



BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

USER INTERFACE FOR ARTABLE AND MICROSOFT HOLOLENS

UŽIVATELSKÉ ROZHRANÍ PRO ARTABLE S BRÝLEMI MICROSOFT HOLOLENS

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR

AUTOR PRÁCE

Bc. DANIEL BAMBUŠEK

SUPERVISOR

VEDOUCÍ PRÁCE

Ing. MICHAL KAPINUS

BRNO 2018

Brno University of Technology - Faculty of Information Technology

Department of Computer Graphics and Multimedia

Academic year 2017/2018

Master's Thesis Specification

For: **Bambušek Daniel, Bc.**

Branch of study: Computer Graphics and Multimedia

Title: **User Interface for ARTable and Microsoft Hololens**

Category: User Interfaces

Instructions for project work:

1. Study the available sources of information about Augmented Reality (AR) and user interface design for AR. Familiarize yourself with the ROS middleware and Microsoft Hololens glasses.
2. Familiarize yourself with the experimental platform ARTable and its application programming interface.
3. Select suitable methods and propose user interface that will display current state of the system, allow user to change parameters and settings of the system and warn him of errors.
4. Implement proposed interface on the Microsoft Hololens glasses and integrate it to the ARTable robotic system.
5. Perform experiments, demonstrate and discuss functionality of the solution and discuss opportunities for further development.
6. Create a poster or a video to present your thesis, its goals and results.

Basic references:

- According to the supervisor's instruction

Requirements for the semestral defense:

- Items 1-3 of the assignment

Detailed formal specifications can be found at <http://www.fit.vutbr.cz/info/szz/>

The Master's Thesis must define its purpose, describe a current state of the art, introduce the theoretical and technical background relevant to the problems solved, and specify what parts have been used from earlier projects or have been taken over from other sources.

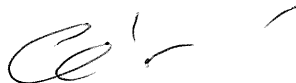
Each student will hand-in printed as well as electronic versions of the technical report, an electronic version of the complete program documentation, program source files, and a functional hardware prototype sample if desired. The information in electronic form will be stored on a standard non-rewritable medium (CD-R, DVD-R, etc.) in formats common at the FIT. In order to allow regular handling, the medium will be securely attached to the printed report.

Supervisor: **Kapinus Michal, Ing.,** DCGM FIT BUT

Beginning of work: November 1, 2017

Date of delivery: May 23, 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2
L.S.



Jan Černocký

Associate Professor and Head of Department

Abstract

This thesis focuses on usability of mixed reality head-mounted display – Microsoft HoloLens – in a human-robot collaborative workspace – the ARTable. Use of the headset is demonstrated by created user interface which helps regular workers to better and faster understand the ARTable system. It allows to spatially visualize learned programs, without the necessity to run the robot itself. The user is guided by 3D animation of individual programs and by device voice, which helps him to get a clear idea of what will happen if he runs the program directly on the robot. The solution also provides interactive guidance for the user when programming the robot. Using mixed reality displays also enables to visualize valuable spatial information, such as robot perception.

Abstrakt

Tato práce se zaměřuje na použitelnost brýlí Microsoft HoloLens pro rozšířenou realitu v prototypu pracoviště pro spolupráci člověka s robotem – „ARTable“. Použití brýlí je demonstrováno vytvořeným uživatelským rozhraním, které pomáhá uživatelům lépe a rychleji porozumět systému ARTable. Umožňuje prostorově vizualizovat naučné programy, aniž by bylo nutné spouštět samotného robota. Uživatel je veden 3D animací jednotlivých programů a hlasem zařízení, což mu pomůže získat jasnou představu o tom, co by se stalo, pokud by program spustil přímo na robotovi. Implementované řešení také umožňuje interaktivně provést uživatele celým procesem programování robota. Použití brýlí umožňuje mimo jiné zobrazit cenné prostorové informace, například vidění robota, tedy zvýraznit ty objekty, které jsou robotem detekovány.

