of possibilities. Consider an

algorithm to compute which arm poses allow hand interaction with virtual interfaces in 3D space from the user's current position. If the algorithm cannot evaluate which poses are ergonomically better than others or make an assessment based on other human factors, its utility is limited. This brings us to the next requirement to enable usable 3D user interfaces discussed in this dissertation: computational interaction methods. Computational interaction methods allow computers to formally represent a design space, understand it, and identify desirable solutions [198]. Algorithms capable of comparing the quality of different solutions facilitate or make it straightforward to pick across the many possible designs often encountered in large design spaces common in Mixed Reality applications.

Context and its understanding are critical for enabling adaptive user interfaces, but there is still a gap in using context and computation interaction methods to adapt interfaces in MR applications. It is essential to make these approaches accessible in tools to create MR applications so creators can use these techniques in UI adaptations. Adapting interfaces can be more intricate than it seems - algorithms can have contradictory objectives that might conflict with one another - for example, ergonomic positions for hand input are typically poor for visibility. These difficulties bring us to the final requirement for enabling usable MR user interfaces discussed in this thesis: tools and methods to support development. Support for developers to implement complex context-aware adaptations is crucial to facilitate the development of MR applications.

There is another important human-computer interaction aspect not discussed yet: how to interact, or in other words, what input methods will allow MR experiences to be something people would like to use daily. While it is unclear which input methods will succeed, there is an opportunity to adopt the ones that provide immediate usability, such as direct manipulation. Existing devices support different combinations of hand tracking, speech recognition, and eye tracking out of the box. Such input methods have advantages over traditional controllers for all-day MR, as users would not need to carry any additional hardware to use the system. Moreover, humans already know how to use their hands, eyes, and speech - using these to interact removes the necessity for learning new controls and allows developers to use interaction metaphors with virtual content similar to how people manipulate objects in the real world. While exploring which input methods and interaction techniques are the most adequate is not the goal of this dissertation, input is inherently related to adaptive user interfaces. Examples and prototypes presented in this dissertation mostly use hand input due to these advantages for all-day MR.

Human-Computer Interaction (HCI) researchers have explored various methods to address the requirements and technical challenges discussed. Context and its importance is an active topic of research since the early 90s - a widely accepted definition was proposed by Dey and Abowd in 2000, where context is "(...) any information that can be used to characterize the situation of an entity", and categorized context types relevant for systems at the time. Since then, HCI researchers have explored various techniques to capture, represent, and use context. More recently, Grubert et al. [101] classified context sources and targets relevant to all-day MR.

Meanwhile, other fields revolutionized what is possible to achieve nowadays in terms of context representation - breakthroughs in Artificial Intelligence (AI) made it possible for computers to quickly understand what objects are in a room, their pose, and the room geometry. These advancements allow us to understand the user's environment and other relevant context sources of interest, such as human poses (including their arms), emotions, and tasks. This dissertation proposes methods to capture and represent context sources based on specific needs and how their usage can be streamlined for creators (addressed in papers A and B).

Computational interaction methods to improve human-computer interaction are also an active research area getting more attention over the last decade. Computational interaction, defined by Oulasvirta et al. as "(...) the use of algorithms and mathematical models to explain and enhance interaction", has been used to solve a variety of problems, such as user interface management and interface design. These approaches played an important role in the success of touchscreen-based devices, such as smartphones, through keyboards that use statistical models to improve text entry. Computational interaction methods can improve the efficiency and usability of MR interfaces, and prior research has explored this direction. In this thesis, I propose novel computational methods that use optimization techniques to improve the ergonomics of mid-air interaction (paper C) and adapt interfaces in real-time (paper D).

Finally, building upon work that explores the development of context-aware systems and adaptive user interfaces in MR, this dissertation proposes a separation of concerns of design concepts in an adaptation and a toolkit that instantiates such a framework to facilitate the creation of MR adaptive user interfaces (paper D).

To summarize, this thesis complements existing research on adaptive user interfaces for MR. Starting by considering the context-sensitiveness of experiences that blend virtual and real worlds, it explores context retrieval and representation, how computers can understand context, and what it takes to use this information to adapt MR user interfaces. These concepts are then consolidated into a toolkit to make the creation of MR adaptations more accessible and flexible to creators.

1.1 Mixed Reality: Past, Present, and Future

The pioneering work of Caudell and Mizzel demonstrated in 1992 how MR could support the manufacturing industry with co-located instructions [45]. Later, others proposed systems where MR enables workers to perform various tasks such as maintenance [118, 234], inspection [211, 267], and assembly [251]. Training is another topic of interest for the manufacturing industry where MR can enhance existing approaches [266], but the technology has also shown great potential for education in general [25]. Researchers have explored MR applications in other fields, such as medicine [86, 274], navigation [75, 129], and telepresence [171].

As shown, researchers have been demonstrating the utility of MR systems over the last 30 years in a variety of applications. During these years, we have gone through many technological advancements in tracking, displays, batteries, interfaces,

computing, and the form factor of head-mounted displays (HMDs). However, there are few actively used MR applications nowadays. Most related consumer devices focus on VR experiences, where entertainment, such as VR games, has been the main driver for adoption. A likely reason for this is that VR presents fewer challenges to proportionate high-fidelity experiences. Virtual environments are easier to control and are less context-dependent than MR applications. Furthermore, displays for VR experiences do not have to bother with the real world. See-through displays for MR still have limited Field-of-View and color accuracy (e.g., HoloLens 2), while video pass-through displays require a high-quality capture of the real world. The ecosystem might be changing soon, however. Companies have been pushing hybrid devices with more MR capabilities, such as the Quest Pro and the Pico 4 - VR headsets with video pass-through for MR experiences. At the same time, we are witnessing a surge in productivity and telepresence applications (e.g., Mesh and Horizon Workrooms) and investments in researching technology to enable visions of all-day MR from big technology companies such as Microsoft, Apple, Google, and Meta.

So far, all-day MR remains a futuristic vision of what could be the next general computing platform. What would it take for MR to take over? As Michael Abrash puts it, there are several technologies where innovations and developments are necessary [1]. Another perspective from Hrvoje Benko is that the adoption of new computing eras was always greatly influenced by the hardware available at the time and by breakthroughs in interfaces, input methods, and interaction techniques. Looking back at the history of computing, the mouse and WIMP interfaces played a key role in the shift from command-line interfaces to graphical user interfaces. Similarly, touch and gesture-based interfaces enabled the success of tablets and smartphones [22]. The interfaces and input methods that could enable all-day MR experiences have yet to be discovered - perhaps the lack thereof is the missing piece for truly compelling MR experiences.

1.2 Thesis Goal and Research Questions

As discussed so far, the success of MR depends on many factors affecting the quality of MR experiences. The research in this thesis is motivated by the interest in one of these - user interfaces and interactions. Existing literature has demonstrated how interfaces in MR must consider diverse context categories to provide good usability. This dissertation takes a holistic view of all the considerations necessary to provide context-aware adaptations, from context capture and representation to tools that abstract relevant concepts to facilitate their creation, exploring three requirements to enable context-aware adaptive user interfaces in MR (some presented previously by Benko [22]):

1. Context acquisition and representation
2. Computational interaction
3. Methods and tools to create adaptive UIs

Challenges related to each of these requirements are investigated in this thesis, raising the following research problems [197]:

- **RP1:** How to make relevant context categories available to creators during the development of adaptive MR UIs?
- **RP2:** How can computational interaction methods provide a further understanding of context and optimize MR UIs?
- **RP3:** What are the components of a context-aware MR UI adaptation?
- **RP4:** How to support creators in applying computational methods and facilitate the creation of context-aware adaptive user interfaces?

I address each research problem in this thesis through a combination of the following: 1) reviewing related work and existing practices, considering established interaction paradigms and MR research, 2) exploration and proposal of optimization techniques or algorithms to solve a problem, 3) development of systems to address a specific problem or demonstrate how to overcome existing technical challenges, 4) implementation of toolkits to support the development of adaptive user interfaces, 5) technical evaluations in terms of performance and empirical evaluations with potential users of the system/toolkit.

Some of these research problems were investigated to a large extent in existing literature when considering other interaction paradigms - related work has explored context-awareness, computational methods, and tools and abstractions to support development. Although there is research exploring these topics in an MR setting, it is a relatively new topic. In the last decade, there have been many hardware improvements, and companies present new XR headsets with more capabilities every year. At the same time, the technology is finally reaching consumers - many households have access to VR headsets, and smartphones have several MR capabilities. It is an exciting time to research the field, and I hope you are excited too to learn how this dissertation attempts to push the boundaries of what adaptive interfaces can do in MR and how to facilitate their development.

1.3 Structure of Part 1

Followed by this introduction chapter, chapter 2 will provide an in-depth overview of related work exploring topics crucial to enable MR adaptive user interfaces, starting with an overview of context-awareness literature and existing methods to capture and represent context. Then, I present existing computational interaction work, such as optimization methods and algorithms, followed by computational methods to tackle MR-related problems. The chapter ends with an outline of existing adaptive UIs in MR systems and tools to support their development. Chapters 3, 4, and 5 discuss the contributions of this thesis and where each publication sits in the big picture concerning the requirements to enable adaptive UIs highlighted earlier. Chapter 3

looks into the challenges related to context capture and representation, proposing methods to overcome specific challenges to represent context and facilitating its usage in adaptations. Chapter 4 introduces novel computational methods to understand aspects of interaction in MR, such as ergonomics. I also present the multi-objective optimization problem formulation for adapting MR UIs. Chapter 5 starts with a holistic view of the concepts MR adaptations comprise. Chapter 6 discusses the challenges for creators during the development of MR interfaces and how tools can support this process. Chapter 7 provides an overview of my research methodology and details the experimental procedures used in this dissertation. Finally, chapter 8 summarizes the contributions of this thesis and takes a glimpse into the various possible directions for future work in this exciting research topic.

Chapter 2

Background

All the work presented in this dissertation builds upon extensive research in the intersection of Human-Computer Interaction and Artificial Intelligence. This chapter is structured over three topics that build upon one another when considering MR adaptations: context-awareness (section 2.1), computational interaction (section 2.2), and adaptive user interfaces (section 2.3). Due to obvious overlap, parts of this section have similarities to related work sections in the publications from this thesis.

2.1 Context-awareness

Initial attempts to define what context and context-awareness are date back to 1994, when Schilit and Theimer define context-aware computing as the ability of a system to understand and react to its environment (objects of interest nearby, people, and location) [232]. Since then, others have refined this definition, extending it to include other categories such as the time of the day, temperature, user's identity, and emotional state [37, 227]. Situated computing explored the ability of systems to sense the local environment, interpret sensor data, and use this data to improve applications [128]. Dey and Abowd proposed perhaps the most established definition for context-awareness as of today [60], "A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task" and note that some context-categories are more relevant than others, such as location, identity, activity, and time. Others have researched the topic extensively since. [13, 24, 46, 65, 67, 207, 233]. More recently, Grubert et al. proposed the concept of pervasive augmented reality (AR) [101], which uses context-awareness to adapt the AR system based on changing requirements and constraints. The authors also propose a taxonomy to classify different context sources into three high-level categories: human, environmental, and system factors. The authors also explore how context-awareness can adapt the output of context-aware AR systems, such as content, which can be shown or filtered depending on context, while making it possible to adjust how much information to display. Information presentation is another relevant output category where the interface can adapt to context.

One of the first requirements to create context-aware applications is context capture and representation [59] because access to context data is crucial to develop them. Fortunately, it is a research field that went through several breakthroughs since the '90s for two reasons: hardware such as sensors has improved substantially over the years, and there have been substantial developments in the field of artificial intelligence that allow computers to process these sources of data into abstraction levels that are more accurate and accessible to developers. I will now cover the most relevant advances related to the context sources explored in this thesis: image recognition to detect objects and their poses and models to assess the ergonomics of the upper limbs.

Image Recognition

In the late '80s, LeCun et al. [152] used a convolutional neural network and gradient-based learning to classify images of digits. Two decades later, the approach gained popularity again when Krizhevsky [146] et al. used it on a much larger scale to develop a CNN capable of classifying images in the ImageNet dataset [55] with much better performance than existing techniques at the time. Since then, CNNs benefited from several speed and accuracy improvements due to various architectural breakthroughs [113, 161, 240].

Breakthroughs in image classification inspired various new approaches for object detection. Detectors consisting of two stages, such as R-CNN [95], start by extracting candidate region proposals using traditional CV techniques and then classify these proposals using a CNN. Successive works improve the approach, making it faster [94] and more accurate [112]. Ren et al. [221] used a CNN to suggest potential object proposals and combined it with a second-stage classifier, resulting in real-time object recognition. Additionally, researchers explored other object recognition techniques for predicting objects in a single stage [162, 167, 217, 218, 237]. One-stage methods are faster but less precise in comparison to two-stage approaches.

Regression of object poses is another visual recognition task witnessing breakthroughs due to the advances of CNNs. Toshev et al. [259] proposed a model to regress the keypoints of humans in image space back in 2014. As in the object recognition task, researchers presented other CNNs with multiple stages and modules to make better 2D keypoint predictions [42, 191, 269]. Object detection, approaches such as Mask R-CNN [111] demonstrated how to successfully support additional capabilities, such as human 2D pose estimation, using another prediction branch. However, in MR scenarios, keypoint estimation in image space has lower utility than 6D pose estimation, which predicts the position in 3D and rotation of the object in the scene. Recently other works have been exploring CNN approaches for 6D pose estimation [206, 242, 242].

Ergonomics of the upper limbs

Ergonomics is a relevant factor in mid-air interaction, which is common in MR applications. Arm fatigue is such a well-known issue that fatigue in the upper limbs due to mid-air interaction has been coined as the *gorilla-arm effect* [34]. HCI research has employed a variety of methods to evaluate ergonomics. Perhaps the most common are the qualitative methods to assess subjective fatigue, such as the Likert scale [43], the NASA Task Load Index (NASA-TLX) [109], and the Borg CR10 scale [33]. While these questionnaires have the advantages of not being invasive and not requiring specialized hardware, they require substantial preparation and only provide a coarse estimation of fatigue. On the other hand, objective methods used in biology and sports science tend to rely on external measures that make them impractical for HCI studies, such as muscle activations [48], blood pressure [241], and heart rate [236].

To overcome the need for specialized and intrusive hardware, HCI researchers have proposed methods to quantify fatigue and other mid-air interaction considerations. Hincapié-Ramos et al. proposed Consumed Endurance, a metric that tracks the user's arm to quantify fatigue of mid-air interactions [125] by computing the arm's center of mass. Bachynskyi et al. successfully used simulations [9] that show how biomechanical simulations correlate with EMG data. Follow-up work used biomechanical simulations to create guidelines for interactions involving mid-air interactions [10]. Jang et al. modeled cumulative fatigue through a model that also estimates muscle states, such as rest, making it possible to model interaction and rest [135].

2.2 Computational Interaction

As described by Oulasvirta et al., "computational interaction (...) focuses on the use of algorithms and mathematical models to explain and enhance interaction"[198]. Capturing and representing sources of context does not necessarily mean the system can identify good designs with desirable properties or evaluate their design quality. For instance, algorithms to detect the pose of the user's arm do not provide data to support the creation of ergonomic user interfaces. There is extensive computational interaction work on various of topics, such as text entry, input recognition, and interface design. The following section describes optimization approaches used successfully for computational interaction. I review methods applied to problems related to UI design afterward.

Optimization methods

There are two categories of optimization methods: exact and heuristic methods. Exact methods guarantee all the optimal solutions to the problem. Such a characteristic is highly desirable, and this would typically be the choice of approach. However, once the size and difficulty of the problem increase, exact methods might require too much processing power or memory. Addressing this challenge is possible by using heuristic methods to find solutions with approaches that settle for a compromise. This

compromise could be to randomly search for solutions or use domain knowledge to exploit specific structures of the search space.

The simplest exact approach is an exhaustive search, which evaluates all the possible solutions. However, this approach is usually impractical since execution time increases exponentially with the dimensions of the problem. The simplex method, developed by Dantzig, is an efficient exact method capable of solving linear problems by reducing the feasible region of the problem to the simplest possible polytope. Interior point methods complemented the simplex algorithm years later, allowing the algorithm to execute in polynomial time (in contrast to exponential time). Branch-and-bound methods are another technique that recursively decomposes the main problem into sub-problems. It introduces constraints in the decision variables (branching) and discards some of the sub-problems if the quality of solutions is below a threshold (bounding). Dynamic programming and cutting plane methods are examples of other successful approaches (see Rothlauf's work [226] for an overview).

Unfortunately, there are many problems for which no efficient exact solvers are known. Moreover, optimization problems common in HCI often have computational and runtime constraints. This is where heuristic optimization methods that trade-off optimality, completeness, accuracy, or precision for speed are relevant. Such methods tend to be problem-specific because of the *no free lunch theorem* (NFL) [276], which says that when an algorithm performs better over a class of problems, it performs worse over others. Heuristic methods can be *construction methods*, that build a solution from scratch by performing iterative construction steps, or *local search methods*, that iteratively improve upon some initial solution. Genetic algorithms, simulated annealing, and tabu search are examples of heuristic optimization algorithms. When heuristics have provable guarantees on the solution quality, they are approximation algorithms. The methods discussed so far typically use problem-specific information. *Metaheuristics* are a higher-level procedure that makes fewer assumptions about the problem, making these usable for a wider variety of optimization problems.

Optimization in HCI

Approaches that use optimization for input recognition and user interface design have a long history (related works contain an in-depth overview [188, 198, 199]). Although optimization plays an important role when fitting machine learning models commonly used in input recognition, most of the optimization procedures at these stages are standardized and are similar to what machine learning literature applies. On the other hand, the problem formulation and optimization methods in interface design have several problem-specific considerations [198, sec. 4.2]:

- Representation of design decisions in the problem as decision variables;
- Definition of design heuristics and human factors in objective functions;
- Modeling of technical constraints of hardware and interfaces;

Pioneering studies applying combinatorial optimization to user interface design date back to 1977 and focused on keyboard layout optimization [38]. Later, other researchers improved on Burkard and Offerman's work through better objective functions and more evidence of the performance gains from optimized keyboard layouts over the established Qwerty layout [158, 286]. Since then, researchers have used combinatorial optimization in other domains related to interface design. Francis proposed a method to optimize the mapping of hierarchy labels to buttons to reduce search time [85]. Follow-up work explored optimizing menu designs to reduce user search time [165] and other factors, such as the consistency of grouped items [11, 99, 173]. Another area where optimization had promising results is interface layout design [88, 91, 235, 256], with some research focusing on concrete challenges such as accessibility [90]. Although the application is not necessarily interface design, O'Donovan et al.'s work on optimizing graphic designs proposed several notable methods to evaluate visual importance, alignment, and hierarchical segmentation [195, 201].

These works played an important role in achieving three milestones crucial for model-based interface optimization, identified by Oulasvirta and Karrenbauer [198, sec. 4.2]:

1. Formal definition of user interfaces as the object of optimization.
2. Objective functions that represent design heuristics and human factors.
3. Usage of predictive models and simulations of users.

2.3 MR User Interfaces

As motivated in the introduction, usable Mixed Reality interfaces must adapt to the user's context. Visibility is one of the basic requirements to fulfill - Höllerer et al. [130] underlined the importance of actively managing UIs in MR, proposing three interface techniques to do so: *information filtering* to select which information to show to the user, *UI component design* to adapt the format to present information to the user, and *view management techniques* to ensure virtual content is arranged appropriately in the user's view.

Pioneering work on information filtering for MR dates back to 1993 when Feiner et al. [74] used a rule-based approach to select which information to present in a printer maintenance task based on the user's position and orientation. Techniques to avoid cluttering were investigated later on, with the development of algorithms that consider the user's position, intent, and state of objects in the environment to adapt the displayed information [137]. Later, Tatzgern et al. investigated hierarchical clustering to create a level-of-detail structure using a cost and benefit metric that considers spatial displacement and semantic similarity [254].

In terms of UI component design, DiVerdi et al. proposed level of detail (LOD) interfaces [62], which allow applications to adapt the content displayed to the user

based on contextual information, such as the position of the user. More recently, Lindlbauer et al. [163] explored an online optimization approach to adapt which applications to display, their LOD, and where to position them according to the user's cognitive load.

Work from Bell et al. on view management techniques [16] was an important starting point on the topic in an MR context, where the position, size, and transparency of UI elements would adapt to prevent occlusion and place related objects next to each other. This approach starts by satisfying constraints and then sequentially determines the properties to adapt following an object priority order defined by the developer. The method considers the layout from previous frames to avoid visual discontinuities. Considering that virtual content in MR overlays the physical world, it is relevant to consider what the user sees to make decisions about the interface layout. Azuma and Furmanski explored how different view management algorithms fare in terms of usability [8], and Rosten et al. proposed an image-based view management technique that analyzed the video feed to look for the most suitable zones to render virtual content. Follow-up work from Grasset et al. [100] used visual saliency and edge analysis to inform layout decisions, identifying important image regions and geometric constraints for placing labels. The view management techniques discussed so far operated in 2D image space, while MR applications co-exist in 3D environments and render 3D virtual content. Tatzgern et al. proposed a method to place the labels in 3D space [253], bringing advantages in terms of consistency when the user's viewpoint changes. Ens et al. introduced a layout manager to achieve spatial constancy across environments [70]. Follow-up work explored temporal coherence further [170], while other studies revealed a preference for view management techniques with a limited update rate in contrast to continuous update strategies [254]. SemanticAdapt [47] is another optimization-based approach that leverages semantic connections between virtual and physical content to adapt the layout of MR applications.

MR Interaction Techniques and Input Devices

Along with UI layout management and presentation, interaction techniques and input devices are essential to support user interaction. Among the most common input devices in consumer-grade MR devices are 6 degrees of freedom (DoF) tracked controllers, typically bundled with VR and MR headsets. Researchers have proposed improvements beyond visual and vibration feedback, such as controllers that provide haptic feedback [23, 273, 284, 285]. However, such controllers must be hand-held and are impractical in MR scenarios involving hand work or all-day MR, requiring the user to grab the controllers to interact with virtual content. Alternatives that could reduce friction are natural input modalities that integrate the sensing hardware into the HMD. This thesis prioritizes a subset of such modalities: hand, gaze, and tangible input.

Early work exploring real-time hand input in MR contexts dates back to 2007 when Lee and Höllerer presented a method using an RGB camera for fingertip tracking that allowed users to inspect virtual objects using their hands [154]. Later on, Harrison

et al. proposed a method using the skin as an input surface, analyzing mechanical vibrations propagated through the body [107]. Other approaches explored depth and RGB cameras for hand-tracking and supporting multitouch applications on everyday surfaces [105, 108, 278]. Nowadays, advances in computer vision allow for high-accuracy real-time hand-tracking [93, 186, 238], and many HMDs reliably support hand input out of the box (e.g., Oculus Quest 2, the Hololens 2). Other interesting examples of related work have explored methods to provide passive haptics [7] and reduce arm fatigue [80].

Gaze interaction is another input modality showing promising results for XR interaction. Although early eye-tracking research for interaction dates back to the early 80s and 90s [29, 30, 133, 245, 265], it was in the 2000s that Tanriverdi and Jacob demonstrated its potential for VR [252]. Later on, Park et al. proposed a dwell-time gaze interaction technique that allowed users to select art in an MR gallery [203]. Ajanki et al. used gaze to infer the user's interest in the environment and determine how much information to provide [3]. McNamara and Kaberdoss proposed a management system to arrange AR labels based on user attention [175], a topic explored later in complex virtual environments [176]. SmoothMoves correlated head movements to moving MR content for interacting with smart home devices [71]. Recent works provide a detailed comparison of methods for precise target selection in MR [72, 147].

Mark Weiser's vision of ubiquitous computing [270] where "technologies (...) disappear and weave themselves into the fabric of everyday life" inspired many of today's computational devices and ideas. Notably, Ishii and Ullmer proposed tangibles, a vision that couples everyday objects and architectural surfaces with digital content [132], enabled by three key concepts: 1) *interactive surfaces*, architectural spaces such as walls become interfaces between virtual and real words; 2) *coupling of bits and atoms*, by coupling everyday graspable objects with related digital information; 3) *ambient media*, the usage of ambient media (e.g., sound, light, airflow) for background interfaces. Tangibles relate to MR, where graspable UIs can support interaction with virtual content - Fitzmaurice et al. explored such a concept, using physical objects as input to manipulate coupled virtual objects [84]. A few years later, Billinghurst et al. used tangibles to interact and enhance collaboration in MR [26] and demonstrated how tangible UIs support seamless MR interaction [27]. However, these works relied on interaction with fiducial markers instead of everyday objects already present in the physical world, as envisioned by Ishii and Ulmer. Henderson and Feiner overcome this limitation through Opportunistic Controls to couple affordances from the physical world with virtual buttons to provide passive haptics [117, 119]. Opportunistic Controls relied on developers to create physical-digital pairs - a limitation addressed by Hettiarachchi and Wigdor in Annexing Reality [122], a system to annex virtual to physical objects. More recently, Gupta et al. explored how digital experiences can become more physical using MR to overlay physical objects [103]. For example, users could browse a collection of digital photos overlaid on a physical album.

2.4 Development of MR Interfaces

While MR devices are improving and getting more accessible, development remains challenging for creators. Among the eight key barriers in authoring MR/VR applications identified by Ashtari et al. [6], three of them relate to UI development: 1) lack of concrete design guidelines and examples; 2) difficult to design for the physical aspect of immersive experiences; and 3) difficult to plan and simulate motion in AR. This is where user interface tools and computational support can shine, as noted by Myers et al. back in 2000[188], supporting creators through the design process. Researchers have proposed approaches to provide computational support throughout the design of applications in established UI paradigms such as the PC [11, 256], but such tools for MR are still in their infancy. A recent example is the rule-based framework proposed by Krings et al. to support the development of context-aware MR applications and trigger adaptations based on context changes [144]. Existing commercial frameworks support UI adaptations in MR to some extent. Microsoft's Mixed Reality Toolkit (MRTK) [180] contains solvers [179], a component to adapt virtual content's position and orientation according to a predefined algorithm. Unity Mars [262] allows developers to have virtual content representing real-world objects and create rule-based adaptations depending on the geometry of the physical world.

Chapter 3

Context Awareness in Mixed Reality

As discussed in the introduction of this thesis, context-awareness is crucial to enable usable MR user interfaces. This chapter aims to advance the understanding of context-awareness in MR scenarios - during runtime and development. It starts with an overview of relevant context categories, followed by current practices and challenges for context acquisition and representation. Then, I discuss the approaches used for context retrieval and abstraction proposed in Papers A and B and how papers C and D use context to adapt MR UIs. Then, I conclude this chapter with a brief discussion on related topics.

3.1 Categories of Context for Context-Aware MR

As Dey and Abowd put it, "a categorization of context types will help designers uncover the most likely sources of context that will be useful in their applications" [60]. They also identify which context categories have higher relevance: location, identity, activity, and time. However, Dey and Abowd's work dates back to 1999, inspired by the era's technology and vision of ubiquitous computing. Although some of these categories are relevant in MR, the fact that content in this interaction paradigm blends with the physical world makes some context categories more important than others. An example is the geometry of the physical world, which constraints the positions virtual content can occupy. Grubert et al. explored context awareness for MR and identified three high-level categories [101]: human, environmental, and system factors. This taxonomy includes sub-categories crucial in MR applications, but they do not discuss their respective relevance. Although dependent on the scenario, some context categories have higher relevance for MR UI adaptations. Benko's categorization [22] identifies such high-level categories:

- Environment
- Task

- User's actions
- User's mental state

This categorization overlaps many of the context sources explored in this thesis. Regarding the environment, Papers A and B explore the pose and recognition of objects in detail, while the toolkit proposed in paper D supports UI adaptations that avoid occlusion and collisions with people and world geometry. In the task category, papers A and B propose systems using the task stage to provide step-by-step instructions, while the former investigates assembly and inspection in detail. For user traits and actions, paper C explores how to consider arm dimensions to find ergonomic positions for interaction. Both papers C and D showcase how the user pose can drive UI adaptations. Refer to figure 3.1 for an overview.

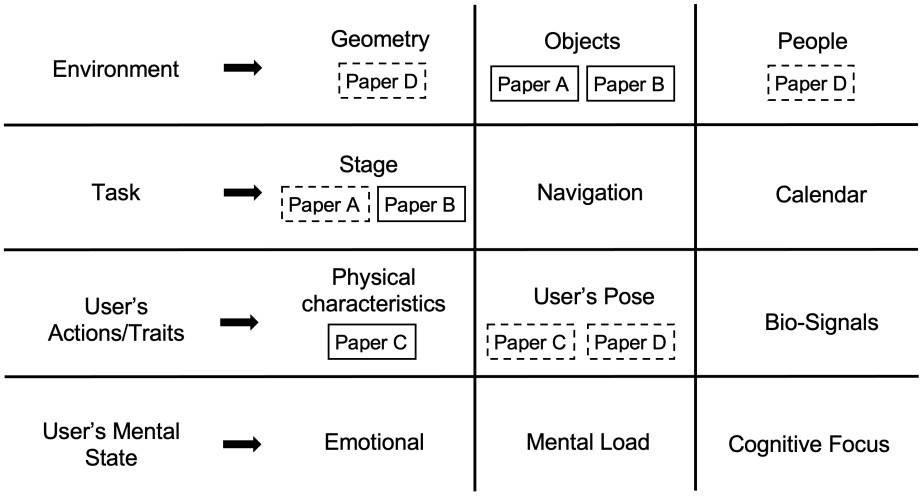


Figure 3.1: High-level context categories (left) and relevant examples of their sub-categories (right) adapted from Benko's categorization [22]. Papers in this thesis explore some of these categories in detail (solid line) or use them to some extent to adapt or demonstrate UI adaptations (dashed line).

3.2 Context Acquisition and Representation

For context to drive adaptations in MR, data that allows the system to interpret it into an adequate abstraction level is necessary. Data acquisition occurs using sensors (e.g., cameras, microphones, electrocardiography sensors) or by accessing data collected in advance (e.g., user's gender, calendar, or visual/movement impairments). However, most raw data sensors collect must be processed into a higher abstraction level before a system can use it. For example, while detecting objects in a picture is easy for humans, the same task is not trivial for a computer. From a computational perspective, the raw data from an RGB camera (a picture with colors) consists of a 3-dimensional matrix

where each position contains an integer ranging from 0 to 255. For an MR UI to adapt considering object semantics it is, therefore, necessary that the system processes the corresponding context information into an appropriate abstraction level using the sensors available. In their conceptual framework for context-aware applications, Dey and Abowd [59] identify this as one of the requirements for dealing with context. They propose context widgets, a category of components to address this requirement and highlight their corresponding benefits. Context widgets simplify context acquisition by *hiding the complexity* of how the system collects sensor data, *processing the data into suitable abstraction levels*, and providing developers with reusable and customizable building blocks to sense context.

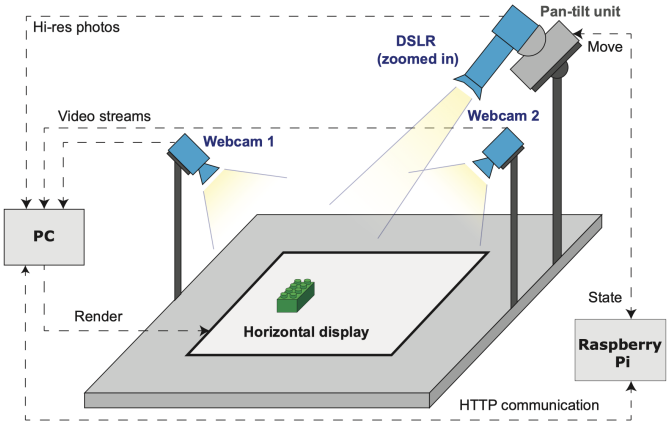


Figure 3.2: Setup for the tracking pipeline proposed in paper A.

Another proposed component in their framework related to context representation is the interpreters, which process context data from one or multiple widgets into higher representation levels, or in other words, into a more abstract or general format. For example, an interpreter can use video and sound to predict the current user task. In this case, an algorithm processes a video and an audio feed from pixels and wave amplitudes into a number with semantic meaning. The tracking pipeline we propose in paper A (see Figure 3.2) illustrates a similar scenario where it processes data from multiple sensors into a higher abstraction level - in this case, the orientation of an object.

While all context categories can be relevant for adapting MR UIs, some arguably play a bigger factor regarding usability. Because MR interfaces coexist in the physical world, understanding it is crucial to avoid usability breakdowns. For example, virtual interfaces colliding with objects in the environment raises coherence issues and is an obstacle to touch interaction. Designing for physical aspects of MR experiences, such as human motion, is another challenge in MR [6] since interaction occurs in 3D and tends to rely on motion (e.g., body and hand movement). All the papers in this thesis are related to the environment and user's pose context categories. Papers A and B explore approaches for tracking near-symmetrical objects. Paper C considers the

user's arm dimensions to identify positions for comfortable interaction. The toolkit proposed in Paper D considers context data such as the space geometry or the user's position and head orientation to adapt MR interfaces.

3.3 Abstracting Context Data

As motivated above, it is often necessary to process context data into higher representation levels to make it usable by MR systems. Data for abstracting context, such as the user's environment and pose, typically comes from sensors such as RGB or depth cameras. However, the raw data from such sensors is an n-dimensional matrix, depending on the camera used. In such cases, information about objects in the environment seen by the camera, such as their position in the 3D space or their semantics has more utility to a developer when creating an MR application than only having access to the video feed. The breakthroughs in computer vision achieved through deep learning discussed in section 2.1 brought improvements for bringing such data to higher abstraction levels. More concretely, identifying objects in a picture is a task that detectors can perform accurately and efficiently [287]. Other tasks where deep learning thrives are 6D pose estimation of objects [148, 206], humans [83, 157], and hands [93] - notably, the pose of these categories is highly relevant for user interfaces in MR systems, enabling hand tracking, tangible interaction, and interfaces anchored to objects in the real world. In papers A and B, we explore the challenge of abstracting raw data from RGB cameras into the pose of near-symmetrical objects.

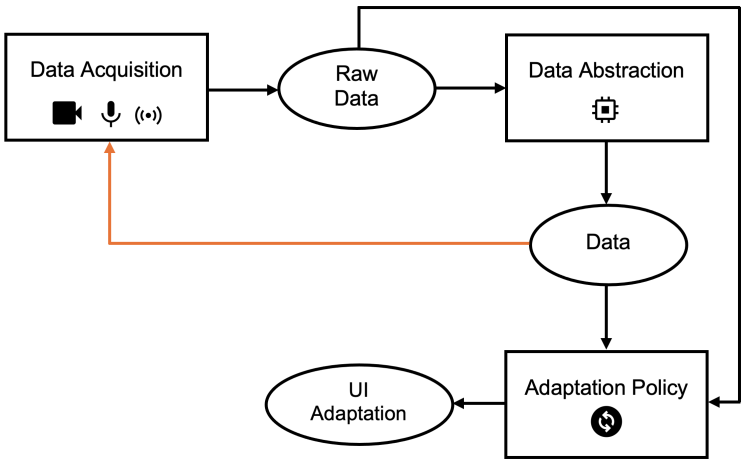


Figure 3.3: The flow of context data in context-aware applications from the moment it is acquired until it is used in a UI adaptation. Connections in black represent the conventional data flow. Connection in orange highlights where our approach in paper A differs from typical architectures.

The approach from paper A follows a slightly different flow from what is typically encountered in context-aware systems - in most cases, data is collected from sensors,

optionally processed into a higher abstraction level, and used in the application (see figure 3.3). In paper A, our method interprets data from webcams and uses it to guide

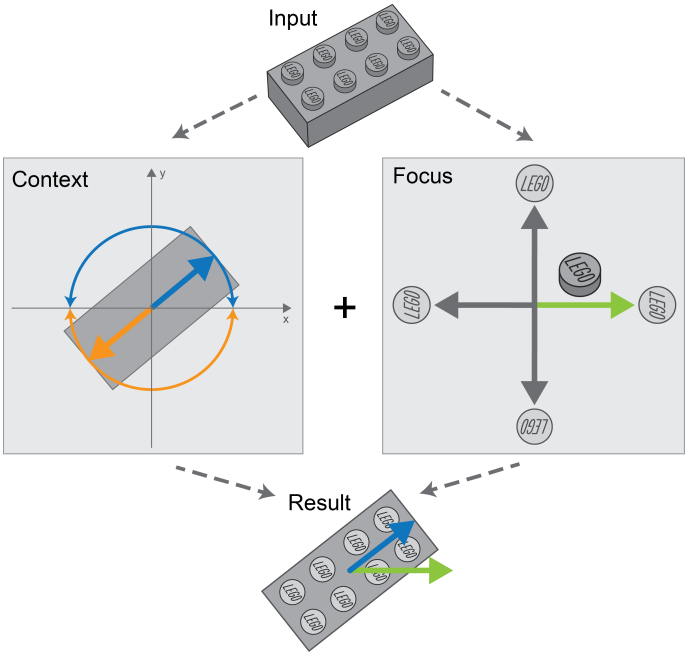


Figure 3.4: The focus+context tracking approach proposed in paper A is inspired by focus+context displays.

a DSLR camera, using an approach we refer to as focus+context tracking. From a high-level perspective, the system uses data abstracted from one sensor to guide another (in orange, figure 3.3) and combines both data streams to bring the data into a higher abstraction level. Once the DSLR camera sees the object, our proposed approach combines the data from all the sensors to make a final prediction of the object's position and orientation (see Figure 3.4). In paper B, we also contribute an approach to track near-symmetrical objects. However, in this work, we prioritize having a less complex setup and a faster vision pipeline, consisting of a single CNN.

Automated Data Abstraction

Most state-of-the-art context abstraction methods rely on deep learning-based approaches. These approaches typically use supervised learning, requiring considerable amounts of labeled data. For example, in paper A, a dataset of 100 labeled images was necessary to achieve high accuracy (98%) for detecting the orientation of a single object. This can be a barrier to creating context-aware applications. Consider the systems explored in papers A and B based on the real-world challenge of assisting LEGO metrologists in their measurement tasks. For such a system to support co-located instructions on all the 3700+ different LEGO elements someone would have to capture and annotate around 370000 images - a laborious and tedious task that is a

clear obstacle to creating such systems. In paper B we demonstrate a pipeline capable of automating this process when digital models of the object are available. Instead of capturing and annotating a dataset, the creator annotates the digital model of the object once. In our use case, the annotation consists of keypoints that allow a CNN to detect the pose of the object. We use these annotations to generate an annotated

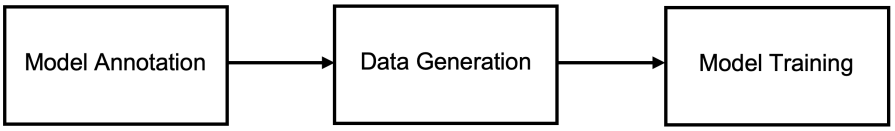


Figure 3.5: Pipeline demonstrated in paper B to avoid manual data labeling in data-driven approaches for data abstraction.

synthetic dataset, which our pipeline uses to train a CNN to abstract RGB pictures into the object's rotation (figure 3.5). This process's outcome is a model that developers can integrate into context-aware systems.

3.4 Using Context in UI Adaptations

Once context data is at an appropriate abstraction level, MR systems can use it to adapt UIs accordingly. Dey and Abowd identify a related component - *services*, responsible for executing actions depending on context. When considering adaptive MR user interfaces, there are many properties that a developer might be interested in for designing UI adaptations. The position of UIs in 3D space is a fundamental property crucial to determine the usability of MR UIs and the focus of this thesis. In papers A and B, co-located instructions adapt according to the position of objects. In paper C, we demonstrate how to use our approach to position UI elements more ergonomic to the user. Paper D allows creators to adapt UI elements' position, orientation, and scale according to a customizable combination of adaptation objectives. However, it is worth mentioning other UI properties that can improve usability further. Lindlbauer et al.[163] explore how the level of detail of MR UIs can adapt according to the user's cognitive load. Interaction modalities of UIs are another property that could adapt to the user's context to facilitate interaction, but as far as I know, the adaptation of modalities is unexplored. For example, a touch-based button could adapt its appearance to support gaze-based interaction from a distance.

So far, I presented examples of *what* properties can adapt based on context. However, using context data in UI adaptations is not always straightforward. While it is trivial to use the pose of objects in the real world for virtual instructions (papers A and B), contextual data such as the user's arm dimensions and pose are insufficient to adapt UIs to be more ergonomic. At the same time, multiple context categories and adaptation goals can dictate how a UI property should adapt, and such policies can conflict with one another. In the following chapter, I revisit these challenges, which we investigate in papers C and D.

3.5 Discussion

So far, I covered relevant context categories in MR, processes for capturing and abstracting context, and context usage in UI adaptations. Here, I briefly discuss other related topics not investigated in this dissertation.

Complex apparatus for context collection

In my work, I have focused either on using fixed setups tailored to gather specific contexts (e.g., paper A) or using sensors/context available through the hardware used to implement the MR systems we demonstrate. However, in many cases, other sensors available in the environment could contribute towards a better understanding of the user's context. Infrastructures to detect what context sources are available to the system and how these can provide context to applications is a relevant aspect of context collection not discussed so far. Dey and Abowd's framework [59] explores and proposes components to address these concerns (aggregators and discoverers). It is a complex topic with other implications (e.g., privacy), but intelligent usage of the resources available in one's environment can give access to context categories that would otherwise not be possible or make context collection more robust.

Explainable AI for context collection

Explainable AI is not a topic explored in this dissertation. However, when adaptive UI systems rely on AI-based approaches for context sensing, it is relevant to bring explainable AI capabilities to the XR system to inform the user when breakdowns occur. It is a topic we explore in paper E (not part of this dissertation) - this publication proposes a design framework to address when, what, and how to provide explanations in MR. Such explanations can also be relevant for AI and optimization methods used at other stages of an adaptation.

3.6 Summary

In this chapter, I highlighted relevant context categories for MR applications and presented an overview of the stages important from the moment systems capture raw context data to the moment they use it to drive UI adaptations. I presented the methods developed to collect and understand context proposed in papers A and B. Although the contributions in these papers retrieve a specific context category, these approaches showcase how to use similar strategies for context collection in MR applications. Paper A contributes a multi-sensor tracking approach for near-symmetrical objects and a technique to bring this sensor data into a higher abstraction level. In paper B we reduce the tracking zone to accommodate a simpler setup and contribute a pipeline that overcomes laborious data capture and labeling processes encountered in similar CNN-based vision systems. However, context data at a high abstraction level is not always sufficient for making an appropriate adaptation decision - or more ambitiously,

finding the optimal adaptation. Computational interaction methods can help achieve this goal, which I will explore in greater detail in the next chapter.

Chapter 4

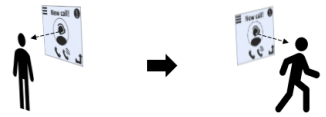
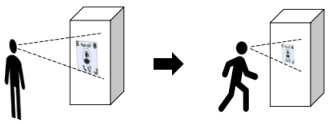
Computational Interaction Approaches

As highlighted in the previous chapter, processing context categories to a higher abstraction level is often insufficient for systems to make appropriate adaptation decisions. For example, knowledge of the environment geometry and user's pose does not inform the system of what positions are adequate for virtual content. Existing practices in MR application development to address the issue are rule-based adaptations. An example of such a rule would be a virtual menu that moves at a fixed offset from the user in their view frustum. However, as discussed in chapter 3, many context categories are important to consider in MR adaptations. Adapting a UI considering multiple contextual sources using rule-based approaches is difficult to scale, requiring considerable engineering efforts as highlighted in interviews conducted in paper D. MRTK [180], a popular open-source framework, contains components (solvers [179]) to facilitate the construction of such rule-based adaptation policies, where developers can combine multiple adaptation rules. These come with severe limitations, however. Rules are applied sequentially - if there is a conflict, the one applied last will disregard the others. For instance, imagine a scenario with rule A, which places the UI in the user's view frustum, and rule B, which prevents the UI from colliding with the environment geometry. When both rules run sequentially and independently, if rule B executes after rule A, there is no guarantee that rule B will move the UI in the user's frustum direction when avoiding a collision. This example highlights a common challenge creators face when developing MR interfaces - how to consider the wide range of relevant context categories and usability objectives common in MR scenarios. In this chapter, I address model-based methods to evaluate how adaptation objectives are met, techniques to find adaptation solutions that overcome the limitations of sequential rule-based approaches, and how I explore these topics in the publications from this dissertation.

4.1 Evaluating the Quality of UI Configurations

MR systems must be able to assess to which extent a solution meets different adaptation criteria to overcome the limitations of rule-based approaches. Along with an abstraction of the design space, cost functions to model adaptation objectives are an established method to do so [198] (in some literature referred to as objectives, loss functions, criteria) - a function that receives a proposal as input, in this case, a UI configuration, and outputs a numerical value. Typically the value increases in proportion to how much the input violates the criterion. While cost functions do not directly provide a solution, an adaptation policy can use them to assess the quality of multiple UI configurations and pick one that is optimal or contains the best trade-off. When considering multiple criteria, it is naturally more straightforward to implement one cost function per objective and combine these into a single cost function using weighted global criterion or weighted sum methods [172]. For example, consider an adaptation goal to maintain a UI element anchored at a fixed distance from the user's torso. In this case, the relevant context source is the position and orientation of the user's torso. A simplistic cost function to evaluate this goal would be to compute the distance of the UI element from this anchored position, where the output gets higher the further away the UI element is from the desired position. We include such a cost function in the toolkit proposed in paper D with six other cost functions to evaluate adaptation objectives related to fundamental MR interaction requirements such as UI visibility, reachability, and consistency. The main contribution in paper C can also be interpreted as a cost function to assess upper limb ergonomics, where we assign an interaction cost to each reachable position in the user's interaction space.

Incorporating multiple cost functions into an adaptation policy raises additional concerns, especially in scenarios without a solution that fulfills all the adaptation objectives. First, for a system to make an adaptation decision and consider trade-offs, the cost functions must output numerical values on a similar scale. For example, a cost function to evaluate the distance from a desired position which outputs values from 0 to 200, cannot be directly compared to a cost function to evaluate ergonomics which outputs values from 0 to 1, when these operate on different scales. To do so, these must go through a normalization stage to bring the values from the different scales into a common one. Scalarization, such as the weighted global criterion method [172], can address this challenge but does not guarantee that it normalizes the cost functions to similar quality levels. In other words, adjustments to the formulation of the cost functions might be necessary for these to output values in the same quality range. Papers C and D use normalized formulations to achieve this - see Figure 4.1 for some examples. Secondly, some UI configurations are undesirable - solvers should not consider them. While the poor quality of such solutions can be represented in a cost function, reducing the dimensions of the solution space brings several computational benefits. In paper C, we constrain the design space to positions reachable by the arm, and we discretize the interaction space to make the problem computationally tractable. Creators can constrain the design space further to remove undesirable interaction zones. Optimization methods to find adaptation solutions also benefit from constraints

Objective	Illustration of the Concept	Cost function (refer to paper D for more info.)
a) Anchor to Target		$l \leftarrow csTRS^{-1} \cdot uiPos$ ▷ get UI position in cs local coord. $d \leftarrow \text{distance}(l, o)$ ▷ distance from offset to UI $c \leftarrow \min(d/t, 1)$ ▷ normalize cost according to t **return** c
b) Distance Interval		$difXZ \leftarrow csXZ - uiXZ$ $dXZ \leftarrow \text{magnitude}(difXZ)$ ▷ dist. from cs in XZ plane $dt \leftarrow \text{abs}(dXZ - gXZ)$ ▷ UI XZ distance from goal $cXZ \leftarrow \max(0, dt - iXZ)$ ▷ no penalty if dist. $\leq iXZ$ $dY \leftarrow \text{abs}(csY - uiY)$ ▷ UI Y distance from goal $cY \leftarrow \max(0, dY - iY/2)$ ▷ no penalty if dist. $\leq iY/2$ $c = cXY + cY$ $c \leftarrow \min(c/t, 1)$ ▷ normalize cost according to t **return** c
c) Field of View		$l \leftarrow csTRS^{-1} \cdot uiPos$ ▷ get UI position in cs local coord. $a \leftarrow \text{angle}([0, 0, 1], l)$ ▷ angle between gaze and UI $a \leftarrow \text{abs}(a - bo)$ ▷ distance from desired origin $a \leftarrow \max(0, a)$ ▷ no penalty if angle $\leq i$ **return** $\min(a/t, 1)$ ▷ return normalized cost

d) Avoid Occlusion		$c \leftarrow 0$
		for all k in ks **do**
		$wK = uiTRS \cdot k$ ▷ get k in world coord.
		if raycast($csPos, wK$) hits **then**
		$c \leftarrow c + 1$ ▷ increase cost
		end if
		end for
		return $c/\text{length}(ks)$ ▷ return normalized cost

e) Look Toward Object		$lA \leftarrow uiPos - csPos$ ▷ Vector from cs to ui $a \leftarrow \text{angle}(lA, uiZ)$ ▷ angle difference **return** $\min(a/t, 1)$ ▷ return normalized cost
f) Constant View Size		$d \leftarrow \text{magnitude}(uiP - csP)$ ▷ get distance from ui to cs $i \leftarrow dS * (d/iS * sF)$ ▷ ideal scale based on d $c \leftarrow \text{magnitude}(uiS/i)$ ▷ compute scale difference **return** $\min(c/t, 1)$ ▷ normalize according to t

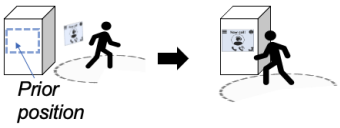
g) Spatio-Temporal Coherence		**if** vs has voxel containing $uiPos$ **then**
		return 0
		else
		return 1
		end if

Figure 4.1: Overview of the adaptive behaviors proposed in paper D and their respective cost functions. Refer to paper D and the source code for further implementation details and clarification on the pseudo-code of the cost functions.

as searching smaller solution spaces tends to be more efficient.