Keywords

ARTable, Microsoft HoloLens, Augmented reality, Mixed reality, mixed reality head-mounted display, human-robot collaborative workspace, human-robot interaction, program visualization, PR2, Unity

Klíčová slova

ARTable, Microsoft HoloLens, Rozšířená realita, Smíšená realita, brýle pro smíšenou realitu, pracoviště pro spolupráci člověka s robotem, interakce člověka s robotem, vizualizace programů, PR2, Unity

Reference

BAMBUŠEK, Daniel. *User Interface for ARTable and Microsoft HoloLens*. Brno, 2018. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Michal Kapinus

Rozšířený abstrakt

Se stále větší dostupností průmyslových robotů je pravděpodobné, že malé a střední podniky si začnou tyto roboty pořizovat, za cílem zvýšení produktivity podniku. Avšak v takových podnicích jsou výrobní šarže menší a produkty bývají často přizpůsobeny pro konkrétní zakázku. To vyžaduje přeprogramování robotů pro konkrétní úkoly, ke kterému je často potřeba odborník disponující nutnými znalostmi konkrétního robota, který by mohl takové podniky vyjít draho. Proto by bylo prospěšné umožnit běžnému pracovníku přeprogramovat tyto roboty bez nutnosti hlubokých znalostí. Proto byl vytvořen prototyp pracoviště pro spolupráci člověka s robotem – „ARTable“, který umožňuje běžnému uživateli robota programovat na vysoké úrovni abstrakce pomocí prostorové rozšířené reality a multimodálního vstupu a výstupu.

Takové programování je založeno na nastavování parametrů konkrétních instrukcí, které tvoří celkový program. Avšak jednotlivé instrukce jsou popsány pouze jednoduchými textovými popisky a jak testy ukázaly, většina uživatelů měla potíže s orientací v topologii celého programu. Uživatelé byli nejistí ohledně toho, co která instrukce znamená, co přesně udělá, pokud bude puštěna na reálném robotu a co se od něj vůbec očekává v rámci nastavování parametrů pro konkrétní program.

Cílem této práce je tedy odstranit uživatelské nejistoty tím, že uživateli nabídne možnost vizualizovat existující programy prostřednictvím brýlí Microsoft HoloLens. Aplikace dále interaktivně provede uživatele samotným procesem programování robota. Celkově tak rozšíří možnosti 2D promítaného uživatelského rozhraní tím, že umožní vizualizovat prostorové informace prostřednictvím těchto brýlí.

Teoretická část práce popisuje základy rozšířené reality, porovnává ji s virtuální realitou, popisuje nejčastější typy zařízení pro rozšířenou realitu, detailně popisuje brýle Microsoft HoloLens a systém ARTable. Tento zahrnuje robota, několik senzorů, projektor a stůl s dotykovou vrstvou.

V praktické části práce je navrženo a dále implementováno uživatelské rozhraní pro Microsoft HoloLens, které umožní vizualizovat programy systému ARTable a provede uživatele procesem programování robota. Práce je implementována prostřednictvím herního enginu Unity, který využívá jazyk C# pro skriptování. Bylo maximálně využito existujících balíčků a zdrojových souborů, jejichž autoři jsou řádně ocitováni, pro urychlení implementace a umožnění se soustředit na jádro práce samotné. Prvním problémem, který bylo nutné vyřešit, byla komunikace mezi ARTable a HoloLens. Obě platformy jsou postaveny na rozdílných operačních systémech, ARTable je postaven na Ubuntu spolu s ROSem (Robotický Operační Systém), kdežto HoloLens běží pod Windows 10. Tento problém řeší nástroj ROSu – *rosbridge*. Ten poskytuje JSON API rozhraní k funkcionalitám ROSu pro externí programy jiných operačních systémů. Systémy si tedy prostřednictvím nástroje *rosbridge* vyměňují zprávy, čímž spolu komunikují. Po úspěšném zprovoznění komunikace je potřeba systémy vzájemně zkalibrovat, jinými slovy sladit počátky souřadných soustav. ARTable má tento počátek po kalibraci umístěn v levém dolním rohu stolu. Brýlím je tedy nutné sdělit, kde se tento roh nachází. Toho je docíleno pomocí detekce markeru z obrazu přední kamery brýlí. Polohu takto detekovaného markeru jsou brýle schopny uložit prostřednictvím prostorové kotvy (angl. spatial anchor), což zajistí, že nebude nutné kalibrovat brýle pořád dokola při opětovném spouštění aplikace. Prostorová kotva rovněž slouží jako „rodič“ všech dalších objektů ve scéně.