4.2 Problem Definition and Design Space

Finding appropriate UI adaptation solutions while attempting to minimize multiple cost functions while respecting inequality and equality constraints is a well-known

multi-objective optimization problem that one can formulate as follows [172]:

$$\begin{aligned} &\text{Minimize}_{\mathbf{x}}\mathbf{F}(\mathbf{x}) = [F_1(\mathbf{x}), F_2(\mathbf{x}), ..., F_k(\mathbf{x})]^T \\ &\text{subject to } g_j(\mathbf{x}) \leq 0 \text{ for } j = 1, 2, ..., m \\ &\text{and } h_l(\mathbf{x}) = 0 \text{ for } l = 1, 2, ..., e, \end{aligned} \tag{4.1}$$

where k is the number of F cost functions, m the number of g inequality constraints, and l the number of h equality constraints. Following Marler's notation [172], $\mathbf{x} \in E^n$ is the vector of decision variables, which in an MR UI adaptation setting would be the UI configuration, and n is the number of decision variables, or in our case, the number of configurable properties in the adaptation. All the possible solutions which are feasible (comply with constraints g and h) and attainable (solution is feasible and exists in the criterion space) are part of the feasible criterion space $\mathbf{Z}$. In UI adaptations, $\mathbf{Z}$ consists of all the possible adaptations that meet the constraints specified by the developer. In contrast to problems with a single cost function, multi-objective problems do not tend to have a single global solution. Instead, multiple points in the design space can be optimal. In this thesis, I use the predominant concept of Pareto optimality for defining optimal points in the design space [172], which consists of all the solutions where it impossible to make an objective better off (achieve a lower cost) without making another worse off. All these points lie in the boundary of the $\mathbf{Z}$ space, forming the Pareto frontier.

Finding optimal solution points that approximate or are in the Pareto frontier is relevant to find a UI configuration that meets the specified objectives. However, in practice, a system only requires one solution. In other words, from all the optimal adaptation proposals in the Pareto frontier, the system will only adapt to one of these UI configurations. Therefore, it is important to constrain the solution space by articulating the goals or relative importance of objectives that reflect user preferences. Most practices use the weighted sum method to do so [172], which reduces all the objective functions into a single utility function U:

$$\mathbf{U} = \sum_{i=1}^{k} w_i F_i(\mathbf{x}) \tag{4.2}$$

This method is considered an a priori approach to articulate preferences and how developers can articulate the relative importance of the objectives in paper D (refer to Marler's survey [172] for an overview of other methods). This approach allows creators to overcome the limitations of most rule-based methods, as it can find adaptation proposals where objectives trade off with each other. The minimum of U Pareto optimal [283] - the next section presents how we search for such solutions in our work.

4.3 Computing UI adaptations in real-time for MR experiences

In section 4.2, , I detailed how to formulate UI adaptations as a multi-objective optimization problem. Here, I discuss methods to find optimal or approximate adaptations in MR, a setting with different requirements from conventional optimization problems. In this case, MR adaptations must run in real-time to be useful to users. Therefore, the optimization methods must be sufficiently fast for the contextual information used as an input to be consistent when the optimizer returns a solution, operating within an upper bound of some seconds. Figure 4.2 provides an overview of how we address this problem by considering the efficiency of cost functions or models to assess the quality of the adaptation (if it exists) and how computationally tractable is the input space. While an exhaustive search might seem inadequate considering

Evaluation of adaptation / Input space	**Yes, in real time**	**Yes, computationally costly**	**No, but we have data**
Computationally tractable	Real-time opt. or exhaustive search w/ indexing	Exhaustive search w/ indexing Paper C	Indexing (if data covers whole input space)
Computationally intractable	Real-time optimization Paper D	Exhaustive search w/ encoder Paper B	Encoder Paper A

Figure 4.2: Overview of methods to find UI adaptations in real-time explored in this thesis. Columns depict how the quality of the adaptation is assessed (e.g., cost function or model that evaluates in real-time). Rows refer to the dimensions of the input space and if the input space is computationally tractable (e.g., the system can discretize and query it in real time). Papers in this thesis explore some of these methods to evaluate and propose UI adaptations (solid line), and others explore them for abstracting context (dashed line).

real-time requirements, it can address some problems. Deterministic models, which we use in paper C, produce the same results for sets of inputs in the input space. We leverage this to compute the interaction cost for a tractable representation of the interaction space in a pre-processing stage. We index these values in a database that the system can query in realtime. To do so, we propose a pipeline consisting of three stages: 1) Discretization of the interaction space, 2) Computation of arm poses, and 3) Computation of the interaction cost. From a high-level overview, this pipeline is an instance of a three-stage method applicable to similar deterministic problems which can be solved using exhaustive search methods (see Figure 4.3):

1. Abstracting and streamlining the design space.

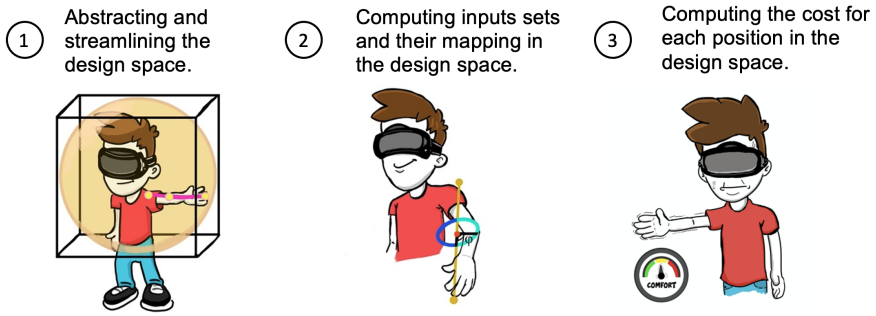


Figure 4.3: Approach to support optimization of computationally expensive cost functions using exhaustive search in real-time. We illustrate each stage with the corresponding step in Paper C.

2. Computing input sets and their mapping in the design space.

3. Computing the cost for each position in the design space.

Deterministic biomechanical models to assess ergonomics are an example of a computationally expensive method where such an approach is valuable, as demonstrated in paper C. However, other factors tend to be static over time in MR settings that could benefit from this method (e.g., optimal color schemes for UIs in a room).

It is possible to circumvent the computational overhead of some deterministic problems using methods such as the pipeline presented earlier (Figure 4.3). However, realistic settings often have large input spaces, impossible to discretize and query any input set in real-time. The interplay between context categories affecting MR experiences is complex and high-dimensional. Therefore, MR systems must assess the UI regularly according to the user's changing context. This is challenging for adaptive UIs in MR with real-time constraints. MR adaptation problems can involve multiple conflicting objectives, where certain positions in the design space meet each criterion to different extents. Such a combination of requirements often leads to a compromise between methods to find an adaptation that is good enough and the system can compute sufficiently fast to be relevant to the user. Real-time optimization, such as local search methods, is highly relevant for solving this problem. For instance, in paper D, we use simulated annealing [4] to iteratively improve solutions by searching the design space for a customizable amount of time. To make this process more effective, each objective in our work contains problem-specific information to steer the optimization process in a direction that benefits that criterion.

Although our work in papers A and B focuses on bringing context to higher abstraction levels, these approaches illustrate how similar methods could find MR UI adaptations. In the case of paper A, we use a labeled dataset to train an encoder to determine the orientation of near-symmetrical objects. This encoder takes the input data (RGB picture) and converts it into the encoded representation of interest (object rotation). Here, the encoder is a CNN that meets real-time requirements, and the

training procedure uses labeled data. Although this CNN is not used to propose UI adaptations, this procedure illustrates how a developer can use a similar model to do so - the only requirement would be a dataset containing information for adapting to a specific input space. Paper B explores a similar problem as paper A, but without having access to a labeled dataset. We use a digital model of the object of interest to generate labeled data to address this limitation. Again, while the work in this paper does not directly propose UI adaptations, this can be seen as an analogy to the scenario where developers have access to computationally costly models or cost functions to evaluate UI adaptations. Here, developers can generate a dataset using such a model to train an encoder capable of meeting real-time requirements.

4.4 Discussion

The following discussions reflect on the limitations and venues for future work on the topics presented in this section.

Limitations of weighted sum approaches

As highlighted in section 4.2, weighted sum approaches are the most common method to articulate preferences in existing optimization techniques. They are an important part of our approach in paper D. While such a method successfully narrows down the design space of the optimization problem, as our system searches for adaptation solutions in specific zones of the Pareto frontier, there is an assumption that the combination of weights is an appropriate representation of the user's preference function, which will not always hold. Furthermore, even when considering a wide range of adaptation objectives, additional contextual factors are neglected at design time or not sensed by the system. Therefore, it would be interesting that MR systems could support more than one solution by exploring other solution zones in the Pareto frontier or adjusting and learning new objectives based on user interactions. Although not part of this thesis, such questions provoked other researchers and collaborations to investigate this topic further (Paper H).

Optimization of UI adaptations in complex design spaces

When solving multi-objective optimization problems in large design spaces and in real-time, the solver can struggle to find solutions close to the Pareto frontier. Therefore, it is crucial to assess trade-offs between the quality of the results and how fast the system computes these. Existing works have explored user preferences in relation to the automation level of such MR adaptations [168], pointing towards users preferring semi-automated adaptations. However, it would be interesting to investigate how the quality of the adaptations affects users' preferences and how error recovery techniques mitigate the cost of problematic adaptations. Solver efficiency can also be improved using more rigorous formulations of the optimization problem - for example, in paper D we formulated the optimization problem by allowing creators to build a cost function

by selecting from various adaptation objectives. In this case, the normalization of these objective functions acts as an alternative to constraints, where the cost reaches its highest value once it passes a threshold. However, some solvers could leverage formally defined constraints to search the design space more effectively.

Stationarity of adaptation preferences

Stationarity of adaptation preferences Using static weights to articulate preferences when optimizing MR adaptive UIs assumes that user preferences towards these adaptation objectives remain constant over time. Relying on the assumption that user preferences do not change can be another pitfall. If users act according to what is best for them, behaviors and objective preferences will likely change over time. Furthermore, memory limits, perceptual bounds, motor bounds, and the environment [200] will likely shape preferences over time. Reinforcement learning approaches can deal with such non-stationary problems and are a promising topic to explore in this context.

4.5 Summary

In this chapter, I presented the foundations and the mathematical formulations that make our contributions in papers C and D possible. I described the computational interaction approach proposed in paper C which facilitates the understanding of ergonomics of the upper limbs and can be used to inform the design of MR adaptive UIs. Furthermore, I introduced the problem formulation we use in paper D, which allows for flexibly mixing and matching different adaptation objectives, overcoming the limitations of rule-based approaches. This serves as the basis for solving adaptive UI problems in the toolkit we contribute in paper D, which I elaborate on in the next chapter.

Chapter 5

Adaptive User Interfaces for Mixed Reality

Adaptive UIs play a crucial role in improving the usability of MR applications, adjusting the UI to context changes, and seamlessly integrating virtual content in the physical environment. Although researchers explored adaptive UIs over the last decades, most works investigated adaptations in 2D applications with different requirements from MR applications. In particular and as motivated in earlier chapters, the types of MR adaptations explored in this thesis have real-time requirements, large design spaces, and hard-to-find solutions. Furthermore, while optimization problem formulations and cost functions can enable sophisticated adaptation behaviors, there is still a gap between the topics discussed in the previous chapter and how these make into concepts that constitute an adaptation. In this chapter, I discuss what comprises MR adaptations and the requirements to implement them in MR applications, paving the way for the framework presented in paper D.

5.1 Understanding MR UI adaptations

To better comprehend adaptations of MR UIs, it is important to understand what parts do adaptations comprise off. Analyzing existing work that focuses on MR real time adaptations [47, 144, 163, 246], I identify four stages across these approaches:

- **Sensing context** - Sensors capture and use data for context understanding (e.g., eye tracking [163] or environment [47, 246]).
- **Context abstraction** - Context data processed into higher representation levels (e.g., cognitive load [163], objects [47], or people [246]).
- **Finding a UI adaptation** - The system uses abstracted context data to find an appropriate UI adaptation (e.g., using optimization [47, 163, 246] or rule-based approaches [144]).

- **Adapting the UI** - UI adapts to the new state (e.g., changes in the level of detail [163], the position of the UI [47], or input modality [144]).

While these stages are fundamental in any context-aware adaptive UI, these are challenging to reuse and modify when creating new MR applications. Existing approaches tend to be developed from scratch and focus on specific context sources and adaptations. In the following sections, I present our findings concerning what is necessary to make the creation of adaptive UIs more accessible and propose a framework to support creation processes, one of paper D's contributions.

5.2 Scalable and flexible UI adaptations

To flexibly support various MR scenarios, it is important that tools for developing adaptive UIs allow creators to use different context sources. Furthermore, they must allow customization of how the UI adapts to a wide range of contextual information. When considering multiple adaptation objectives, allowing creators to combine them into a single adaptation is paramount to support various adaptation behaviors. In section 4.2, I presented a general multi-objective problem formulation that meets such requirements: multi-objective optimization. Formulating adaptation objectives as cost functions allows creators to build a vector of objective functions iteratively - doing so gives creators more flexibility in which adaptation objectives to use and lets systems scale to a wide range of objectives since cost functions operate independently from each other in such a formulation. However, it is necessary to have solvers capable of finding solutions to the optimization problem in real-time to enable this approach. Methods for the system to gracefully handle cases without adequate solutions are also necessary. Moreover, it is critical to allow for flexibility towards what adaptation behaviors drive the adaptation and to enable creators to decide when and why the UI adapts. Finally, when adaptations trigger, the creator must be able to customize how the properties to adapt transition from the current to the new state - when the UI can assume the new configuration, the properties to adapt can change in different ways, notifying the user or transitioning over time. These requirements can be summarized in the following five design goals, which we identify in paper D:

- *D1:* Support a range of adaptation behaviors.
- *D2:* Allow combining multiple adaptation objectives in one adaptation.
- *D3:* Support for context collection and interpretation.
- *D4:* Methods to customize when and why an adaptation occurs.
- *D5:* Support for a variety of property transitions.

5.3 A framework to support the creation of adaptive UIs

In paper D, we propose five concepts to organize the key components of adaptive UIs. In this section, I present the high-level ideas of each, how they address the requirements from section 5.2, and how they relate to each other:

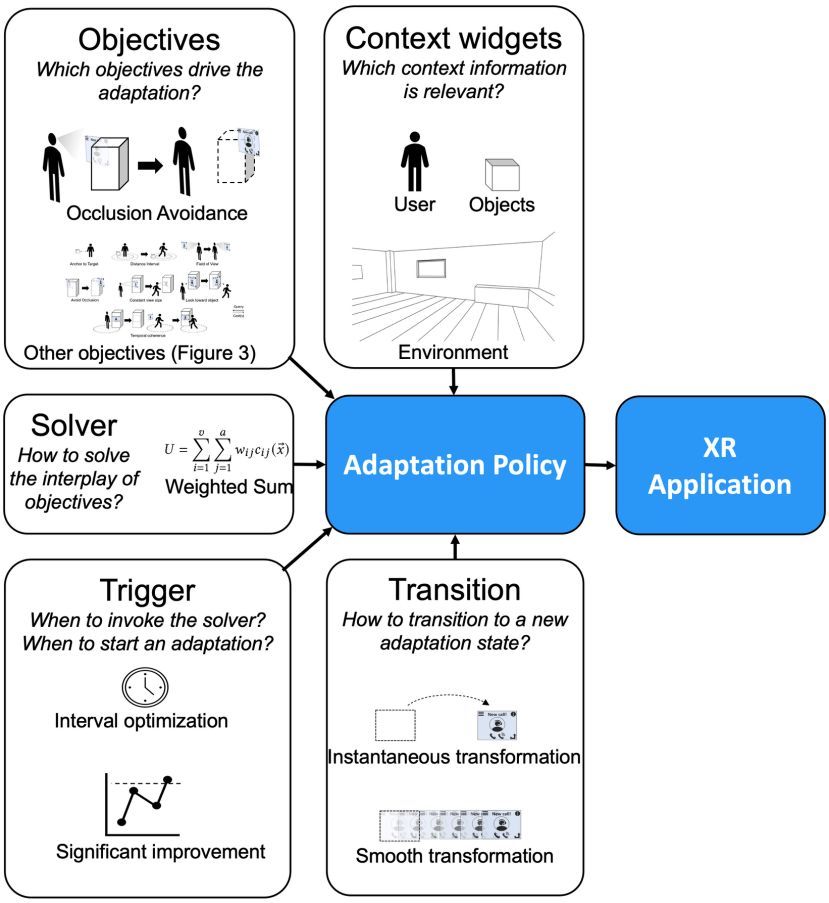


Figure 5.1: Overview of the framework to create adaptive UIs proposed in Paper D.

- **Adaptation Objectives** are the core component of our approach, represented through a cost function each. When put together, they act as the building blocks of the vector of objective functions making up the optimization problem we want to solve. Solutions that minimize this problem are UI adaptations that approximate the Pareto frontier, or in other words, fulfill these objectives so they cannot be improved further without worsening other adaptation objectives. Adaptation objectives should be independent of each other, indivisible into multiple objectives, and represent intuitive adaptation goals. Such characteristics allow for more flexibility and make it possible to achieve a wider variety of adaptation goals (*D1*).

- **Solvers** are algorithms that solve the optimization problem creators formulate by combining adaptation objectives. These approaches solve the optimization problem introduced in section 4.2, solving conflicts between objectives (*D2*). There are various established methods to do so, such as linear programming, genetic algorithms, and simulated annealing, which is the approach we use in paper D. Solvers and adaptation objectives address the *Finding a UI adaptation* stage identified in section 5.1.

- **Context Widgets** are a component we reuse from Dey et al's work on context-aware applications [59]. We broaden the scope of responsibilities of context widgets so these also process context data to higher abstraction levels. Doing so simplifies our framework further and provides additional flexibility to update the context widget an adaptation objective consumes - as long as the context source is the same. For example, an adaptation objective that attempts to position a virtual element close to some entity can generalize to any entity with a context widget providing its position, no matter how it is retrieved. Context widgets fulfill *D3* and the *sensing context* and *context abstraction* stages identified in section 5.1.

- **Adaptation Triggers** are the strategies for invoking the solver and executing UI adaptations. Existing works implement logic to compute and apply adaptations based on application requirements. However, such adaptation strategies are often similar across applications. Adaptation triggers allow creators to reuse and customize them across MR applications (*D4*) and support two of the adaptation stages identified earlier: *finding a UI adaptation* and *adapting the UI*.

- **Property Transitions** determine how different properties of a UI adapt to a new adaptation proposal. As MR interfaces blend with the physical world adaptations must be clear to the user. MR experiences can benefit from smooth transitions over time or subtle notifications when the UI adapts. Property transitions allow creators to reuse and decide how adaptations occur, meeting *D5* and fulfilling the *adapting the UI* stage.

5.4 Framework implementation

Implementing the framework proposed in section 5.3 is the next step towards making it available to creators and assessing its utility. In this section, I describe how we implement these concepts into the toolkit components (AUIT) presented in paper D.

Overview of the architecture and implementation

In this section, I overview the toolkit's architecture, describe how components interact, and go over general implementation details. AUIT is implemented for Unity using the C# programming language. Each concept of the framework translates directly to a component in the toolkit. Figure 5.2 depicts an overview of the processing flow.

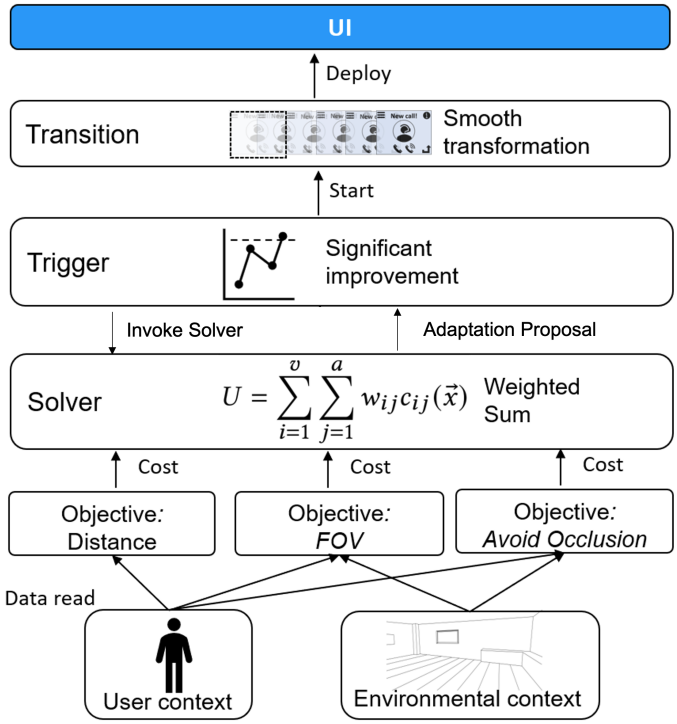


Figure 5.2: The processing flow of the toolkit (from paper D). Context widgets provide data used to compute the cost functions of adaptation objectives. The solver aggregates adaptation objectives and uses these to propose adaptation proposals. The adaptation trigger invokes the solver and decides when to apply the adaptation. When the trigger applies adaptations, the correspondent property transition executes by adapting the UI.

Context widgets make context data available on request for different data sources. Adaptation objectives use these data sources to evaluate the quality of adaptation proposals. Creators can change the context source an adaptation objective consumes if the context data is in the same format. For example, an adaptation objective to anchor UI elements to entities in 3D space can have the user's hand or head as a target as long as such context sources exist and provide data in a format readable by the objective. The solver aggregates the adaptation objectives assigned to the UI optimization problem into a vector of objective functions, which it uses to approximate optimal adaptation proposals when invoked by the adaptation trigger. Besides invoking the solver, the adaptation trigger is also responsible for starting adaptations. When an adaptation starts, properties adapt using their corresponding property transition. I discuss each component in greater detail as follows.

While Figure 5.2 details the processing flow, it might be unclear how each component connects to UI elements in our software architecture. Figure 5.3 depicts AUIT's software architecture. Creators can assign multiple adaptation objectives and property

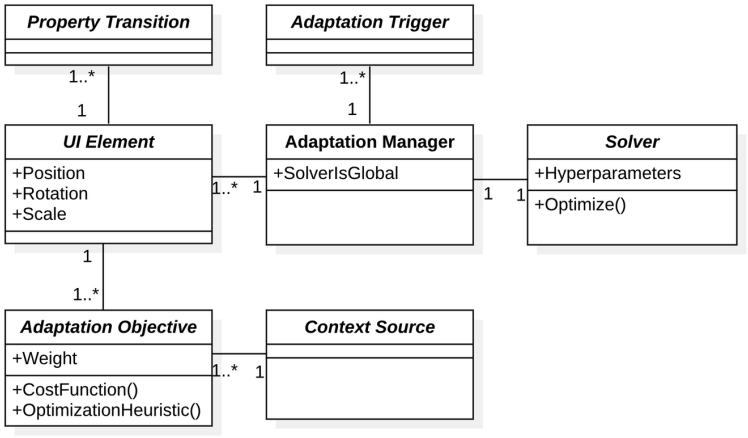


Figure 5.3: UML diagram of AUIT's architecture (from paper D).

transitions to a UI element. Property transitions must not overlap in relation to the property they adapt (e.g., a UI element supports a property transition for position and another for rotation, but not two different property transitions for position). To support multiple adaptation triggers and adaptations containing multiple UI elements, we created the adaptation manager - an auxiliary class to gather all the adaptation objectives, invoke the solver and apply adaptations when requested by adaptation triggers. All the core components of the framework inherit from abstract classes, making it straightforward to add new functionality to the toolkit (e.g., new adaptation objectives and adaptation triggers).

Adaptation Objectives

Each adaptation objective in our toolkit requires the implementation of two important modules: a cost function and corresponding heuristics. The cost function represents to which extent a UI fulfills the objective considering its corresponding context source. Because our current solver uses a weighted sum approach, all the costs are normalized to facilitate weight selection. To avoid additional complexity in AUIT, we opted for a formulation of the UI adaptation problem without explicitly defining constraints (refer to section 4.2). Instead, we allow creators to customize a threshold for the objective to return the highest cost when the solution violates it. The heuristics nudge the solver into an adaptation proposal that will likely benefit the objective. Every adaptation objective contains intentional randomness to broaden the search space of the solver. See Figure 5.4 for an example, which depicts how we implement both modules for the anchor to target objective.

Solvers

A solver for MR adaptations must fulfill several requirements. First, for the type of adaptations we desire, it must run online, or in other words, it must be able to

```
1  o ← offset vector provided by creator
2  t ← threshold for highest cost
3  csTRS ← context source TRS matrix
4  uiPos ← UI position
5  function Cost
6     l ← csTRS^-1 · uiPos      ▷ get UI position in cs local coord.
7     d ← distance(l, o)              ▷ distance from offset to UI
8     c ← min(d/t, 1)               ▷ normalize cost according to t
9     return c
10 end function
11 function Heuristics
12    s ← random value ∈ [0, 1]
13    opt ← csTRS^-1 · o              ▷ compute optimal position
14    if s ≤ 0.33 then                ▷ pick heuristic at random
15       return opt                   ▷ return optimal position
16    else
17       ou ← normalize (opt − uiPos)
18       uv ← random unit vector                ▷ add randomness
19       ou ← ou + uv ∗ random value ∈ [0, 0.3]
20       return uiPos + ou ∗ N(1, 0.5)
21    end if
22 end function
```

Figure 5.4: Source code for the anchor to target objective. Adaptation objectives in AUIT contain two modules: a cost function and a heuristics function. For this objective, the cost increases as the UI element gets further away from its anchor. The heuristics propose solutions that bring the UI element closer to the anchor point.

compute good adaptation proposals in a short period of time (ideally under 1 second). Second, for greater flexibility on what adaptation objectives it supports and to support the problem formulation introduced in section 4.2, it must solve non-linear and non-convex optimization problems. We opted for implementing a local search method as the first supported solver in AUIT - simulated annealing [4]. We use the heuristics from the adaptation objectives to make the solver faster and prioritize specific zones of the design space. The solver returns a solution once it finds an optimal adaptation proposal (early stopping) or the best solution after running for a customizable number of iterations.

Context widgets

The initial iteration of our toolkit uses contextual data from context sources fundamental to MR experiences related to 3D registration (user's position and head orientation, real-world geometry). Therefore, the context widgets which process and abstract this data are already available in Unity. When an adaptation objective is selected, it will consume data from a corresponding context source by default (e.g., an adaptation objective to place content in the user's field of view will have the user's head orientation

as the default context source). Creators can change the context source an objective consumes if it supports the format, supporting fast and flexible development processes. This allows creators to use AUIT without knowing context widget implementation details or even having to configure these components.

Adaptation triggers

In the initial release of the toolkit, we have two adaptation triggers to support basic adaptation strategies: a trigger that invokes and applies unconditionally the solutions from the solver at a fixed rate and a trigger that considers the quality of the UI and only applies adaptations when the new proposals improve the UI by a specified threshold (refer to paper D for further details). Although not currently part of the toolkit, rule-based adaptation triggers could open up several venues for further customization of adaptive MR UIs created using our toolkit. Adaptation triggers are directly associated with an adaptation manager (which manages the adaptation loop of one of multiple UI elements).

Transition Properties

Adaptation triggers are directly associated with UI elements. This design decision allows multiple UI elements in the same optimization loop to adapt in different ways if desired. Furthermore, property transitions adapt a single property type (e.g., position or rotation), allowing for various configurations. For example, creators can combine a rotation with a 3D movement over time or a fade-out fade-in effect from the previous to the new position. The design space of transition properties is enormous, and this is yet another venue with potential for further exploration. Refer to paper D for more detail on what transition properties AUIT supports.

Adaption at runtime in AUIT

Figure 5.5: Context-Aware UI Adaptation in AUIT. 1) UI must adapt to fit new context; 2) Plethora of possibilities for an adaptation that are context-dependent; 3) Adaptation picks a suitable solution;

To clarify how AUIT operates at runtime, I will go revisit steps that result in an adaptation. The adaptation trigger checks following its corresponding logic if the current UI layout meets the requirements (Figure 5.5, 1). To do so, it evaluates how well each adaptation objective is fulfilled in the current context (gathered from context widgets). If the trigger determines the UI must adapt, it calls the solver to find adaptation proposals (Figure 5.5, 2). If the solver computes a suitable adaptation proposal, the adaptation trigger starts the adaptation using the UI element property transitions (Figure 5.5, 3).

5.5 Discussion

In this section, I presented a framework with five components to separate concerns of MR UI adaptations. We bring context collection methods from chapter 3 and computational interaction concepts from chapter 4 into some of these components to make them more accessible during development. Considering the former, context collection and abstraction methods make their way into UI adaptations through context widgets. Regarding the latter, cost functions are the concept behind adaptation objectives, while solvers gather and provide solutions to the UI optimization problem. Our next step was to instantiate the framework through a toolkit with flexibility as the main objective. The outcome is AUIT, a toolkit to support the creation of adaptive user interfaces. AUIT demonstrates how it is possible to implement such a framework, bringing several adaptation methods together and allowing for quick prototyping of adaptations without coding required. Initial prototyping and a study with experts indicate that both the framework separation of concerns and the toolkit support the creation of adaptive MR UIs (refer to paper D for more details on the evaluation). In the following sections, I discuss topics related to the framework and MR UI adaptations.

Creator-designed adaptations

In AUIT, creators design UI adaptations for end users to experience. In contrast to established interaction paradigms, such as the smartphone or personal computer, where applications live in virtual environments, MR applications cross this boundary and live in the user's physical environment. Therefore, the gap between what the creator designs and what the user experiences is wider as there are more unknowns about how the end user will experience the application during design. Furthermore, creators can approximate end-user preferences such as ergonomics for a general population of end users but will ultimately vary on a personal basis (e.g., arm movement impairment). In AUIT, designers create adaptations and prioritize objectives by defining weights and approximating end-user preferences. Although the toolkit is an important step towards facilitating the creation of adaptive MR UIs, one can foresee how such an approach can fall short of providing the best usability, as creator-defined weights and the formulation of the objectives might not reflect end-user preferences. These limitations motivated follow-up work, where we propose methods to bring more

control to end-users. One way to overcome the shortcomings of creator-defined weights is to suggest multiple adaptation proposals in the Pareto-frontier, which we demonstrate in paper H.

Refining and learning new adaptation objectives

Adaptation objectives in our framework may not perfectly align with the end user preferences. For example, consider the objective to place virtual content in ergonomic positions for hand interaction - if the user is injured, the cost function should penalize uncomfortable interaction zones. Moreover, techniques to learn new objectives not initially included in the optimization problem could enhance usability further. In theory, systems could leverage usage data such as manual adjustments to the UI by the user to refine existing adaptation objectives and learn new ones. Our framework can support such capabilities through a new component that collects usage data and adjusts/creates adaptation objectives or new solvers that learn over time.

Learning user preferences for adaptive UIs

Figure 5.6: Adjustment of adaptation objectives and weights in immersive environments using AUIT.

As motivated in section 4.4, while it is unlikely that creators will design adaptations that match users' preferences perfectly, it is also unlikely that these remain the same over time and in different settings. For example, a worker using an MR application to display assembly instructions in the workplace might prioritize the visibility of instructions when unfamiliar with the procedure, but with more experience on the task, ergonomics can get more relevant. Therefore, methods to support the adjustment of preferences over time are promising. AUIT supports tweaking weights in MR (see Figure 5.6), but such an approach results in an additional burden to the

user. Ideally, such preferences would be approximated automatically by the system, another topic we are starting to explore in paper H.

5.6 Summary

In this chapter, I identified the parts UI adaptations consist of, which motivate the main contributions of paper D. Then, I presented these contributions in detail: a framework to separate concerns of MR UI adaptations and how we instantiate it through a toolkit for a widely used development platform. In the next chapter, I will elaborate on how creators can use the toolkit and other tools proposed in this thesis.

Chapter 6

Tools to Ease the Creation of Adaptive UIs

In the previous chapters of this thesis, I went over three major topics crucial to implement adaptive MR UIs:

1. Context-awareness in MR
2. Computational interaction approaches
3. Adaptive UIs for MR

Chapter 3 discussed context categories relevant to MR and presented methods to capture and abstract some of these to use them for UI adaptations. Chapter 4 introduced computational interaction approaches to assess the quality of UI adaptations and retrieve adaptation proposals. Chapter 5 proposes a framework to separate concerns of UI adaptations and presents how we implement it as a toolkit. While the aim of the work presented in these chapters is to enhance the comprehension of adaptive MR UIs and explore new adaptation methods, end-users will only ever benefit from such advancements if the techniques and methods we propose are accessible to creators. In this chapter, I will discuss steps taken to make our research accessible to creators in relation to each topic discussed so far.

6.1 Context collection

In order to enable MR adaptive UIs, access to contextual data is crucial. Therefore, lowering the barrier to accessing this data during development is essential. Additionally, facilitating the expansion of the capabilities of context widgets to retrieve new contextual data can unlock new functionalities in MR applications. Throughout our work, we attempted to make progress in both directions. An example of the former is the design of the context widgets and adaptation objectives in the toolkit (AUIT) introduced in chapter 5. AUIT can hide the complexity of capturing and processing raw context data if creators desire so since most adaptation objectives already

have a default context source assigned (see Figure 6.3, c) - this means that during development creators already have access to context just by selecting adaptation objectives. Regarding the goal of expanding the capabilities of existing context widgets,

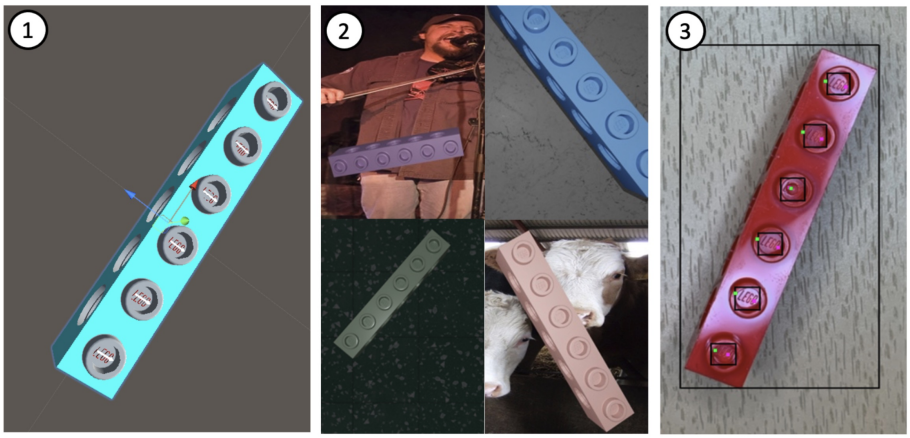


Figure 6.1: Pipeline for tracking new near-symmetrical objects using a CAD model (from paper B). In stage 1, the creator labels all the points that allow symmetry disambiguation. Stage 2 uses the model and annotations from the previous stage to generate a synthetic dataset, which is labeled automatically. Stage 3 uses the synthetic dataset to train a CNN capable of disambiguating the object's orientation.

the approach we propose in paper B is an example of how a context widget could generalize to more contexts with the intervention of creators. In this case, the pipeline we introduce allows users to label a digital representation of a near-symmetrical object (Figure 6.1, stage 1), generate synthetic data and train a CNN to detect the object's orientation (Figure 6.1, stage 2 and 3). In AUIT, the approach we present in Figure 6.1 could be part of a context widget. In this case, the CNN would be a context source for object keypoints in supported objects. The pipeline to track new objects would be an additional capability of the context widget to generalize beyond its default tracking capabilities.

6.2 Informing the design of Adaptive UIs

One of the barriers to authoring XR applications identified by Ashtari et al. is the lack of design guidelines and examples [6]. In paper C, we developed a tool that allows creators to visualize how each position in the interaction space fares in terms of ergonomics (see Figure 6.2). This visualization is part of the XRgonomics toolkit we make available for creators. While creators can use the approach as if it was a cost function for an ergonomics adaptation objective, the graphical user interface allows creators to visualize the ergonomic cost of different zones in the interaction space, considering various established metrics. It is possible to compute these costs considering different arm dimensions and add constraints to disregard zones irrelevant

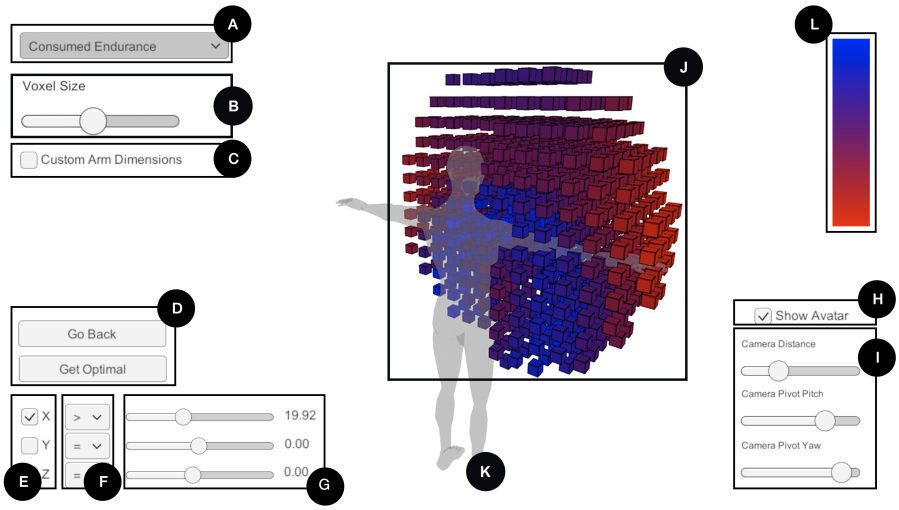


Figure 6.2: Toolkit to inform creators on ergonomics of the upper limbs proposed in paper C. Creators can select and visualize different metrics in real-time (A), change visualization (J) parameters such as the voxel size (B), constraints for the interaction space (E, F, G), display of the avatar (H), and camera controls (I). The toolkit allows creators to input custom arm dimensions (C) and analyze positions of the interaction space in greater detail (D). The interaction cost of each voxel is displayed using a gradient (blue - lowest cost; red - highest cost).

to the problem. While this work is also directly related to the challenge of designing for the physical aspects of XR experiences [6], it is a clear example of how methods grounded in computational interaction can make their way into informed design decisions. A study with designers conducted for paper C showed that our visualization supports the understanding of ergonomics and helps from the early stages of design and development.

6.3 Creating Adaptive UIs

In chapter 5, I presented a framework to support the creation of adaptive UIs for MR and the implementation details to instantiate it through a toolkit (AUIT) that we make available for Unity. In this section, I briefly discuss the steps we took to make AUIT accessible to creators and how they can use it to design adaptive UIs, complementing the content available in paper D.

To use AUIT, creators associate framework components to the UI element they want to add adaptive behavior. Creators can achieve this by selecting the object they wish to adapt and then dragging and dropping script components in the inspector or typing in their names. Figure 6.3 depicts what these components look like when associated with a UI element. The adaptation manager auxiliary class is the starting point for creating an adaptation using AUIT (Figure 6.3, 1). The initial release

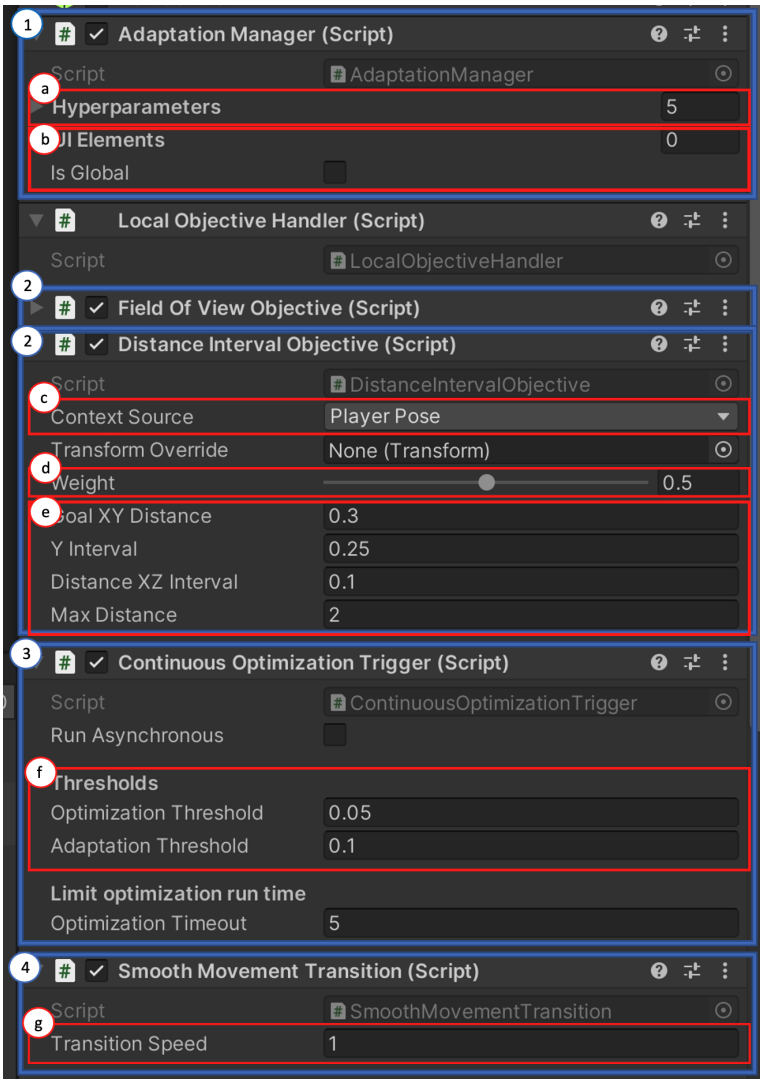


Figure 6.3: Creators can use AUIT by configuring each component to adapt in the Unity inspector. The adaptation manager (1) contains the solver hyperparameters (a) and manages the optimization problem, making it possible to optimize multiple UI elements in the same optimization loop (b) - in that case, an auxiliary Unity component must hold the adaptation manager. Multiple adaptation objectives (2) can be added to a single UI element. Creators can change the context source they consume through a drop-down (c), weights can be adjusted (d), and other objective-related parameters can be customized (e). Parameters of adaptation triggers (3) and adaptation properties (4) can also be adjusted (f and g).

of AUIT only contains one solver, which the adaptation manager uses by default - creators can optionally configure its hyper-parameters (Figure 6.3, a). Creators can optimize multiple UIs in the same optimization loop by listing each element in a global solver (Figure 6.3, b). As mentioned earlier, adaptation objectives (Figure 6.3, 2) use a context source by default that creators can customize (Figure 6.3, c). Each objective has a customizable weight (Figure 6.3, and most components support further customization (Figure 6.3, e, f, and g). Once an adaptation trigger and a property transition are added (Figure 6.3, 3 and 4), creators can visualize how the UI adapts in different contexts, by pressing the play button in Unity and simulating the user and respective environment in the scene. Changes to the weights, solver hyperparameters, and component configurations are applied in real-time if updated during simulation, allowing creators to experiment with a wide range of settings during development.

6.4 Discussion

Now I discuss the potential next steps for improving these tools' relevance and accessibility. This section is an opinion piece with origins in discussions with creators that could inspire directions for future work.

Relevance, accessibility, and usage of tools

The usage of a tool is directly affected by how relevant and accessible the functionality is for the creator. If the functionality is crucial to achieving a specific requirement, its usage is likely even if it is cumbersome to integrate into existing workflows (and there are no better alternatives). An example of such a tool would be the approach we propose in paper B. Currently, it does not integrate seamlessly into XR development tools. In scenarios where this context is required, such inconvenience is easily justified. In contrast, the visualization we propose in paper C can be relevant during the design process of mid-air interactions, but it is not required to design said interactions. Discussions with creators in our expert studies revealed how requiring them to deviate from their typical workflow (e.g., using a new application) could be a barrier to adoption. Making this integration seamless (e.g., through easily accessible plugins to existing development tools) is crucial to increase the adoption of new tools.

Making tools more accessible

The tools described in this chapter were our first attempt at making our methods accessible to creators, but much can improve. In particular, when considering context-aware adaptive UIs, it would be relevant to have all these methods available out of the box in a unifying toolkit such as AUIT. For example, the approach we propose in paper B could be a context widget with an additional module for tracking new objects using a CAD model. The ergonomic cost model for the upper limbs proposed in paper C could exist in AUIT as an adaptation objective. Allowing creators to access the visualization of the cost (Figure 6.2) in AUIT as an adaptation objective. Allowing creators to

access the visualization of the cost in the adaptation objective itself could support the understanding of how these work further - a feature we could extend to other objectives if such functionality turned out to be popular during development. Finally, AUIT is already available as a plugin for a popular development tool. However, its integration (or of a similar toolkit) could be even more seamless if directly accessible in the editor.

6.5 Summary

In this chapter, I described the tools we proposed to make the methods from this dissertation more accessible to creators. These are related to different stages in the development process of adaptive UIs. Developers can use the pipeline and development tools proposed in paper B to support more context (in this case, tracking of near-symmetrical objects). The tool presented in paper C can inform creators on how to design ergonomic MR UIs, and creators can use the model at runtime through the API we make available to optimize UIs. Finally, we propose a toolkit in paper D to unify all the concepts that constitute MR UI adaptations, allowing creators to combine adaptation policies considering various context categories in a flexible way.

Chapter 7

Methodology

The methodology employed in this thesis is grounded on problem-solving approaches advocated by Oulasvirta and Hornbæk [197], where scientific progress in HCI starts with the ability to improve the human use of computers. As the authors put it, once researchers define a research topic and identify its potential improvements, they formulate a research problem. Research to address the problem results in a solution evaluated to assess how it improves problem solving capacity in terms of five criteria that originate from Laudan's philosophy of science [150]: significance, effectiveness, efficiency, transfer, and confidence.

The research problems explored in this thesis fall into the category of constructive research from a problem-solving perspective, complemented by empirical contributions through evaluative studies with potential users of the solutions we propose in our works. From the viewpoint of Wobdbrock and Kientz [275], our HCI contributions combine artifact/system contributions with empirical research contributions that tell us how people use a system. In this chapter, I will start by describing the motivations behind the definition of the research problems explored in this thesis, the process that brought us from the problem formulation to solutions, and how we decided which evaluation methods to use for each solution in our work.