Vizualizace je realizována formou animace, která se sestaví „za chodu“ z předem připravených C# tříd. Každá třída reprezentuje jednotlivou instrukci systému ARTable. Díky tomuto přístupu je možné poskládat na straně systému ARTable jakoukoliv variaci pro-

gramu z existujících instrukcí, která by se měla v brýlích korektně vizualizovat. Brýle kreslí v rámci vizualizace virtuální objekty a virtuální robotickou ruku přímo do scény. Ta objekty přemísťuje, pokládá na stůl a celkově „animuje“ to, co by udělal reálný robot. Vizualizace je ovládatelná hlasem nebo tlačítky promítaného rozhraní na stole. Celý proces je rovněž komentován řečovým syntetizátorem brýlí v angličtině.

Mimo vizualizaci programů byl implementován interaktivní pomocník, který uživatele pomocí řečového syntetizátoru a virtuální ruky navede na správné naprogramování instrukcí. Byla implementována podpora pouze pro dvě základní instrukce – *zvedni z polygonu* a *polož na pozici*. Přínos brýlí je také demonstrován už jen možností vidět to, co vidí robot. Stávající 2D promítané rozhraní vykresluje ohraničení okolo robotem detekovaných objektů položených na stole. Senzory však tyto objekty dokáží detekovat i v prostoru, což lze snadno vizualizovat prostřednictvím 3D ohraničení objektů v brýlích.

Implementované řešení bylo řádně otestováno. Experimentu se zúčastnilo 12 zájemců, přičemž byli rozděleni do dvou skupin. Jedna skupina nastavovala parametry pro základní program typu „zvedni z místa A a polož na místo B“ bez použití brýlí. Dále dostala za úkol pochopit program „zvedni z podavače a polož na stůl“ s již nastavenými parametry, opět bez použití brýlí. Druhá skupina dělala to samé s tím rozdílem, že používala brýle, a tedy i interaktivního pomocníka a vizualizační systém. Všem účastníkům byly měřeny časy dokončení úkolu a počet nutných zásahů z mé strany. Dále byl každý požádán o vyplnění dotazníku „System Usability Scale“ (SUS), který měří použitelnost daného systému. Veškeré hodnoty byly porovnány. Následně si všichni vyzkoušeli brýle Microsoft HoloLens v rámci vizualizace složitějšího programu, kde byli posléze požádáni o vyplnění dotazníku zkoumajícího použitelnost těchto brýlí. Z výsledků vyplynulo, že vizualizace programu v brýlích zkrátila čas potřebný k jeho pochopení o 50 %. Naopak použití brýlí pro programování robota tento čas prodloužilo o téměř 35 %. Skóre ze SUS dotazníku se pro obě skupiny nijak zásadně nelišilo, při použití brýlí byla jeho hodnota nižší – 67.5, což se ale stále pohybuje zhruba v průměru standardních hodnot. Největším problémem bylo nízké zorné pole brýlí a jejich váha, která zapříčinila značné nepohodlí na hlavě. Uživatelé naopak ocenili ovládání hlasem a vizualizaci samotnou.

User Interface for ARTable and Microsoft HoloLens

Declaration

Hereby I declare that this master's thesis was prepared as an original author's work under the supervision of Ing. Michal Kapinus. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....

Daniel Bambušek

May 23, 2018

Acknowledgements

I would like to give my thanks to supervisor of my thesis, Ing. Michal Kapinus for his guidance, patience and dedicated time. I would like to also thank all participants who took part in the user experience testing of this work.

Contents

1	Introduction	3
2	Augmented Reality	4
2.1	AR Definition	4
2.2	Virtual vs. Augmented vs. Mixed Reality	4
2.3	Major AR Components	5
2.4	Types of AR Devices	7
3	Microsoft HoloLens	11
3.1	Key Advantages and Disadvantages	11
3.2	Interaction with Device	12
3.3	Hardware Details	12
3.4	Possible Future Usages	13
4	The ARTable – Augmented Reality Collaborative Workspace	15
4.1	ARTable Description and Motivation	15
4.2	PR2	16
4.3	Used Sensors	17
4.4	ROS - Robot Operating System	18
5	Proposed Augmented Reality-based User Interface	19
5.1	Existing Solutions	19
5.2	Motivation for Use of the Microsoft HoloLens in the ARTable Setup	20
5.3	User Interface for Programs Visualization	20
5.4	User Interface for Robot Programming	21
5.5	Robot Learning Using Augmented Reality and Other Possible Use Cases . .	22
6	Implementation	24
6.1	Integration of Microsoft HoloLens into the ARTable System	24
6.2	Visualization of Robot’s Perception	26
6.3	User Interface for Robot Programs Visualization	27
6.4	Interactive Helper for Robot Programming	30
6.5	Application Components	30
7	Experiments	34
7.1	Experiments Description	34
7.2	Results	38
7.3	Suggestions for Future Work	42