7.1 Formulating the research problems

The research problems explored in this thesis are motivated to a large extent by real-world challenges in the Danish manufacturing industry. For context, MADE is a research platform for the manufacturing industry in Denmark and funded the research in this thesis. Through meetings with MADE partners, such as LEGO, we identified research problems based on real-world challenges, such as the need for MR-based digital assistance for quality assurance procedures. Combined with the gap in the literature on real-time tracking for near-symmetrical objects, we formulated the problems explored in paper A. This initial work sparked interest in improving context capture and facilitating context availability to developers. The approach we propose in paper A requires laborious data acquisition and labeling processes to track

new objects, a pain point highlighted by our industry partners. However, digital models (CAD models) of manufactured objects typically exist in this setting since companies create them as part of the product design process. With the necessity of validating a digital assistance approach for providing instructions on tasks involving near-symmetrical objects, these two aspects motivated the research problems we investigate in paper B. Another factor influencing the problems we investigate is existing research identifying challenges for creating XR applications. As a developer of XR prototypes, the lack of guidelines to design ergonomic XR interfaces motivated the exploration of computational interaction methods to improve understanding of this human factor. These barriers are highlighted further in the work of Ashtari et al. [6] (lack of concrete design guidelines and examples; difficult to design for the physical aspect of immersive experiences), culminating in the research problems explored in paper C. Finally, analyzing the landscape of tools and state-of-the-art approaches for creating MR adaptive UIs revealed several limitations. For instance, popular tools to create adaptive MR interfaces, such as MRTK solvers [179] and Unity Mars [262], do not support multi-objective optimization. While state-of-the-art techniques [47, 163] propose multi-objective optimization, these approaches lack flexibility in the optimization problem formulation, limiting their significance to creators. Motivated further by the absence of a framework to provide a clear separation of concerns for MR UI adaptations, these research problems led to the work conducted in paper D.

7.2 Designing solutions

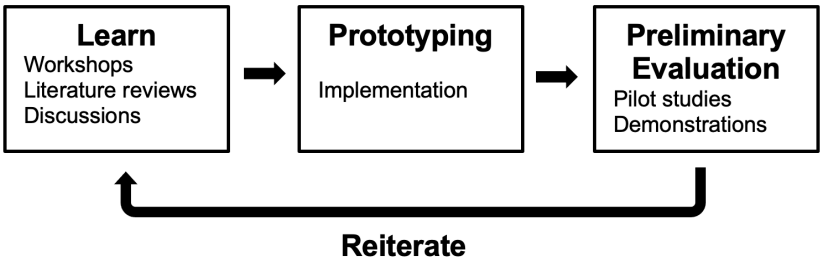


Figure 7.1: Design process employed in this thesis.

As the contributions in this dissertation are primarily constructive [197], prototyping plays a major role in reaching the solutions proposed. Prototypes enhance understanding of the research problem, and empirical evaluations, when appropriate, allow us to learn about how it can change the world [229]. Their importance over demonstrations and descriptions was discussed back in the ’90s by Bødker and Grønbæk [28]. Although not rigorously employed, participatory design [40] inspired the design process of our prototypes. In addition to literature reviews of existing methods related to our research problems, we incorporated feedback from potential users and stakeholders at LEGO during meetings to shape the designs, particularly for

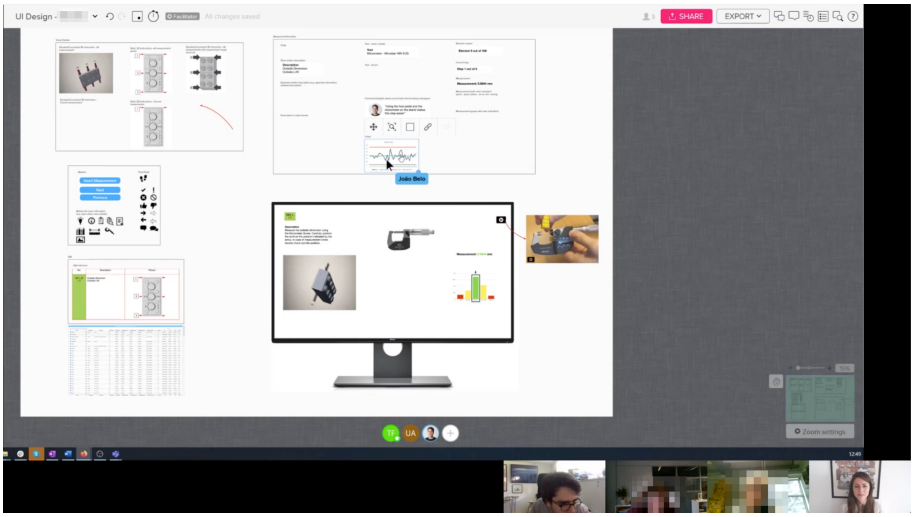


Figure 7.2: Remote collaborative design workshop conducted with workers at LEGO.

papers A and B (see Figure 7.2). Although we conducted initial conversations with creators of XR applications throughout the design of the solutions proposed in papers C and D, we conducted more comprehensive discussions with stakeholders later in the formal evaluation stage. Once a minimum viable prototype for a solution was operational, we ran preliminary testing stages. These took the form of pilot studies or demonstrations and were crucial for finding potential improvements to the proposed approaches and the prototype itself [40]. When the findings gathered during this stage revealed flaws or potential improvements, we reiterated the design process just described in an attempt to address them (depicted in Figure 7.1).

7.3 Evaluation

Following the problem-solving research approach [197], we assess the problem-solving capacity of each of our solutions through a formal evaluation. In our work, we were primarily interested in evaluating the effectiveness of our solutions (i.e., to which extent the solution solves the stated research problems). However, we are also interested in significance (i.e., the solution is relevant to stakeholders), and efficiency (i.e., the costs of applying the solution are proportional to the benefits). I will now provide an overview of how we evaluate each of the papers in this thesis and the literature that influenced such evaluation decisions - for a more in-depth overview I refer the reader to the respective evaluation section of each publication.

Paper A *Digital Assistance for Quality Assurance: Augmenting Workspaces Using Deep Learning for Tracking Near-Symmetrical Objects* - To evaluate the effectiveness of our tracking system and identify which components in our approach were relevant to the model's performance, we performed ablation studies typically conducted in AI literature [111, 113]. Such studies consist of removing certain components of a CNN

to assess how performance is affected.

Paper B *CADTrack: Instructions and Support for Orientation Disambiguation of Near-Symmetrical Objects* - The evaluation we conducted in paper B aimed to assess two criteria: 1) the effectiveness of the tracking pipeline; 2) the effectiveness and significance of the digital assistant to provide instructions containing near-symmetrical objects. To assess the former, we conducted a technical evaluation using two different objects where we gathered qualitative and quantitative data on how the tracking performs. For the former, we conducted a controlled lab user study following procedures established in HCI research [51, 64].

Paper C *XRgonomics: Facilitating the Creation of Ergonomic 3D Interfaces* - The main outcome of paper C is a toolkit, therefore we follow the guidelines Ledo et al. advocate for in their work on evaluation strategies for HCI toolkit research [153]. To show the toolkit capabilities, we demonstrated two application scenarios enabled by the toolkit's capabilities (guiding the placement of UI elements and dynamic adaptation of ergonomic 3D UIs). To evaluate the utility of the toolkit, we conducted a walkthrough demonstration with experts (UI designers and HCI practitioners).

Paper D *AUIT – the Adaptive User Interfaces Toolkit for Designing XR Applications* - The solution from paper D is a toolkit (as in paper C), so we follow the same literature to design the evaluation [153]. In this case, we also conduct an evaluation with experts. However, as we also wanted them to try out the toolkit, we designed a study that combines a usability study and a walkthrough demonstration. Here, participants had the opportunity to design adaptive MR UIs with guidance on using the toolkit along the way.

Chapter 8

Conclusion

It is indisputable that for everyday MR to succeed as the next mainstream Human-Computer interaction paradigm, UIs must adapt to the user's context [1, 101, 163]. The comprehensive approach to the design of context-aware adaptive UIs employed in this thesis made it possible to consider their whole design and development process while exploring some parts of the problem in greater detail. Our initial work on paper A presented a novel approach to gathering a new category of context (near-symmetrical objects), combining information from multiple context sources to disambiguate the symmetry of objects. At the same time, it highlighted challenges in capturing context - in particular, how context availability can be a barrier to developing context-aware UIs. More concretely, complex setups and development procedures that do not easily generalize make it harder for creators to get started and pose limitations to the functionality they can achieve. This hinders creativity and innovation. Paper B was our attempt at addressing some of these difficulties - our proposed system relies on a simpler setup that uses off-the-shelf components and the tracking pipeline it uses does not require data manual acquisition and labeling typically necessary when extending computer vision models. These contributions advance the ability to capture and use context in the design of adaptive UIs, and I cover this topic extensively in chapter 3. However, as motivated in chapter 5, access to context does not mean the data contains information that can directly translate into an adaptation. This is where computational interaction approaches such as the ergonomic model proposed in paper C can be valuable. Furthermore, computational interaction approaches are crucial to scale up adaptive UIs that consider multiple objectives. Paper D proposes a flexible approach for such multi-objective UI adaptations, along with a framework that identifies the components of an adaptive UI. It provides a holistic overview of adaptive UIs for MR that I describe in greater detail in chapter 5. The framework is the culmination of the work conducted throughout this thesis, providing a separation of concerns for UI adaptations and a starting point for creators through the toolkit we develop (AUIT). AUIT is one of the initiatives among others in this thesis to make UI adaptations more accessible to creators. In chapter 6, we describe the steps taken in this direction in papers B, C, and D. In the following section I will revisit the research problems

investigated in this thesis and discuss how each paper and chapter contribute to solving them. Finally, I'll reflect on how my work can inspire future research.

8.1 Revisiting the research problems

In this section, I will revisit the research problems identified in the introduction chapter of this thesis. Although I discussed these throughout the dissertation, I will summarize how each publication contributes to advancing knowledge and understanding of each problem.

RP1: How to make relevant context categories available to creators during the development of adaptive MR UIs? Context-awareness is a requirement for enabling MR adaptive user interfaces. In chapter 3, I overview relevant context categories for MR and propose novel methods to capture and make context available to creators. In paper A, we contribute a multi-camera tracking pipeline to disambiguate the orientation of near-symmetrical objects. At first glance, this might seem like a niche application, but a focus+context approach to context sensing can generalize to other context retrieval problems. Another barrier to data-driven context-collection methods we encountered while working on paper A (reinforced by our industry partners) are the laborious data capturing and labeling processes that these entail. The pipeline we contribute in paper B addresses this challenge - here we focus on making these context collection methods more accessible to creators by automatizing the data capture and labeling process. Another step towards making context data more accessible during the development of adaptive UIs is the combination of context widgets and adaptation objectives proposed in AUIT, the toolkit we contribute in paper D. Here, we allow creators to seamlessly use context data by selecting which adaptation objectives the UI must fulfill.

RP2: How can computational interaction methods provide a further understanding of context and optimize MR UIs? In many cases, access to context data is insufficient to make good adaptation decisions, as motivated in chapter 4. Computational interaction methods - where algorithms, tools, and mathematical models allow artifacts to adapt to the user's context - can help address this problem. In paper C, we propose an approach to analyze the user's interaction space regarding ergonomics. This contribution demonstrates how a model can support context understanding of a context source that would be hard to leverage for informing UI adaptations - the user's arm dimensions. Furthermore, we contribute an optimization approach to identify the most ergonomic interaction zones when considering constraints that can guide adaptive MR interfaces at runtime. We contribute to this research problem further in paper D, proposing a flexible real-time multi-objective optimization approach to create adaptive MR UIs.

RP3: What are the components of a context-aware MR UI adaptation? In chapter 5, based on the framework proposed in paper D, we separate the concerns of MR UI adaptations and identify five components. These are:

- **Context widgets**, to process sensor data into relevant abstraction levels.

- **Adaptation objectives**, which describe desired adaptation behaviors.
- **Solvers**, to find appropriate adaptation proposals.
- **Adaptation triggers**, which contain the logic for invoking the solver and adapting the UI.
- **Property transitions**, to define how UI elements transition from one adaptation state to another.

In paper D we instantiate these components through a toolkit, and a study with experts demonstrates how these are conceptually clear and can be used to create adaptive UIs in MR scenarios.

RP4: How to support creators in applying computational methods and facilitate the creation of context-aware adaptive user interfaces? Supporting creators of MR content during development is crucial for MR to succeed as the next computing platform. Making our approaches available to creators was a priority in the publications that constitute this dissertation, as described in chapter 6. In paper B, our proposed pipeline allows creators to extend context retrieval to other near-symmetrical objects. Paper C supports the understanding of various ergonomic metrics through an interaction space visualization and makes these accessible during development through an API. Finally, the toolkit we developed in paper D is available as a plugin for Unity, a popular development platform for XR applications. The contributions in these publications are examples of how we can support creators by allowing our methods to generalize to more use cases (paper B), supporting understanding of computational models and making them accessible during development (paper C), and enhancing existing development tools with approaches that simplify the formulation of optimization problems and make a wide range of techniques available and possible to test during development, fostering creative exploration of ideas (paper D).

8.2 Future Work

The work presented in this dissertation allowed us to identify gaps in the literature and new directions for future work. Analyzing context awareness in MR, it is possible to notice that many context categories are difficult to infer from the user's environment even when using specialized equipment. Therefore, exploring new methods to access context that are easy to integrate into MR systems is crucial. This involves advancements both only in terms of sensing and abstracting sensor data. In paper A, we proposed a two-stage approach that combines data from multiple sensors to make a final assessment. However, it requires a complex and specific setup - automatizing context collection using multiple sensors is a promising research direction.

Optimization of adaptive MR UIs is a research topic still in its infancy. The approaches we use throughout this dissertation reveal many other underexplored possibilities. A clear example is how UI adaptations nowadays reduce to adapting to a single specific solution. However, in multi-objective problem formulations,

it is possible to have multiple optimal solutions. In particular, when objectives are conflicting and optimal solutions consist of a trade-off between objectives, it is relevant to explore techniques to allow users to choose the preferred solution. This is a topic I started exploring recently with a PhD student that recently joined the group, and the first outcome of our collaboration is an extended abstract (paper H). A close topic I plan to explore relates to how user preferences influence UI adaptations. I envision intelligent UIs that learn from the user's behavior and adapt according to the user's preferences. Reinforcement learning is a promising approach to achieving this goal that I explored during my stay abroad at Aalto University, but this direction is still a work in progress. Models to make informed adaptation decisions based on context, such as the work in XRgonomics (paper C), are also a promising direction for further exploration. In paper C, we explore the interaction cost in different zones of the interaction space in terms of ergonomics of the upper limbs for static poses, but it would be great to consider human motion too. Another extension would be to add support for other factors relevant to MR, such as visibility.

Finally, we implement many components in the toolkit proposed in paper D, prioritizing functionality. In such a big project, we had to allocate how much effort to put into some components over others. Therefore, another promising direction for future work would be implementing more adaptation triggers or property transitions. It would also be valuable to learn how other solvers fare in terms of performance and quality of solutions, a topic we already started exploring in paper H with genetic algorithms. Last, out-of-the-box support for more context categories could make the toolkit more valuable and open up new possibilities for implementing new adaptation objectives.

8.3 Final Remarks

Although extremely hard to predict how long it will take, I am an avid believer that MR will be the user interface of the future. If used for good, such an interaction paradigm can give users super-human capabilities, such as enhanced memory and cognition or improved senses. It is clear, however, that we are still far away from such a reality - many technological advancements will be necessary: displays, compute power, batteries, tracking, and of course, interactions and interfaces. This dissertation is my attempt to advance our knowledge of MR user interfaces, which as motivated throughout this work, must be context-aware and adaptive. Last, out-of-the-box support for more context categories could make the toolkit more valuable and open up new possibilities for implementing new adaptation objectives.

Part II

Publications

Chapter 9

Paper A

This chapter presents the paper *Digital Assistance for Quality Assurance: Augmenting Workspaces Using Deep Learning for Tracking Near-Symmetrical Objects*, published in *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces*: ISS 2019. This paper won a best application paper award.

[17] João Marcelo Evangelista Belo, Andreas Fender, Tiare Feuchtner, and Kaj Grønbæk. Digital assistance for quality assurance: Augmenting workspaces using deep learning for tracking near-symmetrical objects. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces*, pages 275–287, 2019.

Digital Assistance for Quality Assurance: Augmenting Workspaces Using Deep Learning for Tracking Near-Symmetrical Objects

João Marcelo Evangelista Belo, Andreas Fender, Tiare Feuchtner, and Kaj Grønbæk

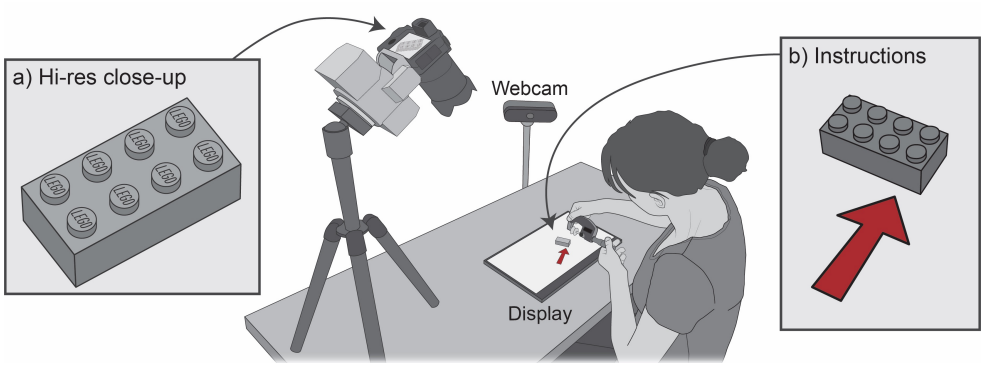


Figure 9.1: In the explored use-case, a worker needs to measure exact distances between different pre-defined points on a near-symmetrical LEGO brick. We present digital assistance for this metrology task by displaying situated step-by-step measurement guides on a tabletop-display. (a) While webcams locate the brick, a zoomed-in camera on a pan-tilt unit rotates towards the brick to identify its unique orientation based on fine-grained features (a LEGO logo in this case). (b) Based on the tracked unique orientation, situated guides can indicate the correct points to measure.

Abstract

We present a digital assistance approach for applied metrology on near-symmetrical objects. In manufacturing, systematically measuring products for quality assurance is often a manual task, where a main challenge for the workers lies in accurately identifying positions to measure and correctly documenting these measurements. This paper focuses on a use-case, which involves metrology of small near-symmetrical objects, such as LEGO bricks. We aim to support this task through situated visual measurement guides. Aligning these guides poses a major challenge, since fine grained details, such as embossed logos, serve as the only feature by which to retrieve an object's unique orientation. We present a two-step approach, which consists of (1) locating and orienting the object based on its shape, and then (2) disambiguating the object's rotational symmetry based on small visual features. We apply and compare different deep learning approaches and discuss our guidance system in the context of our use case.

9.1 Introduction

In manufacturing, metrology is the activity of measuring objects as a part of standard Quality Assurance (QA) procedures. Nowadays, even though industrial metrology tasks are increasingly automated, many still require manual work. While robots can support the overall procedure, e.g., by pre-sorting the objects, many of the actual measurements are conducted by workers, as was observed in real-world use cases in companies associated with the Manufacturing Academy of Denmark (MADE)[1]. The current procedure in these companies involves following electronic instruction manuals that are viewed on a desktop screen. Some measurement tools provide the capability of digitally transmitting data, whereas others require manual input of measurements to a computer database. Conventional systems are not aware of what instruction is being followed, hence even when the measurement tool can send the value to the system, the worker must still specify what measurement position the value corresponds to. In other words, the worker must associate the schematic drawing in the instruction manual to the object being measured and determine the respective mental rotation to know which positions to measure. In particular, near-symmetrical objects pose challenges, since it is difficult to identify the measurement points quickly and accurately with the human eye.

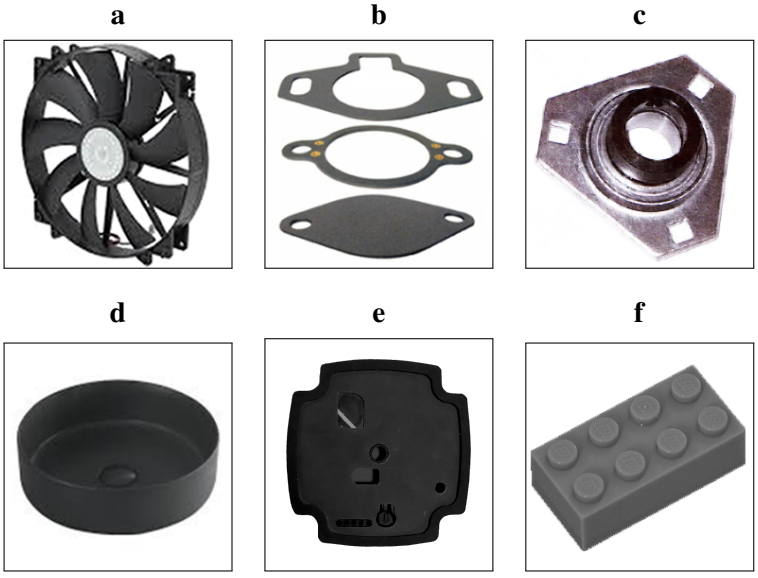


Figure 9.2: Examples of near-symmetrical objects, including components of fans (a), thermostats (b), pumps (c,e), plumbing (d), and a LEGO brick (f). The shape of each object has different degrees of rotational symmetry. Only a couple small visual features on each object allow to determine its unique orientation.

[1]Manufacturing Academy of Denmark: https://www.made.dk/

In this context, "near-symmetrical" means, that the overall shape of the object has rotational symmetry. Its unique orientation can be identified only by small visual features: either by their locations on the object, or by determining the orientation of a non-symmetrical feature. We have come across a multitude of such near-symmetrical objects in industrial manufacturing (for examples see Figure 9.2). Each of these has at least one visual feature that, through careful inspection, permits identification of the object's unique orientation. Such features may be a single adjustment screw on one side of a shaft, a notch or pin that prevents wrong insertion of a component, a serial number, etc. In the example of a 2x4 LEGO brick (Figure 9.2, f), the symmetry-breaking feature is the LEGO logo (Figure 9.1, a).

To better facilitate metrology of such small and near-symmetrical objects, we propose the digital guidance system shown in Figure 9.1, which dynamically provides step-by-step instructions for a given metrology task. We further propose to provide these measurement instructions as visual guides that are situated in close proximity to the measured object. This aims to prevent the repeated attention-switches between object and desktop display, which can increase the worker's time and energy demand [140, 223]. Furthermore, by aligning the guides with the object's current orientation, we strive to reduce workers' cognitive load, decreasing the need of mental rotations [50, 239].

In this paper we discuss the visualization of measurement guides situated in the task space and present a tracking technique for near-symmetrical objects that need to be measured. Our approach supports automatic detection of small symmetry-breaking features through computer vision and deep learning, which allows us to identify a object's unique orientation. We devised a two-step tracking solution that computes the position of the near-symmetrical object in the whole tracking area (context), and resolves the ambiguity of its rotation (focus). We refer to this as *Focus+Context tracking*, analogous to Focus+Context output [14].

We aim to support workers performing a metrology task by:

1. Providing assistance in disambiguating the object's orientation, which is challenging due to its near-symmetrical characteristics.

2. Displaying situated measurement guides superimposed on or in close proximity to the object, to reduce the frequency of switching between *information* and *workpiece* tasks [190].

3. Presenting the measurement guides corresponding to the object's current orientation, to reduce the cognitive load of applying mental transformations [288].

In the following sections, we elaborate on our Focus+Context pipeline and its application to the described real-world use case. We first present a pipeline for tracking a LEGO brick on a horizontal display to render co-located instructions. Thereafter, we present a generalized variation of the pipeline for tracking a handheld brick and discuss the deep learning techniques that both pipelines utilize. We test a number

of hypotheses about training procedure refinements for orientation disambiguation, through ablation studies. Finally, we discuss the generalizability of our approach to objects with different shapes, sizes, and visual features, and present the tracking results for the pump component *e* in Figure 9.2 as an additional example.

9.2 Related Work

The term "Focus+Context Tracking" used in this paper, is inspired by the work of Baudisch et al. [14], where a screen consists of low-resolution regions providing context and high-resolution regions for focus information. Focus+Context tracking can be seen as a metaphor of the same concept, applied to input devices used for tracking, instead of output devices.

The general approach of using multiple cameras to capture different levels of detail has already been investigated [2]. We follow an approach similar to the one used for marker tracking by Rekimoto et al. [220], where a fixed camera is responsible for tracking an entire tabletop surface and a high-resolution pan-tilt camera performs marker recognition. However, while markers are optimized for tracking, we tackle the more challenging problem of estimating the orientation of a marker-less, near-symmetrical object. In other words, we detect the position and orientation of an object based on its shape and small visual features in an image. Previous work has investigated detecting the 2D orientation of a texture, or parts of a texture, e.g., based on gradient vectors [32], or principal directions [134]. These techniques work for 2D rotations in image space, which implies that their applicability is limited, when trying to detect the orientation of a texture seen from an oblique angle. Furthermore, in our case the object features a slightly reflective material that causes view-dependent highlights in the image.

In the scenario presented in this paper, estimating a 2D rotation of the visual feature in image space is not sufficient. We therefore devised a solution with deep learning-based vision techniques.

Computer Vision and Deep Learning

To detect and identify the object's orientation, we apply deep learning in our tracking pipeline. In this regard, the work of Krizhevsky et al. [146] has led to significant breakthroughs in image recognition using Convolutional Neural Networks (CNNs). Since then, CNNs have proven to be highly successful in other image recognition tasks, such as object detection [217, 221], instance segmentation [111], and pose estimation [268]. The accuracy and efficiency of CNNs have increased substantially over the years, due in part to improvements in the architectures of these networks [110, 240]. Furthermore, techniques such as transfer learning [66, 216, 281] allow for improved generalization when the size of the dataset is small, and Kornblith et al. [143] found a strong correlation between accuracy on the ImageNet dataset [146] and

transfer learning accuracy, when fine-tuning or using pre-trained networks as feature extractors.

Augmented Workspaces

Augmented environments that seamlessly combine the virtual and real world have been envisioned since the early 90's, exploring how everyday environments could be augmented to improve people's lives and the way they work [215, 271]. Since then, researchers have proposed systems like the *DigitalDesk* [272], where the user can interact with digital information that is superimposed on conventional paper. *Augmented Surfaces* [220] follows up on the idea of projecting virtual content onto a desk to augment a meeting room, allowing users to utilize their environment as an extension of their laptops and attach data to physical objects.

Even though our solution is technically not augmented reality (AR), there are many related AR systems with similar goals and characteristics [45, 202, 210, 219]. The effectiveness of AR in industry has become an active topic of research over the past few years. For example, Baird and Barfield [12] showed that workers using AR would complete assembly tasks faster and with fewer errors. A study on object assembly [251] provided additional evidence for this and demonstrated that AR can also reduce cognitive load of the worker performing the task. Henderson and Feiner [116] similarly demonstrated that AR assistance in a procedural task can increase the workers' performance and that co-located instructions lead to fewer head movements. Furthermore, they found similar benefits of using AR during maintenance tasks [118]. More recently, Uva et al. [263] conducted a study on the effectiveness of spatial augmented reality in manufacturing, providing evidence that co-located technical information greatly reduces the complexity of the tasks, improving completion times and lowering error rates, when compared to paper-based instructions. Finally, Polvi et al. [211] confirmed that an AR interface can also be beneficial in inspection tasks, resulting in lower completion times, fewer errors, fewer gaze shifts, and a lower subjective workload.

To our knowledge, there is no existing research on augmenting the workplace to specifically support metrology tasks. Assembly and maintenance tasks are related, in that most activities are performed in a predictable environment and are part of a procedural task. Furthermore, inspection tasks entail a similar step of information matching as in metrology. However, our use case of manual metrology poses the need for accurate pose estimation of near-symmetrical objects, which goes beyond related research.

9.3 Use case

In connection with the MADE project, we explore QA processes at multiple manufacturing companies, where workers manually conduct metrology on various near-symmetrical objects. In this paper, we focus on a single use-case of applied metrology

at the LEGO Group. In the presented use-case, workers employ a range of specialized tools for measuring objects. Some of these tools are still analog and require manual input of numbers into the database. Digital measurement tools allow to directly transmit the measured values to the database. However, the worker still has to indicate which measurement step (i.e., which field in the database) the value corresponds to. Thus, to ensure correct recording of measurements, the workers currently measure certain positions, following a strict order. This order is indicated in an electronic instruction manual (i.e., pdf), which includes a schematic drawing of the object with numbered measurement positions. A computer is used to display this manual and the database with measurement entries. Mouse and keyboard serve as input devices for navigation and entry of measured values.

Since in this use-case most products are small and near-symmetrical, the worker must carefully inspect each object to correctly orient it, before being able to accurately identify the next position at which to take a measurement. Within the LEGO Group, we focus on a common near-symmetrical object that undergoes rigorous QA procedures - a 2x4 LEGO brick, which is simply referred to as *brick* in the remainder of this paper.

9.4 Digital assistance: user interface

Our digital assistant displays situated instructions for metrologists. To ensure that the workers obtain all required information about the task at hand, the interface features an *overview panel* (see white panels in Figure 9.3, left). This contains textual information similar to the original instruction manual, i.e., describing the type of measurement to take and what tool to use. Furthermore, it communicates how many measurements are left in the current stage, and shows the last saved measurement. This panel further contains a schematic 3D representation of the object (e.g., the LEGO brick), which reflects the orientation of the tracked object. Measurement guides on this representation indicate which point currently needs to be measured.

The remainder of the screen surface is reserved for displaying co-located measurement guides when the object is placed on the screen. In this manner they indicate measurement positions directly on the physical object (see Figure 9.3, top). Both, the co-located guides and the oriented schematic are only displayed when the system is certain about the actual orientation of the tracked physical object, since it is crucial for the instructions to always be displayed on the correct side.

Measurement guides consist of a pair of arrows. Whenever co-location of guides is not possible, the worker can instead refer to the guides on the schematic representation in the left part of the GUI. This occurs either when the brick is handheld, or when the current instructions would need to display arrows on top of, or underneath, the object (e.g., when measuring height). For instance, in Figure 9.3 (bottom), both of these conditions are met. We will elaborate on this in the *Handheld mode* section. In each step, only one pair of arrows is displayed at a time, indicating the measurement that should be taken. When measuring with an analog caliper, the worker can input the

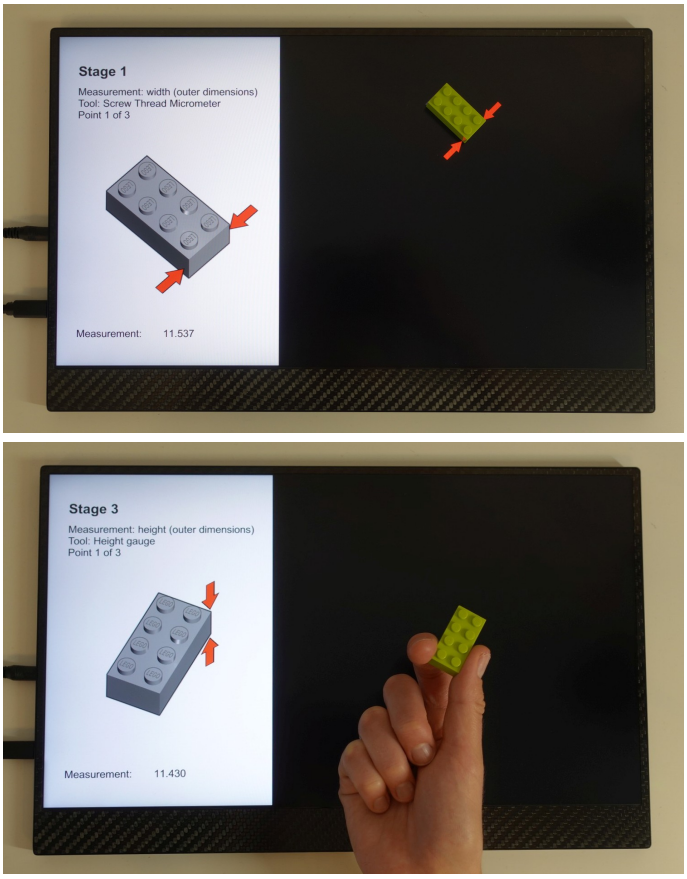


Figure 9.3: User interface of the digital assistant. The left panel shows textual instructions and an enlarged schematic representation of the object, which always reflects the orientation of the actual tracked object. Measurement guides in form of red arrows indicate the current points to measure. Top: In the right part of the screen, the guides are shown co-located with the physical object. Bottom: If the current instructions cannot be co-located or the object is handheld, the left panel still shows the instructions. In both cases, rotating the tracked object will rotate the schematic representation on the left.

measured value with a keyboard, and hit Enter to save it. When using a digital caliper that is connected to the system, the current measurement is saved automatically upon pressing a button on the device. The system then automatically transitions to the next step showing guides for a new point to measure.

9.5 Architecture

In this section, we provide an overview of the hardware and software components of our prototype, and describe the interplay between these.

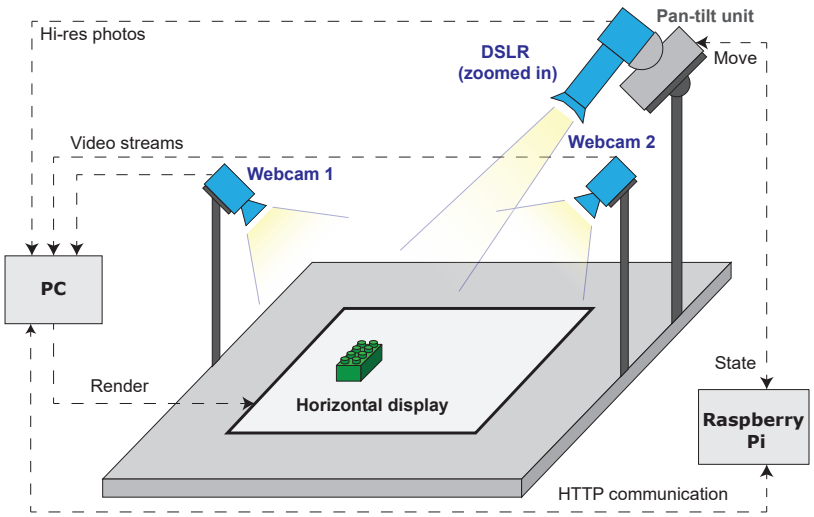


Figure 9.4: Overview of hardware and software components. Two webcams cover the entire tracking area. A zoomed-in DSLR camera provides high-resolution pictures of the tracked LEGO brick. The camera is mounted on a pan-tilt unit, which is controlled by a Raspberry Pi. This allows keeping the brick in focus even when it is moved. The PC controls the overall system flow and renders instructions on a horizontal display, so that they are co-located with the brick when it is placed on the screen.

Overview of the tracking pipeline

An overview of our hardware and software components is provided in Figure 9.4. The video streams from two webcams are used to track the location of the brick on a tabletop screen surface. Furthermore, the brick's ambiguous orientation can be retrieved from these video streams: at this point the orientation can only be defined up to symmetry due to the 180° symmetrical shape of the brick. In a second step, the DSLR camera is oriented towards the brick's position with the help of an underlying pan-tilt unit. To do so, the main PC calculates the necessary rotation and forwards these values to a Raspberry Pi via network. This in turn controls the pan-tilt unit, to ensure continuous tracking of the brick. The overall camera setup can be described as a *master-slave configuration* [2, Ch.8.4], with two webcams as *master* and the DSLR camera as *slave*.

The DSLR camera periodically takes pictures of the brick. The zoom level and resolution of these pictures is sufficient to identify small symmetry-breaking features on the brick, such as a LEGO logo (see Figure 9.6). Such features allow to disambiguate the orientation of the brick. Once the 2D position and unique rotation of the object are known, measurement guides can be displayed accordingly.

The following section provides further details on the individual steps of our tracking pipeline. Additional information on the specific hardware and software

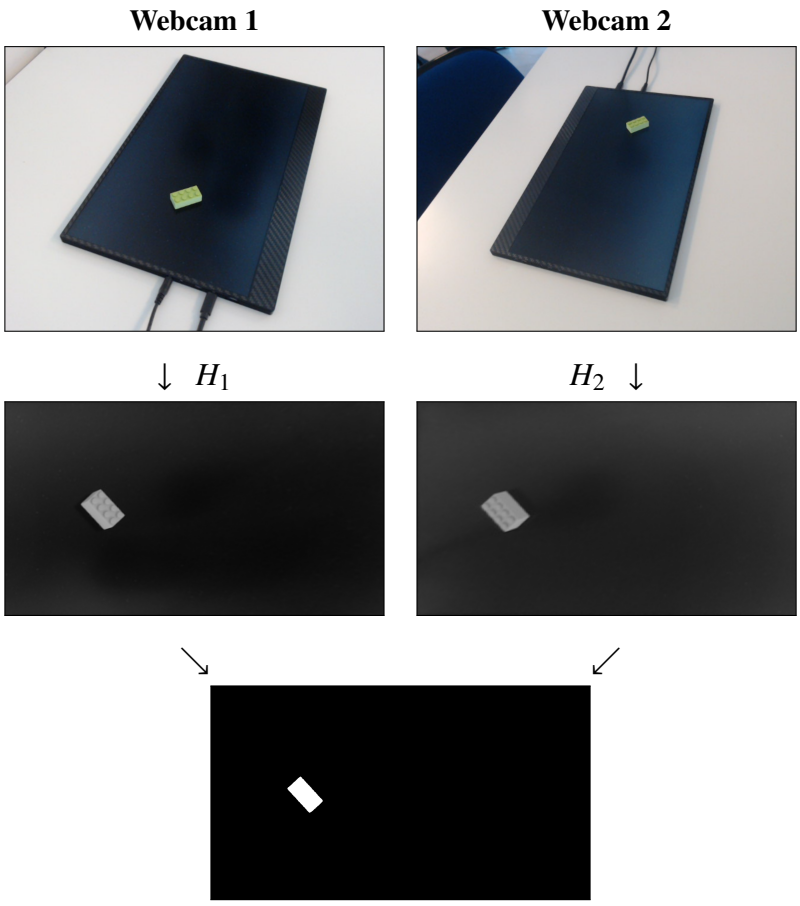


Figure 9.5: Our context tracking sub-pipeline is based on conventional image processing techniques. Polarizing filters make the screen contents appear black (top). Both video feeds are perspectively unwarped using homographies H_1 and H_2 (middle). This brings both feeds into the screen space of the horizontal display. The images are then thresholded and combined, to create the mask of the brick in image space (bottom). The output is the position and ambiguous orientation of the brick.

components that we used may be found in the *System Implementation* section.

9.6 Focus+context tracking

The idea of the pipeline shortly described above is to divide the tracking task into two separate steps: (1) The *context* step tracks the location and symmetric orientation of the brick continuously, based on a simple and fast approach using conventional computer vision techniques. (2) The *focus* step disambiguates the brick's orientation. It is triggered less frequently and is based on deep learning. This section explains

each of these steps in detail.

Context tracking

To locate the brick, two context cameras stream their video feeds to the main PC (see Figure 9.5, top). We attached polarization filters to the cameras, so that all content on the tabletop screen appears black in the video feeds [214]. This way, co-located instructions will not interfere with the tracking. By applying (pre-calibrated) homographies to each stream, both video feeds are warped into screen space (see Figure 9.5, middle). The brick can then be segmented in each feed simply by binary thresholding. These thresholded images are combined with an AND-operation on each pixel. The resulting binary image is then searched for a mask that has 4 corners and a (rotated) rectangular shape, to exclude other potential objects on the screen. The center of this mask corresponds to the brick's position, and its rotation can be identified by averaging the angles of its two long edges. However, as mentioned before, the rotation is still ambiguous at this point, since the brick yields identical thresholded images when it is rotated by 180° (see "Context" image in Figure 9.7). Overall, this approach for context tracking requires very little computational power and can therefore output the position and orientation of the brick in real-time.

Focus tracking

Once the position of the brick is known, the pan-tilt unit can orient the DSLR camera towards it. This camera takes a picture every 2 seconds and transmits it to the main PC. Figure 9.6 shows two examples of raw images provided by the camera, of a 2x4 brick that is rotated by 180°. As with the context cameras, a polarization filter makes the screen beneath the brick appear black. The raw image is then passed to a Mask R-CNN [111] instance segmentation model. In contrast to simpler techniques such as chroma keying, this approach allows the system to effectively detect the brick even when other objects are in the picture, or when the object is partially occluded.

After performing instance segmentation, the picture is cropped and the values of all pixels outside the segmented mask are set to 0. The cropped picture is then fed into an additional CNN to disambiguate the object's orientation. This problem was solved using a 50-layer residual network architecture [110]. As shown in the "Focus" illustration in Figure 9.7, the classifier returns one of 4 different classes, depending on the orientation of the LEGO logo: up, down, left, and right.

Combining focus and context

In the final step of our tracking pipeline, the outputs of *focus* and *context* are combined. While the position of the brick can be retrieved directly from the context tracking step, the orientation results from a combination of both sub-pipelines, as is illustrated in Figure 9.7. The output of the context tracking consists of two vectors, indicating two possible orientations (the blue and orange arrows in the "Context" image of

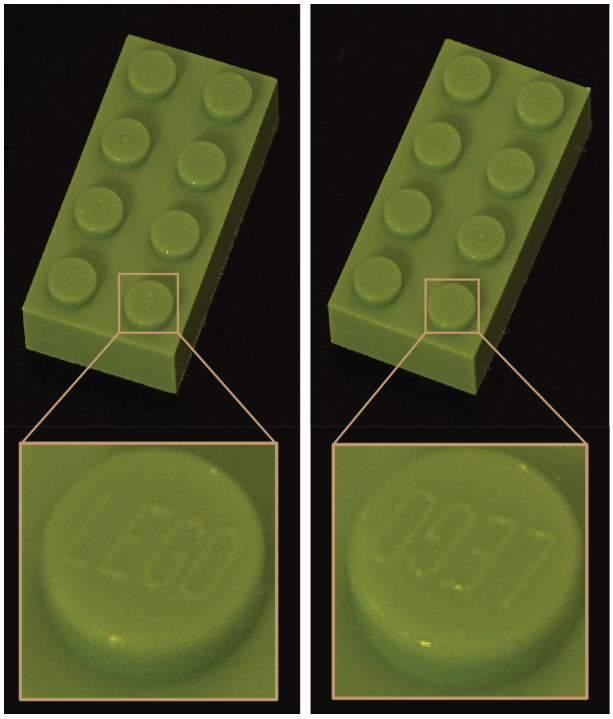


Figure 9.6: Raw pictures from the zoomed-in DSLR camera. For better print quality, we adjusted the aperture and increased the exposure time compared to the values we use at run-time. Furthermore, we cropped the resulting images. The left and right picture lead to the same orientation in the *context* tracking step. However, the upright LEGO logo on the left and the upside-down logo on the right allow disambiguation between both possible orientations in the *focus* step.

Figure 9.7). The output of the focus tracking is one vector, indicating one of four main directions (see Figure 9.7, "Focus" image). We then form the dot product of each vector from context tracking and the single vector resulting from focus tracking, and we choose the context vector that results in a positive dot product (blue arrow in the "Result", Figure 9.7). Even in the rare case when the output of the context tracking is in between classes (e.g., exactly between pointing up-right and down-left) and the focus tracking is undecided between two classes, the end-result from the dot product will still be valid. We will elaborate on this in the *Discussion and future work* section.

The resulting direction vector is used to calculate the unique orientation of the brick, to properly align the measurement guides.

9.7 Handheld mode

In the previous sections, we described an easy-to-replicate setup, which utilizes the polarized light from a horizontal display to segment the object from the background.

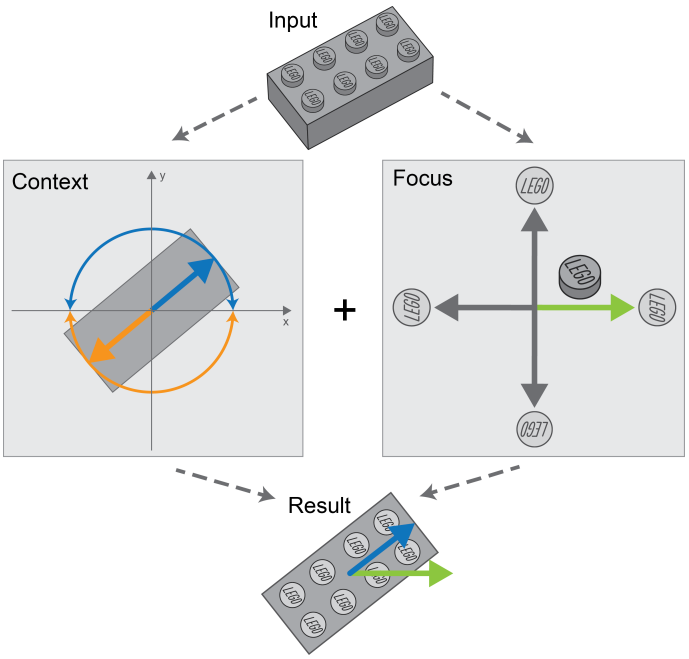


Figure 9.7: Combination of Focus+Context to retrieve the unique orientation of the brick. The output of the context pipeline is an ambiguous orientation: in the "Context" image the orientations marked by blue and orange arrows lead to equivalent results. The output of our focus classifier is one of four directions ("Focus" image). We then choose the direction from the *context* output that has a positive dot product with the *focus* output. In the illustrated example, the brick's orientation corresponds to the blue vector in the "Result" image.

This makes the context pipeline simple and computationally fast. However, an obvious limitation of this approach is that the object must always be placed on top of the display, in order to receive situated instructions. Furthermore, in a preliminary interview, workers were concerned that continuously gazing down at the tabletop throughout an entire work session might cause neck strain. With this limitation and the workers' concerns in mind, we created a variation of the pipeline that allows the object to be handheld and instructions to be displayed on a separate vertical screen.

To achieve this, we generalized the context tracking step of the pipeline, by basing it on the Mask R-CNN instance segmentation model, instead of simple binary thresholding. This makes it possible to correctly detect the object in more challenging scenarios, e.g., when it is partially occluded due to being held by the worker, or being partly encompassed by a measuring tool. The same model, which we use for segmenting the object in the DSLR camera image, can directly be applied in real-time to the footage of the two webcams.

Figure 9.8 (top) shows tracking of a handheld object by combining the segmentation

results from both webcams. Based on the camera positions, intrinsic parameters, and bounding boxes in each camera stream, we can estimate the position of the object. For each webcam, we create a ray from the camera's position through the center of the detected bounding box in the image plane. This creates skewed 3D rays, i.e., they are neither parallel, nor do they intersect, since the centers of the bounding boxes are rarely located at the exact same points on the brick. Based on the line equations, we then find the point with minimal distance to both rays. This gives us the object's position, which is forwarded to the pan-tilt unit controller for orientation of the DSLR camera.

The next step is to identify the ambiguous orientation of the object. In the image of the the zoomed-in DSLR camera, we approximate a polygon around the segmented object in image space and take the longest edge as orientation indicator. We then calculate the orientation in world space by making two assumptions: (1) Due to the fact that the DSLR camera is zooming in on the small object, we can assume an almost orthographic projection of the object in the segmentation. (2) With our chosen set of instruction steps, the orientation will vary only around the y-axis (up-axis). Based on these assumptions, we can simply transform the direction of the longest edge into world space, using the known extrinsic parameters of the DSLR camera. Finally, we resolve the near-symmetry as in the previously presented pipeline.

With this approach, the non-co-located instructions for a handheld brick can be presented in correct orientation corresponding to that of the tracked object. This is illustrated in Figure 9.3 (bottom) and Figure 9.8 (bottom). Workers can switch between these modes as desired: they can trigger co-located guides by placing the brick on the horizontal screen, or they can look at the vertical screen to reduce neck strain.

9.8 System Implementation

This section provides details about the frameworks, engines, and hardware that our particular implementation of the architecture is based on. While the pipeline is not bound to the specific set of software and hardware components described here, these choices were useful for an effective proof-of-concept setup.

The overall pipeline and the rendering is implemented in *Unity 3D*. The measurement guides are displayed on a 15.6" portable screen, which can be used as a horizontal tabletop display, or positioned vertically. The worker can input and save measurements in the system through a digital measuring device, such as the *Mitutoyo* micrometer (series 406), or a traditional keyboard. Pressing a button on the digital measuring device emulates keyboard inputs with the digits of the measurements followed by Enter. Alternatively, a foot pedal could be used to perform this button press.