8	Conclusions	44
	Bibliography	45
	Appendices	48
	List of Appendices	49
A	Contents of the DVD	50
B	System Usability Scale and Custom Questionnaires	51

Chapter 1

Introduction

As industrial collaborative robots are getting more affordable, it is likely that small and medium enterprises will soon adopt such robots in order to increase productivity. However, in such enterprises, production batches are smaller and products may be customized for a specific contract. This requires reprogramming robots for particular tasks, which could be challenging due to necessity of robot-specific knowledge. Thus it would be beneficial to enable ordinary-skilled worker to program these robots easily. Therefore a prototype of a human-robot collaborative workspace – the ARTable was created, which presents a novel approach to programming robots based on cognition, spatial augmented reality and multimodal input and output [17].

Programming of collaborative robots in this approach is based on setting up parameters for specific instructions which forms the whole program. These instructions are described only with text identifiers and as experiments showed up, users had difficulties in orienting in such program topology. There were uncertainties of how instructions follow, what exactly will specific program do or what is expected from user to program.

This work aims to clarify user’s uncertainties in programming robots by providing program visualization system in spatial augmented reality with use of Microsoft HoloLens. Being able to see what will happen when specific program runs, what specific instruction means and what is expected from the user to program should rapidly help in understanding of the whole ARTable system. Further on, it extends capabilities of current 2D projected user interface by visualizing valuable spatial information through the headset.

Chapter 2 introduces basics of augmented reality (AR), compares virtual reality with augmented, describes required components for the complete AR system and presents most common types of devices capable of AR. The Microsoft HoloLens is introduced in Chapter 3, its advantages, disadvantages or possible use cases are described. Chapter 4 describes mentioned ARTable and its core components, both hardware and software. In Chapter 5 I propose an AR-based interface for the HoloLens and the ARTable, which should improve effectiveness of a collaboration of the user with the robot. Chapter 6 covers implementation part of proposed interface and Chapter 7 evaluates it.

Chapter 2

Augmented Reality

This chapter covers basic concepts of the augmented reality (AR), introduces its definition (Section 2.1) and compares basic differences between virtual, augmented and mixed reality (Section 2.2). Section 2.3 analyzes three major components of the complete AR system – tracking, registration and visualization. Last but not least, section 2.4 describes most common AR devices and display technologies.

2.1 AR Definition

Furht et al. [6] defines AR as “a real-time direct or indirect view of a physical real-world environment that has been enhanced/augmented by adding virtual computer-generated information to it”. AR aims to simplify the user’s life by bringing virtual information to his immediate surroundings or to any indirect view of the real-world environment, such as live-video streams. AR enhances the user’s perception and interaction with the real world [6].

First use of the term “Augmented Reality” dates back to 1950s, since then this technology has grown on interest and nowadays could be considered as a next step to a modern future. AR can be used in various fields, from army (head-up displays) to entertainment (AR games for handheld devices) and its possibilities are endless.

2.2 Virtual vs. Augmented vs. Mixed Reality

Virtual reality replaces real world with a virtual one using 360-degree video, photospheres or computer-generated environments. The goal is to completely immerse a user into the virtual world. Bishop et al. [2] defines virtual reality as a “real-time interactive graphics with three-dimensional models, when combined with a display technology that gives the user immersion in the model world and direct manipulation.”

On the contrary, augmented reality just adds computer-generated information into the real world instead of completely replacing it. The goal is to enhance the existing environment with some computer-generated graphics. However mixed reality mixes real world with virtual one and creates new environments where physical and digital objects can coexist and interact with each other.