We use the *Velt* Framework [78] to handle the data flow of the system and the communication between its various components. This node-based framework is a *Unity 3D* plugin and simplifies the creation and inspection of data flow pipelines, including

Figure 9.8: Top: Alternative context tracking pipeline. When tracking the object based on image segmentation within each of the two webcam streams (left and right), the 3D position of the object can be estimated with the intrinsic and extrinsic parameters of the two webcams. Bottom: Alternative display setup. The system can alternatively be used with a vertical screen, or both a vertical and a horizontal screen simultaneously. In each arrangement, the displayed guides are always presented in accordance with the object's current orientation.

pre-processing, network communication, etc. The context tracking is implemented as a specialized Velt extension, but also based on built-in nodes, e.g., nodes that wrap *OpenCV* functionalities.

We use a *Raspberry Pi Model 3 B* for receiving HTTP requests and for interfacing with a *Maxwell MPR-202* pan-tilt unit. This serves to correctly orient the attached focus camera, which is a *Sony RX10 II* DSLR camera. To trigger rotations of the

DSLR camera, the Raspberry Pi controls relays, opening and closing circuits on the pan-tilt unit's DIN7 socket. Since the pan-tilt unit only supports relative movements and does not have a built-in sensor to provide its pan and tilt values, we attached an accelerometer (*MPU-9160*) to calculate its current orientation. Thus, when an absolute desired orientation is forwarded to the Raspberry Pi (based on the tracked object's position relative to the pan-tilt unit), it rotates the pan-tilt unit until the requested orientation is reached, so that the DSLR camera is oriented towards the tracked object. We take the high-resolution pictures with an ISO value of 640, an exposure time of 0.1 seconds, and an f-number of 3.2. These values only serve as an orientation, as the robustness of the pipeline does not heavily depend on the camera settings, as long as the symmetry-breaking features (e.g., LEGO logo) are visible in the picture. We then use the *Sony Imaging Edge Remote* tool [243] to automatically take pictures and periodically transmit them to our system via USB. Another specialized Velt node receives these pictures and triggers the focus part of our tracking pipeline.

All deep learning components are implemented in Python and the central pipeline communicates with these via HTTP. We use the *PyTorch* framework [213] to implement and train our models and we follow a training procedure inspired by He et al. [115]. All evaluations of our system were conducted on machines with two *Nvidia RTX2080ti* graphics cards.

9.9 Technical Evaluation

In this section, we evaluate the accuracy of our deep learning models. To train our models we gathered two different datasets, which are described in the following subsections.

Instance Segmentation Model

We start by describing the training procedure for the Mask R-CNN model that was used for instance segmentation. For this problem we used a training dataset with 90 pictures and a validation dataset with 20 pictures, which were annotated using the VIA annotation tool [69]. Our Mask R-CNN model uses a Feature Pyramid Network [159] backbone architecture based on a 50-layer residual network [110]. We used a model that was pre-trained on the COCO dataset [160]. This model was trained over 50 epochs using stochastic gradient descent with momentum, at an initial learning rate of 0.005 divided by 10 every 13 epochs, a weight decay of 0.0005, and a batch size of 2. After training, our Mask R-CNN detector achieves a segmentation mAP of 88% and a mask mAP of 87% on the validation dataset, which is robust enough for our segmentation needs.

Orientation Model

The orientation model is responsible for disambiguating the orientation of the tracked object. For the orientation problem in our specific use case we had a training dataset

	Classification			
	D=50 (120e)	D=100 (60e)	D=200 (30e)	D=400 (15e)
Baseline	0.45 ± 0.02	0.46 ± 0.03	0.45 ± 0.03	0.46 ± 0.03
+ Transfer learning	0.82 ± 0.06	0.96 ± 0.02	0.95 ± 0.02	0.96 ± 0.01
+ Rotation	0.91 ± 0.04	0.98 ± 0.01	0.97 ± 0.01	0.97 ± 0.01

	Regression			
	D=50 (120e)	D=100 (60e)	D=200 (30e)	D=400 (15e)
Baseline	0.35 ± 0.04	0.31 ± 0.02	0.31 ± 0.04	0.29 ± 0.01
+ Transfer learning	0.66 ± 0.10	0.85 ± 0.06	0.85 ± 0.04	0.82 ± 0.05
+ Rotation	0.79 ± 0.03	0.86 ± 0.06	0.90 ± 0.03	0.85 ± 0.05

Table 9.1: Evaluation results. D stands for size of the dataset, followed by the number of epochs. Each experiment was executed 5 times and we report the average accuracy. The best results were obtained when using a classifier and a training procedure using transfer learning and rotation as part of the augmentation techniques. Higher accuracy was obtained when the dataset had at least 100 samples.

with 400 images and a validation dataset with 192 images. Each image had the ground truth of the 2D pose of the brick. In this section we will test the following three hypotheses related to this model:

(H1) *With a small dataset, using transfer learning improves accuracy.*

Models pre-trained on ImageNet [54] tend to lead to improved performance for diverse image classification tasks [66, 216]. However, recent research [143] has demonstrated that, for some small fine-grained image classification datasets, the benefits of transfer learning are minimal.

(H2) *Augmenting data with random rotations leads to higher accuracy.*

Rotation in image space is an augmentation technique that has been used successfully in previous work [208]. We expect that such an augmentation is particularly beneficial when training a model that predicts the orientation of an object.

(H3) *Solving our problem using a regression loss function leads to better performance compared to a classification loss function.*

Since the goal of regression is to predict the exact orientation of the object, we expect it to be more accurate when comparing to solving the problem for classification.

Considering that classification alone would not be sufficient to get the orientation of circular objects, we solved the problem using regression to estimate the 2D rotation unit vector of the object. To test H3, we compared the accuracy of regression models to classification models by assigning a class from the rotation vector estimated through regression. This can be obtained by normalizing the output vector and assigning it to its corresponding class.

We performed various experiments to test our hypotheses (see Table 9.1). We tried different sizes of training datasets, since it is not only relevant to know how large the dataset has to be in order to solve the orientation problem, but also to explore the efficiency of the different refinements in the training procedure when the size of the dataset varies. We used stochastic gradient descent with momentum to train the orientation models and used a ResNet-50 architecture [110] in all the experiments, due to its simplicity and accuracy on the ImageNet dataset. For transfer learning, we used weights pre-trained on the ImageNet dataset [54]. The number of epochs was adjusted according to the size of the dataset.

Each experiment was executed 5 times, and we report the average accuracies in Table 9.1. In preliminary experiments we obtained the best results with a learning rate of 0.001, a batch size of 8, and a weight decay of 0.00004. Therefore, we used these hyperparameters for all further experiments. We did not decay the learning rate for the experiments in Table 9.1, since for the bigger dataset sizes the number of epochs is low. When solving our problem using regression, we used the mean squared error loss function. For the classification problem we used cross entropy loss. In all experiments, after cropping the image to the bounding box from our detector, we cropped the pictures with an aspect ratio randomly sampled in $[\frac{3}{4}, \frac{4}{3}]$ and an area distributed between 8% and 100%, finally resizing them to the input size of the network (224x224). This method has been used successfully in previous work [115, 250] and also worked well for the brick. However, this may be facilitated by the fact that the symmetry-breaking LEGO logo is present on most of the brick's surface. For objects where fine details are important it might be necessary to keep the image aspect ratio unchanged and add padding to the image, or make changes to the CNN architecture to support a larger input size.

To test H1 and H2, we conducted a baseline training experiment where we did not use transfer learning. We added each of the refinements incrementally, hence in the *Transfer learning* row of experiments in Table 9.1 we used a pre-trained model, and in the *Rotation* row we added rotation as a data augmentation technique. For the latter, we randomly rotated the image by an angle between $[-30°, 30°]$ and adjusted the ground truth accordingly. Intrigued by the lower accuracy obtained when solving the problem with a regression loss function, we decided to run additional experiments for longer with the larger training dataset (90 epochs). The learning rate was adjusted to 0.002, but decayed at a rate of 0.1 every 30 epochs. Results thereof are shown in Figure 9.9.

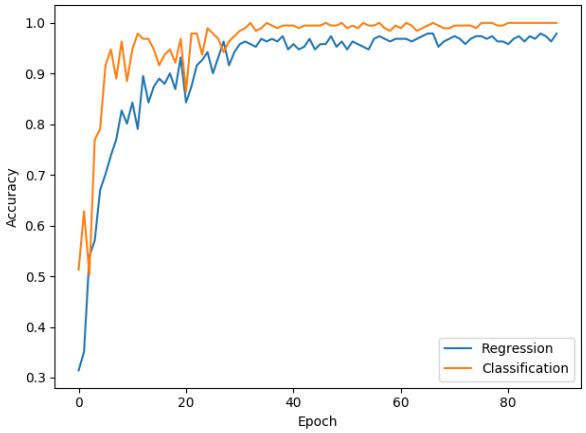


Figure 9.9: Accuracy comparison between a model using classification and a model using regression, trained with the complete training dataset containing 400 pictures.

	Classification (D=100, 60e)
Baseline	0.91 ± 0.05
+ Transfer learning	0.95 ± 0.02
+ Rotation	1.00 ± 0.00

Table 9.2: Results of experiments using object e from Figure 9.2. D stands for size of the dataset, followed by the number of epochs. Each experiment was executed 5 times and we report the average accuracy.

Discussion of results

The results in Table 9.1 are in line with H1 and H2. For this use case, transfer learning always resulted in substantial improvements in accuracy. Using rotation as an augmentation technique also resulted in better accuracy, in particular in cases when the dataset was small. These results provide evidence that it is possible to perform the rotation disambiguation with a very small dataset. Contrary to H3, our results indicate that classification always performed better than regression in this particular task. The graph of accuracies shown in Figure 9.9 also suggests that when solving the problem using a classifier, the model was able to learn faster than with regression. However, these findings are closely related to the choice of architecture, loss function and training procedures. Hence, further research is needed to understand why approaching the problem from a classification perspective results in better performance.

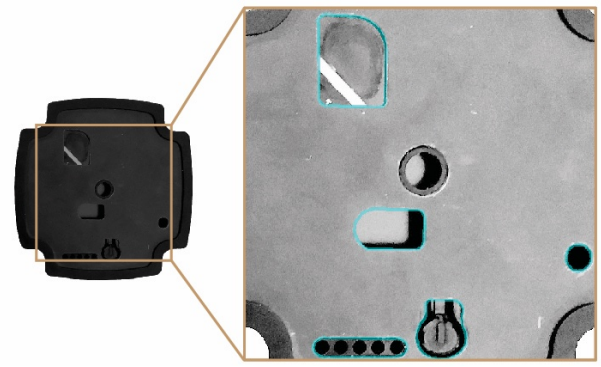


Figure 9.10: Near-symmetrical pump component made of black metal and plastic. It measures 8.5x8.5cm. The zoomed in image on the right is enhanced to highlight the symmetry-breaking features of the component (outlined in blue).

Other materials and shapes

In this paper we apply well established deep learning algorithms that have been used successfully to accomplish different visual recognition tasks [110, 111, 159] in a variety of complex datasets [54, 160]. Therefore, we speculate that our approach is generalizable for most near-symmetrical objects that require such QA procedures in industry. To support this argument, we further tested the orientation model with the pump component depicted in Figure 9.2 (e). This component has 4 degrees of symmetry and is composed of black plastic and metal. The zoomed in image in Figure 9.10 shows its symmetry-breaking features, which consist of several holes of different shapes and sizes, as well as a bright vertical element. The experiment was conducted using a training dataset with 100 images, trained over 60 epochs, with the same hyper-parameters as described in Table 9.1. We used 8 classes, spanning 45° each. The results of this experiment, given in Table 9.2, show similarly high accuracies as earlier experiments with the brick (see Table 9.1, column with classification, D=100, 60e). While these results support that our method is generalizable, further research is necessary to confirm this assumption.

9.10 Discussion and future work

Our system is inspired by metrology practices in QA at several manufacturing companies associated with the MADE project. While such practices involve various types of objects and different measurement tools, we focus merely on a subset of a metrologist's task space. We hope that in the future the concepts presented in this paper can be applied to more varied measurement activities. In this section, we reflect on essential parts of our pipeline, discuss limitations and give considerations for future work.

Degrees of symmetry

In the presented work, we primarly focused on 180°-symmetric objects (e.g., a 2x4 LEGO brick). This means that after context tracking, there are two possible rotations to choose from (see "Context" in Figure 9.7). We then use four classes at 90° to each other (i.e., up, down, left, right), to resolve uncertainties, as is shown in Figure 9.7 ("Focus"). Even if the orientation is close to the boundaries between two classes and the classifier is undecided, the end-result remains valid. For instance, if the detected direction is exactly between "up" and "left", it does not matter whether the classifier outputs "up" or "left", since in both cases the resulting vector based on the dot product will be the same. We can therefore argue that for a 180° symmetrical object, the minimum number of classes for resolving ambiguities orientation is three, i.e., each spanning 120°. In our example we use four classes, to increase the stability and yield a more intuitive set of directions for output and training.

From this we can go on to surmise more generally, that the minimum number of classes to disambiguate orientation is the degree of rotational symmetry plus one (i.e., with 180° symmetry, a resulting vector can stem from exactly 2 different orientations, ergo $2+1=3$ classes). These classes must be evenly distributed around a full circle (360°). For instance, a 90°-symmetric object, such as a the pump component in Figure 9.2 (e), would require a minimum of five classes, spanning 72° each. This approach is limited regarding round shapes, like discs, since these have no discrete set of rotations to disambiguate from. For instance, for a round object with a small non-symmetric logo in the middle the context tracking pipeline in our setup could merely provide the object's location for the focus camera to orient towards, but all orientation information would need to be provided by focus tracking. This can be achieved with the orientation model that estimates the exact rotation, as was discussed in the *Technical Evaluation* section.

Limitations and alternatives

There is room for improvement in several parts of the Focus+Context tracking pipeline. Our current scope covers measurement steps when the object is oriented so that the symmetry breaking feature faces up towards the cameras. With small adjustments to the setup, the same principles may be used to cover further cases (e.g., a side-ways brick) and support a larger variety of measurement steps. In more general terms, in the future we intend to integrate our orientation model in the Mask R-CNN framework to explore real-time 3D pose estimation using deep learning. Other promising approaches could involve continuously tracking the object using information from the previous known pose, or designing a deep learning framework that uses the input of both context and focus cameras to improve accuracy. This could also help cope with the issue of occlusion, which persists in particular when measuring small objects. As of now, the object has to be visible to the focus camera so the system can provide instructions with the correct orientation. However, in these situations, the instruction could still be visualized in an initial default pose or the last known one.

Currently, although the deep learning algorithms run in real time, the system has some latency caused by the pan-tilt unit and DSLR camera. Rotating the unit and taking a picture takes some seconds before it is received by the main PC. One way to circumvent this practical limitation would be to use multiple focus cameras. A faster pan-tilt unit, or industrial cameras with zoom lenses and high resolution video streams would also reduce the system's latency.

Alternative solutions could also be explored in regards to the display technology. In our system, we currently use LCD screens to prevent worker instrumentation. However, projectors and head-mounted displays could allow co-location of measurement guides even in a hand-held tracking scenario. We aim to explore further display options and their trade-offs in the future.

Future long-term evaluation with experts

To assess the practical value of our proposed solution for digital assistance in applied metrology, a long-term evaluation of our system at manufacturing companies is required. Arguably our guidance system can lead to performance advantages in QA. In an unaided scenario, workers currently need to closely inspect an object to identify and adjust its orientation manually, before referring to the measurement instructions, and must then manually enter the values in the correct field. Our algorithm detects the object's orientation for them and presents the measurement guides accordingly, which removes the need for close scrutiny and mental rotations. Furthermore, our proposed approach entails that both the guides and the object are always visible in the worker's field of view, which reduces task complexity [222]. While a field study is beyond the scope of this paper, it would allow us to explore the efficacy of our approach, identify further limitations, and help us to better address the workers' needs. The results of such a study would also lead to further development of our system. For example, this could involve a step for verification of measurements - i.e., tracking the worker and the measurement tool to verify what position is measured, to ensure that it is measured correctly and allow automatic recording of values. By providing the technical details involved in tracking objects for digital assistance during applied metrology, this paper forms the ground work for further development and evaluation during deployment in the field.

Applications beyond metrology

The presented approach is aimed at industrial metrology tasks that are executed in a conventional work space (consisting of a desk, chair, measurement tools and a computer). This arguably makes our solution easily transferable to similar activities beyond metrology, e.g., in a play context. For instance, Miller et al. [182] track the building process of a colored brick construction to create a virtual replication thereof. This could be extended through our concept, by additionally tracking the unique orientation of each new brick whenever the user attaches it to the construction. This would add degrees of freedom to the building process without requiring specialized

bricks: the orientation of a brick could alter the local appearance of a virtual texture that spans across the construction, or it could be used to define the *inside* and the *outside* of the construction.

9.11 Conclusion

In this paper, we present the basic concepts for providing digital assistance for metrology during quality assurance in manufacturing. In particular, we propose a two-step approach for pose estimation of near-symmetrical objects, which we call Focus+Context tracking. By combining (1) coarse-grained object recognition with context cameras and (2) precise pose estimation based on fine-grained features with a focus camera, we leverage (1) fast computer vision techniques and (2) accurate deep learning strategies. We describe the tracking pipeline we implemented and elaborate on how this was applied to a typical metrology scenario, using a 2x4 LEGO brick as an example use case. The results show that the fine-grained features on a brick are sufficient to successfully estimate its pose, including its unique orientation, with very high accuracy. We further apply our framework to the example of tracking a pump component and argue that the presented concept for pose-estimation can be extended to a wider range of applications with near-symmetric objects, or more generally, nearly identical objects with small distinguishing features.

Chapter 10

Paper B

This chapter presents the manuscript *CADTrack: Instructions and Support for Orientation Disambiguation of Near-Symmetrical Objects*, currently in submission to *ISS 2023*.

CADTrack: Instructions and Support for Orientation Disambiguation of Near-Symmetrical Objects

João Marcelo Evangelista Belo, Jon Wissing, Tiare Feuchtner, Kaj Grønbæk

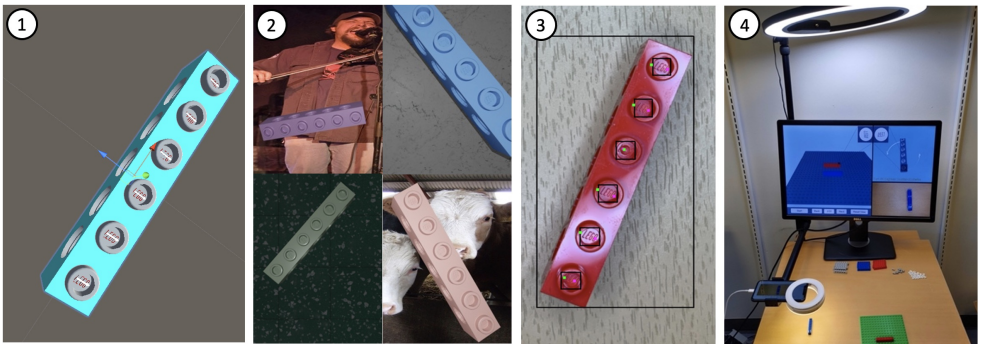


Figure 10.1: We propose CADTrack, a system to detect the orientation of near-symmetrical objects through a pipeline consisting of 4 stages: 1) Annotation of the points of interest in the object's digital model (CAD); 2) Generation of a synthetic dataset; 3) Training a convolutional neural network 4) Real-time tracking.

Abstract

Determining the correct orientation of objects can be critical to succeed in tasks like assembly and quality assurance. In particular, near-symmetrical objects may require careful inspection of small visual features to disambiguate their orientation. We propose CADTrack, a digital assistant for providing instructions and support for tasks where the object orientation matters. Additionally, we present a deep learning pipeline for tracking the orientation of near-symmetrical objects. In contrast to existing approaches, which require labeled datasets involving laborious data acquisition and annotation processes, CADTrack uses a digital model of the object to generate synthetic data and train a convolutional neural network. Furthermore, we extend the architecture of Mask R-CNN with a confidence prediction branch to avoid errors caused by misleading orientation guidance. We evaluate CADTrack in a user study, comparing our tracking-based instructions to other methods, to confirm the benefits of our approach in terms of preference and lower effort requirements.

10.1 Introduction

The orientation of near-symmetrical objects with small distinctive features can be critical for several tasks. For example, orienting components correctly, such as transistors and capacitors, is crucial in the assembly procedures of circuit boards. Quality assurance [17] is another domain where a correct orientation is essential to achieve precise measurements in specific positions of near-symmetrical objects. Identifying the visual features that disambiguate the object's symmetry in such tasks can require substantial effort, as these can be small and difficult to see.

In this paper, we propose CADTrack, a system to support users in disambiguating the orientation of objects while providing instructions for the task at hand. It supports visualization of the instructions in 3D and real-time tracking of the object's orientation, overcoming two challenges not addressed in state-of-the-art solutions [17]:

1. Existing approaches rely on deep learning pipelines requiring labeled datasets involving laborious data acquisition and annotation processes. CADTrack requires a single annotated digital model of the object, which it uses to generate a synthetic dataset.

2. Digital assistance to support tasks such as orientation detection must be highly accurate, or it can introduce errors making it counterproductive. However, existing systems do not inform users when the guidance they provide might be misleading. We address this challenge with a confidence prediction branch in a modified architecture of the Mask R-CNN framework [111], which CADTrack uses to indicate when users must proceed with caution.

To address these challenges, we also propose a tracking pipeline we use in CADTrack, consisting of four stages (see Figure 10.1):

1. Annotation of the visual features in the object's digital model to disambiguate the orientation.

2. Generation of an automatically annotated synthetic dataset.

3. Training our modified architecture of Mask R-CNN using the synthetic dataset.

4. The CNN is added to CADTrack and can be used to track the object.

Our work is highly relevant for manufactured components, where typically corresponding CAD models are already created during product design. Furthermore, the setup of CADTrack only requires off-the-shelf components, making it easy to add to surfaces such as office desks, allowing users to receive instructions with clear visual guidance when placing an object in a designated tracking zone. Once an object is detected, CADTrack indicates its current orientation and how to rotate it to fulfill the current instruction.

We evaluated our approach in a user study with 13 participants, confirming the potential of our system to support dynamic instructions and live object tracking during

3D manipulations, as users preferred these over static instructions. We also found that, compared to static instructions, instructions with tracking reduced the difficulty and physical load of verifying the orientation of the near-symmetrical object without negatively impacting task completion time.

10.2 Related work

In this section, we will cover deep learning-based approaches, which have replaced traditional computer vision techniques in various computer vision tasks in the last decade, followed by a review of prior work about digital assistance.

Deep learning for computer vision

Image classification: In the late eighties, LeCun et al. [152] applied gradient-based learning to classify images of digits with a convolutional neural network. The approach saw a resurgence in 2012, when the work of Krizhevsky et al. [146] revolutionized computer vision research, achieving considerably better results using a CNN over traditional computer vision approaches in image classification of the ImageNet dataset [55]. Since then, CNNs have seen considerable speed and accuracy improvements over the years, due to various architectural breakthroughs [113, 161].

Object recognition: Object detection also benefited from considerable improvements in accuracy over the last years. R-CNN [95] kicked off a series of discoveries in object recognition with a detector consisting of two stages: 1) extracting candidate region proposals through traditional CV techniques; and 2) classifying each proposal as an object or background using a CNN. Other works improve the approach, making it faster [94] and more accurate [112]. Later, Ren et al. [221] used a CNN to propose candidate proposals and integrated it with the second-stage classifier into a single CNN, making object recognition possible in real-time. Researchers have also explored object recognition approaches to perform object predictions over a single stage [162, 167, 217, 218, 237]. One-stage methods are faster but less accurate than two-stage approaches. In our work, we use a two-stage detector because we prioritize accuracy.

Pose estimation: Advances in deep learning have also contributed to breakthroughs in pose estimation of objects from RGB images. Toshev et al. [259] proposed a CNN model that directly regresses the coordinates for 2D pose estimation. Since then, novel architectures with multiple stages or modules have improved the accuracy and performance of 2D keypoint predictions [42, 191, 269]. Recent work leverages a CNN to predict keypoints, edges, and symmetries [242] and combines it with EPNP [156] to perform 6D pose estimation. Mask R-CNN [111] adds a branch to R-CNN to perform instance segmentation while demonstrating how to modify the architecture to perform other tasks like keypoint estimation. In our work, we are interested in tracking near-symmetrical objects with small distinguishing features, so we use Mask R-CNN to detect the object, its symmetry-breaking features, and respective keypoints.

Training and Synthetic Data Generation: CNNs excel in various visual recognition tasks but need annotated data to train. Capturing and annotating data is time-consuming and expensive. To address this issue, Tobin et al. [255] propose domain randomization, a technique to generate synthetic images with a high degree of variation in a simulator. The goal is to create images that are so diverse that real-world data will appear to the CNN as another variation. Adding distractors to synthetic data such as geometric shapes, random backgrounds, and different rendering settings (e.g., lighting, different object textures) can improve performance [260]. To generate synthetic data, tools such as BlenderProc [57] can render photorealistic images, reducing the domain gap further. Another factor that can improve the performance of CNNs is to employ tricks such as data augmentation and optimization methods. We use BlenderProc in our pipeline to generate a diverse training dataset from a digital model of the object of interest and apply several data augmentations and established optimization techniques to achieve high accuracy when tracking near-symmetrical objects.

Digital assistance

Extensive work has explored digital systems to support users in tasks such as assembly and inspection. Some work tracks objects for different goals (e.g., validation, error detection), which we emphasize in this section.

Assistance with object tracking: Anderson et al. [5] proposed a system where the assembly parts are instrumented with sensors, making it possible to detect the assembly process of the model. DuploTrack [104] is a system to track and support the assembly of a LEGO construction, capable of providing feedback every step while detecting mistakes. This work suggests that dynamically updating the pose of a virtual model by tracking its counterpart in the physical world can improve the user's speed and accuracy. Miller et al. [182] track LEGO bricks similarly to infer how users build a model. Sukan et al. [247] explored how different visualization approaches can assist users in orienting a tracked object. In contrast to these works, our goal is to support users in finding the orientation of objects where doing so is difficult and propose a convenient method to track them.

Augmented Reality for assembly and inspection: Augmented Reality (AR) techniques can mitigate gaze shifts through an immersive environment where virtual instructions are presented directly in the physical world. Henderson et al. [121] explored the benefits of using AR in procedural tasks, which allows for speed and accuracy improvements. Several works explore the benefits of AR in assembly and inspection tasks or compare AR to other methods, such as paper instructions [87, 120, 211, 280]. However, AR systems typically require user instrumentation, complex apparatus, expensive hardware, or a combination of these - tracking near-symmetrical objects presents additional challenges in AR because of lighting and image definition requirements. In our work, we don't need user instrumentation and only use off-the-shelf components: a smartphone and a computer.

Assistance for tracking near-symmetrical objects: Belo et al. [17] proposed a system to detect the orientation of near-symmetrical objects. However, it consists of a complex multi-camera setup, requires manual data labeling, and uses two standalone CNNs for pose regression that output a single orientation vector without evaluating features individually. Moreover, they do not assess the utility of the system for users. Our work overcomes these limitations, having a smaller tracking zone as a trade-off.

10.3 Tracking near-symmetrical objects

The architectural breakthroughs of Krizhevsky et al. [146] made Convolutional Neural Networks (CNNs) the most successful technique for solving various visual recognition tasks. The main challenge when using such approaches resides in collecting and labeling data, a tedious and laborious task when done manually. In this work, the pipeline we propose uses the object's digital models (CAD models) that tend to exist for objects used in assembly and inspection tasks since they are necessary for other manufacturing stages. Moreover, we explore architectural modifications for an existing recognition framework to handle the specific case where objects are near-symmetrical.

Our pipeline consists of 4 stages, shown in Figure 10.1. In the first stage, the developer provides the CAD model and annotates the keypoints that disambiguate the object's orientation. Then, there is a data generation stage, where a 3D engine automatically generates and annotates a synthetic dataset. In stage 3, we train our customized Mask R-CNN model on the generated data. Finally, in stage 4, the model is ready to make predictions and identify keypoints to predict the orientation of the object of interest.

We start by describing the model we use and how we modify its architecture to predict the orientation of near-symmetrical objects. Then, we go over each stage of the training pipeline in greater detail.

Model architecture

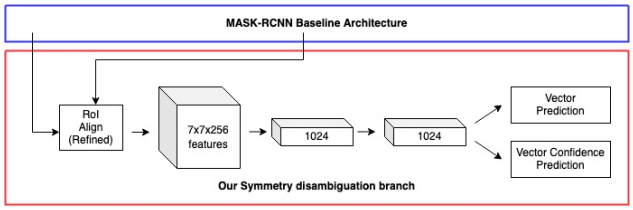


Figure 10.2: Model Architecture - in red, the custom branch we propose to disambiguate the object's orientation using Mask-RCNN.

In our use case, it is crucial to assess the quality of predictions the CNN outputs - many factors influence the predictions and can vary from frame to frame. For

example, manipulating the object can result in poorly focused pictures. Meanwhile, its position in the tracking zone affects the visibility and lighting of the features that make the object's rotation disambiguation possible. Our model extends the Mask R-CNN framework with an additional branch for predicting the orientation vectors of the symmetry-breaking features and the confidence of these orientation predictions (see fig. 10.2). It uses the RoIAlign features from the original architecture [111], which consists of a $7x7x256$ feature map for the detected symmetry-breaking feature, and executes after the classification step. This branch contains two fully connected layers of size 1024 with Relu activation functions and only runs for bounding boxes classified as a symmetry-breaking feature. We use the last feature map (the second fully connected layer) to predict the orientation vector and the confidence of the orientation prediction. The vector output has no activation function - it simply outputs four values - the coordinates for the initial and terminal points $[i_x, i_y, t_x, t_y]$ of the vector. The confidence output uses a sigmoid activation function to output the prediction accuracy.

Tracking pipeline

CAD model + keypoint annotation

The first step in our pipeline is to annotate the symmetry-breaking features in the digital model of the object to track. Here, a developer inspects the CAD model and annotates visual features that disambiguate the object's orientation. In contrast to standard procedures where hundreds of pictures are captured and labeled per object, this is only done once per object. To do so, the user inspects the object in a 3D engine (such as Unity) and annotates the positions of these keypoints into a configuration file.

For some near-symmetrical objects, locating the symmetry-breaking features is sufficient to disambiguate their orientation, but this is not always the case, like the LEGO element we use in our study. Therefore it is possible to annotate each of these symmetry-breaking features with a vector, indicating their orientation. In the next section, we discuss how the symmetry-breaking features might only be visible from an interval of camera perspectives. If that is the case, the user can indicate at this stage which perspective interval is appropriate to ensure the data generation step creates sufficient data where the symmetry-breaking features are visible.

Data generation

We generate the data to train our model with BlenderProc [56]. BlenderProc can generate datasets for several computer vision tasks, including object detection, instance segmentation, and pose estimation. To do so, it creates synthetic data and its respective ground truth annotations (e.g., bounding boxes, segmentation masks, keypoints, and 3D poses).

In our problem, we are interested in predicting the symmetry-breaking keypoints that allow the system to disambiguate the object's orientation. For each keypoint, we annotate a bounding box to identify its location and a vector to indicate its orientation

(see figure 10.1, stage 1). Using the positions annotated in stage 1 BlenderProc generates bounding box annotations for the object and each orientation vector. Then, we create vector annotations - to do so, we use the pose of the object and camera intrinsics to compute a model view projection matrix that converts the position of the features in the CAD model coordinate system (3D) to their position in the rendered picture (2D).

The images CADTrack generates have the object in different positions and rotations. Because all our training data is synthetic, we apply domain randomization techniques to create a dataset with enough variability [255]. We achieve this by changing the colors and materials of the objects and using different backgrounds throughout the data generation process (see figure 10.1, stage 2). We randomly select half of these backgrounds from the ImageNet dataset [55], while the other half are textures from ambientCG.com. However, completely random rotations may result in few images with symmetry-breaking features visible. For this reason, half of the generated images are randomized within a fixed rotation interval if provided in stage 1. The result is a dataset where most pictures have symmetry-breaking features visible.

Training the model

The third stage in our tracking pipeline is to train the Deep Learning model that CADTrack uses to predict the position and orientation of the object.

We use several data augmentation techniques from PyTorch[1], specifically *RandomErasing*, *RandomNoise*, *RandomInvert*, *GaussianBlur*, *RandomPosterize*, *RandomSolarize*, and *RandomEqualize*. We create two additional custom augmentations to deal with the fact that symmetry-breaking features can have poor quality in pictures for various reasons, such as lighting or camera focus issues. The first is a blur transformation which targets only a subset of the symmetry-breaking features, leaving the rest of the image unmodified. The second overlays keypoints with another picture using a random transparency level. The goal is to create noisy images that encapsulate various issues common in the real world, such as imperfections in the object, occlusion, and different camera focus.

The model is trained in three steps, using 7600 images for training and 400 for validation. The first step is a simple training round where we do not use any of the described augmentations. Here, we train the model for 10 epochs, using a batch size of 2 and a learning rate of 0.0001. In the second step, we broaden the domain to allow the model to recognize real-world images. To do so, we use aggressive data augmentation techniques that can make symmetry-breaking features hard or impossible to detect, increasing the training duration. However, our ablation studies show that training the model without such data augmentation results in poor performance on real-world images, even though high accuracy is achieved faster in our synthetic dataset. In step two, we use all the data augmentations for 24 epochs, a batch size of 16, and a learning rate of 0.0001.

[1] https://pytorch.org/vision/main/transforms.html

After the second training step is complete, we train the vector confidence branch. To do so, we lock all the parameters in the model except for the confidence branch and make predictions on data samples. Depending on how far these predictions are from the ground truth, we consider these to have good or bad quality and set the confidence ground truth to 1 or 0, respectively. We also use data augmentations at this training step. When the image contains heavy noise or blur augmentations, we rate it low quality and set the prediction confidence to 0. The third training step runs for 10 epochs, with a batch size of 2 and a learning rate of 0.0001.

Tracking the object

After training, the CNN can be used in CADTrack to support instructions containing the object it tracks. We show some prediction examples in Figure 10.3.

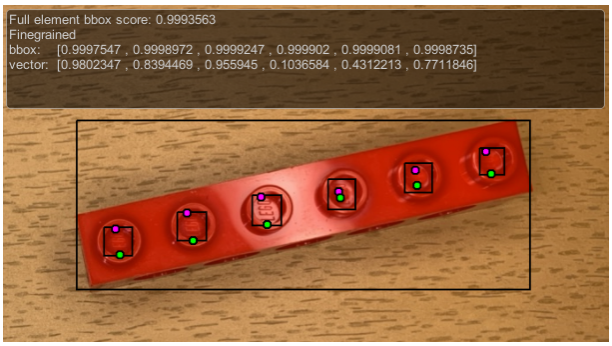


Figure 10.3: Predictions from our CNN. Green dots represent the start of a vector, and purple dots represent the end. Scores for bounding boxes and vectors are displayed in the system on demand, s.t. the left-most value corresponds to the left-most bounding box in the image.

10.4 System

We have proposed a tracking pipeline for near-symmetrical objects to provide accurate predictions of their position and orientation. However, bounding box coordinates and vectors in image space have limited utility in tasks involving orientation disambiguation since they are not easily readable by humans. Furthermore, orientation disambiguation tasks are typically sub-tasks of more complex procedures, such as Quality Assurance [17]. Therefore, we propose CADTrack, a system to provide instructions for assembly and inspection tasks while assisting users in disambiguating the orientation of corresponding near-symmetrical objects.

CADTrack uses only off-the-shelf components (see Figure 10.1, stage 4), making it accessible. It consists of a tracking zone, where a smartphone captures pictures of the object, and a computer runs the system, allowing the user to visualize instructions and interact. The architecture is straightforward: we run a Python server that receives

the video from the smartphone and runs the latest frame through our model. A Unity application uses these predictions in a desktop user interface.

Processing model predictions

To use CADTrack, the user will typically manipulate or place the object in a tracking zone, leaving it under the camera for consecutive frames. Again, in a system such as CADTrack, it is crucial to provide orientation information as accurately as possible and notify the user when the tracking is unreliable - otherwise, using such a system can result in errors limiting its utility. As our model outputs predictions per image without taking previous frames into account, we make the system more robust by keeping state information to provide accurate instructions when consequent frames have lower confidence predictions. Poor predictions may happen for various reasons - lighting conditions, the camera cannot focus on the symmetry-breaking features, or the user's hand occludes the object.

The visibility of symmetry-breaking features is typically constrained to specific intervals of camera perspectives, so we approximate fixed camera perspectives depending on the features detected in the picture. To find the element's 2D rotation, we use the center of the bounding boxes for the symmetry-breaking features as keypoints. The relationship between these points results in a range of possible orientations easily disambiguated when vector predictions have sufficient quality. The strategy to decide the object's rotation from the CNN output depends on the amount of symmetry-breaking features. For example, for the LEGO element (figure 10.3), we consider the vector predictions good in any of these cases:

1. The CNN predicts a vector with a confidence higher than 80% without other disagreeing features (with 60% confidence or higher).

2. Two vectors pointing in matching directions with at least 60% confidence are detected, and no disagreeing vector is detected (with 50% confidence or higher).

We determined these threshold values after experimentation to achieve robust and consistent tracking.

Whenever CADTrack obtains a prediction with high confidence, it keeps track of it through an internal state. CADTrack uses the locations of the features to compute the potential rotations in 2D every frame. If a follow-up prediction has low confidence, the system assumes the object orientation is the closest to the previous state. Because the CNN runs at an average of 5 frames per second, this approach rarely results in problems, making the system more robust to lower-quality images resulting from partial occlusion, poor lighting, or camera focus issues.

User interface

We split the user interface into two main zones: tracking and instructions. Separating these zones was a deliberate decision, so instructions are not constantly affected when the user uses the tracking functionality.

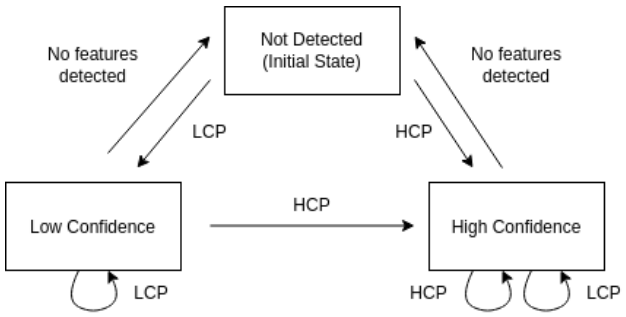


Figure 10.4: State machine for the confidence. HCP: High confidence prediction; LCP: Low Confidence prediction.

Presentation of tracking information

The pose of the tracked object currently relevant for the task is reflected in real-time by the orientation of its virtual representation (see Figure 10.6, right). The concept of digital shadows [145] is related, where a virtual object uses real-world data (typically in real time) to create an enhanced virtual representation of the object. In this case, the real-world information is the pose captured by an external sensor (smartphone camera). When the user places the physical object in the marked tracking zone on the desk, its digital shadow updates the pose as seen from the user's perspective. The top left corner of the screen shows the symmetry-disambiguating feature orientation (the LEGO logo, in this case). Although the model is accurate, adverse conditions such as poor light, wrong camera focus, or occlusion may lead to low-confidence predictions. Therefore, tracking accuracy is reported as a text label in the tracking area, allowing the user to take corresponding measures and adjust their level of trust towards low confidence tracking results (see Figure 10.6, right, green text).

Presentation of instructions

The left side of the screen contains task instructions. The top right corner shows the disambiguating feature (LEGO logo) to indicate the desired object orientation. Users can align the physical object orientation to this target orientation by following the system's instructions, which vary by task (*Inspection task* and *Assembly task*) and are described further in section 10.5.

10.5 Evaluation

To evaluate our approach, we benchmark our prediction pipeline with real-world data and conducted a user study to assess the utility and usability of our system. We chose the LEGO brick from Figure 10.3 as the primary object for our evaluation for multiple reasons: 1) it is challenging to see and track - the logo is inside each of the brick knobs, and it is hard to disambiguate its symmetry under a variety of conditions (e.g.,

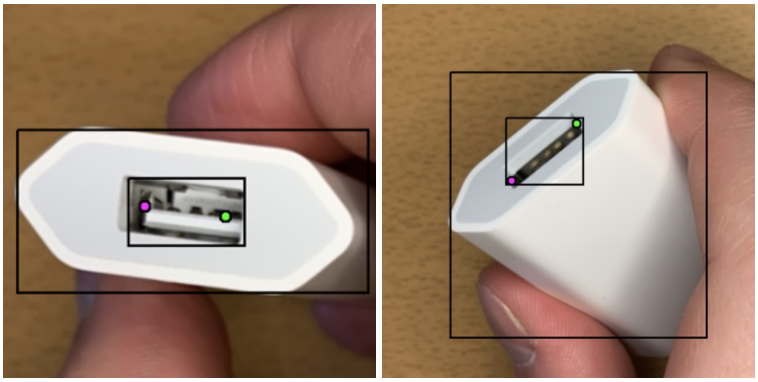


Figure 10.5: Tracking results on an Apple charger

lighting, distance, camera focus), 2) it is easily accessible and can be used to simulate tasks such as assembly and inspection.

Performance of the tracking pipeline

To assess the performance of our CNN, we focus on criteria we consider crucial to make CADTrack a reliable system: the accuracy of high-confidence predictions. We capture 1200 validation images from the real world of the LEGO element we use in our study through 3 video sequences - each with a LEGO element of a different color (red, blue, and grey). Note that in some of these pictures, it is impossible to discern the object orientation through visual inspection - this is intentional since real-world settings have low-quality pictures where CADTrack should notify the user that the tracking is unreliable. We label this validation dataset semi-automatically, using a Mask R-CNN model to detect the location of the symmetry-breaking features. Then, we disambiguate the symmetry through manual inspection. Because the initial rotation in each video is known and the difference in the position in each consequent video frame is minor, we can annotate the object 2D rotation for every image semi-automatically. We validate our model with the dataset we captured. Using our approach, we label each prediction as a confidently correct prediction, a confidently incorrect prediction, or a poor-quality image. From 1200 captured images, our CNN predicts correctly and confidently 651, identifies 539 of these images as having poor quality, and 10 are confidently wrong. The result is a true-positive rate of 54.25%, a false-positive rate of 0.83%, and labels the remaining 44.92% as poor-quality images. These results are highly desirable - for a system such as CADTrack, we are interested in achieving a low false-positive rate. An approach that discards a high percentage of images because they have poor quality is advantageous if that results in a lower false-positive rate, even if the trade-off is a lower true-positive rate.

We also evaluate how our approach generalizes to a different near-symmetrical object using our pipeline to track a smartphone charger. A qualitative analysis of some predictions suggests it can perform well with another item (see Figure 10.5). We will

revisit this topic in the discussion section.

In terms of performance, our CNN uses 1.8GB of GPU memory when running in inference mode. We test the model speed on different hardware to examine its performance. To do so, we evaluate 1000 images and calculate how many it processes per second. With an RTX 2080ti, our model processes an average of 9.25 predictions per second. We also evaluate the speed of our system in a low-end GPU. Using a laptop's GTX1060, our model processes an average of 3.25 predictions per second, indicating that CADTrack can run on machines with lower specifications.

User evaluation of CADTrack

We explored the utility and usability of our system in a user study, where participants were tasked to perform inspection and assembly tasks involving near-symmetrical objects. To investigate how real-time object tracking assistance (**Tracked**) affects performance compared to **Interactive** or **Static** guidance, we evaluated task completion time and the number of errors. The latter refers to incorrectly marked positions (e.g., wrong letter or wrong position) in the *Inspection task* or the wrong orientation or location of a placed LEGO brick in the *Assembly task* (see task descriptions in section 10.5), and counted if it was not rectified before clicking "next". To investigate whether our **Tracked** digital assistance effectively supports users in inspection and assembly tasks, we further assess six workload-related [39] items of the NASA-TLX questionnaire [109]. We also collected ratings on the difficulty of determining the correct object orientation and preferences for the different assistance modes. For the **Interactive** and **Tracked** conditions, we asked about the usefulness of being able to interactively adjust the visual perspective of the digital shadow and instruction space (i.e., move the virtual camera). Finally, for **Tracked**, participants were asked to rate the utility of the tracking (Tracking Utility) and their confidence that the system was providing accurate instructions (Tracking Confidence). All subjective ratings were made on a 7-point Likert scale (e.g., 1 - "very low", 7 - "very high").

Study design

All the participants performed an *Inspection task* and an *Assembly task* with LEGO elements, using CADTrack to receive instructions for specific positions and orientations of the LEGO object. Participants perceived these instructions using CADTrack in three different conditions. For a fair comparison with **Tracked**, we provided optimal lighting throughout the study using a light ring on the top of the table (see Figure 10.1, stage 4). The three conditions are:

1. **Static:** The 3D object is depicted from two distinct static perspectives and supplemented by written instructions, as shown in figure Figure 10.6 (left). Paper-based LEGO assembly instructions loosely inspire this.
2. **Interactive:** Using direct manipulation with the mouse, the user can rotate the 3D object representation to explore different visual perspectives.

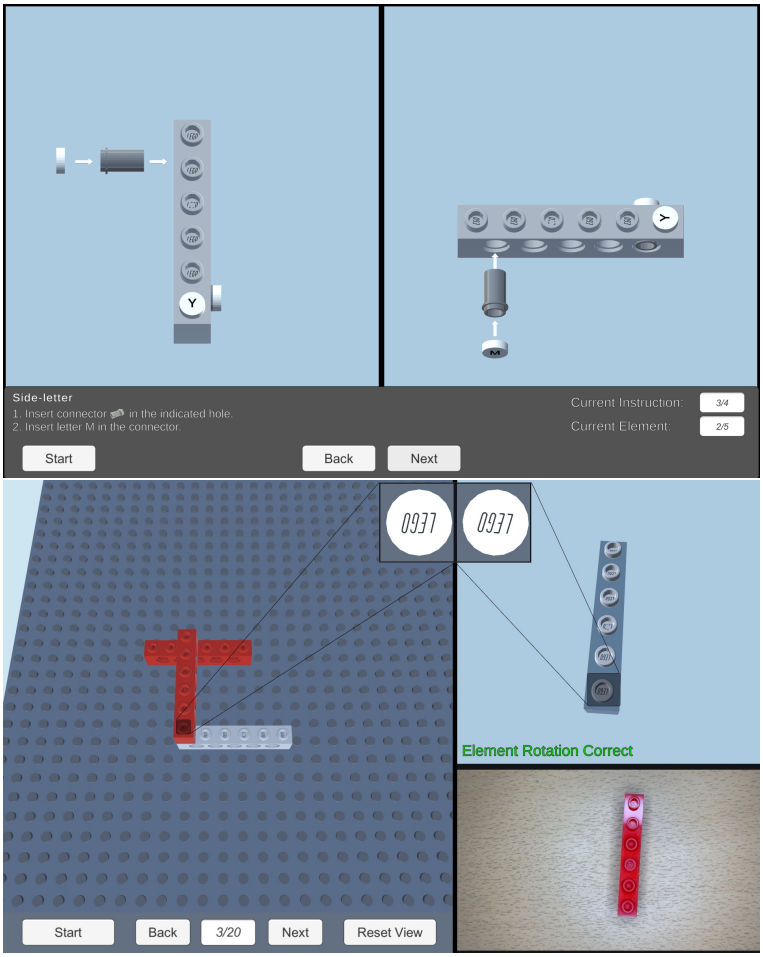


Figure 10.6: Two instruction visualization modes were implemented in CADTrack, to support each of our study tasks. Here we exemplify guidance in the **Static** condition for the *Inspection task* (top) and **Tracked** condition for *Assembly task* (bottom).

3. **Tracked:** The 3D object representation matches its tracked physical counterpart, giving the user feedback in real-time about the orientation of the physical LEGO element in the workspace (figure Figure 10.6, right).

We consider the **Static** condition (1) a ”baseline”, as this is a common way to provide instructions in assembly and inspection nowadays [17], either in a digital document (PDF) or on paper. With the **Interactive** condition (2), we aim to explore the benefits of allowing the user to manipulate instructions in 3D with the mouse.