Milgram [18] describes a taxonomy that identifies how augmented reality and virtual reality are related (see fig. 2.1). He defines the case at the left of the continuum – Real Environment, as any environment consisting solely of real objects that includes whatever might

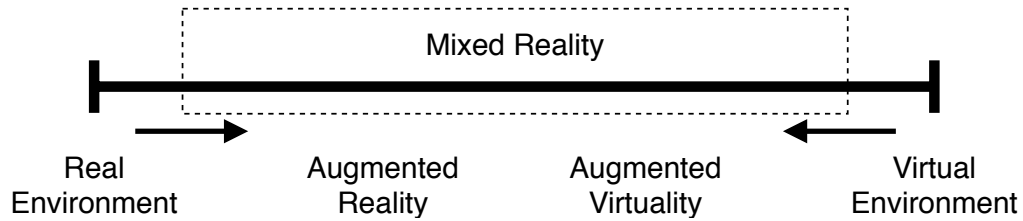


Figure 2.1: Milgram's Reality-Virtuality Continuum

be observed when viewing a real-world scene. The case at the right – Virtual Environment, is an environment consisting solely of virtual objects, examples of which would include conventional computer graphic simulations, either monitor-based or immersive. Within this relation he defines generic Mixed Reality environment as one in which real world and virtual world objects are presented together. Besides displaying real world together with virtual one, mixed reality also involves environmental input, spatial sound and location [28].

2.3 Major AR Components

To have a complete AR system, at least three major components are required – a tracking component, a registration component and a visualization component. In such system, a user's position is tracked and poses of the real world objects are registered with the virtual content that is afterwards presented to the user through visualization device [21, Chapter 1].

Tracking

To display virtual objects registered to real objects in real-world space, the position and orientation of the AR display relative to the real objects must be known. This is accomplished with the tracking technique, which is a process of locating a user in an environment. It continuously measures his position and orientation. Various entities can be tracked, most commonly a user's head, eyes or limbs, or any other object in the scene [21, Chapter 3].

Different techniques of tracking will be used depending if the user is indoor or outdoor. A good tracking option for outdoor environments is considered GPS (Global Positioning System). This technique computes position of the Earth's surface from measured time of flight of coded radio signals emitted by at least four satellites in Earth orbit. The accuracy can vary from 1 to 100 m, depending on the number of visible satellites, the circumstances of signal reception, and the quality of the receiver. Use of differential GPS can assure higher accuracy. For a rotational motions – gyroscopes can be used; for a translational motions – accelerometers can be used. Both are nowadays common part of mobile devices.

Accuracy of aforementioned techniques is usually not sufficient enough for AR. Thus both indoor and outdoor tracking combines vision-based approaches. Based on particular technique, cameras, stereo-cameras or depth cameras are used. Those techniques differ whether they use markers or features for object detection and tracking. Markers are known patterns placed on the surfaces of target objects, designed to make detection as easy and reliable as possible. Marker-based tracking is often used in indoor or preset environments. Natural feature tracking can be also used, however it typically requires better image quality and more computational resources. It searches for points of interest in the image, typically edges or corners, which are used for object tracking [21, Chapter 3].

Well known algorithm SLAM (Simultaneous Localization and Mapping), used in robotics field to localize a robot based on the map which it creates by observing its environment, can be also applied to a camera mounted on the user in an AR context [3]. Lastly, fusion methods that uses different sensors together can be used in order to achieve high accuracy.

Registration

Registration is a process that maps the virtual objects onto tracked physical real-world objects. It converts the poses from tracking into the coordinate system of the rendering application. For proper registration, calibration of the AR device's components needs to be done. This involves camera and display calibration.

Camera calibration is the process of estimating the distortion coefficients, intrinsic and extrinsic parameters of a camera using images of a special calibration pattern. Most popular type of a calibration pattern is a checkerboard. The intrinsic parameters include the focal length, the optical center and the skew coefficient. The extrinsic parameters consist of a rotation matrix and a translation matrix [1]. Proper calibration corrects a lens distortion (pincushion or barrel distortion effects).

Calibrated camera is sufficient for presenting registered AR overlays on a video see-through display. However, for an optical see-through display, where head tracking is used instead of camera tracking, the display calibration needs to be done. Usually, the system displays calibration patterns, where the user is asked to align a physical structure, for example his forefinger, with that specific pattern [21, Chapter 5].

Visualization

Visualization determines how virtual information should be shown. For a realistic feeling, visual coherence needs to be done, which means that real and virtual objects needs to be combined in a way that the virtual objects are indistinguishable from the real world. In order to achieve such coherence, following cues of computer graphics must be respected: *relative size* (farther objects are smaller), *relative height* (distant objects have their base higher), *perspective* (parallel lines converge as they extend farther), *surface detail* (closer objects are more detailed than farther ones), *atmospheric attenuation* (distant objects appear more blurred), *occlusion* (closer objects obscure farther objects), *shading* (illumination of objects according to the position and orientation of light sources) and *shadows* (objects cast shadows on other objects).

To fulfill these cues, geometric and photometric registration must be obtained. With the knowledge of the geometry of the real scene, occlusion of virtual objects that are rendered behind real ones can be solved. Photometric registration is much more complicated, as the transport of light between real and virtual objects must be simulated. The simplest set of techniques deals only with shadows between virtual and real objects. For advanced illumination effects, model of the environment illumination is needed, from which the virtual objects are illuminated. Number of light sources and their characteristics have a strong influence on the computational complexity. Their number for AR can be limited by assuming that all light sources are distant and can be treated as external to the scene. To acquire proper shadows, standard shadow simulation techniques can be used, such as *shadow mapping* or *shadow volumes* [21, Chapter 6].

When decided on how to draw virtual objects to scene, what to draw needs to be answered. Principle, where information is attached to real-world objects is called *situated visualization*. Examples of this principle include annotation and labeling techniques of

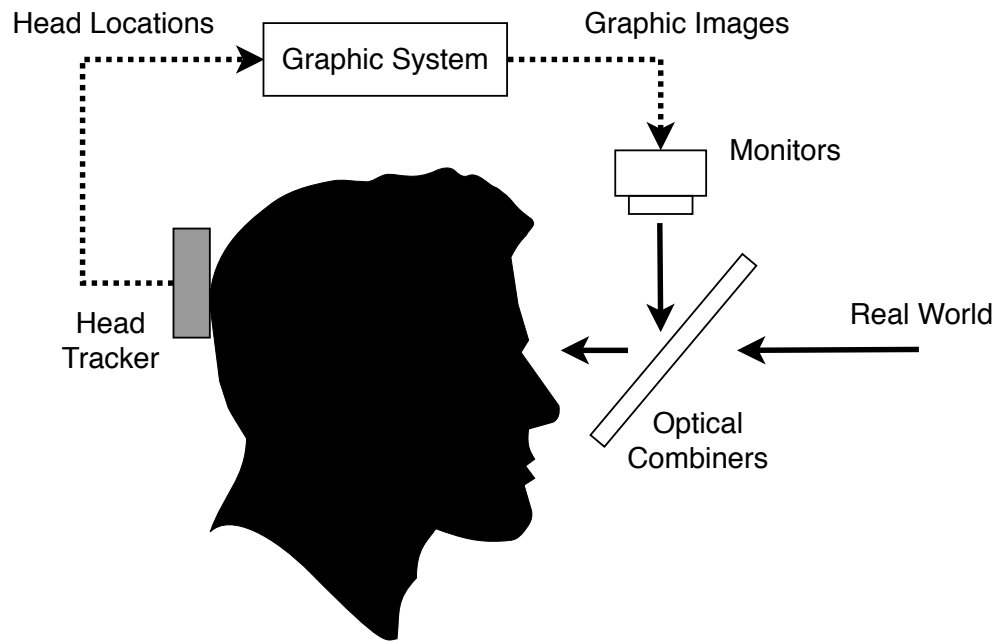


Figure 2.2: Optical see-through display conceptual diagram.

points of interest (historical buildings, model of a human head, etc.) or use of X-ray visualization that uncovers hidden or otherwise imperceptible structures (like buildings, car hoods or human body) [21, Chapter 7].

An interesting part of AR is a *Diminished reality*. While most applications of AR are concerned with the addition of virtual objects to a real scene, *diminished reality* describes the conceptual opposite – it removes real objects from a real scene.

Devices capable of visualization of virtual information will be described further in section 2.4.