Task description

Informed by industrial partners, we designed tasks inspired by two real-world use cases with alignment challenges: product assembly and visual inspection. For exam-

ple, assembling a circuit with the correct orientation of near-symmetrical electronic components, such as transistors or capacitors, can determine whether it works. Further, correctly orienting near-symmetrical objects can also be relevant for inspection tasks, such as in quality assurance procedures, where measurements must be performed precisely in the correct positions [17].

***Inspection task*:** To simulate a realistic inspection task that requires disambiguation of the object orientation (e.g., in metrology [17]), we instruct users to attach round LEGO tiles with letters to a 1x6 LEGO Technic element (see Figure 10.6, left). Users had to identify the tile and determine its intended position on the current LEGO brick, which may be either on one of six top knobs or attached to one of the 10-hole positions (5 holes, but two possible sides) with the help of a cylindrical pin. Hereby, it is critical to identify the orientation of the near-symmetric element based on the LEGO logo stenciled within the rim of each knob, to identify the correct object position. Participants repeated this 20 times in each condition, attaching 4 tiles to each of 5 LEGO Technic elements.

***Assembly task*:** To mimic an assembly procedure involving near-symmetrical components, we designed three different abstract structures, each consisting of 20 6x1 LEGO Technic elements in three different colors (red, blue, and grey). In each assembly step, the instructions indicate the color, target position, and orientation of the next element that participants must place (see Figure 10.6, right). Each structure shares a similar difficulty level with two objects on every layer in different positions and colors - we randomize the structure presented per condition for each participant.

In both tasks, users moved on to the following instruction by clicking on the "next" button on the desktop interface. The interactivity of the 3D visualization depends on the corresponding study condition (see section 10.5), and we determined the order of alternating task and condition pairings with Latin square.

Study procedure

After a briefing about the purpose and procedure of the study and data collection, participants were asked to provide signed consent of voluntary participation and complete a demographics questionnaire. Then they completed the *Inspection task* in each condition, followed by the *Assembly task* per condition. The order of conditions and task was counterbalanced across participants (within-subject design). At the beginning of each condition, participants got time to practice the task. When they started the trial they were reminded to complete it as quickly and accurately as possible. Each participant completed a total of 120 instructions, resulting from 20 instructions per task in each of the three conditions (20x2x3). After each condition, we elicited user feedback through a NASA-TLX questionnaire and Likert scale questions, as reported in the results below.

Participants

We recruited 13 participants (5 female) aged between 23 and 60 years (M = 33 years). All participants reported to have normal or corrected-to-normal vision and medium to high familiarity with building LEGO (M = 4.84, std = 1.23; scale 1 - "very low" to 7 - "very high").

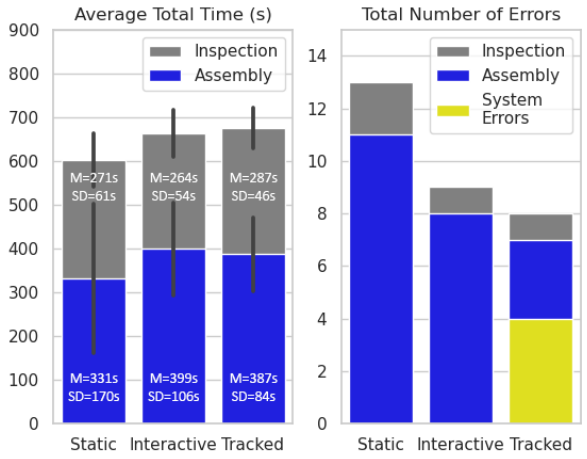


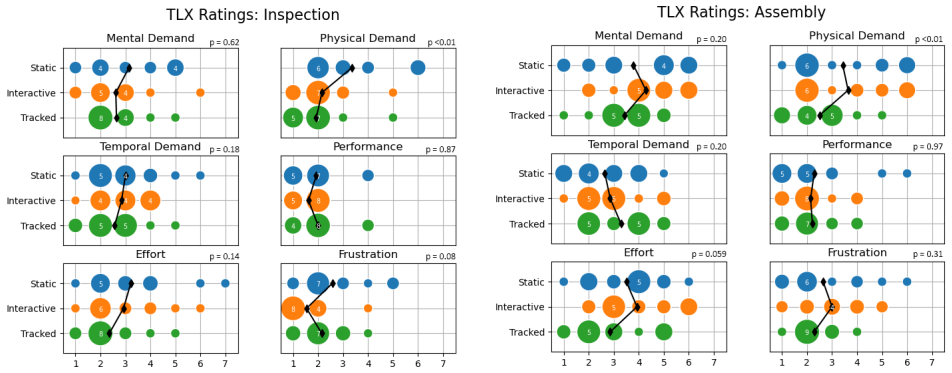
Figure 10.7: Left: Task completion time was similar across all conditions for inspection and assembly. Right: Errors across conditions - false positives from the tracking resulted in some system errors (yellow).

User study results

Task completion time and errors

Across all three conditions task completion times remained similar (figure Figure 10.7), as an ANOVA revealed no significant effect in either the *Inspection task* ($F(2,36) = 0.57$) or the *Assembly task* ($F(2,36) = 0.99$). This indicates that the use of our tracking pipeline does not introduce substantial temporal overhead, supporting the feasibility of its integration into existing workflows without decreasing users' task performance.

Overall, participants were very successful in completing the tasks, with less than one error on average per person. We observed a smaller average number of errors in **Interactive** ($M = 0.62$) and **Tracked** ($M = 0.54$) in comparison to **Static** ($M = 0.85$) (see Figure 10.7), but this difference is not statistically significant (ANOVA: Inspection $F(2,36) = 0.76$; Assembly $F(2,36) = 0.77$). Notably, in the *Inspection task*, half of the errors in **Tracked** occured due to false positives of the tracking system (325 objects processed; four system errors; 1.23% were false positives). In other words, throughout the whole study, the system reported the orientation of the object incorrectly four times, leading participants to make an error (avg. of 0.30 errors per participant due to system errors). Further improving the tracking accuracy will reduce the number of user errors



(a) (b)

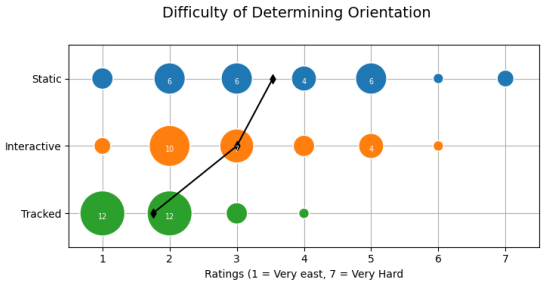
Figure 10.8: **a)** NASA TLX scores given on a 7-point Likert scale (1 - "very low", 7 - "very high", for Performance 1 - "perfect", 7 - "failure") indicate low workload for the *Inspection task*. We only find significant difference of condition in Physical Demand, which was higher in the **Static** condition compared to both others. **b)** NASA TLX scores again show low workload in the *Assembly task*. Again a significant effect is reported only for Physical Demand, where **Tracked** condition is superior to both others.

in such tasks. Although CADTrack only supports the orientation-disambiguation of the objects and does not validate the assembly and inspection itself, tracking acts as a validation step in these processes.

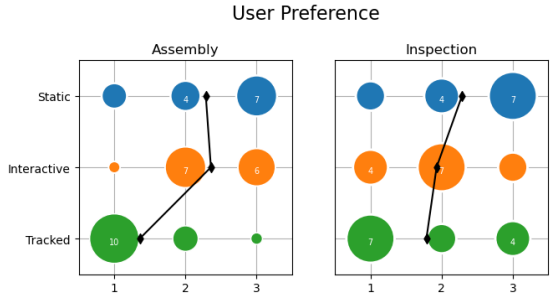
Subjective workload assessment

To evaluate task load, we asked participants to complete a NASA TLX questionnaire, scoring items on a 7-point Likert scale with 1 corresponding to "very low" and 7 to "very high", and for the *Performance* item we mapped 1 to "perfect" and 7 to "failure" respectively. Overall low to moderate scores support that the *Inspection task* was relatively easy, and the instructions were effective in all conditions (see plots in Figure 10.8a). Scores for the *Assembly task* were somewhat higher, particularly in effort and mental demand, as can be seen in Figure 10.8b, which reflects that this task was more complex. Yet, the good performance and low frustration scores again support that our guidance was effective across all conditions.

When comparing the three conditions in both tasks, a Friedman test revealed significant differences only with regards to Physical Demand (Inspection: $\chi^2(2) = 10.12, p < 0.01$, Assembly: $\chi^2(2) = 10.21, p < 0.01$). A post hoc Wilcoxon signed rank test with Bonferroni correction showed that in the *Assembly task*, **Static** was perceived as significantly more physically demanding than **Interactive** ($p < 0.05$) and **Tracked** ($p < 0.05$), while **Interactive** and **Tracked** were similar ($p = 0.12$). In the *Inspection task*, **Static** again scored more poorly than **Tracked** ($p < 0.05$). Here **Tracked** also outperformed **Interactive** ($p < 0.01$), while **Static** and **Interactive** scored similarly ($p = 0.40$).



(a)



(b)

Figure 10.9: **a)** Across both tasks (inspection and assembly), participants reported that object orientation disambiguation was significantly easier with **Tracked** compared to both other conditions. **b)** Preferences of different instruction modes.

It should be noted, that overall Physical Demand scores remain low across all conditions, but the apparent differences may support our expectation that CADTrack requires less eye strain and effort to detect the small symmetry-breaking features on the objects.

User Feedback

With regards to the difficulty of determining the object orientation, a Friedman test shows a significant effect of condition ($\chi^2(2) = 20.86, p < 0.01$). As can also be seen in Figure 10.9a, **Tracked** supports orientation disambiguation more effectively in contrast to **Static** ($p < 0.01$) and **Interactive** ($p < 0.01$), which we confirmed with a post-hoc Wilcoxon signed rank test with Bonferroni correction. **Static** and **Interactive** were rated similarly ($p = 0.07$), though some participants reported that the difficulty in disambiguating the object's orientation was low overall. Users reported that controlling the camera was useful in **Interactive** ($M = 5.56, SD = 1.13$), but not as relevant in **Tracked** ($M = 4.5, SD = 2.06$). A possible reason is the tracked mode allows users to view instructions from the desired viewport by manipulating the object directly, making interaction with the mouse unnecessary. Finally, participants rated the tracking function as useful ($M = 5.19, SD = 1.86$) in the **Tracked** condition, and

indicated high trust in the system's predictions ($M = 6.11, SD = 1.25$).

After completing all tasks, we asked participants to rank the guidance types by preference (see Figure 10.9b). This revealed a preference for **Tracked** for the *Assembly task* (Friedman test: $\chi^2(2) = 8.71, p < 0.05$). A post hoc Wilcoxon signed rank test with Bonferroni correction favors **Tracked** over both **Static** ($p < 0.05$) and **Interactive** ($p < 0.01$), but reveals no significant preference between **Static** and **Interactive** ($p = 0.86$). No significant differences in preference could be found for the *Inspection task* (Friedman test: $\chi^2(2) = 2.625, p = 0.13$). The latter may not be surprising due to the nature of the task, as users only needed to disambiguate the orientation of each of the 5 LEGO elements before attaching the 4 tiles respectively. In contrast, the *Assembly task* involves repeatedly aligning each of the LEGO elements consecutively to succeed, hence the system's tracking functionality was of greater importance.

10.6 Discussion

We presented CADTrack, a system containing a pipeline to track near-symmetrical objects from a digital model. Furthermore, we evaluate its utility in supporting users in assembly and inspection tasks involving near-symmetrical objects.

Our performance evaluation shows that our tracking pipeline is highly accurate (false positive rate of 0.83%), an accuracy level observed throughout our user study (1.23% wrong predictions). Moreover, the pipeline is efficient, allowing CADTrack to run on a mid-low-end GPU such as a laptop's GTX1060. This result is relevant because we can accurately track near-symmetrical objects without expensive data annotations, demonstrating that a similar approach has the potential to track the orientation of objects in various scenarios, such as tangibles [132] in AR.

Through a user study, we learned that a tracked approach has the potential to reduce the number of errors and the difficulty in determining the orientation of near-symmetrical objects. Comparable task completion times to existing methods (**Static** condition) support the possibility of integrating real-time orientation disambiguation using tracking into existing inspection and assembly workflows with no negative impact on performance. Such integration is relevant, considering that participants indicated lower physical load and preferred the condition with tracking.

Static instructions are practical because they present the object from an optimal perspective, meaning that designers can define these optimal settings by default. However, if this is not the case or in situations where different perspectives are valuable to support a task, a 3D digital shadow that participants can view from all sides (**Interactive** and **Tracked** conditions) brings clear benefits. Arguably, the real-time tracking from CADTrack can facilitate tasks with complex instructions, allowing users to focus primarily on manipulating the physical object instead of interacting with the system or paper/pdf instructions. It is also straightforward to combine the benefits of optimal perspectives present in Static instructions into a system like CADTrack, by

providing a predefined optimal view by default, which users could then customize, combining both benefits of static and interactive approaches.

Beyond the explored scenario, we suspect CADTrack may hold promise for supporting users under adverse conditions, such as poor lighting, safety distance requirements, or severe visual impairments. While this remains to be studied, we gathered anecdotal evidence from an informal system evaluation with a participant with severe visual impairments. The participant reported a reduced field of vision (approx. 1/3 of the standard field of view) and similarly reduced capacity of visual focus. They explained that it would be impossible to disambiguate the orientation of the LEGO brick used in the study without additional tools, such as a magnifying glass. Notably, when using our system with tracking enabled, the individual completed the assembly task presented in our study with a similar speed as participants without visual impairments from our formal user study. With this anecdotal observation, we wish to highlight the potential of tracking and guidance systems for empowering humans to perceive and manipulate the world around them and hope to inspire future avenues for accessibility research.

Limitations and future work

In this section, we highlight a number of limitations of our work, which should be taken into consideration when assessing our results and may highlight opportunities for improvement in future work:

Simplicity of the use case

It may be perceived as a limitation of our study that participants continuously inspect only one single type of object. While its orientation requires a careful analysis of the LEGO logo, the distinguishing features are always the same, and participants get familiar with it quickly. However, our collaborating industry partner identified this object as particularly representative and challenging, where workers complete comparable tasks that require repetitive visual inspection. This lends support to the ecological validity of our findings. We further argue that the utility of CADTrack would increase in tasks involving more varied objects that are near-symmetrical. Note that analyzing the element orientation presents only one step in the overall task supported by CADTrack. Beyond this, the system also delivers step-by-step instructions to assist the user in the assembly or inspection procedure.

Making CADTrack more accessible

The primary limiting factor for the adoption of CADTrack may be that the authoring, data generation, and training stages in our pipeline are currently done manually by the developer using development tools such as Blender and Pytorch. Integrating these pipeline steps into CADTrack through a dedicated tool to label the symmetry-breaking features, and enabling users to run the pipeline without programming knowledge, would make the system more accessible.

Tracking accuracy and performance

Although the system accuracy is high for confident predictions and participants reported high trust in the tracking, there were still a few false positives. Some tasks may require high precision, so it is crucial to mitigate false positives. For others, the system must run effectively on a device with limited computational capabilities, such as a smartphone. Improving the accuracy and speed of our tracking pipeline is another venue for future work - improvements in the data generation step, CNN architecture, or training procedure bring clear benefits in this direction.

Tracking generalization

Existing work demonstrates the capabilities of CNNs to generalize to various visual recognition tasks [139, 164]. Moreover, we use our pipeline to track a secondary object, showing promising results that our approach generalizes well. However, an extensive evaluation of other components with different characteristics is necessary to make a stronger claim. Such an evaluation would require the creation of a dataset focusing on near-symmetrical objects and their corresponding CAD model.

Other display methods

The simplicity of our system has the benefit of using off-the-shelf components. However, by showing instructions on a traditional display, our approach requires gaze shifts from the user between the task zone and the screen. A possibility for future work would be to make our system available in AR, to reduce gaze shifts and bring other benefits documented in related work [121, 211, 280].

10.7 Conclusion

We propose CADTrack, a digital assistant for disambiguating the orientation of near-symmetrical objects with camera-based tracking using off-the-shelf components. CADTrack can support typical inspection and assembly tasks through a tracking pipeline that generates training data from the object's CAD model and respective annotations about the symmetry-breaking features. Our benchmarks support the effectiveness of our approach, and a user study reveals a preference for dynamic instructions with live tracking and evidence of reduced physical effort when using our assistant.

Chapter 11

Paper C

This chapter presents the paper *XRgonomics: Facilitating the Creation of Ergonomic 3D Interfaces*, published in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*: CHI 2021. This paper won a best paper award.

[73] João Marcelo Evangelista Belo, Anna Maria Feit, Tiare Feuchtner, Kaj Grønbæk. XRgonomics: Facilitating the Creation of Ergonomic 3D Interfaces. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380966. https://doi.org/10.1145/3411764.3445349.

XRgonomics: Facilitating the Creation of Ergonomic 3D Interfaces

João Marcelo Evangelista Belo, Anna Maria Feit, Tiare Feuchtner, Kaj Grønbæk

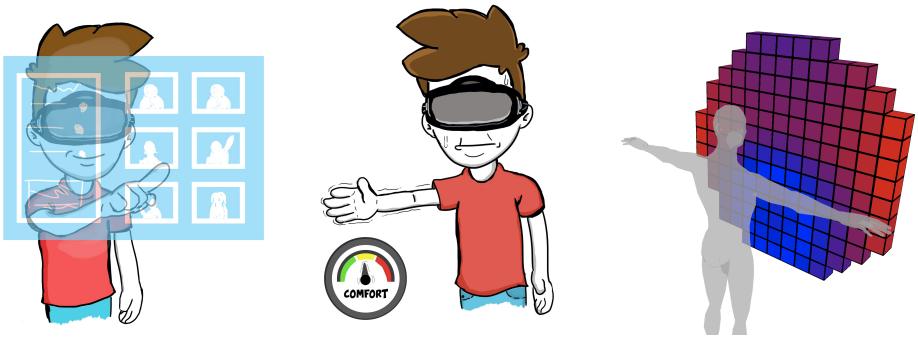


Figure 11.1: The XRgonomics toolkit aims to facilitate the design of ergonomic 3D UIs, common in mixed reality applications (left). We use a user's physiological model to compute the ergonomic cost of interaction at each reachable position in the interaction space (center). In XRgonomics, creators can visualize this cost through colored voxels in the interaction space: red indicates high and blue low-cost areas (right).

Abstract

Arm discomfort is a common issue in Cross Reality applications involving prolonged mid-air interaction. Solving this problem is difficult because of the lack of tools and guidelines for 3D user interface design. Therefore, we propose a method to make existing ergonomic metrics available to creators during design by estimating the interaction cost at each reachable position in the user's environment. We present XRgonomics, a toolkit to visualize the interaction cost and make it available at runtime, allowing creators to identify UI positions that optimize users' comfort. Two scenarios show how the toolkit can support 3D UI design and dynamic adaptation of UIs based on spatial constraints. We present results from a walkthrough demonstration, which highlight the potential of XRgonomics to make ergonomics metrics accessible during the design and development of 3D UIs. Finally, we discuss how the toolkit may address design goals beyond ergonomics.

11.1 Introduction

Cross Reality (XR) technologies are becoming mainstream as hardware gets more accessible, resulting in new applications across different sectors [49]. Despite the shift in interaction paradigms (e.g., from mouse input to mid-air interaction with controllers), interface elements and design guidelines for XR User Interfaces (UIs) are often inspired by 2D UI design. This influence can negatively affect user experience (UX) [151]. In particular, recent literature shows that creators struggle to address the physical aspects of XR experiences [6]. Existing challenges involve designing the posture of users and reducing fatigue. Remarkably, this problem persists even though substantial research in the HCI community has focused on mid-air interactions in the past decade, proposing design guidelines and evaluation metrics. A possible explanation is that these are difficult to apply during design and development of XR applications because:

- Proposed metrics [125] and models [135] focus on evaluating mid-air interactions that already exist but do not directly support the creation of new 3D UIs.
- General guidelines [10, 125] do not apply to the dynamic nature of MR applications that need to adapt constantly to the user's context [163].

To address these issues, we propose a method to make existing ergonomics metrics accessible to creators during design. We use a physiological model of the arm to assign a cost of interaction to any point in the user's reachable 3D space, that we call ergonomic cost. Its computation comprises the following steps:

1. **Discretization of the interaction space** – transfer of the continuous interaction space into a discrete representation.
2. **Computation of arm poses** – computation of multiple arm poses for each position in the interaction space using Inverse Kinematics (IK).
3. **Computation of ergonomic cost** – calculation of the ergonomic cost for each arm pose using existing metrics and heuristics that assess ergonomics.

To make our method accessible to creators, we introduce XRgonomics - a toolkit to compute and visualize the ergonomic cost of the user's 3D interaction space. It comprises two major components: A Graphical User Interface (GUI) and an Application Programming Interface (API). The GUI allows creators to visualize the ergonomic cost associated with each position in the interaction space. The API gives access to this data at runtime to support development of adaptive interfaces. XRgonomics does not require any specifications about the XR application, making it easy to use during various design processes. To achieve this, we simplify the computation of the ergonomic cost by considering only static arm poses. We disregard

users' arm motion between points of interaction which is difficult for creators to predict [6].

Two scenarios show how the toolkit can support the design of static UI elements and dynamic adaptation of UIs based on spatial constraints. To assess the usefulness of the toolkit, we present our findings from a walkthrough demonstration conducted with UI design experts. Finally, we discuss the potential of XRgonomics to address design goals beyond ergonomics. All the source code is available at: https://github.com/joaobelo92/xrgonomics.

11.2 Background and Related Work

Designing for ergonomics

Ergonomic factors in physical workstation design

Assessing ergonomic factors plays an important role when designing physical spaces, such as workstations, cars, and terminals. Much prior work in this domain estimates discomfort and ergonomic issues based on simple heuristics, such as joint angles. An example is RULA, a survey method for investigating work-related upper limb disorders [174]. RULA records working postures and attributes scores depending on risk factors. It assesses the risk for upper limb disorders considering aspects such as arm poses, movements, and forces.

Analysis of robot workspaces shares challenges also found in ergonomics design. For instance, Zacharias et al. [282] proposed a method to show which positions are easy to reach for robot arms. A significant challenge inherent to these scenarios is the limitation imposed by the agent's physical environment. For example, the physical space within a car cockpit limits the possibilities for where to mount a dashboard. In contrast, virtual workspaces are more flexible and allow the 3D user interface to adapt continuously to the user's context.

Ergonomic factors in mid-air interaction

Ergonomics are a significant factor in the design of virtual user interfaces, particularly in 3D UIs. Arm fatigue is one of the main issues designers must consider [151]. It is a common problem in interaction with vertical screens, also known as the *gorilla-arm effect* [34]. Researchers have proposed novel approaches to address this issue, ranging from novel interaction techniques [34, 80, 166] to UI optimization methods [184] to reduce muscle strain and fatigue. What these approaches have in common is that they intend to reduce fatigue in interaction. Among the most prominent qualitative methods to assess subjective fatigue are Likert scales [43], the NASA Task Load Index (NASA-TLX) [109], and the Borg CR10 scale [33]. HCI studies usually apply these approaches because they are non-invasive and do not require specialized equipment. However, substantial work must go into preparation and user studies, and these techniques provide only a coarse estimation of fatigue. While objective methods overcome some of these limitations, techniques used in biology and sports

science often rely on external measurements, such as muscle activations [48], blood pressure [241], and heart rate [236]. Because these methods require specialized equipment and might interfere with the user's task, they are often inappropriate for HCI studies.

The HCI community has proposed alternatives to objective methods that are not intrusive. For example, Consumed Endurance (CE) [125] is a metric that tracks the user's arm pose to quantify arm-fatigue. CE computes the center of mass of the arm over time and uses that information to predict how long the user can continue interaction before the shoulder muscles need rest. The authors show that the metric correlates well with the Borg CR10 scale and propose several guidelines for the design of mid-air UIs. Other studies use muscle activations from biomechanical models as indicators of fatigue. Bachynskyi et al. [9] show that predictions of muscle activation from static optimization correlate well with EMG data. In subsequent work, Bachynskyi et al. [10] applied biomechanical simulations to create a set of heuristics for designing 3D pointing interfaces, highlighting the potential of biomechanical simulations in UI design. Later, Jang et al. [135] proposed a method for modeling cumulative fatigue. Their approach quantifies arm fatigue by introducing a model for estimating muscle states (active, rest, fatigue) and uses a biomechanical arm model to estimate maximum shoulder strength. This approach makes it possible to consider periods of both interaction and rest.

Another ergonomic issue interlinked with fatigue is user comfort [151]. User comfort covers both physical and psychological dimensions, encompassing broader aspects such as posture and social awkwardness that may arise from using gestures in public spaces. Although there is work exploring subtle mid-air interaction [166], we are not aware of guidelines or objective metrics to evaluate this issue.

The primary goal of XRgonomics is to make ergonomics metrics accessible to creators during design. We use established objective metrics as heuristics for comfort, to assess the quality of positions in the interaction space regarding ergonomic factors, such as fatigue and comfort. Our toolkit supports several of the metrics introduced above, namely RULA, Consumed Endurance, and biomechanical simulations.

Computational support for UI design

Already 20 years ago, Myers, Hudson and Pausch highlighted the need for toolkits to support the creation of user interfaces [188]. Since then, researchers have proposed several computational methods to support UI design (see survey by Oulasvista et al. for an overview [199]). Some of these methods focus on ensuring user performance, while others make suggestions to improve the aesthetic qualities of an interface. Such computational methods often differ in the degree of involvement of the designer. At one end of the spectrum, tools automatically create UI designs and do not require designer involvement [91]. Other toolkits support the creator by observing their design process, evaluating manually created solutions, and generating alternative designs or changes, which the creator can choose to follow [11, 256]. Studies show that such

tools improve the quality of designs and inspire creators, ultimately resulting in a collaborative environment involving the designer and the toolkit [142].

The support of computational methods is crucial for MR applications where the context of the user continuously changes. Existing work has explored methods that automatically determine *where* to place virtual content [79, 92, 194]. Others have investigated *how* to display virtual content to the user [63, 138, 254], or a combination of multiple aspects [163]. However, none of these automated approaches considers ergonomics. Designing for the physical aspects of interaction is one of several common difficulties during the creation of XR applications, as highlighted recently by Ashtari et al [6]. Among the key challenges identified by the authors, we aim to address the lack of concrete and accessible design guidelines.

In this work, we use computational methods to support 3D UI design for MR and VR applications. XRgonomics is a toolkit that supports the visual exploration of the design space in terms of ergonomics, enabling creators to make informed decisions about where to place UI elements as part of their standard design process. Also, creators can use XRgonomics to guide the layout of 3D UIs at runtime and specify areas of interaction to avoid or prioritize.

11.3 Ergonomic cost pipeline

When designing XRgonomics, our primary goal was to create a method that supports the design of ergonomic user interfaces during the early design stages of XR applications. To facilitate accessibility, we did not want to impose constraints on the application itself, nor require the content creator to provide extensive input about the to-be-designed interface (e.g., usage data, user profiles, or physical environment). For that reason, we developed an approach that does not make assumptions about the interaction space or interaction techniques involved. Another noteworthy aspect is that interaction in XR applications is often context-dependent. Consider a typical Hololens 2[1] application, where the UI comprises virtual mid-air interaction with buttons and sliders. Contextual aspects such as the task, environment, or user's pose can limit interaction with the system. For this reason, general guidelines for ergonomic 3D interface design are often inappropriate for XR applications. To overcome this challenge, we facilitate exploration of the interaction space during UI design and allow developers to use ergonomics metrics at runtime. In our approach, we analyze the entire interaction space and assign a cost of interaction at each reachable position in 3D space. We call this the ergonomic cost. For it to be accessible in real-time, we propose a pipeline that shifts the computationally intensive tasks to a pre-processing stage. This ergonomic cost pipeline comprises three steps that we describe in the following sections. Our approach allows the comparison of distinct reachable positions regarding different ergonomic aspects, opening novel possibilities for designing and optimizing user interfaces.

[1] https://www.microsoft.com/en-us/hololens/hardware

Discretization of the interaction space

In the initial step of the pipeline, we transfer the continuous interaction space into a discrete representation. This is necessary to make the problem computationally tractable. Hence, we represent the interaction space as a 3D Cartesian grid and call each element a voxel - a common term in computer graphics. We define the interaction space based on the positions a human can reach and manipulate objects with his hands from a fixed torso position, a concept also known as the reach envelope [52]. We use a simple kinematic chain between the shoulder and hands. A user representation that includes both arms requires a fixed offset between the shoulders and thorax. However, the shoulder's mechanics are complex, and shoulder joint motion depends on its component joints [131]. Hence, this simplification results in some loss of precision, but not enough to justify a more complex kinematic chain for our use case.

To generate the interaction space's voxel representation, we start by setting up voxel dimensions with a default side length of 10cm. Creators can adjust the voxels' side length to change the granularity of the interaction space representation. We use a simple algorithm that iterates through an overestimated 3D Cartesian grid in a cube (Figure 11.2, black cube). Its side length is equal to the kinematic chain dimensions, which delimit the arm's reach. Applications can include a calibration step, so these dimensions accurately reflect the user. In our standard implementation, we use the arm dimensions of the 50th percentile male [98]. Then, we verify which voxels belong to the interaction space, removing the voxels outside of the reach envelope (Figure 11.2, yellow sphere). We do this by checking whether the distance from the shoulder to the center of a voxel is smaller or equal to the user's arm length.

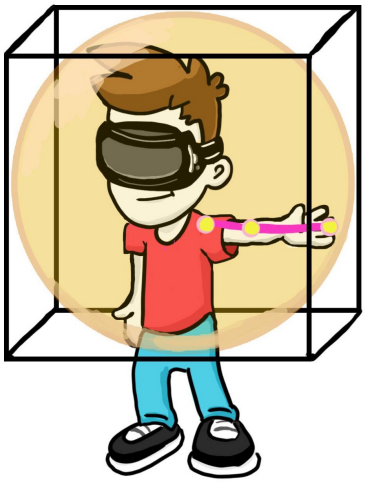


Figure 11.2: The interaction space is computed from an overestimated 3D Cartesian grid (black cube) and delimited by the user's reach envelope (yellow sphere). A simple kinematic chain representing the user's arm can be seen in pink.

Computation of arm poses

At the end of the pipeline, the result will be the cost of interaction for each position in the user's reach. But first, we must compute multiple poses the arm can take to reach each voxel in the interaction space. Related work points out lower risks of injury and reduced muscle load for postures with the wrist in a neutral position [174] (deviation and twist is 0 degrees). We aim to find the pose that minimizes discomfort, and because postures of the wrist in neutral positions are considered optimal, we simplify the kinematic chain further by removing this degree of freedom. This results in a two-segment body of the arm, where the forearm and wrist constitute a single segment (see Figure 11.2, pink kinematic chain). While this approach considers fewer possible arm poses, it significantly reduces the complexity of the inverse kinematics (IK) process and computation time of the pipeline. We base our IK process on the work of Tolani et al [258]. Considering fixed end-effector and shoulder positions, the elbow is free to swivel on an axis between these two points (see Figure 11.3), allowing us to express the elbow position as a function of ϕ about the $\hat{\mathbf{u}}$ axis:

$$e = r[cos(\phi)\ \hat{\mathbf{u}} + sin(\phi)\ \hat{\mathbf{v}}] + c$$

Where r denotes the radius and c the center of the circle described by the swiveling elbow joint [258]. The variable ϕ controls the elbow position, which is at its lowest height when $\phi = 0$. Note that the arm's physiology constrains the ϕ value and we disregard unreasonable postures of the arm, based on impossible joint angles and elbow positions. Therefore, to generate arm poses, we increase ϕ by a constant value ψ, which determines how much the elbow rotates until it reaches an anatomically impossible threshold (e.g., 150°). Here, the ψ value determines how fine-grained the discretization of the elbow position is. At this stage, it is possible to customize thresholds for ϕ and create additional rules, to consider factors such as a user's physical impairments or constraints imposed by hardware.

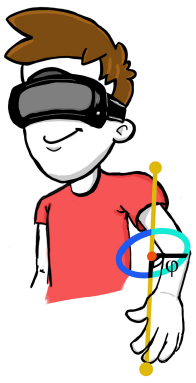


Figure 11.3: Our inverse kinematics approach considers the shoulder and end-effector (i.e., hand) positions to be fixed, and the elbow is free to swivel about the shoulder-hand axis.

Computation of the ergonomic cost

Our toolkit implements established metrics from HCI and ergonomics research to assess the ergonomic cost of each reachable voxel. In theory, any metric that considers arm poses to assess ergonomic factors is appropriate to compute this ergonomic cost. XRgonomics currently supports consumed endurance (CE) [125], Rapid Upper Limb Assessment (RULA) [174], and muscle activations from biomechanical simulations. Notice that some of these metrics, such as CE, consider motion. In those cases, we adjust the metric to consider only static arm poses and use the result as a heuristic for strain. In other words, the ergonomic cost is a measure of how comfortable it is to maintain interaction at a specific position in the interaction space.

In the previous step of the pipeline, the toolkit generated several arm poses for reaching each voxel. We then compute the ergonomic cost for each of these poses, and assign the one of these costs (i.e., of the pose with least discomfort) to the corresponding voxel. We base this strategy on findings that humans tend to use more efficient poses [224].

In XRgonomics, a creator can compute the ergonomic cost using one of the supported metrics, or combine multiple metrics by assigning a weight to each. In the next sections we will describe how we applied each metric in our pipeline.

Consumed Endurance (CE)

To quantify fatigue in mid-air interaction, CE [125] considers endurance of the shoulder in terms of torque as ratio to the interaction time. We follow the authors' approach and use shoulder torque as an index for muscle strain. As the authors mention, when there is no motion, the shoulder torque has to match the gravity torque $\vec{g}$:

$$\left\|\vec{T}_{shoulder}\right\| = \left\|\vec{r} \times m\vec{g}\right\|$$

Where $\vec{r}$ is the distance from the shoulder joint to the center of mass of the arm, and m is the mass of the arm. Since we are working with static poses, we can directly compute the center of mass of each pose and apply the formula above. The result is a heuristic for the ergonomic cost based on the CE approach to estimate muscle contraction.

Rapid Upper Limb Assessment (RULA)

RULA [174] assigns posture scores to the upper limbs, neck, trunk and legs, depending on joint angles. Combining that information with muscle and force scores, the method results in a final score to assess risk factors associated with upper-limb disorders. Even though our approach only considers arm poses, RULA posture ratings convey relevant information about postures that prevent or might result in upper limb disorders. Hence, we use RULA's posture scores to compute a score based on the joint angles of the upper and lower arms' joint angles (see posture scores for group A [174]). Low posture scores reflect a working posture with minimal risk factors, while higher

numbers indicate an increasing presence of risk factors. The final score can indicate which positions in the interaction space are preferable to avoid upper-limb disorders.

Muscle activations from Biomechanical Simulations

Biomechanical simulations can estimate muscle activation for a motion, which can indicate energy consumption and fatigue [9]. Therefore, this method has great potential as a heuristic for the design of 3D UIs. Typical biomechanical simulation pipelines use experimental motion data, which typically involve mapping physical to virtual markers, scaling the model to match the subject dimensions, using inverse kinematics to compute joint angles, and a final step to estimate muscle activations [10]. For our simulations we use OpenSim 4.1[2], an open-source tool for biomechanical modeling and simulation [53], and the upper extremity model created by Saul et al. (MoBL) [231]. This model has the dimensions of the 50th percentile male, and must be scaled to support other arm dimensions. Because we generate arm poses in the previous step of the pipeline, we only use OpenSim to estimate muscle activations. However, we must convert our vector representation of the arm's pose into OpenSim's generalized model coordinates and generate corresponding motion files where each arm pose remains static over a short time (refer to the source code for more details). We use static optimization[3] to estimate muscle activations for each pose, which is a fast and efficient method. To run our simulations, we follow Hicks et al.'s recommendations [123]. We used reserve actuators to prevent the model from being under-actuated and avoid failures in static optimization. These reserve actuators complement the model's muscles when these cannot generate sufficient forces to achieve a pose. It is important that reserve moments are small or non-existent [123], so that the model's muscles exert most of the forces necessary to maintain each pose. Hence, we use low optimal forces in our reserve actuators, to ensure the cost function in the static optimization algorithm prioritizes muscle forces. Because MoBL is a complex model and static optimization can converge to different results, we analyze each pose over time and save the timeframe that minimizes reserve actuation for each pose. This results in an activation value for each muscle and reserve actuator in the model. To facilitate comparison with other metrics, we combine these into a single ergonomic cost value. To do so, we average the muscle activations and sum all the reserve actuators. To prioritize results that mostly use muscle forces, we penalize cases where reserve moments are high. Note that while muscle activation ranges from 0 to 1, the same does not apply to reserve actuators. Therefore, we set a threshold for the maximum acceptable reserve forces ($T_{reserve}$), based on the net joint moments [123]. Then, we divide the sum of the reserve forces by $T_{reserve}$, which will always result in a higher value than the average muscle activation, if it the reserve forces are above the

[2] https://simtk.org/projects/opensim

[3] https://simtk-confluence.stanford.edu/display/OpenSim/How+Static+Optimization+Works

threshold. This results in the following ergonomic cost function:

$$erg\ cost = \frac{\sum_{n=1}^{M} n_{activation}}{M} + \frac{\sum_{n=1}^{A} a_{activation}}{T_{reserve}}$$

Where M is the number of muscles and A the number of reserve actuators in the model.

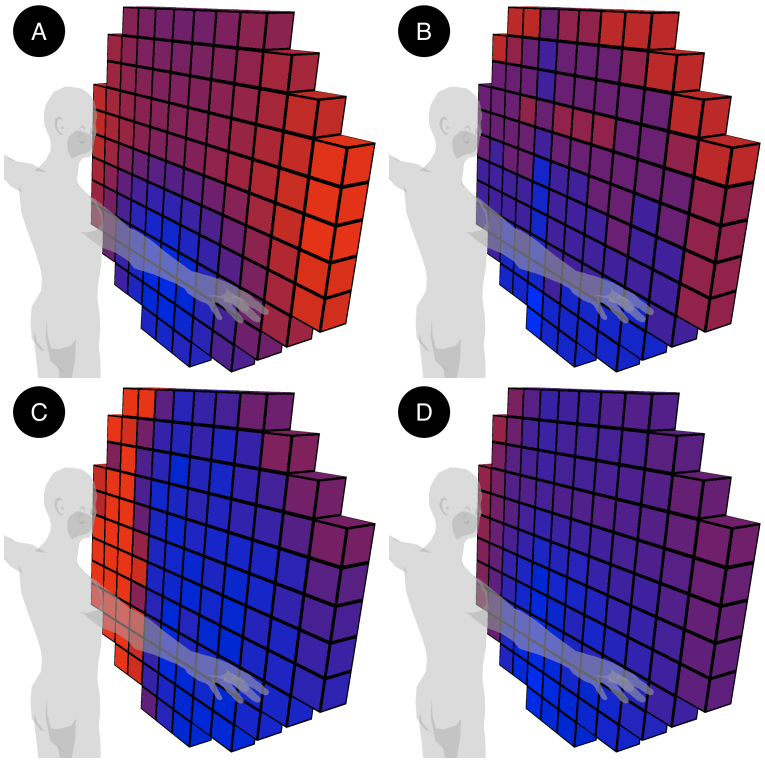


Figure 11.4: Visualization of the interaction space of the right arm using the supported metrics: A) Consumed endurance, B) RULA, C) Muscle activation, D) Weighted average (arithmetic mean in this case). The image shows only the voxels at the 40 cm slice (x-axis)

11.4 The XRgonomics toolkit

The pipeline described in section 3 constitutes the central part of XRgonomics, a toolkit that gives creators of 3D applications easy access to ergonomics metrics during design and development of 3D adaptive UIs. The toolkit comprises two major components: A Graphical User Interface (GUI) and an Application Programming Interface (API). The GUI allows creators to visualize the interaction space and each voxel's ergonomic cost. It can support the design of static interfaces (e.g., positioning virtual buttons on a desk) or give an overview of different metrics and their correspondent

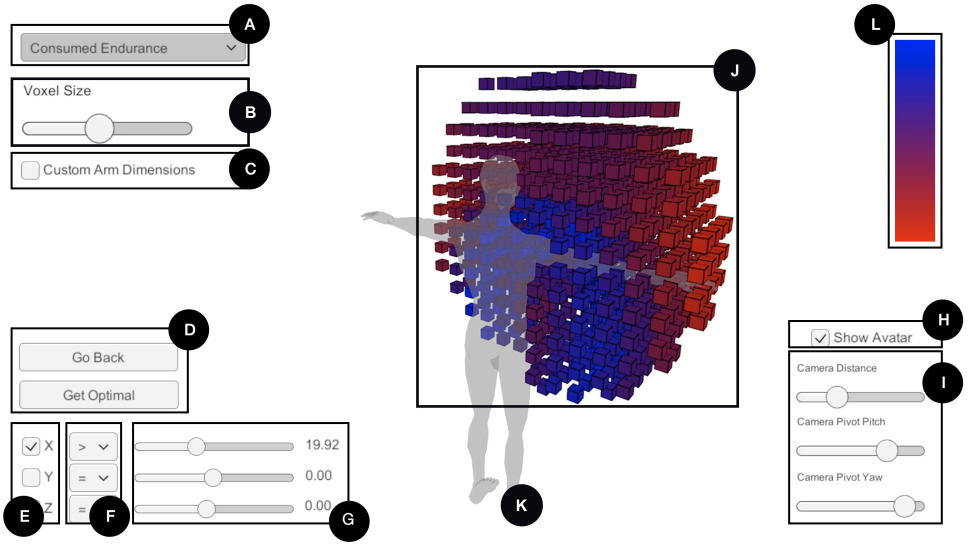


Figure 11.5: The XRgonomics GUI allows creators to visualize the interaction space and ergonomic cost of each voxel according to different ergonomic metrics. We will briefly describe each UI element: A) Dropdown menu for metric selection; B) Slider for voxel size setting; C) Menu to run computation pipeline for different arm dimensions; D) Buttons for retrieving the "optimal" voxel with the lowest ergonomic cost; E) Checkboxes for enabling/disabling spatial constraints; F) Dropdown list with comparison operators (=, >= or <=); G) Sliders for setting constraint values; H) Checkbox to toggle display of the avatar as visual reference for the shoulder position; I) Camera controls; J) Visualization of the interaction space and ergonomic cost in form of colored voxels; K) Avatar; L) Color mapping for the ergonomic cost, from blue (most comfortable) to red (least comfortable).

ergonomic cost for different positions in the interaction space (Figure 11.4). The API allows developers to use the ergonomic cost at runtime. This feature allows developers to create adaptive 3D UIs that consider user comfort as a criterion in the formulation of the optimization problem. For example, developers can retrieve voxels that minimize the ergonomic cost under specified spatial constraints in real-time. In this initial version of the toolkit, we consider only the right arm. Therefore, we set the center of the interaction space on the shoulder instead of the user's thorax. The source code for XRgonomics is available at `https://github.com/joaobelo92/xrgonomics`.

Graphical User Interface (GUI)

The GUI is implemented in Unity and uses the API to retrieve the ergonomic cost data. By default, it supports the arm dimensions of the 50th percentile male [98]. Creators can directly change parameters, such as the user's arm and voxel dimensions (Figure 11.5, C). Modifications to other parts of the pipeline require updates in the source code (see API section for more details). The GUI allows creators to visualize

the interaction space and each voxel's ergonomic cost (Figure 11.5, J). The user can select between different ergonomic metrics supported by the toolkit (Figure 11.5, A). XRgonomics supports the metrics described in section 3.3 and a weighted average of those three metrics. Because the interaction space is a sphere, the voxels in the interior might be occluded. For that reason, the GUI features controls to apply spatial constraints on each coordinate axis (Figure 11.5, E), to limit the range of visible voxels. For example, it is possible to visualize a "slice" of voxels by adding an equality constraint on one axis (Figure 11.4). Creators can also reduce the rendering dimensions of the voxels to visualize data through more than one "slice" (Figure 11.5, B). An avatar is depicted in the center of the GUI, as a reference for the user's shoulder position in the interaction space (Figure 11.5, K).

Each voxel is colored according to the selected metric and the arm pose with the minimum ergonomic cost. As previously mentioned, we base this design choice on the principle that humans tend to use efficient poses [224]. Because CE and Biomechanical simulations output continuous results, we normalize all the ergonomic cost data using a simple feature scaling formula:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

The color mapping is a linear interpolation from blue to red, representing low to high ergonomic cost, respectively (Figure 11.5, L). This mapping allows creators to visualize and compare voxels with similar values. Note that computed muscle activations from biomechanical simulations differ by small values when not influenced by reserve forces, which may result in identical voxel colors, even though there is a difference in average muscle activations. Therefore, we use a different normalization strategy to facilitate the visualization of this metric. We normalize the average muscle activations, multiply them by a scaling factor, and sum it with the ergonomic cost previously computed. This makes voxels with high reserve forces appear red, while smaller differences in the average muscle activations remain visible.

Finally, creators can click on a single voxel to visualize the arm poses generated by the IK process and their correspondent ergonomic cost (Figure 11.6). Since we are using a simplified kinematic chain, only the elbow positions differ in each pose.

Application Programming Interface (API)

We implemented the API in Python and used NumPy for most mathematical operations. The API has an endpoint to run the ergonomic cost pipeline for different arm and voxel dimensions, but developers must update the source code to change parameters in the inverse kinematics step, such as joint rotation limits or complex spatial constraints. While we implemented the algorithms for CE and RULA, the toolkit uses the OpenSim 4.1 Python bindings to run biomechanical simulations. However, XRgonomics does not directly support scaling of the arm or other changes to the biomechanical model, and the OpenSim tool is necessary for such tasks. We store voxel and ergonomic cost data in R*Trees [15], using an SQLite database. This makes it fast to find positions in

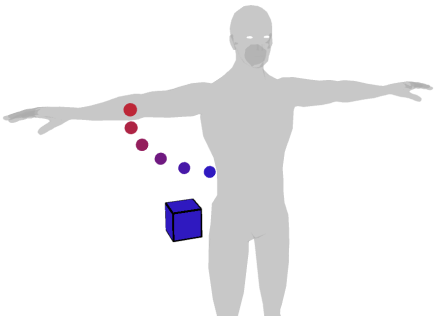


Figure 11.6: Creators can click on a voxel to visualize the possible positions of the elbow and the pose's ergonomic cost.

3D space that minimize a particular ergonomic metric or meet specific requirements. R*trees allow developers to query voxels within a bounding-box or arbitrary shapes like the area visible to a 3D camera. For networking, the API uses the ZeroMQ[4] framework, a fast messaging library. These networking features allow the API to communicate with the GUI (Unity), and enable developers to integrate XRgonomics in their applications. For example, developers can run the ergonomic cost pipeline for custom arm dimensions and retrieve data under specified spatial constraints. In our tests, the response time for such queries was less than 10ms, showing that the API can process requests in real-time and is suitable for XR applications.

11.5 Evaluation of the toolkit

Ledo et al. introduced a categorization of evaluation strategies for HCI toolkit research [153]. We applied two strategies identified in this work to evaluate XRgonomics. First, we illustrate what the toolkit might support by discussing the usage of XRgonomics in two distinct scenarios: ergonomically optimized placement of static 3D UI elements, and runtime adaptation of a 3D UI based on dynamic constraints. Then, we collect feedback from potential toolkit users to explore its utility through a walkthrough demonstration.

Demonstration of application scenarios

To demonstrate the functionality of XRgonomics, we implemented two application scenarios for 3D UI design that we describe in the following sections:

Guiding the placement of static UI elements

Consider a "traditional" AR application, as Grubert describes it [101], where a designer defines the position of UI elements based on the user's pose. Creators can

[4] https://zeromq.org/

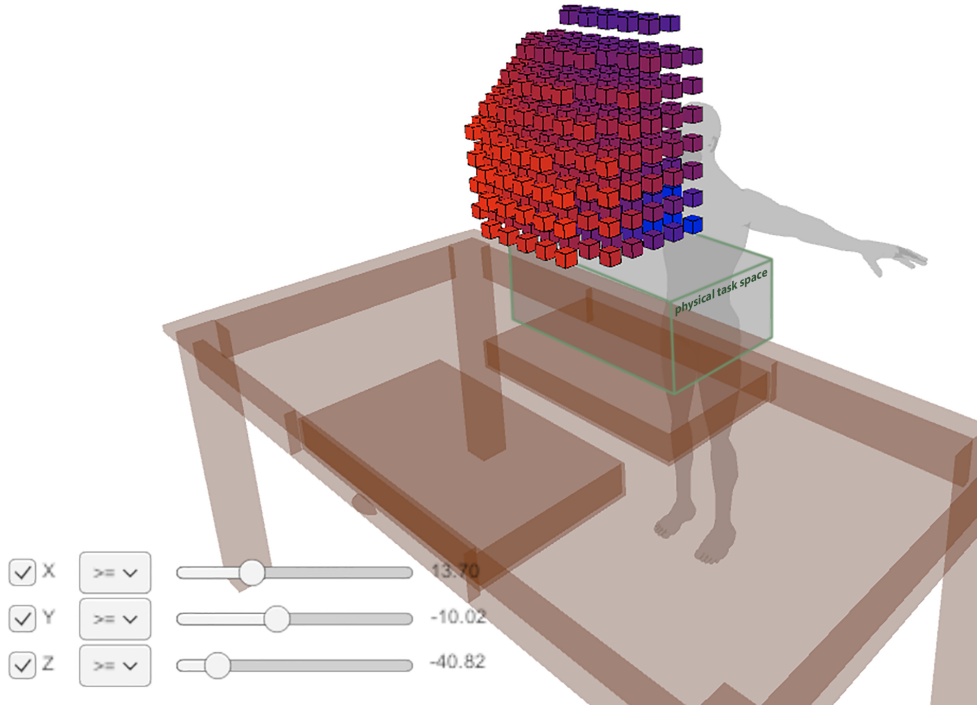


Figure 11.7: Creators can use the XRgonomics GUI to guide the design of static UI elements in traditional AR applications. Representations of physical/virtual objects can be added in the Unity scene to facilitate the task. In this case, the creators use constraints on the x and y axis to visualize the interaction space above and to the right zone of a table.

use the XRgonomics GUI to visualize the ergonomic cost for each position in the user's interaction space and guide the placement of 3D UI elements under specified constraints. For example, consider positioning virtual input elements on an office desk. The designer can analyze all the positions above the desk by setting constraints on different axes (see Figure 11.7), and use this information to design virtual elements such as a calculator or a drawing-board.

Dynamic adaptation of 3D UIs

Changes in the user's task, environment, and pose can limit interaction in MR applications. Because context changes are difficult or impossible to predict during design and development, a solution is to adapt the UI to the user's context at runtime. We implemented a prototype to show how creators can use XRgonomics to design adaptive ergonomic UIs. In this simplified MR scenario, we use the XRgonomics API to adapt the placement of a virtual music player menu in a Hololens 2 application. The UI consists of a virtual 3D menu with buttons to play, stop, or change songs. Although the controls are easily accessible when the menu is visible, the limited field of view (FoV) of the Hololens can make interaction challenging. To overcome

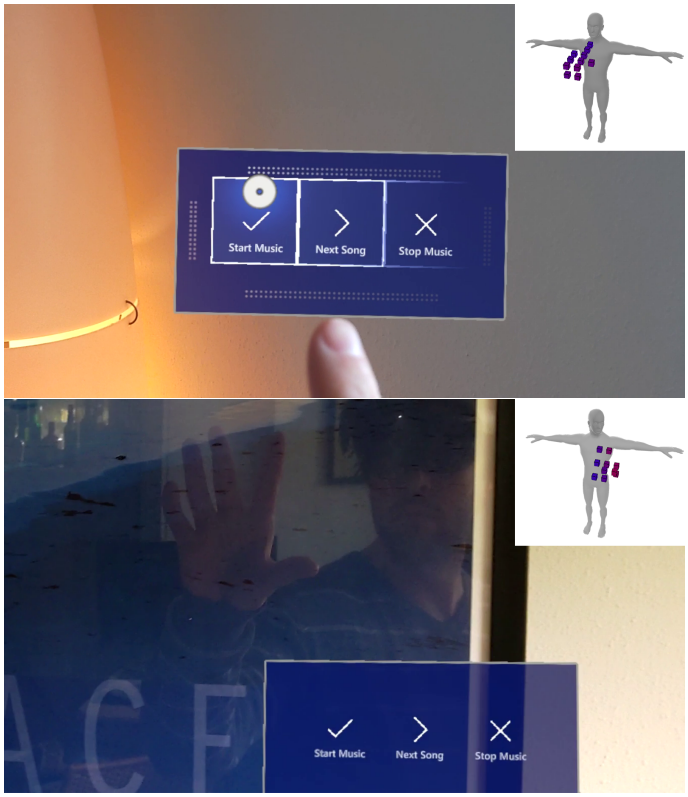


Figure 11.8: A proof-of-concept application on the Hololens allows automatic placement of a virtual menu (music player) in the ergonomically optimal position within the user's FoV. The left image shows the user interacting with the virtual menu while looking straight ahead. The upper right visualization shows the ergonomic cost within the user's current FoV. When turning the head in other direction, the constraints are updated based on the new FoV (right picture). With a gesture, the user can summon the menu to reappear in the most comfortable position within this new zone of the interaction space.

this issue, the user can request the menu to move into his FoV with a gesture (see Figure 11.8). The prototype then uses the XRgonomics API to identify the most comfortable and reachable position in the user's FoV and moves the virtual menu there. To achieve that, we use the view frustum of the Hololens as a spatial constraint. This application was implemented in Unity, using the Mixed Reality Toolkit (MRTK) and XRgonomics API. MRTK provides algorithms to facilitate the positioning of virtual menus (solvers[5]). However, these are limited to behavior like surface magnetism or following a virtual object and do not consider ergonomics.

[5] https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/README_Solver.html

Walkthrough demonstration of toolkit

To explore the utility of the toolkit, we conducted a walkthrough demonstration [153] with representatives from our target group. The study comprised six phases (Table 11.1). In each phase, we conducted semi-structured interviews with open-ended questions, rather than using for example questionnaires, to gain more in-depth insights. We will discuss the goals and findings from each phase in the following sections. Due to the COVID-19 pandemic, the study was conducted online through a video conferencing tool with screen sharing. For further reference, the study protocol, interview transcripts, and questionnaires are available in the project repository. We recruited eight participants (2 female; age: $M = 31.3, SD = 2.8$). All participants were familiar with UI design and XR technology, as they were professional software developers or VR/MR researchers. Most participants were uncertain about the concept of ergonomics, and none had prior knowledge of the metrics RULA, CE, or muscle activations.

1. Review of 3D UIs	2. Introduction of GUI	3. Design Task	4. Metrics overview	5. Introduction of API	6. Conclusion
Topic introduction and examination of prior knowledge	Instruction on how to use the GUI and visualization	Participants use the tool to design a static 3D UI with 3 elements	Demonstration and explanation of different ergonomics metrics	Demonstration and discussion of API features	General feedback and final remarks

Table 11.1: Overview of the study procedure consisting of a walkthrough demonstration of XRgonomics and a design task (phase 3), where participants used the GUI to create a static 3D UI.

Review of 3D UIs

In an initial discussion about 3D UI design, we aimed to learn about participants' prior knowledge and concerns regarding ergonomics. Three participants could relate directly to ergonomics and fatigue issues (P1, P3, P7) and all acknowledged the importance of the topic. However, none had addressed the problem in practice and they were not aware of existing metrics or strategies to use, apart from referring to guidelines for particular tools (e.g., ARKit[6] and ARCore[7]) (P3). These insights highlight one of the key barriers identified by Ashtari et al. [6], about the difficulties in designing for the physical aspects of AR/VR applications.

Introduction of GUI

In this phase, we introduced the XRgonomics prototype, explained the interaction space voxel representation, and instructed the participants on how to use the GUI (see section 11.4). To confirm that participants understood the visualization based

[6] https://developer.apple.com/design/human-interface-guidelines/ios/system-capabilities/augmented-reality/

[7] https://designguidelines.withgoogle.com/ar-design/

on our explanations, we prompted them to describe the ergonomic characteristics of the interaction space when referring to a "slice" of voxels using the CE metric, as illustrated in Figure 11.5 (J). All participants showed an intuitive understanding of blue areas being "most comfortable" (P1), "easiest" (P5), and "most relaxed" (P7) to reach with the hand. We then challenged their understanding of this visualization, by pointing out some questionable CE results for positions above head-level (see Figure 11.5). Several participants expressed some uncertainty and even disagreement with the values in this area, which they perceived as hard to reach and therefore expected a higher ergonomic cost (e.g., P0, P3, P4, P7). However, instead of doubting the metric, they came up with likely explanations for why their opinions were wrong (e.g., P1, P3, P4, P5). We conclude that while the ergonomic cost and the toolkit visualization are easy to understand, creators might over-trust the tool, interpreting the visualizations as the ground-truth instead of reflecting on the validity of the metric. Hence, such tools should encourage creators to be critical and clarify the the metrics strengths and weaknesses.

Design Task

To evaluate the usefulness of XRgonomics in UI design, participants completed a design task using the toolkit. It consisted of planning the layout of three UI elements with different usability aspects (e.g., usage frequency) in a workstation (similar to Figure 11.5). At this stage, we enabled remote control of the mouse cursor, and the participants could use the toolkit running on the experimenter's PC. We asked them to think aloud while exploring the visualization, and show their desired UI element locations by pointing with their mouse or selecting a particular voxel. For a UI element that required frequent hand manipulation, all the participants used the toolkit to locate voxels with a low ergonomic cost. When deciding on the position for a rarely used element, with which inadvertent interaction is undesirable (e.g., "delete all"), participants pursued different strategies. To ensure the user makes a deliberate choice, some participants selected areas with a high ergonomic cost (P0-P3), while others also considered the workspace layout (P4-P7). All the participants stated that the visualization of the interaction space and ergonomic cost informed their decisions. Finally, when placing a non-interactive display element, participants pointed out that the supported metrics were not relevant, revealing an opportunity to integrate other metrics beyond ergonomics, such as visibility and consistency.

To explore the potential benefits of using XRgonomics in contrast to formulated guidelines from existing work, we quoted two design guidelines from the CE paper [125] and asked participants how they would apply these in the previous task. Participants highlighted several limitations of written guidelines, such as verbal statements being ambiguous or open to interpretation (P2-P6), whereas XRgonomics allows the designer to visually explore the interaction space (P0, P5, P7). Further, written guidelines may not apply if the recommended area is unavailable (e.g., because of physical restrictions). In contrast, setting constraints in XRgonomics allows the creator to analyze voxels in specific zones, compare, and identify locations that may

not be the best overall but are optimal for a particular scenario (P0, P1, P3, P4).

Metrics overview

Next, we showed the ability to visualize different metrics in XRgonomics, briefly explaining the underlying theory and how the ergonomic cost is computed for CE, RULA, and muscle activation, respectively. All participants agreed that the visualizations aided their understanding of the underlying concepts, while appreciating that the toolkit makes the metrics accessible and useful without knowledge of the formal details.

Introduction of API

To explore the potential of XRgonomics to develop adaptive UIs, we explained the features accessible through the API and showed a video of the AR prototype described in section 4. Overall, participants appreciated the idea of generating constraints automatically and proposed several use cases for adaptive UIs. However, some participants mentioned that the toolkit should allow the designer or end-user to modify these constraints (P1, P2, P3) to address personal preferences, implicit spatial requirements, or a physical disability.

Conclusion

To collect general feedback and identify limitations of the toolkit, we concluded the study with some final questions. Participants agreed that the visualization provided understandable information about ergonomics in the interaction space, and mentioned that XRgonomics would help 3D UI design from early stages of design and development. They also proposed support for additional metrics beyond ergonomics, such as spatial relations between (physical/virtual) objects (P3, P5, P6), eye strain (P6), and visibility (P0, P1, P4-P7). A participant asked about having XRgonomics integrated into development tools, such as Unity (P4), which would facilitate access to it.

We conclude the results from this walkthrough demonstration highlight the potential of XRgonomics to make ergonomics metrics accessible during the design and development of 3D UIs.

11.6 Discussion

In this work, we propose a method to estimate the ergonomic cost at each reachable position in the user's interaction space. We make this cost available to creators during design and development through XRgonomics, a toolkit to facilitate the creation of ergonomic 3D UIs. We demonstrated its potential through two examples: guidance for placement of static UI elements, and dynamic adaptation of 3D UIs optimized for comfort. Finally, we presented a walkthrough demonstration that highlights how

XRgonomics can support UI design experts. We will now discuss the limitations of our approach, avenues for future work, and other relevant findings.

Limitations of XRgonomics

To create a method that runs in real-time, we simplified multiple steps of the pipeline. A simple kinematic chain limits the number of possible poses represented by the model to allow for a simple and fast inverse kinematics algorithm. Although in most cases a fixed wrist angle in the kinematic chain results in an ergonomic position, interaction with complex 3D input does not always work under such conditions, and environmental constraints might require poses with different wrist angles. Further, modeling the shoulder mechanism and its relation with the torso would require more complex IK.

Another design trade-off we made, was to ignore motion. Without context, existing models that analyze movement and fatigue are difficult to use, because it is hard to forecast certain aspects of interaction, like movement [6]. Therefore, we consider only static poses, allowing creators to easily use XRgonomics at design time.

Finally, XRgonomics currently supports metrics related to the ergonomics of the upper limbs. However, several other factors impact interaction in XR applications, such as visibility and consistency.

Future work

Improvements to the IK implementation can result in higher accuracy and support more arm poses. In particular, extending the kinematic chain to consider the wrist angle is a natural improvement to the work we present. A possible approach would be to use a spiral point algorithm at each voxel to compute possible wrist positions. Another related improvement is to consider the user's torso position, with a model that represents the shoulder mechanism. This would result in more realistic arm poses and an improved representation of the interaction space. Another avenue for future work is to consider motion and model fatigue over time, based on the movement between voxels, their ergonomic cost, and muscle endurance. It will also be interesting to expand XRgonomics to consider other human factors beyond ergonomics of the upper limbs, such as vision, cognition, and spatial relations of objects.

Integration in existent MR and VR toolkits, such as MRTK solvers or Unity's IDE, is another important direction that would make our method more accessible to creators, as mentioned by study participants. The walkthrough demonstration also revealed several opportunities for GUI improvements, such as improved camera controls and better control for voxel selection (e.g., selecting between a range of values).

On a different topic, the user study revealed that participants' may over-trust the metrics when using the GUI. When we encouraged further reflection, participants reported doubts and treated the visualization more critically, after receiving explanations about how the metrics worked and their limitations. This highlights a potential issue of trust-calibration, which is an ongoing research topic in Visual Analytics [106]. To

address this, XRgonomics could provide explanations for each metric and a disclaimer of their limitations.

Supported ergonomics metrics

In our current implementation, we incorporate three existing metrics to compute the ergonomic cost of interaction: CE, RULA, and muscle activations. While these represent important research in ergonomics, we discovered limitations throughout development and the user study. For instance, Hincápie-Ramos et al. proposed CE as a metric to quantify fatigue of mid-air interactions [125]. However, the main scenario considered in their work is interaction with vertical displays. We assume this influenced the design of the metric, which is base on the cross-product of the gravity vector and the center of mass of the arm for static poses. This results in questionable results when reaching overhead (Figure 11.4, A), which was a common topic of discussion in our study.

Then, RULA investigates risk factors associated with work-related disorders [174]. Although such information is relevant to the design of ergonomic 3D UIs, it does not consider poses where the arm is at rest. Moreover, it relies on wide-angle ranges for scoring arm poses, resulting in similar values for several voxels (Figure 11.4, B).

In our biomechanical simulations, the optimization algorithm did not always converge. Without inspecting each individual case, it was impossible to discern whether this was due to poses being physiologically impossible, or caused by issues with the model or optimization step. Nevertheless, we argue that biomechanical models can represent important information which other metrics cannot, such as physical constraints, muscles, and tendon length.

We believe that XRgonomics can support the understanding of existing ergonomic metrics and the development of new ones by offering a simple way to inspect, compare, and debug them.

11.7 Conclusion

In this paper, we proposed a method to estimate the ergonomic cost of interaction at each reachable position in the user's environment. We make it available through the XRgonomics toolkit, which aims to support the design of ergonomic 3D UIs by making existing ergonomics metrics accessible to creators. The GUI allows creators to visualize the user's reachable interaction space and the ergonomic cost in each position. The API allows the creation of complex and dynamic constraints, which enable real-time adaptation of 3D UIs (e.g., repositioning the UI to avoid hitting physical obstacles). We illustrate functionalities XRgonomics can support through two scenarios. Finally, a walkthrough demonstration of the prototype shows the usefulness of our approach and highlights its potential to integrate additional factors beyond ergonomics.

Chapter 12

Paper D

This chapter presents the paper *AUIT – the Adaptive User Interfaces Toolkit for Designing XR Applications*, published in *The 35th Annual ACM Symposium on User Interface Software and Technology*: UIST 2022.

[19] João Marcelo Evangelista Belo, Mathias N. Lystbæk, Anna Maria Feit, Ken Pfeuffer, Peter Kán, Antti Oulasvirta, Kaj Grønbæk. AUIT - the adaptive user interfaces toolkit for designing XR applications. In *The 35th Annual ACM Symposium on User Interface Software and Technology*, UIST 2022, Bend, OR, USA, 29 October 2022 - 2 November 2022, pages 48:1–48:16. ACM, 2022. doi: 10.1145/3526113.3545651. https://doi.org/10.1145/3526113.3545651

AUIT – the Adaptive User Interfaces Toolkit for Designing XR Applications

João Marcelo Evangelista Belo, Mathias N. Lystbæk, Anna Maria Feit, Ken Pfeuffer, Peter Kán, Antti Oulasvirta, Kaj Grønbæk

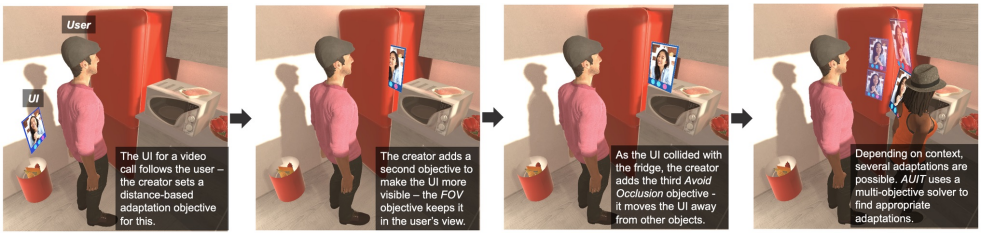


Figure 12.1: AUIT supports creators defining adaptation policies for UI elements that combine multiple objectives for XR interfaces. In this example, a video call UI is gradually extended with adaptation objectives to render it visible and within reach. Complexity rises with more potentially *competing* objectives and context changes. AUIT simplifies the design of adaptations by finding the best compromise via a multi-objective solver.

Abstract

Adaptive user interfaces can improve experiences in Extended Reality (XR) applications by adapting interface elements according to the user's context. Although extensive work explores different adaptation policies, XR creators often struggle with their implementation, which involves laborious manual scripting. The few available tools are underdeveloped for realistic XR settings where it is often necessary to consider conflicting aspects that affect an adaptation. We fill this gap by presenting AUIT, a toolkit that facilitates the design of optimization-based adaptation policies. AUIT allows creators to flexibly combine policies that address common objectives in XR applications, such as element reachability, visibility, and consistency. Instead of using rules or scripts, specifying adaptation policies via adaptation objectives simplifies the design process and enables creative exploration of adaptations. After creators decide which adaptation objectives to use, a multi-objective solver finds appropriate adaptations in real-time. A study showed that AUIT allowed creators of XR applications to quickly and easily create high-quality adaptations.

12.1 Introduction

Extended Reality (XR) is a medium that has gotten more widespread over the past years and will likely continue growing in the years to come [49]. Hardware improvements push the boundaries of what these applications can achieve, and sectors such as entertainment and manufacturing contribute towards this computing platform increasing popularity. However, easy-to-use XR applications are still challenging to develop. In contrast to traditional desktop or mobile applications, they are not confined to a 2D screen, but merge with the user's real-world environment to different extents. A key challenge of XR applications is how well they adapt to changes in the user's situation and surroundings [163].

The design of an adaptive UI for XR applications involves a high degree of complexity. As the user moves in the environment, considering context changes like his position or surrounding objects is crucial for creating an adaptation policy that provides a usable UI. Figure 12.1 illustrates one example scenario where a user has a floating video call interface close to him. The user might be unable to reach the virtual buttons, and positions outside his field of view or colliding with physical objects are inappropriate. Thus, the environment's geometry and the user's position constantly affect the *visibility* and *reachability* of the UI element - two fundamental usability factors of XR applications.

To address both requires considering multiple adaptation objectives [76]. However, these are typically not independent and might compete with each other. For example, moving the video call to prevent occlusion might position it outside the reach of a user. Such interactions grow as the number of UI elements and the complexity of the environment increase. They are hard for developers to foresee and resolve, increasing the difficulty of creating adaptive XR interfaces.

Over the last years, HCI researchers have proposed various methods to adapt interface elements in XR applications. They were concerned with the visibility and integration of virtual elements into the physical environment [47, 92, 163, 194] and their reachability or ergonomics [73, 126]. When considering criteria to adapt, these typically address independent aspects of the interface, such as position and content [163]. However, these methods tend to be custom tailored to specific applications and are difficult for creators to implement in practice. Existing tools for developing XR applications [180] only offer naive adaptation policies that are ineffective when multiple usability aspects come together.

To close this gap, we propose AUIT, the **A**daptive **U**ser **I**nterfaces **T**oolkit for supporting the design of XR applications. AUIT simplifies the adaptation of virtual elements to users' contexts and enables the combination of multiple adaptation objectives. It also offers a general framework that unifies prior research to make it available to practitioners. We achieve this goal by identifying five design concepts that adaptive user interfaces must implement:

Adaptation objectives Describe adaptation behaviors to address, such as visibility and reachability of UI elements.

Solvers Algorithms to compute adaptation candidates for UIs, resolving conflicts between objectives.

Context widgets Process raw sensor data into higher abstraction levels to inform adaptations.

Adaptation triggers The logic for when to invoke solvers and when to apply the adaptation proposals to the UI.

Property transitions How properties of virtual content transition to a new state when adaptations are triggered.

AUIT implements seven *adaptation objectives* that creators can flexibly assign to UI elements to address two fundamental usability issues of XR interfaces: visibility and reachability. AUIT automates conflict resolution by continuously optimizing the interface and determining the best trade-off between the chosen adaptation objectives using a multi-objective *solver*. Creators can choose between different *adaptation triggers* for initiating the adaptation, either at fixed time intervals or when the solver finds substantial UI improvements. They can also select *property transitions* to decide how the UI transitions to its new state. AUIT is implemented as a Unity extension that creators can easily import to develop adaptive UIs without drastic changes in their current workflow.

We evaluate AUIT's usefulness through a user study with eight experts who actively create XR applications as part of their jobs. We found that the design concepts in AUIT were easy to understand for participants, allowing for a clear separation of concerns in adaptive UIs. The process was fast, and participants designed adaptive user interfaces for two different scenarios in less than 25 minutes. They appreciated how easy it was to combine adaptation objectives and the quality of the results, while feeling efficient considering the time spent and the adaptations obtained. Participants also discussed the importance of adaptive UIs and pointed out that their current practice was limited by manual scripting, highlighting the need for tools to facilitate their development.

To summarize, this paper proposes AUIT, a toolkit based on a conceptual framework to support the creation of user interfaces that adapt to the user's context. It allows for 1) combining different adaptation objectives, 2) resolving conflicts between objectives using multi-objective optimization, and 3) customizing how and when XR content adapts. We demonstrate the utility of the toolkit through a study where experts successfully create high-quality adaptations for two applications using AUIT. We make the toolkit available through a Unity package that can be extended and customized by creators to fit their needs. AUIT makes existing research on adaptation methods for 3D interfaces available to creators, and offers a unifying framework for future work. Source code is available at https://github.com/joaobelo92/auit.

12.2 Related Work

Over the last decades, researchers have proposed different methods to adapt interfaces to improve usability. We start with a brief overview of adaptation and optimization techniques for 2D user interfaces. Then, we move on to research focusing on XR, starting with view management techniques, followed by adaptation techniques focusing on other usability goals. Finally, we give an overview of related frameworks and toolkits.

Adaptation and Optimization of 2D Interfaces

The increasing availability of mobile devices has motivated significant work on adaptive UIs. Researchers have proposed model-based approaches allowing developers to adapt applications across devices based on rules and logic (e.g., MARIA/TERESA [185, 205]) and methods that dynamically generate interfaces for multiple devices [192]. Gajos and Weld introduced SUPPLE, an approach that uses optimization to design UIs [88]. Similarly to SUPPLE, we use cost functions in our optimization procedure to represent objectives that guide adaptations in XR.

To support designers of 2D applications, researchers have explored genetic algorithms [228], other combinatorial optimization approaches [199], and data-driven optimization [89] based on user preferences to compute an optimal UI. The UI can also use optimization in real-time to dynamically adapt to users' preferences, context, or a device's capabilities (e.g. [44, 58, 77, 97, 204]). Such adaptations of user interfaces are relevant on mobile devices [35, 193] that typically have small screen sizes [82]. Recently, Todi et al. [257] presented a method for adaptive user interfaces based on reinforcement learning. There is extensive research on adaptive 2D UIs, and we refer to Miraz et al. [183] for a broader discussion of related work in this area.

View Management Techniques

View management techniques address how to maintain virtual objects in the user's view plane. These techniques focus on visibility aspects, such as avoiding occlusion and maintaining spatial relationships of virtual objects. Pioneering work focused on algorithms that use the upright rectangular extents of content in the view plane to avoid occlusion, adapting object properties such as their position, size, and transparency [16]. Grasset et al. focused on optimizing layouts of elements during run-time in the 2D view plane [100], while Tatzgern et al. explored this issue in 3D space [253]. Spatio-temporal coherence is another relevant factor to consider in XR experiences. Using spatial information from previous frames can help reduce visual discontinuities. Experiments suggest that users prefer limited update rates over continuous update rates for adaptations [170].

Other work investigated adaptive UIs to manage information density in AR and avoid information overload, which might affect task performance depending on the user's cognitive load. Therefore, researchers have developed adaptive level-of-detail

(LOD) methods for AR interfaces. Tatzgern et al. [254] proposed an adaptive information density display for AR using hierarchical clustering. Their approach automatically groups UI elements to reduce information overload and provides the user the control to unfold the level of detail. Several works use special sensors, such as eye-tracking technology, to adapt how and which content to present to the user [163, 169, 209].

View management techniques are closely related to UI adaptations. AUIT aims to make this line of work available to practitioners through a tool to adapt UIs that they can extend to support other sources of context (e.g., eye-gaze) and objectives (e.g., less-cluttered UI).

Adaptive User Interfaces in XR

UIs in XR pose additional challenges compared to traditional UIs because of the higher-dimensional design space, context changes, and broader range of interaction metaphors. Adaptive UIs are particularly important in XR scenarios using wearable [149], ubiquitous [124, 230], and mobile [127] computing platforms. Oliveira and Araujo [196] developed a context-aware AR system that adapts its interface based on changing contexts. Their system uses adaptation rules which select an appropriate UI pattern according to the current context. To improve the usability of XR applications, creators must consider factors such as real-world geometry [92, 194], cognitive load [163], or ergonomics [73]. Gal et al. [92] presented a method to automatically generate object layouts in AR applications, where the virtual elements in the AR application adapt to real-world geometry.

Ens et al. proposed a body-centric layout management technique that keeps layouts consistent across multiple environments while adapting to local geometric and visual features [70]. The work from Xiao et al. [277] explores various interaction techniques that use spatial awareness and optimization to adapt to different work surfaces. Later on, Lindlbauer et al. proposed an optimization-based approach to automatically control when and where mixed reality (MR) applications are shown and how much information they display, depending on the user's cognitive load [163]. Lu and Xu [168] studied different levels of automation and control of adaptive UI in AR. Their results suggest that users prefer and perform better when adaptations are semi-automated.

All these works show how different adaptation factors can improve usability in XR applications. AUIT provides a novel platform for creators to experiment with several adaptation goals and can be extended to support many more.

Frameworks and Toolkits

As XR technology is becoming widely available, researchers called for better support for developers across various stages of the design process of creating XR experiences [6]. As such, there has been a surge in research for tools that can ease the design and development of augmented [155, 187, 249] and virtual reality applications [102, 189]. We extend existing work with a toolkit to create adaptive UIs.

There is limited work on frameworks to facilitate the creation of adaptive UIs. Bonanni et al. [31] presented a framework for adaptive UI design focused on an AR kitchen scenario to support cooking, a scenario we build upon in our user study. Krings et al. implemented context-aware UI adaptations with a rule-based framework in which any change in context can trigger adaptation actions [144].

There are also other frameworks with some support for creating UI adaptations. For example, MRTK [180] has solvers [179] that use algorithms to calculate the position and orientation of UI elements. Existing solvers in MRTK focus on fundamental usability issues, such as visibility and reachability, as we do in the initial iteration of our toolkit. Although MRTK allows creators to chain multiple solvers with different adaptation objectives, it runs these sequentially without support for multi-objective optimization. Moreover, each solver in MRTK is tied to specific transitions (e.g., smooth movement over time, triggered every frame), limiting the design space of adaptations for XR.

Unity Mars [262] is another authoring tool that provides proxies to represent real-world objects, allowing creators to design UI adaptations based on rules relative to these proxies.

These frameworks focus on rule-based adaptations or algorithms to address specific adaptation objectives and lack the flexibility to combine multiple adaptation goals. We propose a framework that can integrate existing research and is easy to extend and generalize various scenarios encountered in the XR landscape. We use this framework as the foundation for AUIT, giving flexibility to creators by allowing them to combine different adaptation objectives, find adaptations using multi-objective optimization, and customize when and how UI elements transition from one state to another. AUIT separates adaptation concerns [61] into components, providing a modular approach where it is straightforward to customize different aspects of an adaptation.

12.3 AUIT: Design Concepts

To facilitate the design of adaptive UIs, we propose a clear separation of concerns [61] of the different design concepts present in an adaptation. Throughout this paper, we refer to the **creator** as the individual responsible for the application's development or design and the **user** as the end-user that will use the application. Consider the video call scenario presented in Figure 12.1. A creator develops an application that consists of a single UI element with a live video call and some controls to interact with it. The creator wants the video to be visible to the user and follow him as he moves around without interfering with his tasks and the environment. Such a scenario can quickly become complex, with various considerations about what kind of adaptive behavior is required, what contextual information it depends on, the conditions that cause an adaptation, and how to execute the UI adaptation. Such questions are not specific to this example, and are relevant to consider for many types of UI adaptations in XR. Therefore, we formulate these concerns as design goals for UI adaptations, followed by a framework to address them.

Design Goals

D1: Support a range of adaptation behaviors XR applications are not limited to the dimensions of a screen, in contrast to the GUIs present in traditional applications. In this setting, the design space tends to be broad and challenging to predict at design time. The design space also changes at runtime due to context changes such as the user's position or moving real-world objects. Consider the scenario in Figure 12.1 - the creator designs an adaptive UI that is 1) in reach, 2) in the user's field of view, and 3) not occluded by other objects. However, different scenarios have different requirements, and no specific combination of adaptation behaviors addresses the needs of the wide variety of applications possible in XR. Lindlbauer et al. [163] explored multiple adaptation objectives, but their approach focus on specific scenarios and fixed adaptation objectives, limiting generalization to other settings. Flexibility to combine different adaptation objectives across various UI elements allows creators to develop a wider variety of designs.

D2: Allow combining multiple adaptation objectives in one adaptation Multiple adaptation objectives can conflict with each other. For example, in the scenario from Figure 12.1, the adaptation objective to position objects in a specific zone of the user's field of view (FoV) can conflict with the objective to avoid collisions. Although related work has explored methods to find suitable solutions when considering multiple adaptation objectives through multi-objective optimization [92, 163], the support for combining adaptation objectives is still limited in existing tools. Therefore, the framework must be capable of finding a compromise between multiple objectives at runtime without constraining which objectives are possible to select.

D3: Support for context collection and interpretation The lack of standard methods to acquire and handle context is one of the barriers identified by Dey et al. [59] for using context in applications. In the scenario from Figure 12.1, context plays a crucial role in providing appropriate UI adaptations - it is necessary to know the user's position, where he is looking, and the environment geometry. In the past, applications would retrieve the user's context with custom implementations that processed sensor data into applications. Nowadays, this issue is not as prominent for XR applications. Game engines and software development kits already support some contextual information at a high abstraction level. Nonetheless, methods to interpret context at higher levels of abstraction that are generalizable across different applications are still a requirement to facilitate the creation of adaptive UIs in XR.

D4: Methods to customize when and why an adaptation occurs Depending on the application, creators might require different strategies for triggering UI adaptations. For example, in the video call scenario encountered in Figure 12.1, a naive approach that adapts the UI at a constant update rate might be sufficient, but the creator could be interested in a different strategy such as adapting the UI only when the quality of the layout goes below a certain threshold. Lindlbauer et al. [163] propose temporal

smoothing to improve transitions through adaptations, while Krings et al. [144] decide when to adapt the UI based on rules. The framework must allow creators to customize why and when UI adaptations occur to increase flexibility.

D5: Support for a variety of property transitions When considering the position of a UI element in an XR application, there are many possible ways it can adapt from one state to another. For example, in Figure 12.1, a creator can choose to update the position of the video call by moving it over time in 3D or instantly. These are just a few of the many possible transitions a creator could use for adapting the position of an object, one of the properties to adapt in XR. Consider now other properties of UI elements, like size, rotation, or modality. In such a vast design space, a framework to facilitate the creation of adaptive user interfaces must allow creators to choose from multiple property transitions.

Design Concepts for Adaptation Policies

We propose five design concepts for the development of adaptation policies to address the design goals we just presented. We provide an overview in Figure 12.2. Creating an adaptation policy that considers multiple adaptation objectives for an XR application should incorporate these to some extent:

Adaptation objectives (D1) are goals that guide the UI adaptation. For greater flexibility, an objective should only have one goal, allowing creators to combine objectives with different goals in one UI adaptation. For example, a creator might want a button to be reachable to the user while avoiding collisions with the real-world environment. By abstracting these goals into two separate objectives, creators can use them modularly for other adaptations throughout the application. Although an adaptation objective must have a single goal, it is worth noting that it can refer to a set of UI elements. For example, an objective to avoid clutter can have multiple UI elements as a target, but it is still a single goal.

Solvers (D2) are approaches that try to find the optimal solution to a stated optimization problem [199]. In our framework, solvers generate adaptation proposals to optimize the UI according to the adaptation objectives selected by creators.

Context widgets (D3) encapsulate how context is retrieved and make that data accessible to applications. Dey et al. [59] proposed such a component, and we refer to their work for a more in-depth overview. In short, context widgets process raw data and make it available at higher levels of abstraction, allowing creators to reuse and customize the usage of context data throughout the application. For XR applications, development tools such as MRTK [180] have some context widgets available. An example is the spatial awareness system in MRTK [178], a feature to provide real-world environmental awareness through a collection of meshes representing the environment geometry, which demonstrates how raw sensor data from the device is converted into a higher level of abstraction (in a mesh format), facilitating its integration in XR applications.

Adaptation Triggers (D4) are responsible for the logic to invoke solvers and if the solver proposals are applied. For example, creators can save computational resources by invoking the solver only when the layout quality goes below a certain threshold. Then, adaptations might be applied if the improvements from a new proposal are sufficient to justify the adaptation. The framework should allow creators to customize adaptation triggers and use or implement different strategies.

Property Transitions (D5) address how virtual content adapts to its new state when adaptations occur. Once an adaptation trigger executes an adaptation, property transitions define how the relevant properties of the UI element adapt from the previous to the new state. For example, there are different ways a UI element can move from position x to position y, such as moving over time from one position to another or fading out from the previous to the new position.

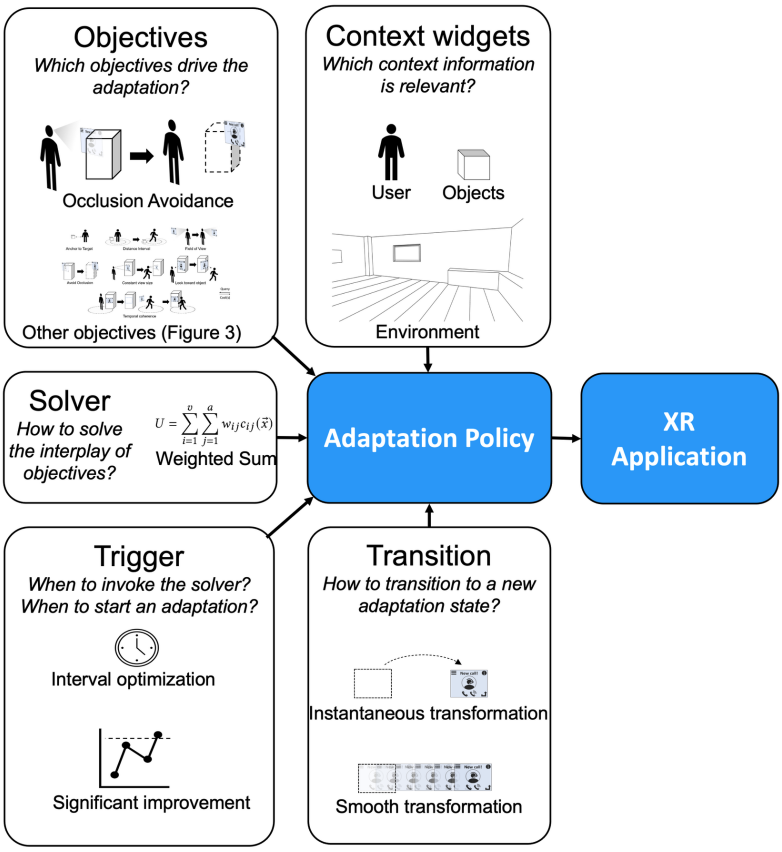


Figure 12.2: Overview of the design concepts for adaptive UIs. MAUI proposes 5 concepts to design adaptation policies for XR applications.

12.4 AUIT: Toolkit Implementation

We implement the framework introduced in the prior section through AUIT, a toolkit to facilitate the creation of adaptive user interfaces for XR applications. To optimize XR interfaces considering a combination of adaptation objectives, we formulate a cost minimization problem and solve it using multi-objective optimization. Adaptation objectives are formulated mathematically through a cost function representing how much the current layout fulfills that objective.

From an optimization perspective, we are dealing with a multi-objective optimization problem to optimize multiple objective functions simultaneously. In this case, objectives can contradict each other such that improving the solution towards one will worsen any of the others. Non-trivial problems have a set of optimal solutions that form the Pareto optimal frontier [172] instead of a single global optimal solution. To simplify picking a desirable solution in our toolkit, we opt for a weighted sum method [172], where creators articulate their preferences about the relative importance of different objectives using weights. We describe how we implement each design concept we proposed in Section 3 as a component of our toolkit:

Adaptation Objectives

Adaptation objectives are the criteria the UI adapts to and represent atomic adaptation behaviors that accomplish usability goals, such as visibility and reachability of the UI.

We define each adaptation objective through a cost function. To facilitate the customization of weights by creators, each adaptation objective we implement has a normalized cost function that outputs a cost from 0 to 1, reaching the highest value when the current layout infringes the adaptation objective beyond a customizable threshold. For example, consider an adaptation objective to keep virtual content from colliding with objects in the physical world. In this context, such an objective would return a value of 0 when applied to a hologram occupying a position that results in no collisions. This value would increase when the hologram starts colliding with environment geometry, reaching the value of 1 when the whole area of its virtual content is colliding. We implement heuristics for each adaptation objective so the solver can find improvements more efficiently. In addition, the solver can still search for new solutions following a random approach to avoid getting stuck in local minima.

We include seven adaptation objectives in AUIT that we illustrate in Figure 12.3, that identifies the high-Here, we briefly describe the adaptation objectives AUIT supports and refer the reader to the appendix for a more detailed description of cost functions and optimization heuristics.

Field of View Objective (Figure 12.3a)

Ensures the UI element is within a specific region of the user's field of view. Creators can select a pre-defined peripheral view interval or create a custom one by defining its inner and outer boundaries.

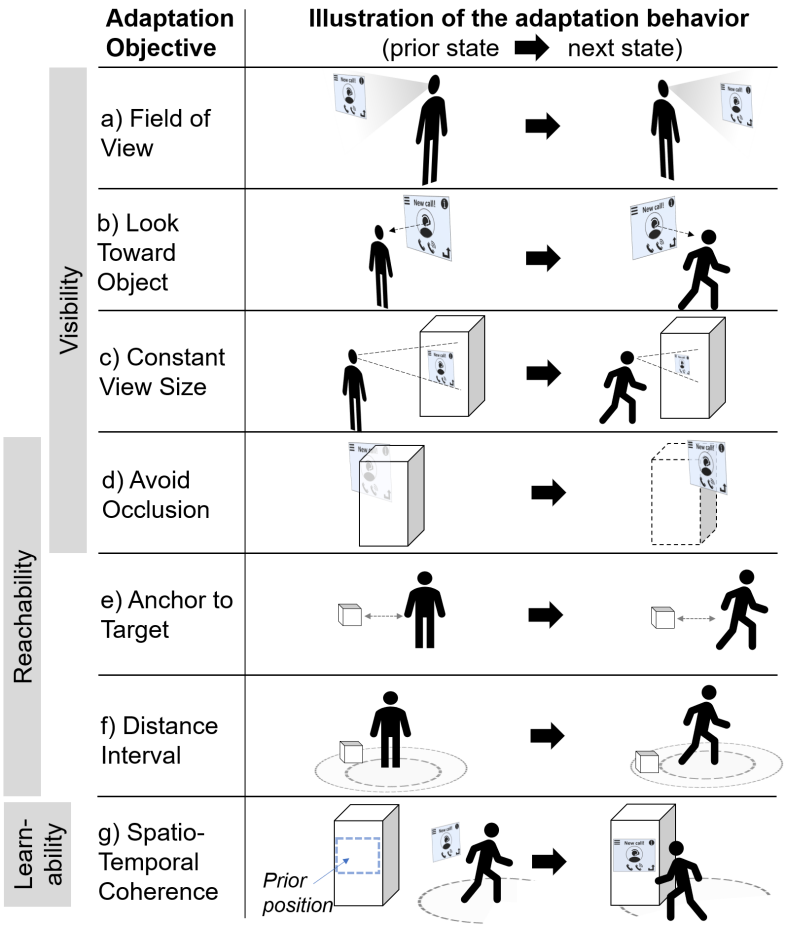


Figure 12.3: Adaptation objectives that are currently supported in AUIT. Creators can customize and combine them to design adaptation policies.

Optimization heuristic: attempt to move the UI element towards the FoV interval selected by the creator.

Look Towards Objective (Figure 12.3b)

Rotates the UI element towards a selected context source. It defaults to the user's position. This objective can contribute to visibility, as content such as text and images will become easier to see when rotated towards the user.

Optimization heuristic: rotate the UI towards the optimal rotation.

Constant View Size Objective (Figure 12.3c)

Scales the UI element depending on its distance from a target. This objective aims to maintain a constant view size to a context source, typically the user. We determine the optimal size of the UI using a configurable linear function dependent on the distance

from the UI to the context source. This is relevant to improve visibility of the UI without updating its position.

Optimization heuristic: scale the UI towards the optimal scale according to its distance from the context source.

Avoid Occlusion Objective (Figure 12.3d)

Avoids positions where the environment geometry or other virtual elements in the scene would occlude the object (typically to the user). Avoiding occlusions also prevents collisions with other virtual or physical objects. To check for occlusion, we dynamically add a configurable set of points in a grid composition to the UI element. Then, we draw rays [225] from the context source to each point and increase the cost for each ray hitting other content.

Optimization heuristic: move the UI in the direction of the surface normal hit by one of the obstructed rays.

Anchor to Target Objective (Figure 12.3e)

Aims to position a UI element at an offset from a selected context source. It selects the rotation and position of the user's head by default and the creator must provide the offset. Relevant when a UI element should follow the user or a virtual object.

Optimization heuristic: move the UI in the direction of the anchor point by a random distance.

Distance Interval Objective (Figure 12.3f)

Keeps UI elements positioned within a customizable distance interval in the shape of a vertical hollow cylinder from a selected context source - typically the user's position. The creator can set the inner and outer boundary of the cylinder area. This objective is relevant in cases where UI elements must be reachable to support hand input or at a distance that allows the user to see their content.

Optimization heuristic: move the UI towards the hollow cylinder by a random distance.

Spatio-Temporal Coherence Objective (Figure 12.3g)

Prioritizes adaptations where the UI element adapts to positions where it has been before, avoiding updates unless there are substantial improvements. Relevant to improve usability by leveraging the user's spatial memory [170].

Optimization heuristic: pick the closest previously visited position or other visited position at random.

Solvers

Solvers are the algorithms responsible for computing adaptation proposals. As mentioned earlier, we formulate a cost minimization problem and allow creators to articu-

late their preferences in terms of the relative importance of each adaptation objective using weights. An XR application typically contains v virtual elements. Each virtual object can have multiple adaptation objectives a - each having a correspondent weight w and cost function c - we can articulate the problem with a weighted sum:

$$U = \sum_{i=1}^{v} \sum_{j=1}^{a} w_{ij} c_{ij}(\vec{x}) \tag{12.1}$$

where $\vec{x}$ is the decision vector that consists of UI configuration parameters for all the UI elements to optimize and the minimum of U is Pareto optimal [172]. One of the goals of our framework is to generalize to a wide range of objectives, so we are particularly interested in methods capable of solving non-linear optimization problems. For that reason, the solver we implement uses simulated annealing [141, 264], which gradually converges to a near-optimal solution [4]. To find appropriate solutions more efficiently, our solver uses heuristics implemented for each objective and some randomness to avoid local optima. The solver uses early stopping to stop searching for proposals when it finds a suitable candidate and Unity coroutines [261] to distribute the computational load across multiple frames.

Context Widgets

Context widgets are the components responsible for processing raw sensor data into higher levels of abstraction. An advantage of building our toolkit for Unity is that several context widgets are available out of the box. It is trivial to retrieve fundamental context data for XR applications such as the position and gaze of the user from the Unity Camera component, which follows the user's head movement and rotation when using an HMD. Other relevant context information, such as the user's environment geometry or hand tracking, can be available depending on the development platform. For example, when developing for the Hololens, its SDK provides hand tracking and the environment geometry in Unity [178]. Because the adaptation objectives that are part of our first iteration of the toolkit consider fundamental usability goals, the context widgets already available in Unity are sufficient. For more flexibility, the creator can change the context source of adaptation objectives in the Unity inspector. For example, an adaptation objective that uses a position in 3D can be customized to use the user's pose or another virtual object in the scene.

Adaptation Triggers

Adaptation triggers are the component responsible for the logic to invoke the solver and apply the resulting adaptation proposal. AUIT supports two adaptation triggers to handle 1) when and how frequently to invoke the solver and 2) when to apply the adaptation proposed by the solver:

Interval optimization trigger

A basic adaptation strategy is to use a solver to generate UI proposals and apply them at a fixed rate, which creators can customize. This strategy involves a trade-off between update rate *vs.* usability. If the update rate is too high, it can result in too many adaptations that are a nuisance to the user. If the update rate is too low, the UI might violate adaptation objectives for too long until it adapts. Moreover, higher update rates result in a higher computational load as the solver runs more often.

Significant improvement trigger

This adaptation trigger only adapts the UI if it will result in an improvement. It invokes the solver once the quality of the UI (determined by the correspondent cost functions) is below a configurable threshold. Adaptations are applied if they improve on the previous interface by a ratio customizable by creators. This approach saves computational power because the solver only executes when the quality of the UI declines.

Property Transitions

Property transitions adapt the UI from one state to another. Once an adaptation trigger starts an adaptation for a UI element, its property transitions are applied. Depending on the property to adapt, AUIT invokes the correspondent property transition - different properties such as position and rotation require their respective property transitions. AUIT supports the following:

Instantaneous Movement

UI element moves instantaneous from the current position to the new one.

Smooth Movement

movement animation from the current to the new position over time using linear interpolation.