2.4 Types of AR Devices

As human vision is the most effective sense in receiving surrounding information, many of today's AR devices focus mainly on providing augmentations to a user's visual perception. Some devices combine visual augmentations with spatial sound. According to augmentation methods, there are three types of displays – *optical see-through*, *video see-through* and *spatial projection*.

Optical see-through (OST) displays use an optical element that is partially transmissive and partially reflective to achieve the augmentation of the real world. Simple example of such element is a half-silvered mirror, which lets a sufficient amount of light from the real world pass through in order to overlay it with reflected computer-generated images that are projected from monitors placed above the mirror (see Figure 2.2).

Video see-through (VST) displays capture the real world with a video camera. Such captured real-world images are combined with computer-generated images in a video compositor. This is often achieved by simply drawing the computer-generated elements on top of the real-world images. Combined images are then presented to the user using a conventional viewing device (see Figure 2.3).

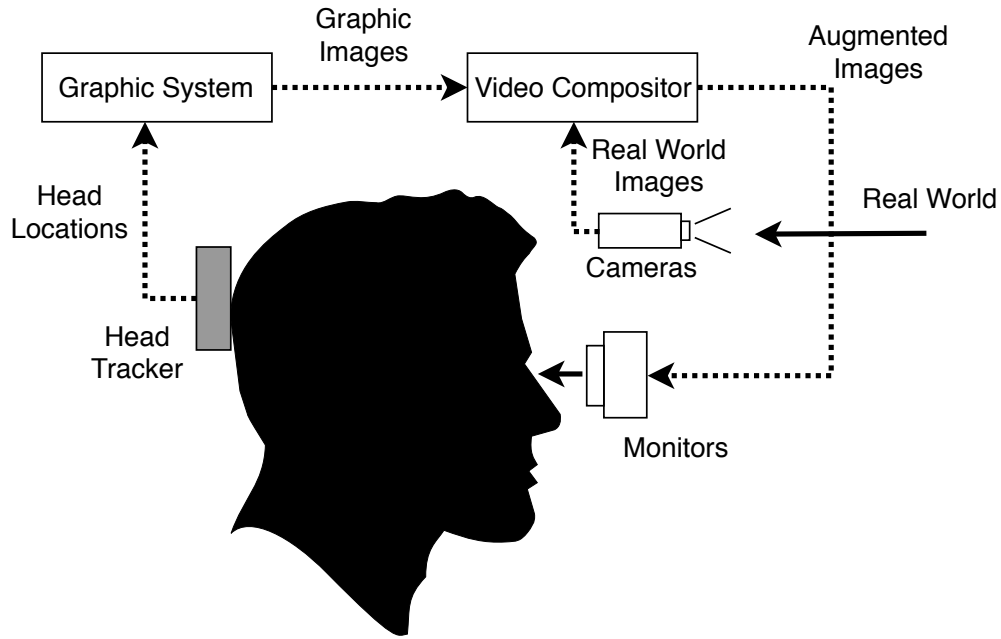


Figure 2.3: Video see-through display conceptual diagram.

Spatial projection casts computer-generated images directly onto real-world objects using a light projector. Conceptual diagram can be seen in Figure 2.4.

Handheld Displays

Smart phones and tablets are probably the most popular platform for AR. Its back-facing camera can provide a video see-through experience. These devices houses both the actual display and the camera rigidly mounted in a casing. Thus the transformation from display to camera can be precalibrated. Tracking of the device's pose in the world will be performed through the camera in most cases, which can be improved with built in accelerometers and gyroscopes [21, Chapter 2].

Nowadays, there are plenty of AR applications for handheld displays. Most of them focuses on entertainment, like the widely known Pokémon GO¹, developed by Niantic for iOS and Android devices. In this game, players are walking in the real world, trying to catch pokémon creatures that are augmented into the real-world environment. An example of this application in use can be seen in Figure 2.5. All applications don't have to necessarily target the entertainment industry, for example, Google Translator has a feature of real-time signs translation from foreign languages.