Smooth Rotation

rotation animation from the current to the new rotation over time using linear interpolation.

Smooth Scaling

scaling animation from the current to the new scale over time using linear interpolation.

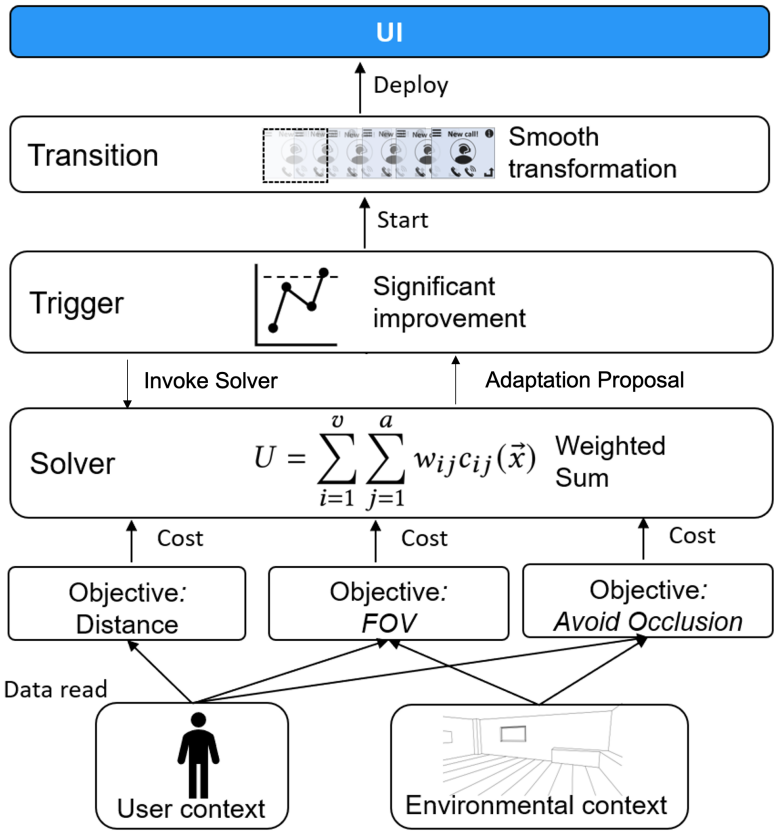


Figure 12.4: The processing flow of AUIT for the example adaptation in Figure 1. The cost from each objective is calculated using context. When invoked, the solver computes a new adaptation proposal considering all adaptation objectives – if this significantly improves the adaptation state, the UI uses a smooth movement transition to accomplish the UI adaptation. A solver can find proposals for multiple UI elements in the same optimization loop.

Architecture

Now we describe the software architecture to put all the components in AUIT together (see Figure 12.5).

Adaptation objectives access their context widgets directly to compute cost functions. The toolkit allows creators to change the context widget that is used by an adaptation objective through the Unity inspector, as long as it is compatible (e.g., anchor to target objective can use the user's position or the position of another virtual object as its context source). Adaptation objectives are then associated with UI elements. It is possible to associate the same adaptation objective to different UI elements, and each instance supports different configurations. Creators must assign property transitions for each property the UI elements adapt to (e.g., adaptations that involve the position and rotation of the UI will require corresponding property

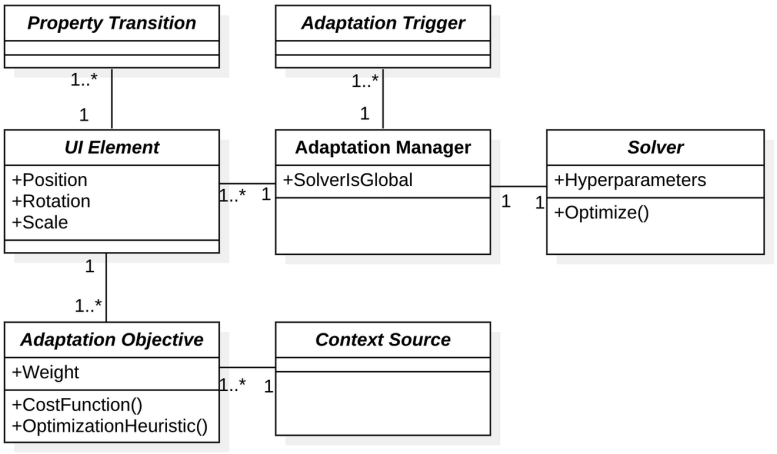


Figure 12.5: AUIT software architecture

transitions).

To manage adaptation triggers and solvers, we created an auxiliary class named Adaptation Manager. Adaptation managers gather all the objectives in the UI elements to optimize, invoke the solver using the logic in the adaptation trigger to compute adaptation proposals, and apply adaptations using the property transitions. Note that adaptation managers can optimize multiple UI elements in the same optimization loop by enabling a flag indicating the solver is global and indicating which UI elements it optimizes.

The adaptation objectives, triggers, and transitions derive from a corresponding abstract class. These abstract classes serve as a starting point for creators that want to extend the toolkit with new implementations of the components present in AUIT. Implementing additional components for AUIT, such as new adaptation objectives or property transitions, can be done by following three steps:

1. Create a new script component in Unity
2. Inherit from the abstract class that implements the component of interest
3. Implement the correspondent class abstract methods (e.g., for Adaptation Objectives, implement the *Cost Function* and *Heuristic* methods)

Technical Performance

Although not the focus of this work, it is important to assess how well it performs on common XR devices. Hence, we benchmark how the toolkit runs on a standalone device tailored for MR experiences. We test the Microsoft HoloLens 2 [177], a popular MR device nowadays. Note that the HoloLens 2 uses a mobile Qualcomm Snapdragon 850 with limited performance compared to laptop or desktop processors. Our benchmarking shows that AUIT can run consistently at 60fps in a scene with

(a) In-Editor Work (b) In-scene Menus

Figure 12.6: Users primarily use AUIT through the editor, but it is possible to tweak weights in an immersive setting for online configuration of adaptations.

three UI elements (each with four adaptation objectives) without noticeable frame rate drops.

Using and Extending AUIT in a Project

Creators can add AUIT to an existing Unity project by importing the Unity package we make available. To design adaptations, a creator associates toolkit components to UI elements in the scene - by dragging and dropping or typing their names in the Unity inspector. Creators can customize AUIT components through the Unity inspector without requiring coding. Once adaptation objectives and property transitions are added, alongside a solver and respective adaptation trigger, AUIT is ready to adapt the UI. Creators can immediately visualize the adaptations they create by entering play mode in Unity. It is possible to adjust the weights of adaptation objectives in real-time (see Fig. 12.6) through the Unity inspector or in an immersive setting, allowing adaptive UIs to be experienced immediately through an XR device or simulation. To optimize multiple UI elements in the same optimization loop, creators must add these to an adaptation manager component and enable the option to use a global solver.

12.5 Toolkit Evaluation

To evaluate the utility of AUIT, we conducted a study where creators of XR applications design adaptive UIs for two scenarios using AUIT. Assessing toolkit usage has been identified as a valuable step to evaluate what toolkits can do, whom they can support, and which tasks their users can perform [153]. The study aims to explore three research questions: (1) the *conceptual clarity of toolkit components*, (2) *toolkit usability*, and (3) the *quality of adaptations*.

(a) *Scenario 1: Video Call* (b) *Scenario 2: Interactive Recipe*

Figure 12.7: The UI panels from the two scenarios.

Study design

First, the experimenter introduces participants to the design concepts present in AUIT. Then, they use AUIT to create adaptive user interfaces for two XR scenarios. To facilitate the introduction to the wide variety of features and customization the toolkit supports, the experimenter assists the creator throughout the study by answering questions about specifics of parameters and other technical details they find unclear. For that reason, we consider our study a combination of a usability study and a walkthrough demonstration, according to the toolkit evaluation methods identified by Ledo et al. [153]. Initially we planned to use MRTK's solvers [179] as a baseline. However, in a pilot study, we noticed how sequential optimization was insufficient to successfully fulfill the goals of our scenarios in a satisfactory manner. Such a baseline would require coding and result in an unfair comparison. For these reasons, we only evaluated AUIT.

The two scenarios present in our study are a video call (Figure 12.7a) and an interactive recipe (Figure 12.7b). To increase control and streamline the study, participants start each design task from a scene containing the corresponding UI elements in Figure 12.7.

To allow quick prototyping, we use a simulation of an MR application in Unity based on VirtualHome [212], to model activities occurring in a kitchen. Participants could quickly see the adaptations they create by starting the simulation, which shows what an end-user would see from a first-person point of view when using the application. The user randomly performs different tasks in the kitchen, such as opening the fridge to gather ingredients, moving to the stove or counter to prepare food, or having a break doing something else. We asked participants to consider the kitchen in our simulation as the real-world in an MR application, while the UIs from the scenarios would be the holograms the user sees through the HMD.

Participants

We recruited 8 experts for the study who develop XR applications and have experience with Unity (1 female, age: $M = 32.5, SD = 3.38$). Although there is no meaningful cut-off point for which the sample size is enough [41], note that the goal of our study was to gather qualitative feedback - it is not our intention to draw statistically significant conclusions. Four participants shared an academic background and conducted research in XR, while the remaining four reported jobs in the industry where they actively develop XR applications. On average, participants had several years of experience developing XR applications ($M = 4.5, SD = 2.56$). On a scale from 1 (low) to 5 (high), participants reported familiarity with Adaptive User Interfaces ($M = 3.25, SD = 0.71$) and some regularity in developing them ($M = 2.5, SD = 1.51$).

Procedure

After an introduction to the study and signing the consent form, participants went through the following phases:

Adaptive user interfaces and existing tools

We start the study with a discussion about adaptive user interfaces for XR experiences. Participants reported their opinion on their importance, and those who develop adaptations revealed their current methods and practices to implement them.

Introduction to AUIT

After introducing the concept of adaptive UIs and the aim of the study, the experimenter explained the toolkit components to the participant. Then, six adaptation objectives are showcased through videos, followed by a discussion about breakdowns that might occur when using naive strategies to combine multiple adaptation objectives. Participants reported how they currently handle adaptations with conflicting objectives or how they would do it if faced with such a problem.

Creating adaptive user interfaces using AUIT

Next, we start the main task, where participants use AUIT to create adaptive UIs for two scenarios:

Video call A video call application in XR, with a UI element (Figure 12.7a) that contains the video feed and basic call controls that support hand input. The end-user is performing tasks in the kitchen while on a video call. The goal is to create an adaptation policy where the video call is visible, in the user's reach, and the virtual element is not occluded.

Interactive Recipe A cooking application to provide recipe instructions, interactive videos, and access to the list of ingredients. In this scenario, the application

contains two UI elements: 1) the instruction panel and 2) a panel with controls that support hand input (Figure 12.7b). The user interacts with the system through the control panel, where it is possible to change instructions, control the video, and spawn co-located timers in the kitchen. The goal is to create an adaptation policy where both UIs are visible and do not overlap with each other. In this scenario, only the control panel needs to be reachable.

The order of the tasks represents a learning curve, with a simple video call task first (one UI), followed by the interactive recipe scenario (two UIs). Participants are encouraged to meet other usability criteria they deem relevant. After each task we elicit user feedback by asking them to fill out a questionnaire and open-ended questions.

Feedback and discussion

Finally, we conduct an open-ended interview with the participants on aspects of AUIT, such as whether and how they would use it in their work and potential trade-offs of doing so.

Results

We started the study with a discussion about adaptive UIs for XR. All the participants considered adaptations to be crucial to provide a good user experience in XR applications: "It's really important because you want the user to [be able to] interact with the system when he moves around" (P4); "for XR [experiences] to not be frustrating [...] the interface [should] adapt to the environment" (P1). Creators also mentioned difficulties caused by the lack of resources and tools: "Resources to develop AUIs are limited" (P5); "I don't think [our company] has prioritized creating [AUIs], I think partly due to the [low] availability of tools to make it happen" (P8). Participants that actively develop adaptive user interfaces described their current practices are mostly based on custom rule-based implementations: "we develop our adaptations [...] we try to create our behaviors" (P4); "[we had to] customize MRTK behavior because often it doesn't behave the way we want [...] we faced [usability] issues and have solved them by implementing our own [rule-based] logic" (P6).

All participants successfully designed adaptations that met the requirements of both scenarios. In *Scenario 1: Video Call*, the participants spent between 6 and 18.3 minutes ($M = 11.92, SD = 4.57$), whereas in *Scenario 2: Interactive Recipe* they took between 4.82 and 23.87 minutes ($M = 10.87, SD = 6.26$).

Conceptual clarity of design concepts

After the presentation of the toolkit, participants showed understanding of the different design concepts throughout the study, suggesting it provides a clear separation of concerns for the problem: "the adaptation speed is still slow, [but] that is outside of the scope of the adaptation objective." (P4); "In VR I would use the player's transform as

Table 12.1: The top row shows some of the potential usability breakdowns that can occur when using naive adaptation approaches in scenario 1; The bottom row shows adaptations created by a study participant during scenario 1 where AUIT overcomes some of these issues.

the context [source] for the distance [interval] objective" (P5); "another [adaptation] objective that could be added [... is to] take the [user's FOV], make it into a grid, and then specify which cells you want [virtual objects] to be at" (P6). Moreover, a participant suggested the design concepts in AUIT make development easier: "[the toolkit components] are very much in line with the Unity philosophy of structuring behavior, which would make it easy to implement [XR adaptations]" (P8).

Toolkit usability

After each scenario, participants filled in a questionnaire with questions concerning the difficulty in using the toolkit to create adaptations for the video call (S1) and interactive recipe (**S2**) scenarios. On a scale from 1 (very easy) to 5 (very hard) participants reported that combining objectives was easy, but the difficulty slightly increases with more UI elements (**S1:** $M = 1.25, SD = 0.43$; **S2:** $M = 2.25, SD = 0.66$). Meanwhile, configuring components in the toolkit was reported as easy in both scenarios (**S1:** $M = 2.13, SD = 0.59$; **S2:** $M = 1.88, SD = 0.59$). In regards to finding appropriate weights for each objective, participants rated the difficulty of this task as medium for both scenarios (**S1:** $M = 2.5, SD = 0.86$; **S2:** $M = 2.75, SD = 0.18$). Note that participants used around 4 adaptation objectives in **S1** ($M = 4.13, SD = 0.59$) and 8 in **S2** ($M = 7.5, SD = 1.32$).

To develop adaptations using AUIT, creators followed a similar workflow throughout the study. They would add or remove components to the UI, such as adaptation objectives, tweak parameters, and run the simulation to visualize the resulting adapta-

Table 12.2: Adaptations by participants for scenario 2 (top row - P3); (bottom row - P8). Both solutions fulfill visibility requirements while considering world geometry using different combinations of adaptation objectives, highlighting the toolkit's flexibility. While P3 attempted to group both UI elements, P8 focused on maintaining these in the same zones of the user's FoV over time.

tions. Then, they would repeat this process until they were satisfied with the result, appreciating that they could see the adaptations created. A participant suggested the possibility to add commonly used combinations of toolkit components through a pre-configured and reusable asset to make the current workflow faster - "there are some objectives that often will go together [...] I like the [flexibility] to freely define what you want your behavior to be like [...] but it [requires some setup]" (P4).

Throughout the study, participants would occasionally ask for details about some of the parameters in components of the toolkit - "[Without documentation], it's hard to understand what some of these [parameters] do" (P5), highlighting the importance of resources to support the development and help developers get started with the toolkit. Nonetheless, they immediately understood many existing features due to familiarity with Unity and computer graphics concepts. As AUIT evolves, it is crucial to provide good tutorials and up-to-date documentation. In a few instances, experimenters helped participants debug specific behaviors in adaptations. Going forward, it is important to expand on the existing debugging capabilities of the toolkit, which are limited to console logging.

Quality of adaptations

When asked if they succeeded in creating the adaptations they had initially envisioned, participants reported on a scale from 1 (no, not at all) to 5 (yes, absolutely) that AUIT enabled them to do so in both scenarios (**S1:** $M = 4.38, SD = 0.48$; **S2:** $M =$

$3.75, SD = 0.82$). This feedback is interesting, considering that participants solved both tasks using different combinations of adaptation objectives, different settings for property transitions, and different parameters for the adaptation trigger. Even though all participants met basic usability requirements, such as visibility and reachability, some went a step further and considered factors such as consistency (about the position of the UI in relation to the user) and grouping of related virtual elements (in scenario 2) - see Tables 12.1 and 12.2.

On a scale from 1 (very poor) to 5 (very high), participants rated the adaptations they created to be of high quality (**S1:** $M = 4, SD = 0.5$; **S2:** $M = 4.13, SD = 0.60$). When considering the time spent and the results obtained, creators reported using a scale from 1 (very inefficient) to 5 (very efficient), that they felt highly efficient (**S1:** $M = 4.38, SD = 0.48$; **S2:** $M = 4.25, SD = 0.66$). Participants also mentioned how the toolkit could be valuable in their current work, pointing out how it could make development faster: "This is a challenge that we have every time we create a HoloLens application. Our workflow is pretty much the same [all the time]. We use MRTK and then code until it behaves like we want to, which can take a lot of time" (P6). A participant with less coding expertise appreciated that it was easy to get started with AUIT: "I think it's a huge benefit, particularly from my standpoint where I don't do a lot of coding. Instead of creating a lot of scripts myself, this is a toolkit that would allow me to prototype things very quickly, test things out, and demonstrate [them to my team]" (P7).

12.6 Discussion

We present AUIT, a toolkit to facilitate the design of adaptive UIs for XR applications. AUIT allows creators to combine adaptation objectives and find appropriate solutions using multi-objective optimization. Our approach lowers the barrier for creators to develop and experiment with adaptation policies while making the development process efficient through a clear separation of concerns. AUIT is a unifying toolkit that can bring together different adaptation methods, such as the variety of objectives we already support and other concepts explored in related work.

In our expert evaluation, participants pointed out that adaptive UIs are particularly important for providing good user experiences in XR applications. Moreover, they reported that existing tools lack appropriate support for designing adaptive UIs, requiring repetitive and time-consuming programming tasks. Their input suggests the toolkit components are understandable, and its separation of concerns can facilitate the development of adaptations.

Our study demonstrated how AUIT simplifies combining adaptation objectives and customizing adaptations, allowing creators to finish tasks involving complex behaviors in a short time while reporting feeling efficient doing so. All the participants successfully performed the proposed tasks, and their workflow showcases how AUIT can enable creative exploration of several aspects of adaptive UIs. Participants also reported that the toolkit allowed them to create the adaptations they had envisioned

and rated their creations to be of high quality, suggesting how AUIT can support the creation of adaptive UIs.

AUIT is a plugin for Unity, so it can be used alongside other tools such as MRTK, giving creators more options without requiring drastic changes to existing workflows. Furthermore, it allows for quick prototyping of adaptive UIs - most game engines allow creators to visualize changes throughout development (e.g., play mode in Unity) - AUIT allows similar prototyping, letting users update several aspects of an adaptation without requiring scripting. Ideally, approaches such as our toolkit will be better integrated into XR development tools, reducing the challenges for creators to get started with the development of adaptations. Moreover, adding new components to AUIT to consider other adaptation aspects opens new opportunities to explore the immense design space of adaptive UIs.

Limitations and Future Work

AUIT presents a first step towards a general toolkit for the design of adaptive UIs for XR applications. In this first version we have focused on two fundamental usability issues of XR applications: visibility and reachability of individual UI elements. However, the five design concepts described in this paper allow to extend AUIT in the future, for example to address other usability issues, enable joint optimization of multiple UI elements or allow creators to interactively add constraints to the optimization. Now we discuss these opportunities for future work in more detail and underline limitations of our research.

Adaptation objectives We implemented seven adaptation objectives related to fundamental usability issues in XR applications, but many others can contribute to better usability. Creators can build on existing research to implement new adaptation objectives as part of AUIT, such as ergonomics [73] or surface magnetism [278]. AUIT is currently limited to optimizing properties such as position, rotation, and scale, which are critical to ensure the visibility of UI elements. However, there are other properties of interest to adapt in XR settings, such as the level of detail or the decision to show or hide a UI element. Integrating those into AUIT is possible, but not straightforward, and requires more research to extend our solvers and support the consideration of other factors, such as the utility of a UI configuration and how it affects the user's cognitive load in our cost formulation. In this initial version of AUIT, we limit the optimization of multiple virtual elements to objectives that only consider properties of a single UI element at a time. An interesting direction is to extend AUIT to support global objectives that consider multiple UI elements, for example to avoid clutter in the user's FoV.

Solvers and alternative optimization methods In the current implementation of our toolkit, we use a weighted sum method to combine multiple objectives into one cost function. This cost function is a linear combination of the objective's cost functions that allows a designer to set the weights according to the importance of individual

adaptation objectives. This method works well if the particular costs of objectives have a similar scale, the reason why we designed normalized cost functions. However, an initial selection of weights does not guarantee the desired solution, requiring creators to adjust these during the design process. An additional disadvantage of using a weighted sum method is that it cannot find certain Pareto optimal solutions in the case of a non-convex objective space [172], a problem that other methods such as evolutionary algorithms could overcome [289].

Context widgets We use context widgets that are already part of most XR development tools. Adding methods to retrieve context in high abstraction levels can enable the design of new adaptation objectives. Some options with potential are enhanced scene understanding [96, 114], measuring cognitive load [68], or full-body tracking.

Adaptation triggers Although UI adaptations have the potential to improve usability [163], they can also be counterproductive [81, 257]. For example, adaptations can affect the user's attention or memory of the UI layout. One of the adaptation triggers implemented as part of our toolkit considers that adaptations come at a cost and only adapt when there are considerable improvements. However, other approaches could assess the utility of an adaptation to the user or add support for adding rules before or after running the optimization procedure.

Property transitions Property transitions can avoid detrimental effects when adaptations occur and are currently under-explored in AUIT. Some examples of property transitions that could enhance usability are the replication of UI elements temporarily - to avoid changes in the layout - or anchor UI elements to body parts, such as the hands - a technique commonly used in applications supporting hand input.

Evaluation Although our study suggests how AUIT can be relevant for creators, it has a small sample size of 8 experts. Replicating the study with more participants using a standardized test for usability, such as the System Usability Scale [36], could provide relevant quantitative results.

Other scenarios Although we cover video calls and interactive cooking scenarios in our work, AUIT can generalize to other settings. An example would be sketching in 3D, whether in VR for creating 3D models, or AR for drawing in 3D. Here, a UI could adapt the position of a color palette to be easily reachable by the non-dominant hand. Another example is manufacturing, where workers often need to assemble separate parts into larger machinery. As the hands are busy, an AR interface can provide clear instructions to the user. The UI could adapt to be close to relevant parts of the user's assembling steps while avoid occlusion. Furthermore, it is interesting to consider AR in everyday life as a personal computing device. Users might frequently transition between different locations, rooms, and activities, bringing in a new level of

complexity - tools such as AUIT can support the development of adaptive AR content to fit the user's context.

12.7 Conclusion

We presented AUIT, a toolkit to facilitate the design of adaptive UIs. AUIT implements five design concepts to support the development of such adaptations. The toolkit allows creators to combine multiple adaptation objectives as part of their development process and easily customize each aspect of an adaptive user interface. Our study with experts suggests that the design concepts we propose give creators a valuable separation of concerns for creating adaptations. Furthermore, AUIT allowed participants to efficiently design adaptive user interfaces that they rated to be of high quality. By making our toolkit widely available, we hope not only to lower the barrier for practitioners to get started creating adaptive UIs, but also to enable new workflows that allow for more creativity and require less repetitive and tedious tasks.

12.8 Appendix: Implementation details

In this section we describe how the cost functions and optimization heuristics for each adaptation objective are implemented. Note that the adaptation objectives are described in a different order here to optimize page space.

In this pseudo-code, variables declared before the functions are obtained dynamically or have default values that can be customized in the Unity inspector. We omit some software engineering technicalities - for more technical details please refer to the source code.

Adaptation Objectives

Algorithm 12.1: Distance Interval Objective

$gXZ \leftarrow$ goal distance from context source in XZ plane
$iXZ \leftarrow$ distance interval from gXZ (no cost penalty)
$iY \leftarrow$ height of the hollow cylinder (no cost penalty)
$t \leftarrow$ threshold for highest cost
$uiXZ \leftarrow$ UI XZ coordinates
$uiY \leftarrow$ UI Y coordinate
$csXZ \leftarrow$ context source XY coordinates
$csY \leftarrow$ context source Y coordinate
function Cost
    $difXZ \leftarrow csXZ - uiXZ$
    $dXZ \leftarrow$ magnitude($difXZ$) ▷ dist. from cs in XZ plane
    $dt \leftarrow$ abs($dXZ - gXZ$) ▷ UI XZ distance from goal
    $cXZ \leftarrow \max(0, dt - iXZ)$ ▷ no penalty if dist. $\leq iXZ$

```
14    dY ← abs(csY − uiY)                          ▷ UI Y distance from goal
15    cY ← max(0, dY − iY/2)                       ▷ no penalty if dist. ≤ iY/2
16    c = cXY + cY
17    c ← min (c/t, 1)                             ▷ normalize cost according to t
18    return c
19 end function
20 function HEURISTICS
21    s ← random value ∈ [0, 1]
22    if s ≤ 0.5 then
23       gXZ ← csXZ − uiXZ
24       dXZ ← magnitude(gXZ) − gXZ                ▷ distance from goal
25       guXZ ← normalize(gXZ)
26       nPXZ ← uiXZ + guXZ ∗ N(dXZ, 0.2)          ▷ move to goal
27       gY ← csY − uiY
28       nPY ← uiY + gY ∗ N(0.3, 0.2)              ▷ move to goal
29       return nP                                 ▷ new position likely closer to goal
30    else
31       rU ← random unit vector
32       return uiPos + rU ∗ N(0.3, 0.2)           ▷ move at random
33    end if
34 end function
```

Algorithm 12.2: Avoid Occlusion Objective

```
ks ← UI keypoint array dynamically generated (local coords.)
csPos ← context source position
uiTRS ← UI TRS matrix
uiPos ← UI position
function Cost
    c ← 0
    for all k in ks do
        wK = uiTRS · k                          ▷ get k in world coord.
        if raycast(csPos, wK) hits then
            c ← c + 1                           ▷ increase cost
        end if
    end for
    return c/length(ks)                         ▷ return normalized cost
end function
function Heuristics
    s ← random value ∈ [0, 1]
    if s ≤ 0.5 then                             ▷ pick heuristic at random
        for all k in ks do
            gP = uiTRS · k                      ▷ get k in world coord.
            if raycast(csPos, wK) hits then
                n ← hit normal(csPos, wK)       ▷ surface normal
                return hit pos. + n ∗ N(1, 0.5) ▷ move away
            end if
        end for
    else
        rU ← random unit vector
        return uiPos + rU ∗ N(0.3, 0.2)         ▷ move at random
    end if
end function
```

Algorithm 12.3: Anchor to Target Objective

```
o ← offset vector provided by creator
t ← threshold for highest cost
csTRS ← context source TRS matrix
uiPos ← UI position
function Cost
    l ← csTRS^−1 · uiPos                        ▷ get UI position in cs local coord.
    d ← distance(l, o)                          ▷ distance from offset to UI
    c ← min(d/t, 1)                             ▷ normalize cost according to t
    return c
end function
function Heuristics
```

```
    s ← random value ∈ [0,1]
    opt ← csTRS^{-1} · o                          ▷ compute optimal position
    if s ≤ 0.33 then                              ▷ pick heuristic at random
      return opt                                  ▷ return optimal position
    else
      ou ← normalize (opt − uiPos)
      uv ← random unit vector                     ▷ add randomness
      ou ← ou + uv ∗ random value ∈ [0,0.3]
      return uiPos + ou ∗ N(1,0.5)
    end if
end function
```

Algorithm 12.4: Spatial Coherence Objective

```
u ← adaptations allowed until a position is forgotten
vs ← data structure for visited voxel data
uiPos ← position of the UI
function OnAdapt(pos)                             ▷ called whenever ui adapts
    for all v in vs
      decrease v score by 1
      if v score is 0 then
        remove v from vs
      end if
    end for
    add/update voxel at pos to vs with score u
end function
function Cost
    if vs has voxel containing uiPos then
      return 0
    else
      return 1
    end if
end function
function Heuristics
    s ← random value ∈ [0,1]
    if s ≤ 0.5 then                               ▷ pick heuristic at random
      return voxel in vs closest to uiPos
    else
      return voxel in vs at random
    end if
end function
```

Algorithm 12.5: Constant View Size Objective

```
sF ← scaling factor                               ▷ scaling based on linear function
```

```
2  iS ← scaling difference tolerance (no cost penalty)
3  d ← base scale intended distance for UI
4  t ← threshold for highest cost
5  dS ← UI default scale
6  uiS ← UI current scale
7  uiP ← UI position
8  csP ← context source position
9  function Cost
10   d ← magnitude(uiP − csP)                ▷ get distance from ui to cs
11   i ← dS ∗ (d/iS ∗ sF)                    ▷ ideal scale based on d
12   c ← magnitude(uiS/i)                    ▷ compute scale difference
13   return min(c/t, 1)                      ▷ normalize according to t
14 end function
15 function Heuristics
16   s ← random value ∈ [0, 1]
17   if s ≤ 0.5 then
18     d ← magnitude(uiP − csP)              ▷ dist. from ui to cs
19     i ← dS ∗ (d/iS ∗ sF)                  ▷ optimal scale based on d
20     return i ∗ N(1, 0.3)
21   else
22     r ← N(0.3, 0.2)
23     return uiS ∗ r                        ▷ randomize scale
24   end if
25 end function
```

Algorithm 12.6: Field of View Objective

```
1  bo ← boundary origin for desired region in FoV
2  i ← angle interval from bo (no cost penalty)
3  t ← angle threshold for highest cost
4  csTRS ← context source TRS matrix
5  uiPos ← UI position
6  function Cost
7    l ← csTRS^−1 · uiPos                    ▷ get UI position in cs local coord.
8    a ← angle([0, 0, 1], l)                 ▷ angle between gaze and UI
9    a ← abs(a − bo)                         ▷ distance from desired origin
10   a ← max(0, a)                           ▷ no penalty if angle ≤ i
11   return min(a/t, 1)                      ▷ return normalized cost
12 end function
13 function Heuristics
14   s ← random value ∈ [0, 1]
15   lUI ← csTRS^−1 · uiPos                  ▷ get UI pos. in cs local coord.
16   if s ≤ 0.5 then                         ▷ pick heuristic at random
17     a ← angle ([0, 0, 1], l)              ▷ angle between gaze and UI
```

```
18      dir = 1                                          ▷ move towards cs forward
19      if a − i ≤ 0 then
20          dir = −1                                     ▷ move away from cs forward
21      end if
22      gW = csTRS · [0, 0, magnitude(lUI)]
23      g ← gW − uiPos
24      return uiPos + g ∗ dir ∗ N(0.3, 0.2)
25    else
26      rU ← random unit vector
27      return uiPos + rU ∗ N(0.3, 0.2)                  ▷ move at random
28    end if
29  end function
```

Algorithm 12.7: Look Towards Objective

```
1  csPos ← context source position
2  uiZ ← UI "look" vector                               ▷ UI forward or equivalent
3  uiPos ← UI position
4  t ← angle threshold for highest cost
5  function Cost
6    lA ← uiPos − csPos                                 ▷ Vector from cs to ui
7    a ← angle(lA, uiZ)                                 ▷ angle difference
8    return min(a/t, 1)                                 ▷ return normalized cost
9  end function
10 function Heuristics
11   lA ← uiPos − csPos                                 ▷ vector pointing to target
12   rot ← interpolate between uiZ and lA by N(1, 0.3)
13   return rot as a Quaternion
14 end function
```

Bibliography

[1] Michael Abrash. Creating the future: Augmented reality, the next human-machine interface. In *2021 IEEE International Electron Devices Meeting (IEDM)*, pages 1.2.1–1.2.11, 2021. doi: 10.1109/IEDM19574.2021.9720526. 3, 7, 57

[2] H. Aghajan and A. Cavallaro. *Multi-Camera Networks: Principles and Applications*. Elsevier Science, 2009. ISBN 9780080878003. URL https://books.google.dk/books?id=XA_6o2dhTGEC. 67, 71

[3] Antti Ajanki, Mark Billinghurst, Hannes Gamper, Toni Järvenpää, Melih Kandemir, Samuel Kaski, Markus Koskela, Mikko Kurimo, Jorma Laaksonen, Kai Puolamäki, et al. An augmented reality interface to contextual information. *Virtual reality*, 15(2):161–173, 2011. doi: https://doi.org/10.1007/s10055-010-0183-5. 17

[4] Khalil Amine. Multiobjective simulated annealing: Principles and algorithm variants. *Advances in Operations Research*, 2019:1–13, 05 2019. doi: 10.1155/2019/8134674. 32, 41, 144

[5] David Anderson, James L. Frankel, Joe Marks, Aseem Agarwala, Paul Beardsley, Jessica Hodgins, Darren Leigh, Kathy Ryall, Eddie Sullivan, and Jonathan S. Yedidia. Tangible interaction + graphical interpretation: A new approach to 3d modeling. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, page 393–402, USA, 2000. ACM Press/Addison-Wesley Publishing Co. ISBN 1581132085. doi: 10.1145/344779.344960. URL https://doi.org/10.1145/344779.344960. 91

[6] Narges Ashtari, Andrea Bunt, Joanna McGrenere, Michael Nebeling, and Parmit K. Chilana. Creating augmented and virtual reality applications: Current practices, challenges, and opportunities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367080. doi: 10.1145/3313831.3376722. URL https://doi.org/10.1145/3313831.3376722. 18, 21, 48, 49, 54, 111, 112, 114, 125, 128, 136

[7] Mahdi Azmandian, Mark Hancock, Hrvoje Benko, Eyal Ofek, and Andrew D. Wilson. Haptic retargeting: Dynamic repurposing of passive haptics for enhanced virtual reality experiences. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 1968–1979, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450333627. doi: 10.1145/2858036.2858226. URL https://doi.org/10.1145/2858036.2858226. 17

[8] R. Azuma and C. Furmanski. Evaluating label placement for augmented reality view management. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 66–75, 2003. doi: 10.1109/ISMAR.2003.1240689. 16

[9] Myroslav Bachynskyi, Antti Oulasvirta, Gregorio Palmas, and Tino Weinkauf. Is motion capture-based biomechanical simulation valid for hci studies? study and implications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 3215–3224, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450324731. doi: 10.1145/2556288.2557027. URL https://doi.org/10.1145/2556288.2557027. 13, 113, 118

[10] Myroslav Bachynskyi, Gregorio Palmas, Antti Oulasvirta, and Tino Weinkauf. Informing the design of novel input methods with muscle coactivation clustering. *ACM Trans. Comput.-Hum. Interact.*, 21(6), January 2015. ISSN 1073-0516. doi: 10.1145/2687921. URL https://doi.org/10.1145/2687921. 13, 111, 113, 118

[11] Gilles Bailly, Antti Oulasvirta, Timo Kötzing, and Sabrina Hoppe. Menuoptimizer: Interactive optimization of menu systems. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, page 331–342, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450322683. doi: 10.1145/2501988.2502024. URL https://doi.org/10.1145/2501988.2502024. 15, 18, 113

[12] K. M. Baird and W. Barfield. Evaluating the effectiveness of augmented reality displays for a manual assembly task. *Virtual Reality*, 4(4):250–259, Dec 1999. ISSN 1434-9957. doi: 10.1007/BF01421808. URL https://doi.org/10.1007/BF01421808. 68

[13] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of ad Hoc and ubiquitous Computing*, 2(4):263–277, 2007. 11

[14] Patrick Baudisch, Nathaniel Good, and Paul Stewart. Focus plus context screens: Combining display technology with visualization techniques. In

Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology, UIST '01, pages 31–40, New York, NY, USA, 2001. ACM. ISBN 1-58113-438-X. doi: 10.1145/502348.502354. URL http://doi.acm.org/10.1145/502348.502354. 66, 67

[15] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2):322–331, May 1990. ISSN 0163-5808. doi: 10.1145/93605.98741. URL https://doi.org/10.1145/93605.98741. 121

[16] Blaine Bell, Steven Feiner, and Tobias Höllerer. View management for virtual and augmented reality. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, page 101–110, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 158113438X. doi: 10.1145/502348.502363. URL https://doi.org/10.1145/502348.502363. 16, 135

[17] João Belo, Andreas Fender, Tiare Feuchtner, and Kaj Grønbæk. Digital assistance for quality assurance: Augmenting workspaces using deep learning for tracking near-symmetrical objects. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces*, ISS '19, page 275–287, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368919. doi: 10.1145/3343055.3359699. URL https://doi.org/10.1145/3343055.3359699. vii, 63, 89, 92, 95, 100, 101

[18] João Belo, Tiare Feuchtner, Chiwoong Hwang, Rasmus Lunding, Mathias Lystbæk, Ken Pfeuffer, and Troels Rasmussen. Challenges of xr transitional interfaces in industry 4.0. In *ISS'21: Interactive Surfaces and Spaces*, 2021. viii

[19] João Marcelo Evangelista Belo, Mathias N. Lystbæk, Anna Maria Feit, Ken Pfeuffer, Peter Kán, Antti Oulasvirta, and Kaj Grønbæk. AUIT - the adaptive user interfaces toolkit for designing XR applications. In Maneesh Agrawala, Jacob O. Wobbrock, Eytan Adar, and Vidya Setlur, editors, *The 35th Annual ACM Symposium on User Interface Software and Technology, UIST 2022, Bend, OR, USA, 29 October 2022 - 2 November 2022*, pages 48:1–48:16. ACM, 2022. doi: 10.1145/3526113.3545651. URL https://doi.org/10.1145/3526113.3545651. vii, 131

[20] João Marcelo Evangelista Belo, Felix Izarra, Xuhai Xu, Tanya R. Jonker, and Kashyap Todi. Designing and optimizing adaptive shortcuts for extended reality. 2023. viii

[21] João Marcelo Evangelista Belo, Jon Wissing, Tiare Feuchtner, and Kaj Grønbæk. Cadtrack: Instructions and support for orientation disambiguation of near-symmetrical objects. 2023. vii

[22] Hrvoje Benko. The future of mixed reality interactions. In *Companion Proceedings of the 2020 Conference on Interactive Surfaces and Spaces*, pages 1–1, 2020. 7, 19, 20

[23] Hrvoje Benko, Christian Holz, Mike Sinclair, and Eyal Ofek. Normaltouch and texturetouch: High-fidelity 3d haptic shape rendering on handheld virtual reality controllers. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, page 717–728, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341899. doi: 10.1145/2984511.2984526. URL https://doi.org/10.1145/2984511.2984526. 16

[24] Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, and Daniele Riboni. A survey of context modelling and reasoning techniques. *Pervasive and mobile computing*, 6(2):161–180, 2010. 11

[25] Mark Billinghurst. Augmented reality in education. *New horizons for learning*, 12(5):1–5, 2002. 6

[26] Mark Billinghurst, Hirokazu Kato, and Ivan Poupyrev. Collaboration with tangible augmented reality interfaces. In *HCI international*, volume 1, pages 5–10, 2001. 17

[27] Mark Billinghurst, Hirokazu Kato, and Ivan Poupyrev. Tangible augmented reality. *Acm siggraph asia*, 7(2):1–10, 2008. 17

[28] Susanne Bødker and Kaj Grønbæk. Design in action: from prototyping by demonstration to cooperative prototyping. In *Design at work: cooperative design of computer systems*, pages 197–218. 1992. 54

[29] Richard A. Bolt. Gaze-orchestrated dynamic windows. In *Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '81, page 109–119, New York, NY, USA, 1981. Association for Computing Machinery. ISBN 0897910451. doi: 10.1145/800224.806796. URL https://doi.org/10.1145/800224.806796. 17

[30] Richard A. Bolt. Eyes at the interface. In *Proceedings of the 1982 Conference on Human Factors in Computing Systems*, CHI '82, page 360–362, New York, NY, USA, 1982. Association for Computing Machinery. ISBN 9781450373890. doi: 10.1145/800049.801811. URL https://doi.org/10.1145/800049.801811. 17

[31] Leonardo Bonanni, Chia-Hsun Lee, and Ted Selker. A framework for designing intelligent task-oriented augmented reality user interfaces. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, IUI '05, page 317–319, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1581138946. doi: 10.1145/1040830.1040913. URL https://doi.org/10.1145/1040830.1040913. 137

[32] Rein Van Den Boomgaard and Joost Van De Weijer. Robust estimation of orientation for texture analysis, 2002. 67

[33] Gunnar Borg. Psychophysical scaling with applications in physical work and the perception of exertion. *Scandinavian Journal of Work, Environment & Health*, 16:55–58, 1990. ISSN 03553140, 1795990X. URL http://www.jstor.org/stable/40965845. 13, 112

[34] Sebastian Boring, Marko Jurmu, and Andreas Butz. Scroll, tilt or move it: Using mobile phones to continuously control pointers on large public displays. In *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7*, OZCHI '09, page 161–168, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605588544. doi: 10.1145/1738826.1738853. URL https://doi.org/10.1145/1738826.1738853. 13, 112

[35] Amani Braham, F. Buendia, Maha Khemaja, and Faiez Gargouri. User interface design patterns and ontology models for adaptive mobile applications. *Personal and Ubiquitous Computing*, pages 1–17, 01 2021. doi: 10.1007/s00779-020-01481-5. 135

[36] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996. 156

[37] P.J. Brown, J.D. Bovey, and Xian Chen. Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications*, 4(5):58–64, 1997. doi: 10.1109/98.626984. 11

[38] Rainer E Burkard and Josef Offermann. Entwurf von schreibmaschinentastaturen mittels quadratischer zuordnungsprobleme. *Zeitschrift für Operations Research*, 21(4):B121–B132, 1977. 15

[39] Ernesto A Bustamante and Randall D Spain. Measurement invariance of the nasa tlx. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 52, pages 1522–1526. SAGE Publications Sage CA: Los Angeles, CA, 2008. 99

[40] Susanne Bødker, Kaj Grønbæk, and Morten Kyng. Cooperative design: techniques and experiences from the scandinavian scene. In *Readings in human–computer interaction*, pages 215–224. Elsevier, 1995. 54, 55

[41] Kelly Caine. Local standards for sample size at chi. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 981–992, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450333627. doi: 10.1145/2858036.2858498. URL https://doi.org/10.1145/2858036.2858498. 150

[42] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 12, 90

[43] James Carifio and Rocco J Perla. Ten common misunderstandings, misconceptions, persistent myths and urban legends about likert scales and likert response formats and their antidotes. *Journal of social sciences*, 3(3):106–116, 2007. 13, 112

[44] Eduardo Castillejo, Aitor Almeida, and Diego López de Ipiña. Ontology-based model for supporting dynamic and adaptive user interfaces. *International Journal of Human–Computer Interaction*, 30(10):771–786, 2014. doi: 10.1080/10447318.2014.927287. URL https://doi.org/10.1080/10447318.2014.927287. 135

[45] T. P. Caudell and D. W. Mizell. 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 ii, pages 659–669 vol.2, Kauai, HI, USA, USA, Jan 1992. IEEE. doi: 10.1109/HICSS.1992.183317. 3, 6, 68

[46] Guanling CHEN. A survey of context-aware mobile computing research. *Technical Report TR2000-381*, 2000. 11

[47] Yifei Cheng, Yukang Yan, Xin Yi, Yuanchun Shi, and David Lindlbauer. Semanticadapt: Optimization-based adaptation of mixed reality layouts leveraging virtual-physical semantic connections. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, UIST '21, page 282–297, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450386357. doi: 10.1145/3472749.3474750. URL https://doi.org/10.1145/3472749.3474750. 16, 35, 36, 54, 133

[48] Mario Cifrek, Vladimir Medved, Stanko Tonković, and Saša Ostojić. Surface emg based muscle fatigue evaluation in biomechanics. *Clinical Biomechanics*, 24(4):327 – 340, 2009. ISSN 0268-0033. doi: https://doi.org/10.1016/j.clinbiomech.2009.01.010. URL http://www.sciencedirect.com/science/article/pii/S0268003309000254. 13, 113

[49] Perkins Coie. 2020 augmented and virtual reality survey report. https://www.perkinscoie.com/en/ar-vr-survey-results/2020-augmented-and-virtual-reality-survey-results.html, 2020. Accessed: 2020-06-16. 111, 133

[50] Lynn A Cooper. Mental rotation of random two-dimensional shapes. *Cognitive Psychology*, 7(1):20 – 43, 1975. ISSN 0010-0285. doi: https://doi.org/10.1016/

0010-0285(75)90003-1. URL http://www.sciencedirect.com/science/article/pii/0010028575900031. 66

[51] John W Creswell and J David Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017. 56

[52] Joachim Deisinger, Ralf Breining, and Andreas Rößler. ERGONAUT: A Tool for Ergonomic Analyses in Virtual Environments. In Jurriaan Mulder and Robert van Liere, editors, *Virtual Environments 2000*, pages 167–176, Vienna, 2000. Springer Vienna. ISBN 978-3-7091-6785-4. 115

[53] Scott L Delp, Frank C Anderson, Allison S Arnold, Peter Loan, Ayman Habib, Chand T John, Eran Guendelman, and Darryl G Thelen. Opensim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, 54(11):1940–1950, 2007. 118

[54] J. Deng, W. Dong, R. Socher, L. Li, and and. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. doi: 10.1109/CVPR.2009.5206848. 79, 80, 82

[55] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848. 12, 90, 94

[56] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019. 93

[57] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Dmitry Olefir, Tomas Hodan, Youssef Zidan, Mohamad Elbadrawy, Markus Knauer, Harinandan Katam, and Ahsan Lodhi. Blenderproc: Reducing the reality gap with photorealistic rendering. In *Robotics: Science and Systems (RSS)*, July 2020. URL https://elib.dlr.de/139317/. 91

[58] Tilman Deuschel and Ted Scully. On the importance of spatial perception for the design of adaptive user interfaces. In *2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 70–79, 2016. doi: 10.1109/SASO.2016.13. 135

[59] Anind Dey, Gregory Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16, 04 2001. doi: 10.1207/S15327051HCI16234_02. 12, 21, 25, 38, 138, 139

[60] Anind K Dey and Gregory D Abowd. Towards a better understanding of context and context-awareness. Technical report, Georgia Institute of Technology, 1999. 11, 19

[61] Edsger W Dijkstra. On the role of scientific thought. In *Selected writings on computing: a personal perspective*, pages 60–66. Springer, 1982. 137

[62] S. DiVerdi, T. Hollerer, and R. Schreyer. Level of detail interfaces. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 300–301, 2004. doi: 10.1109/ISMAR.2004.38. 15

[63] Stephen DiVerdi, Tobias Hollerer, and Richard Schreyer. Level of detail interfaces. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 300–301, New York, NY, USA, 2004. IEEE. 114

[64] Alan Dix. Statistics for hci: Making sense of quantitative data. *Synthesis Lectures on Human-Centered Informatics*, 13(2):1–181, 2020. 56

[65] Alan Dix, Tom Rodden, Nigel Davies, Jonathan Trevor, Adrian Friday, and Kevin Palfreyman. Exploiting space and location as a design framework for interactive mobile systems. *ACM Trans. Comput.-Hum. Interact.*, 7(3): 285–321, sep 2000. ISSN 1073-0516. doi: 10.1145/355324.355325. URL https://doi.org/10.1145/355324.355325. 11

[66] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages I–647–I–655. JMLR.org, 2014. URL http://dl.acm.org/citation.cfm?id=3044805.3044879. 67, 79

[67] Paul Dourish. What we talk about when we talk about context. *Personal and ubiquitous computing*, 8(1):19–30, 2004. 11

[68] Andrew T. Duchowski, Krzysztof Krejtz, Izabela Krejtz, Cezary Biele, Anna Niedzielska, Peter Kiefer, Martin Raubal, and Ioannis Giannopoulos. The index of pupillary activity: Measuring cognitive load vis-à-vis task difficulty with pupil oscillation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–13, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356206. doi: 10.1145/3173574.3173856. URL https://doi.org/10.1145/3173574.3173856. 156

[69] Abhishek Dutta and Andrew Zisserman. The via annotation software for images, audio and video, 2019. 78

[70] Barrett Ens, Eyal Ofek, Neil Bruce, and Pourang Irani. Spatial constancy of surface-embedded layouts across multiple environments. In *Proceedings of the 3rd ACM Symposium on Spatial User Interaction*, SUI '15, page 65–68, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450337038. doi: 10.1145/2788940.2788954. URL https://doi.org/10.1145/2788940.2788954. 16, 136

[71] Augusto Esteves, David Verweij, Liza Suraiya, Rasel Islam, Youryang Lee, and Ian Oakley. Smoothmoves: Smooth pursuits head movements for augmented reality. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, page 167–178, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349819. doi: 10.1145/3126594.3126616. URL https://doi.org/10.1145/3126594.3126616. 17

[72] Augusto Esteves, Yonghwan Shin, and Ian Oakley. Comparing selection mechanisms for gaze input techniques in head-mounted displays. *International Journal of Human-Computer Studies*, 139:102414, 2020. 17

[73] João Marcelo Evangelista Belo, Anna Maria Feit, Tiare Feuchtner, and Kaj Grønbæk. Xrgonomics: Facilitating the creation of ergonomic 3d interfaces. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380966. URL https://doi.org/10.1145/3411764.3445349. vii, 109, 133, 136, 155

[74] Steven Feiner, Blair Macintyre, and Dorée Seligmann. Knowledge-based augmented reality. *Commun. ACM*, 36(7):53–62, jul 1993. ISSN 0001-0782. doi: 10.1145/159544.159587. URL https://doi.org/10.1145/159544.159587. 15

[75] Steven Feiner, Blair MacIntyre, Tobias Höllerer, and Anthony Webster. A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. *Personal Technologies*, 1(4):208–217, 1997. 6

[76] Anna Maria Feit, Myroslav Bachynskyi, and Srinath Sridhar. Towards multi-objective optimization for ui design. Presented at CHI 2015 Workshop on Principles, Techniques and Perspectives on Optimization and HCI, April 2015. URL http://annafeit.de/resources/papers/Multiobjective_Optimization2015.pdf. 133

[77] Anna Maria Feit, Lukas Vordemann, Seonwook Park, Caterina Berube, and Otmar Hilliges. Detecting relevance during decision-making from eye movements for ui adaptation. In *ACM Symposium on Eye Tracking Research and Applications*, ETRA '20 Full Papers, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371339. doi: 10.1145/3379155.3391321. URL https://doi.org/10.1145/3379155.3391321. 135

[78] Andreas Fender and Jörg Müller. Velt: A framework for multi rgb-d camera systems. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*, ISS '18, pages 73–83, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5694-7. doi: 10.1145/3279778.3279794. URL http://doi.acm.org/10.1145/3279778.3279794. 76

[79] Andreas Fender, Philipp Herholz, Marc Alexa, and Jörg Müller. Optispace: Automated placement of interactive 3d projection mapping content. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–11, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356206. doi: 10.1145/3173574.3173843. URL https://doi.org/10.1145/3173574.3173843. 114

[80] Tiare Feuchtner and Jörg Müller. Ownershift: Facilitating overhead interaction in virtual reality with an ownership-preserving hand space shift. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, page 31–43, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359481. doi: 10.1145/3242587.3242594. URL https://doi.org/10.1145/3242587.3242594. 17, 112

[81] Leah Findlater and Krzysztof Z. Gajos. Design space and evaluation challenges of adaptive graphical user interfaces. *AI Magazine*, 30(4):68–73, 2009. ISSN 07384602. doi: 10.1609/aimag.v30i4.2268. 156

[82] Leah Findlater and Joanna McGrenere. Impact of screen size on performance, awareness, and user satisfaction with adaptive graphical user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 1247–1256, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580111. doi: 10.1145/1357054.1357249. URL https://doi.org/10.1145/1357054.1357249. 135

[83] Martin Fisch and Ronald Clark. Orientation keypoints for 6d human pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10145–10158, 2022. doi: 10.1109/TPAMI.2021.3136136. 22

[84] George W. Fitzmaurice, Hiroshi Ishii, and William A. S. Buxton. Bricks: Laying the foundations for graspable user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, page 442–449, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0201847051. doi: 10.1145/223904.223964. URL https://doi.org/10.1145/223904.223964. 17

[85] Gregory Francis. Designing multifunction displays: An optimization approach. *International Journal of Cognitive Ergonomics*, 4(2):107–124, 2000. 15

[86] Henry Fuchs, Mark A Livingston, Ramesh Raskar, Kurtis Keller, Jessica R Crawford, Paul Rademacher, Samuel H Drake, Anthony A Meyer, et al. Augmented reality visualization for laparoscopic surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 934–943. Springer, 1998. 6

[87] Markus Funk, Thomas Kosch, and Albrecht Schmidt. Interactive worker assistance: Comparing the effects of in-situ projection, head-mounted displays, tablet, and paper instructions. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, page 934–939, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450344616. doi: 10.1145/2971648.2971706. URL https://doi.org/10.1145/2971648.2971706. 91

[88] Krzysztof Gajos and Daniel S. Weld. Supple: Automatically generating user interfaces. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, IUI '04, page 93–100, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138156. doi: 10.1145/964442.964461. URL https://doi.org/10.1145/964442.964461. 15, 135

[89] Krzysztof Gajos and Daniel S. Weld. Preference elicitation for interface optimization. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, UIST '05, page 173–182, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595932712. doi: 10.1145/1095034.1095063. URL https://doi.org/10.1145/1095034.1095063. 135

[90] Krzysztof Z. Gajos, Jacob O. Wobbrock, and Daniel S. Weld. Improving the performance of motor-impaired users with automatically-generated, ability-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 1257–1266, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580111. doi: 10.1145/1357054.1357250. URL https://doi.org/10.1145/1357054.1357250. 15

[91] Krzysztof Z. Gajos, Daniel S. Weld, and Jacob O. Wobbrock. Automatically generating personalized user interfaces with supple. *Artificial Intelligence*, 174(12):910–950, 2010. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2010.05.005. URL https://www.sciencedirect.com/science/article/pii/S0004370210000822. 15, 113

[92] Ran Gal, Lior Shapira, Eyal Ofek, and Pushmeet Kohli. Flare: Fast layout for augmented reality applications. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 207–212, New York, NY, USA, 2014. IEEE. 114, 133, 136, 138

[93] Liuhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. 3d hand shape and pose estimation from a single rgb image.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 17, 22

[94] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. 12, 90

[95] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 12, 90

[96] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 156

[97] Camille Gobert, Kashyap Todi, Gilles Bailly, and Antti Oulasvirta. Sam: A modular framework for self-adapting web menus. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, page 481–484, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362726. doi: 10.1145/3301275.3302314. URL https://doi.org/10.1145/3301275.3302314. 135

[98] Claire C Gordon, Thomas Churchill, Charles E Clauser, Bruce Bradtmiller, and John T McConville. Anthropometric survey of us army personnel: methods and summary statistics 1988. Technical report, Anthropology Research Project Inc Yellow Springs OH, 1989. 115, 120

[99] Mikhail V. Goubko and Alexander I. Danilenko. An automated routine for menu structure optimization. In *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '10, page 67–76, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300834. doi: 10.1145/1822018.1822030. URL https://doi.org/10.1145/1822018.1822030. 15

[100] Raphaël Grasset, Tobias Langlotz, Denis Kalkofen, Markus Tatzgern, and Dieter Schmalstieg. Image-driven view management for augmented reality browsers. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 177–186, 2012. doi: 10.1109/ISMAR.2012.6402555. 16, 135

[101] Jens Grubert, Tobias Langlotz, Stefanie Zollmann, and Holger Regenbrecht. Towards pervasive augmented reality: Context-awareness in augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 23(6):1706–1724, 2017. doi: 10.1109/TVCG.2016.2543720. 5, 11, 19, 57, 122

[102] Uwe Gruenefeld, Jonas Auda, Florian Mathis, Stefan Schneegass, Mohamed Khamis, Jan Gugenheimer, and Sven Mayer. Vrception: Rapid prototyping of cross-reality systems in virtual reality. 2022. 136

[103] Aakar Gupta, Bo Rui Lin, Siyi Ji, Arjav Patel, and Daniel Vogel. Replicate and reuse: Tangible interaction design for digitally-augmented physical media objects. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367080. doi: 10.1145/3313831.3376139. URL https://doi.org/10.1145/3313831.3376139. 17

[104] Ankit Gupta, Dieter Fox, Brian Curless, and Michael Cohen. *DuploTrack: A Real-Time System for Authoring and Guiding Duplo Block Assembly*, page 389–402. Association for Computing Machinery, New York, NY, USA, 2012. ISBN 9781450315807. URL https://doi.org/10.1145/2380116.2380167. 91

[105] Taejin Ha, Steven Feiner, and Woontack Woo. Wearhand: Head-worn, rgb-d camera-based, bare-hand user interface with visually enhanced depth perception. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 219–228, 2014. doi: 10.1109/ISMAR.2014.6948431. 17

[106] Wenkai Han and Hans-Jörg Schulz. Beyond trust building – calibrating trust in visual analytics. In *Proceedings of the Workshop on TRust and EXperience in Visual Analytics (TREX)*, pages 9–15, New York, NY, USA, 2020. IEEE. to appear. 128

[107] Chris Harrison, Desney Tan, and Dan Morris. Skinput: Appropriating the body as an input surface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 453–462, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589299. doi: 10.1145/1753326.1753394. URL https://doi.org/10.1145/1753326.1753394. 17

[108] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. Omnitouch: Wearable multitouch interaction everywhere. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, page 441–450, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450307161. doi: 10.1145/2047196.2047255. URL https://doi.org/10.1145/2047196.2047255. 17

[109] Sandra G. Hart and Lowell E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In Peter A. Hancock and Najmedin Meshkati, editors, *Human Mental Workload*, volume 52 of *Advances in Psychology*, pages 139–183. North-Holland, 1988. doi: https://doi.org/10.1016/S0166-4115(08)62386-9. URL https://www.sciencedirect.com/science/article/pii/S0166411508623869. 13, 99, 112

[110] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. doi: 10.1109/CVPR.2016.90. 67, 73, 78, 80, 82

[111] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Oct 2017. doi: 10.1109/ICCV.2017.322. 12, 55, 67, 73, 82, 89, 90, 93

[112] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015. doi: 10.1109/TPAMI.2015.2389824. 12, 90

[113] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 12, 55, 90

[114] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 156

[115] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks, 2018. 78, 80

[116] S. J. Henderson and S. K. Feiner. Augmented reality in the psychomotor phase of a procedural task. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 191–200, Oct 2011. doi: 10.1109/ISMAR.2011.6092386. 68

[117] Steven Henderson and Steven Feiner. Opportunistic tangible user interfaces for augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):4–16, 2010. doi: 10.1109/TVCG.2009.91. 17

[118] Steven Henderson and Steven Feiner. Exploring the benefits of augmented reality documentation for maintenance and repair. *IEEE Transactions on Visualization and Computer Graphics*, 17(10):1355–1368, 2011. doi: 10.1109/TVCG.2010.245. 6, 68

[119] Steven J. Henderson and Steven Feiner. Opportunistic controls: Leveraging natural affordances as tangible user interfaces for augmented reality. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology*, VRST '08, page 211–218, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781595939517. doi: 10.1145/1450579.1450625. URL https://doi.org/10.1145/1450579.1450625. 17

[120] Steven J. Henderson and Steven Feiner. Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 135–144, 2009. doi: 10.1109/ISMAR.2009.5336486. 91

[121] Steven J. Henderson and Steven K. Feiner. Augmented reality in the psychomotor phase of a procedural task. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 191–200, 2011. doi: 10.1109/ISMAR.2011.6092386. 91, 107

[122] Anuruddha Hettiarachchi and Daniel Wigdor. Annexing reality: Enabling opportunistic use of everyday objects as tangible proxies in augmented reality. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 1957–1967, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450333627. doi: 10.1145/2858036.2858134. URL https://doi.org/10.1145/2858036.2858134. 17

[123] Jennifer L. Hicks, Thomas K. Uchida, Ajay Seth, Apoorva Rajagopal, and Scott L. Delp. Is My Model Good Enough? Best Practices for Verification and Validation of Musculoskeletal Models and Simulations of Movement. *Journal of Biomechanical Engineering*, 137(2), 02 2015. ISSN 0148-0731. doi: 10.1115/1.4029304. URL https://doi.org/10.1115/1.4029304. 020905. 118

[124] Otmar Hilliges, Christian Sandor, and Gudrun Klinker. Interactive prototyping for ubiquitous augmented reality user interfaces. In *Proceedings of the 11th International Conference on Intelligent User Interfaces*, IUI '06, page 285–287, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595932879. doi: 10.1145/1111449.1111512. URL https://doi.org/10.1145/1111449.1111512. 136

[125] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. Consumed endurance: A metric to quantify arm fatigue of mid-air interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 1063–1072, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450324731. doi: 10.1145/2556288.2557130. URL https://doi.org/10.1145/2556288.2557130. 13, 111, 113, 117, 126, 129

[126] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. Consumed endurance: A metric to quantify arm fatigue of mid-air interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 1063–1072, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450324731. doi: 10.1145/2556288.2557130. URL https://doi.org/10.1145/2556288.2557130. 133

[127] Tobias Hans Höllerer. *User Interfaces for Mobile Augmented Reality Systems*. PhD thesis, Columbia University, 2004. 136

[128] R. Hull, P. Neaves, and J. Bedford-Roberts. Towards situated computing. In *Digest of Papers. First International Symposium on Wearable Computers*, pages 146–153, 1997. doi: 10.1109/ISWC.1997.629931. 11

[129] Tobias Höllerer, Steven Feiner, Tachio Terauchi, Gus Rashid, and Drexel Hallaway. Exploring mars: developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers & Graphics*, 23 (6):779–785, 1999. ISSN 0097-8493. doi: https://doi.org/10.1016/S0097-8493(99)00103-X. URL https://www.sciencedirect.com/science/article/pii/S009784939900103X. 6

[130] Tobias Höllerer, Steven Feiner, Drexel Hallaway, Blaine Bell, Marco Lanzagorta, Dennis Brown, Simon Julier, Yohan Baillot, and Lawrence Rosenblum. User interface management techniques for collaborative mobile augmented reality. *Computers & Graphics*, 25(5):799–810, 2001. ISSN 0097-8493. doi: https://doi.org/10.1016/S0097-8493(01)00122-4. URL https://www.sciencedirect.com/science/article/pii/S0097849301001224. Mixed realities - beyond conventions. 15

[131] Verne T Inman, JB deC M Saunders, and LeRoy C Abbott. Observations on the function of the shoulder joint. *The Journal of Bone and Joint Surgery*, 26 (1):1–30, 1944. 115

[132] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pages 234–241, 1997. 17, 105

[133] Robert J. K. Jacob. What you look at is what you get: Eye movement-based interaction techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, page 11–18, New York, NY, USA, 1990. Association for Computing Machinery. ISBN 0201509326. doi: 10.1145/97243.97246. URL https://doi.org/10.1145/97243.97246. 17

[134] K. Jafari-Khouzani and H. Soltanian-Zadeh. Radon transform orientation estimation for rotation invariant texture analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):1004–1008, June 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.126. 67

[135] Sujin Jang, Wolfgang Stuerzlinger, Satyajit Ambike, and Karthik Ramani. Modeling cumulative arm fatigue in mid-air interaction based on perceived exertion and kinetics of arm motion. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 3328–3339, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346559. doi: 10.1145/3025453.3025523. URL https://doi.org/10.1145/3025453.3025523. 13, 111, 113

[136] Christoph Albert Johns, João Marcelo Evangelista Belo, Ken Pfeuffer, and Clemens Nylandsted Klokmose. Pareto optimal layouts for adaptive mixed reality. 2023. URL https://doi.org/10.1145/3544549.3585732. viii

[137] S. Julier, M. Lanzagorta, Y. Baillot, L. Rosenblum, S. Feiner, T. Hollerer, and S. Sestito. Information filtering for mobile augmented reality. In *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*, pages 3–11, 2000. doi: 10.1109/ISAR.2000.880917. 15

[138] Simon Julier, Marco Lanzagorta, Yohan Baillot, Lawrence Rosenblum, Steven Feiner, Tobias Hollerer, and Sabrina Sestito. Information filtering for mobile augmented reality. In *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*, pages 3–11, New York, NY, USA, 2000. IEEE. 114

[139] Andreas Kamilaris and Francesc X. Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147:70–90, 2018. ISSN 0168-1699. doi: https://doi.org/10.1016/j.compag.2018.02.016. URL https://www.sciencedirect.com/science/article/pii/S0168169917308803. 107

[140] SeungJun Kim and Anind K. Dey. Simulated augmented reality windshield display as a cognitive mapping aid for elder driver navigation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 133–142, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7. doi: 10.1145/1518701.1518724. URL http://doi.acm.org/10.1145/1518701.1518724. 66

[141] Scott Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science (New York, N.Y.)*, 220:671–80, 06 1983. doi: 10.1126/science.220.4598.671. 144

[142] Janin Koch, Andrés Lucero, Lena Hegemann, and Antti Oulasvirta. May ai? design ideation with cooperative contextual bandits. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–12, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359702. doi: 10.1145/3290605.3300863. URL https://doi.org/10.1145/3290605.3300863. 114

[143] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 67, 79

[144] Sarah Krings, Enes Yigitbas, Ivan Jovanovikj, Stefan Sauer, and Gregor Engels. Development framework for context-aware augmented reality applications. In *Companion Proceedings of the 12th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '20 Companion,

New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379847. doi: 10.1145/3393672.3398640. URL https://doi.org/10.1145/3393672.3398640. 18, 35, 36, 137, 139

[145] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11):1016–1022, 2018. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2018.08.474. URL https://www.sciencedirect.com/science/article/pii/S2405896318316021. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018. 97

[146] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. 12, 67, 90, 92

[147] Mikko Kytö, Barrett Ens, Thammathip Piumsomboon, Gun A. Lee, and Mark Billinghurst. Pinpointing: Precise head- and eye-based target selection for augmented reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–14, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356206. doi: 10.1145/3173574.3173655. URL https://doi.org/10.1145/3173574.3173655. 17

[148] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*, pages 574–591. Springer, 2020. 22

[149] Wallace S. Lages and Doug A. Bowman. Walking with adaptive augmented reality workspaces: Design and usage patterns. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, page 356–366, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362726. doi: 10.1145/3301275.3302278. URL https://doi.org/10.1145/3301275.3302278. 136

[150] Larry Laudan. *Progress and its problems: Towards a theory of scientific growth*, volume 282. Univ of California Press, 1978. 53

[151] Joseph J LaViola Jr, Ernst Kruijff, Ryan P McMahan, Doug Bowman, and Ivan P Poupyrev. *3D user interfaces: theory and practice*. Addison-Wesley Professional, Boston, MA, USA, 2017. 111, 112, 113

[152] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition.

Neural Computation, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541. 12, 90

[153] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. Evaluation strategies for hci toolkit research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356206. doi: 10.1145/3173574.3173610. URL https://doi.org/10.1145/3173574.3173610. 56, 122, 125, 148, 149

[154] Taehee Lee and Tobias Hollerer. Handy ar: Markerless inspection of augmented reality objects using fingertip tracking. In *2007 11th IEEE International Symposium on Wearable Computers*, pages 83–90, 2007. doi: 10.1109/ISWC.2007.4373785. 16

[155] Germán Leiva, Jens Emil Grønbæk, Clemens Nylandsted Klokmose, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. Rapido: Prototyping interactive ar experiences through programming by demonstration. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, UIST '21, page 626–637, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450386357. doi: 10.1145/3472749.3474774. URL https://doi.org/10.1145/3472749.3474774. 136

[156] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81 (2):155–166, 2009. 90

[157] Jiefeng Li, Chao Xu, Zhicun Chen, Siyuan Bian, Lixin Yang, and Cewu Lu. Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3383–3393, June 2021. 22

[158] LW Light and Peter Anderson. Designing better keyboards via simulated annealing. *AI Expert*, 8(9):20–27, 1993. 15

[159] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, July 2017. doi: 10.1109/CVPR.2017.106. 78, 82

[160] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1. 78, 82

[161] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 12, 90

[162] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 12, 90

[163] David Lindlbauer, Anna Maria Feit, and Otmar Hilliges. Context-aware online adaptation of mixed reality interfaces. In *Proceedings of the 32Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, pages 147–160, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6816-2. doi: 10.1145/3332165.3347945. URL http://doi.acm.org/10.1145/3332165.3347945. 16, 24, 35, 36, 54, 57, 111, 114, 133, 136, 138, 156

[164] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2017.07.005. URL https://www.sciencedirect.com/science/article/pii/S1361841517301135. 107

[165] Baili Liu, Gregory Francis, and Gavriel Salvendy. Applying models of visual search to menu design. *International Journal of Human-Computer Studies*, 56 (3):307–330, 2002. 15

[166] Mingyu Liu, Mathieu Nancel, and Daniel Vogel. Gunslinger: Subtle arms-down mid-air interaction. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, page 63–71, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450337793. doi: 10.1145/2807442.2807489. URL https://doi.org/10.1145/2807442.2807489. 112, 113

[167] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 12, 90

[168] Feiyu Lu and Yan Xu. Exploring spatial ui transition mechanisms with head-worn augmented reality. In *ACM CHI Conference on Human Factors in Computing Systems*, CHI '22. ACM, 2022. 33, 136

[169] Feiyu Lu, Shakiba Davari, Lee Lisle, Yuan Li, and Doug A Bowman. Glanceable ar: Evaluating information access methods for head-worn augmented reality. In *2020 IEEE conference on virtual reality and 3D user interfaces (VR)*, pages 930–939. IEEE, 2020. 136

[170] Jacob Boesen Madsen, Markus Tatzgern, Claus B. Madsen, Dieter Schmalstieg, and Denis Kalkofen. Temporal coherence strategies for augmented reality labeling. *IEEE Transactions on Visualization and Computer Graphics*, 22(4): 1415–1423, 2016. doi: 10.1109/TVCG.2016.2518318. 16, 135, 143

[171] Andrew Maimone and Henry Fuchs. Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 137–146, 2011. doi: 10.1109/ISMAR.2011.6092379. 6

[172] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6): 369–395, 2004. 28, 30, 141, 144, 156

[173] Shouichi Matsui and Seiji Yamada. Genetic algorithm can optimize hierarchical menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 1385–1388, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580111. doi: 10.1145/1357054.1357271. URL https://doi.org/10.1145/1357054.1357271. 15

[174] Lynn McAtamney and E Nigel Corlett. Rula: a survey method for the investigation of work-related upper limb disorders. *Applied ergonomics*, 24(2):91–99, 1993. 112, 116, 117, 129

[175] Ann McNamara and Chethna Kabeerdoss. Mobile augmented reality: Placing labels based on gaze position. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pages 36–37, 2016. doi: 10.1109/ISMAR-Adjunct.2016.0033. 17

[176] Ann McNamara, Katherine Boyd, Joanne George, Weston Jones, Somyung Oh, and Annie Suther. Information placement in virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 1765–1769, 2019. doi: 10.1109/VR.2019.8797891. 17

[177] Microsoft. About hololens 2, 2022. URL https://docs.microsoft.com/en-us/hololens/hololens2-hardware. Accessed: 2022-07-24. 147

[178] Microsoft. Spatial awareness getting started — mrtk2, 2022. URL https://docs.microsoft.com/en-gb/windows/mixed-reality/mrtk-unity/mrtk2/features/spatial-awareness/spatial-awareness-getting-started?view=mrtkunity-2021-05. Accessed: 2022-07-24. 139, 144

[179] Microsoft. Solver overview — mrtk2, 2022. URL https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/solvers/solver?view=mrtkunity-2022-05. Accessed: 2022-07-24. 18, 27, 54, 137, 149

[180] Microsoft. Mixed reality toolkit (mrtk) for unity, n.d. URL https://github.com/microsoft/MixedRealityToolkit-Unity. Accessed: 2022-07-24. 18, 27, 133, 137, 139

[181] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329, 1994. 3

[182] A. Miller, B. White, E. Charbonneau, Z. Kanzler, and J. J. LaViola Jr. Interactive 3d model acquisition and tracking of building block structures. *IEEE Transactions on Visualization and Computer Graphics*, 18(4):651–659, April 2012. ISSN 1077-2626. doi: 10.1109/TVCG.2012.48. 84, 91

[183] Mahdi H. Miraz, Maaruf Ali, and Peter S. Excell. Adaptive user interfaces and universal usability through plasticity of user interface design. *Computer Science Review*, 40:100363, 2021. ISSN 1574-0137. doi: https://doi.org/10.1016/j.cosrev.2021.100363. URL https://www.sciencedirect.com/science/article/pii/S1574013721000034. 135

[184] Roberto A. Montano Murillo, Sriram Subramanian, and Diego Martinez Plasencia. Erg-o: Ergonomic optimization of immersive virtual environments. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, page 759–771, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349819. doi: 10.1145/3126594.3126605. URL https://doi.org/10.1145/3126594.3126605. 112

[185] G. Mori, F. Paterno, and C. Santoro. Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Transactions on Software Engineering*, 30(8):507–520, 2004. doi: 10.1109/TSE.2004.40. 135

[186] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 17

[187] Leon Müller, Ken Pfeuffer, Jan Gugenheimer, Bastian Pfleging, Sarah Prange, and Florian Alt. Spatialproto: Exploring real-world motion captures for rapid prototyping of interactive mixed reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380966. doi: 10.1145/3411764.3445560. URL https://doi.org/10.1145/3411764.3445560. 136

[188] Brad Myers, Scott E. Hudson, and Randy Pausch. Past, present, and future of user interface software tools. *ACM Trans. Comput.-Hum. Interact.*, 7(1):3–28, March 2000. 14, 18, 113

[189] Michael Nebeling, Katy Lewis, Yu-Cheng Chang, Lihan Zhu, Michelle Chung, Piaoyang Wang, and Janet Nebeling. Xrdirector: A role-based collaborative immersive authoring system. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367080. doi: 10.1145/3313831.3376637. URL https://doi.org/10.1145/3313831.3376637. 136

[190] U. Neumann and A. Majoros. Cognitive, performance, and systems issues for augmented reality applications in manufacturing and maintenance. In *Proceedings. IEEE 1998 Virtual Reality Annual International Symposium (Cat. No.98CB36180)*, pages 4–11, March 1998. doi: 10.1109/VRAIS.1998.658416. 66

[191] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016. 12, 90

[192] Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld, and Mathilde Pignol. Generating remote control interfaces for complex appliances. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, UIST '02, page 161–170, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134886. doi: 10.1145/571985.572008. URL https://doi.org/10.1145/571985.572008. 135

[193] Lauren Norrie and Roderick Murray-Smith. Investigating ui displacements in an adaptive mobile homescreen. 8(3), 2016. ISSN 1942-390X. doi: 10.4018/IJMHCI.2016070101.oa. URL https://doi.org/10.4018/IJMHCI.2016070101.oa. 135

[194] Benjamin Nuernberger, Eyal Ofek, Hrvoje Benko, and Andrew D. Wilson. Snaptoreality: Aligning augmented reality to the real world. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, page 1233–1244, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450333627. URL https://doi.org/10.1145/2858036.2858250. 114, 133, 136

[195] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Designscape: Design with interactive layout suggestions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 1221–1224, New York, NY, USA, 2015. Association for Computing Machinery.

ISBN 9781450331456. doi: 10.1145/2702123.2702149. URL https://doi.org/10.1145/2702123.2702149. 15

[196] Allan Oliveira and Regina B. Araujo. Creation and visualization of context aware augmented reality interfaces. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, page 324–327, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450312875. doi: 10.1145/2254556.2254618. URL https://doi.org/10.1145/2254556.2254618. 136

[197] Antti Oulasvirta and Kasper Hornbæk. Hci research as problem-solving. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 4956–4967, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450333627. doi: 10.1145/2858036.2858283. URL https://doi.org/10.1145/2858036.2858283. 8, 53, 54, 55

[198] Antti Oulasvirta, Per Ola Kristensson, Xiaojun Bi, and Andrew Howes. *Computational interaction*. Oxford University Press, 2018. 5, 13, 14, 15, 28

[199] Antti Oulasvirta, Niraj Ramesh Dayama, Morteza Shiripour, Maximilian John, and Andreas Karrenbauer. Combinatorial optimization of graphical user interface designs. *Proceedings of the IEEE*, 108(3):434–464, 2020. doi: 10.1109/JPROC.2020.2969687. 14, 113, 135, 139

[200] Antti Oulasvirta, Jussi P. P. Jokinen, and Andrew Howes. Computational rationality as a theory of interaction. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391573. doi: 10.1145/3491102.3517739. URL https://doi.org/10.1145/3491102.3517739. 34

[201] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Learning layouts for single-page graphic designs. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1200–1213, 2014. doi: 10.1109/TVCG.2014.48. 15

[202] V. Paelke. Augmented reality in the smart factory: Supporting workers in an industry 4.0. environment. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–4, Sep. 2014. doi: 10.1109/ETFA.2014.7005252. 68

[203] Hyung Min Park, Seok Han Lee, and Jong Soo Choi. Wearable augmented reality system using gaze interaction. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 175–176, 2008. doi: 10.1109/ISMAR.2008.4637353. 17

[204] Seonwook Park, Christoph Gebhardt, Roman Rädle, Anna Feit, Hana Vrzakova, Niraj Dayama, Hui-Shyong Yeo, Clemens Klokmose, Aaron Quigley, Antti Oulasvirta, and Otmar Hilliges. AdaM: Adapting Multi-User Interfaces for

Collaborative Environments in Real-Time. In *SIGCHI Conference on Human Factors in Computing Systems*, CHI '18, New York, NY, USA, 2018. ACM. 135

[205] Fabio Paterno, Carmen Santoro, and Lucio Davide Spano. Maria: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Trans. Comput.-Hum. Interact.*, 16(4), nov 2009. ISSN 1073-0516. doi: 10.1145/1614390.1614394. URL https://doi.org/10.1145/1614390.1614394. 135

[206] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019. 12, 22

[207] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, 2014. doi: 10.1109/SURV.2013.042313.00197. 11

[208] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017. 79

[209] Ken Pfeuffer, Yasmeen Abdrabou, Augusto Esteves, Radiah Rivu, Yomna Abdelrahman, Stefanie Meitner, Amr Saadi, and Florian Alt. Artention: A design space for gaze-adaptive user interfaces in augmented reality. *Computers & Graphics*, 95:1–12, 2021. 136

[210] J. Platonov, H. Heibel, P. Meier, and B. Grollmann. A mobile markerless ar system for maintenance and repair. In *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 105–108, Oct 2006. doi: 10.1109/ISMAR.2006.297800. 68

[211] Jarkko Polvi, Takafumi Taketomi, Atsunori Moteki, Toshiyuki Yoshitake, Toshiyuki Fukuoka, Goshiro Yamamoto, Christian Sandor, and Hirokazu Kato. Handheld guides in inspection tasks: Augmented reality versus picture. *IEEE Transactions on Visualization and Computer Graphics*, 24(7):2118–2128, 2018. doi: 10.1109/TVCG.2017.2709746. 6, 68, 91, 107

[212] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 149

[213] PyTorch. Pytorch - an open source deep learning platform that provides a seamless path from research prototyping to production deployment, 2019. URL https://pytorch.org/. Accessed: 6/25/2019. 78

[214] Roman Rädle, Hans-Christian Jetter, Jonathan Fischer, Inti Gabriel, Clemens N. Klokmose, Harald Reiterer, and Christian Holz. Polartrack: Optical outside-in device tracking that exploits display polarization. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, April 2018. URL https://www.microsoft.com/en-us/research/publication/polartrack-optical-outside-device-tracking-exploits-display-polarization/. 73

[215] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 179–188, New York, NY, USA, 1998. ACM. ISBN 0-89791-999-8. doi: 10.1145/280814.280861. URL http://doi.acm.org/10.1145/280814.280861. 68

[216] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pages 512–519, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-4308-1. doi: 10.1109/CVPRW.2014.131. URL http://dx.doi.org/10.1109/CVPRW.2014.131. 67, 79

[217] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, June 2016. doi: 10.1109/CVPR.2016.91. 12, 67, 90

[218] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 12, 90

[219] Dirk Reiners, Didier Stricker, Gudrun Klinker, and Stefan Müller. 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*, IWAR '98, pages 31–46, Natick, MA, USA, 1999. A. K. Peters, Ltd. ISBN 1-56881-098-9. URL http://dl.acm.org/citation.cfm?id=322690.322694. 68

[220] Jun Rekimoto and Masanori Saitoh. Augmented surfaces: A spatially continuous work space for hybrid computing environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 378–385, New York, NY, USA, 1999. ACM. ISBN 0-201-48559-1. doi: 10.1145/302979.303113. URL http://doi.acm.org/10.1145/302979.303113. 67, 68

[221] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes,

N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf. 12, 67, 90

[222] C. M. Robertson, B. MacIntyre, and B. N. Walker. An evaluation of graphical context when the graphics are outside of the task area. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 73–76, Sep. 2008. doi: 10.1109/ISMAR.2008.4637328. 84

[223] Robert D Rogers and Stephen Monsell. Costs of a predictible switch between simple cognitive tasks. *Journal of experimental psychology: General*, 124(2): 207, 1995. 66

[224] David A. Rosenbaum. Chapter 2 - core problems. In David A. Rosenbaum, editor, *Human Motor Control (Second Edition)*, pages 11 – 41. Academic Press, San Diego, second edition edition, 2010. ISBN 978-0-12-374226-1. doi: https://doi.org/10.1016/B978-0-12-374226-1.00002-4. URL http://www.sciencedirect.com/science/article/pii/B9780123742261000024. 117, 121

[225] Scott D Roth. Ray casting for modeling solids. *Computer graphics and image processing*, 18(2):109–144, 1982. 143

[226] Franz Rothlauf. *Design of modern heuristics: principles and application*, volume 8. Springer, 2011. 14

[227] Nick S Ryan, Jason Pascoe, and David R Morse. Enhanced reality fieldwork: the context-aware archaeological assistant. In *Computer applications in archaeology*. Tempus Reparatum, 1998. 11

[228] Paulo Salem. User interface optimization using genetic programming with an application to landing pages. *Proc. ACM Hum.-Comput. Interact.*, 1 (EICS), June 2017. doi: 10.1145/3099583. URL https://doi.org/10.1145/3099583. 135

[229] Antti Salovaara, Antti Oulasvirta, and Giulio Jacucci. Evaluation of prototypes and the problem of possible futures. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 2064–2077, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346559. doi: 10.1145/3025453.3025658. URL https://doi.org/10.1145/3025453.3025658. 54

[230] Christian Sandor and Gudrun Klinker. A rapid prototyping software infrastructure for user interfaces in ubiquitous augmented reality. *Personal Ubiquitous Computing*, 9(3):169–185, 2005. ISSN 1617-4909. doi: http://dx.doi.org/10.1007/s00779-004-0328-1. 136

[231] Katherine R. Saul, Xiao Hu, Craig M. Goehler, Meghan E. Vidt, Melissa Daly, Anca Velisar, and Wendy M. Murray. Benchmarking of dynamic simulation predictions in two software platforms using an upper limb musculoskeletal model. *Computer Methods in Biomechanics and Biomedical Engineering*, 18 (13):1445–1458, 2015. 118

[232] B.N. Schilit and M.M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, 1994. doi: 10.1109/65.313011. 11

[233] Albrecht Schmidt, Michael Beigl, and Hans-W Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999. 11

[234] Bernd Schwald and Blandine De Laval. An augmented reality system for training and assistance to maintenance in the industrial context. 2003. 6

[235] A. Sears. Layout appropriateness: a metric for evaluating user interface widget layout. *IEEE Transactions on Software Engineering*, 19(7):707–719, 1993. doi: 10.1109/32.238571. 15

[236] Suzanne C. Segerstrom and Lise Solberg Nes. Heart rate variability reflects self-regulatory strength, effort, and fatigue. *Psychological Science*, 18(3):275–281, 2007. doi: 10.1111/j.1467-9280.2007.01888.x. URL https://doi.org/10.1111/j.1467-9280.2007.01888.x. PMID: 17444926. 13, 113

[237] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. 2014. Publisher Copyright: © 2014 International Conference on Learning Representations, ICLR. All rights reserved.; 2nd International Conference on Learning Representations, ICLR 2014 ; Conference date: 14-04-2014 Through 16-04-2014. 12, 90

[238] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, Daniel Freedman, Pushmeet Kohli, Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 3633–3642, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450331456. doi: 10.1145/2702123.2702179. URL https://doi.org/10.1145/2702123.2702179. 17

[239] Roger N. Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 171(3972):701–703, 1971. ISSN 0036-8075. doi: 10.1126/science.171.3972.701. URL https://science.sciencemag.org/content/171/3972/701. 66

[240] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 12, 67

[241] Gisela Sjøgaard, Gabrielle Savard, and Carsten Juel. Muscle blood flow during isometric activity and its relation to muscle fatigue. *European journal of applied physiology and occupational physiology*, 57(3):327–335, 1988. 13, 113

[242] Chen Song, Jiaru Song, and Qixing Huang. Hybridpose: 6d object pose estimation under hybrid representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 12, 90

[243] Sony. Imaging edge remote, 2019. URL https://imagingedge.sony.net/en-us/ie-desktop.html. Accessed: 6/22/2019. 78

[244] Maximilian Speicher, Brian D. Hall, and Michael Nebeling. What is mixed reality? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–15, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359702. doi: 10.1145/3290605.3300767. URL https://doi.org/10.1145/3290605.3300767. 3

[245] India Starker and Richard A. Bolt. A gaze-responsive self-disclosing display. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, page 3–10, New York, NY, USA, 1990. Association for Computing Machinery. ISBN 0201509326. doi: 10.1145/97243.97245. URL https://doi.org/10.1145/97243.97245. 17

[246] Zinovia Stefanidi, George Margetis, Stavroula Ntoa, and George Papagiannakis. Real-time adaptation of context-aware intelligent user interfaces, for enhanced situational awareness. *IEEE Access*, 10:23367–23393, 2022. doi: 10.1109/ACCESS.2022.3152743. 35

[247] Mengu Sukan, Carmine Elvezio, Steven Feiner, and Barbara Tversky. Providing assistance for orienting 3d objects using monocular eyewear. In *Proceedings of the 2016 Symposium on Spatial User Interaction*, SUI '16, page 89–98, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340687. doi: 10.1145/2983310.2985764. URL https://doi.org/10.1145/2983310.2985764. 91

[248] Ivan E. Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), page 757–764, New York, NY, USA, 1968. Association for Computing Machinery. ISBN 9781450378994. doi: 10.1145/1476589.1476686. URL https://doi.org/10.1145/1476589.1476686. 3

[249] Ryo Suzuki, Rubaiat Habib Kazi, Li-Yi Wei, Stephen DiVerdi, Wilmot Li, and Daniel Leithinger. Realitysketch: Embedding responsive graphics and visualizations in ar with dynamic sketching. In *Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20

Adjunct, page 135–138, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375153. doi: 10.1145/3379350.3416155. URL https://doi.org/10.1145/3379350.3416155. 136

[250] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015. doi: 10.1109/CVPR.2015.7298594. 80

[251] Arthur Tang, Charles Owen, Frank Biocca, and Weimin Mou. Comparative effectiveness of augmented reality in object assembly. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, page 73–80, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581136307. doi: 10.1145/642611.642626. URL https://doi.org/10.1145/642611.642626. 6, 68

[252] Vildan Tanriverdi and Robert J. K. Jacob. Interacting with eye movements in virtual environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, page 265–272, New York, NY, USA, 2000. Association for Computing Machinery. ISBN 1581132166. doi: 10.1145/332040.332443. URL https://doi.org/10.1145/332040.332443. 17

[253] Markus Tatzgern, Denis Kalkofen, Raphael Grasset, and Dieter Schmalstieg. Hedgehog labeling: View management techniques for external labels in 3d space. In *2014 IEEE Virtual Reality (VR)*, pages 27–32, 2014. doi: 10.1109/VR.2014.6802046. 16, 135

[254] Markus Tatzgern, Valeria Orso, Denis Kalkofen, Giulio Jacucci, Luciano Gamberini, and Dieter Schmalstieg. Adaptive information density for augmented reality displays. In *2016 IEEE Virtual Reality (VR)*, pages 83–92, 2016. doi: 10.1109/VR.2016.7504691. 15, 16, 114, 136

[255] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. doi: 10.1109/IROS.2017.8202133. 91, 94

[256] Kashyap Todi, Daryl Weir, and Antti Oulasvirta. Sketchplore: Sketch and explore with a layout optimiser. DIS '16, page 543–555, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340311. doi: 10.1145/2901790.2901817. URL https://doi.org/10.1145/2901790.2901817. 15, 18, 113

[257] Kashyap Todi, Gilles Bailly, Luis Leiva, and Antti Oulasvirta. Adapting user interfaces with model-based reinforcement learning. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21,

New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380966. doi: 10.1145/3411764.3445497. URL https://doi.org/10.1145/3411764.3445497. 135, 156

[258] Deepak Tolani and Norman I Badler. Real-Time Inverse Kinematics of the Human Arm. *Presence: Teleoperators and Virtual Environments*, 5(4):393–401, 1996. doi: 10.1162/pres.1996.5.4.393. URL https://doi.org/10.1162/pres.1996.5.4.393. 116

[259] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 12, 90

[260] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018. 91

[261] Unity. Unity coroutines, n.d.. URL https://docs.unity3d.com/Manual/Coroutines.html. Accessed: 2022-07-24. 144

[262] Unity. Unity mars, n.d.. URL https://unity.com/products/unity-mars. Accessed: 2022-07-24. 18, 54, 137

[263] Antonio E. Uva, Michele Gattullo, Vito M. Manghisi, Daniele Spagnulo, Giuseppe L. Cascella, and Michele Fiorentino. Evaluating the effectiveness of spatial augmented reality in smart manufacturing: a solution for manual working stations. *The International Journal of Advanced Manufacturing Technology*, 94(1):509–521, Jan 2018. ISSN 1433-3015. doi: 10.1007/s00170-017-0846-4. URL https://doi.org/10.1007/s00170-017-0846-4. 68

[264] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. 45(1):41–51, January 1985. ISSN 0022-3239. doi: 10.1007/BF00940812. URL https://doi.org/10.1007/BF00940812. 144

[265] Colin Ware and Harutune H. Mikaelian. An evaluation of an eye tracker as a device for computer input2. *SIGCHI Bull.*, 17(SI):183–188, may 1986. ISSN 0736-6906. doi: 10.1145/30851.275627. URL https://doi.org/10.1145/30851.275627. 17

[266] Sabine Webel, Uli Bockholt, Timo Engelke, Nirit Gavish, Manuel Olbrich, and Carsten Preusche. An augmented reality training platform for assembly and maintenance skills. *Robotics and Autonomous Systems*, 61(4):398–403, 2013. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2012.09.013. URL https://www.sciencedirect.com/science/

article/pii/S0921889012001674. Models and Technologies for Multimodal Skill Training. 6

[267] Anthony Webster, Steven Feiner, Blair MacIntyre, William Massie, and Theodore Krueger. Augmented reality in architectural construction, inspection and renovation. In *Proc. ASCE Third Congress on Computing in Civil Engineering*, volume 1, page 996. Citeseer, 1996. 6

[268] S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4732, June 2016. doi: 10.1109/CVPR.2016.511. 67

[269] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 12, 90

[270] Mark Weiser. The computer for the 21 st century. *Scientific American*, 265(3): 94–105, 1991. ISSN 00368733, 19467087. URL http://www.jstor.org/stable/24938718. 17

[271] Pierre Wellner, Wendy Mackay, and Rich Gold. Back to the real world. *Commun. ACM*, 36(7):24–26, July 1993. ISSN 0001-0782. doi: 10.1145/159544.159555. URL http://doi.acm.org.ez.statsbiblioteket.dk:2048/10.1145/159544.159555. 68

[272] Pierre David Wellner. Interacting with paper on the DigitalDesk. Technical Report UCAM-CL-TR-330, University of Cambridge, Computer Laboratory, March 1994. URL https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-330.pdf. 68

[273] Eric Whitmire, Hrvoje Benko, Christian Holz, Eyal Ofek, and Mike Sinclair. Haptic revolver: Touch, shear, texture, and shape rendering on a reconfigurable virtual reality controller. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356206. doi: 10.1145/3173574.3173660. URL https://doi.org/10.1145/3173574.3173660. 16

[274] Kenneth L Wilson, Jayfus T Doswell, Olatokunbo S Fashola, Wayne Debeatham, Nii Darko, Travelyan M Walker, Omar K Danner, Leslie R Matthews, and William L Weaver. Using augmented reality as a clinical support tool to assist combat medics in the treatment of tension pneumothoraces. *Military medicine*, 178(9):981–985, 2013. 6

[275] Jacob O. Wobbrock and Julie A. Kientz. Research contributions in human-computer interaction. *Interactions*, 23(3):38–44, apr 2016. ISSN 1072-5520. doi: 10.1145/2907069. URL https://doi.org/10.1145/2907069. 53

[276] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. doi: 10.1109/4235.585893. 14

[277] Robert Xiao, Scott Hudson, and Chris Harrison. Supporting responsive cohabitation between virtual interfaces and physical objects on everyday surfaces. *Proc. ACM Hum.-Comput. Interact.*, 1(EICS), jun 2017. doi: 10.1145/3095814. URL https://doi.org/10.1145/3095814. 136

[278] Robert Xiao, Julia Schwarz, Nick Throm, Andrew D. Wilson, and Hrvoje Benko. Mrtouch: Adding touch input to head-mounted mixed reality. *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1653–1660, 2018. doi: 10.1109/TVCG.2018.2794222. 17, 155

[279] Xuhai Xu, Mengjie Yu, Tanya R. Jonker, Kashyap Todi, Feiyu Lu, Xun Qian, João Marcelo Evangelista Belo, Tianyi Wang, Michelle Li, Aran Mun, Te-Yen Wu, Junxiao Shen, Ting Zhang, Narine Kokhlikyan, Fulton Wang, Paul Sorenson, Sophie Kahyun Kim, and Hrvoje Benko. Xair: A framework of explainable ai in augmented reality. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2023. Association for Computing Machinery. URL https://doi.org/10.1145/3544548.3581500. viii

[280] Zhen Yang, Jinlei Shi, Wenjun Jiang, Yuexin Sui, Yimin Wu, Shu Ma, Chunyan Kang, and Hongting Li. Influences of augmented reality assistance on performance and cognitive loads in different stages of assembly task. *Frontiers in Psychology*, 10, 2019. ISSN 1664-1078. doi: 10.3389/fpsyg.2019.01703. URL https://www.frontiersin.org/article/10.3389/fpsyg.2019.01703. 91, 107

[281] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3320–3328. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf. 67

[282] Franziska Zacharias, Christoph Borst, and Gerd Hirzinger. Capturing robot workspace structure: representing robot capabilities. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3229–3236, New York, NY, USA, 2007. IEEE. 112

[283] L. Zadeh. Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, 8(1):59–60, 1963. doi: 10.1109/TAC.1963.1105511. 30

[284] André Zenner and Antonio Krüger. Drag:on: A virtual reality controller providing haptic feedback based on drag and weight shift. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–12, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359702. doi: 10.1145/3290605.3300441. URL https://doi.org/10.1145/3290605.3300441. 16

[285] André Zenner and Antonio Krüger. Shifty: A weight-shifting dynamic passive haptic proxy to enhance object perception in virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 23(4):1285–1294, 2017. doi: 10.1109/TVCG.2017.2656978. 16

[286] Shumin Zhai, Michael Hunter, and Barton A Smith. Performance optimization of virtual keyboards. *Human–Computer Interaction*, 17(2-3): 229–269, 2002. doi: 10.1080/07370024.2002.9667315. URL https://www.tandfonline.com/doi/abs/10.1080/07370024.2002.9667315. 15

[287] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019. doi: 10.1109/TNNLS.2018.2876865. 22

[288] Guy W. Zimmerman, Dale Klopfer, G. Michael Poor, Julie Barnes, Laura Leventhal, and Samuel D. Jaffee. "how do i line up?": Reducing mental transformations to improve performance. In *Proceedings of the 14th International Conference on Human-computer Interaction: Design and Development Approaches - Volume Part I*, HCII'11, pages 432–440, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21601-5. URL http://dl.acm.org/citation.cfm?id=2022384.2022435. 66

[289] E. Zitzler, Laumanns, M., and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, EUROGEN*, 2001. 156