Head-Mounted Displays

While handheld displays brought the idea and potential of AR technologies to everybody's awareness, head-mounted displays (HMD), through advanced technological and ergonomic innovations, might represent the next wave of AR. They can differ based on their mounting options:

¹<https://www.pokemongo.com/>

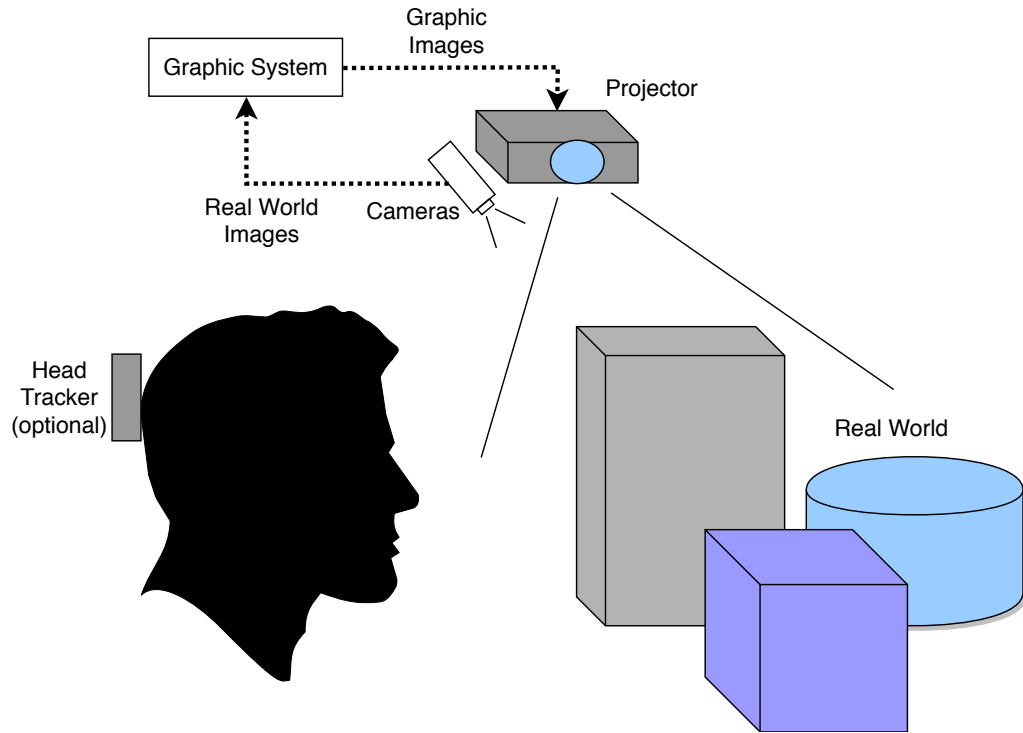


Figure 2.4: Projected spatial AR conceptual diagram. Projector casts images directly onto the real-world objects.

- **Helmet-mounted display** – like Rockwell Collins SimEye that simulates head-up display functions for aircraft trainers.
- **Clip-on display** – like Google Glass.
- **Visor display** – like Microsoft HoloLens or Meta 2.

Based on the method of augmentation, HMD differs to optical see-through and video see-through.

Optical see-through HMD requires an optical combiner placed in front of the user's eyes to mix the real world with computer-generated images. Most recent displays suffer from relatively small field of view. They cover only small portion of a human field of view, which can really break apart desired illusion of joining the real world with the virtual one. Another problem is how to balance the brightness and contrast. With too much incoming light from real world, virtual content won't be visible well. Important factor for OST is to keep the headsets as light as possible, while maintaining performance and widening field of view. Most recent representatives of this technology are for example Microsoft HoloLens (described in detail in Chapter 3) or Meta 2.

Unlike the Microsoft HoloLens, the Meta 2 requires a connection to a traditional computer to function and is designed to be used in a stationary location. Its sensor inputs and image processing are handled by the connected PC's processors. It uses a semi-spherical large combiner optics system, which is able to extend the field of view up to 90 degrees. Meta 2 offers a collection of sensors thanks to which is capable of creating an environment map to allow interaction with the real world [19].



Figure 2.5: **Left:** Handheld display – example of the *Pokémon GO* AR game for smart phones. **Right:** Optical see-through head-mounted display – Meta 2 in use.

Source: <https://www.enkronos.com/why-you-should-start-using-augmented-reality-ar-and-gamification/>, <http://www.metavision.com/>

Video see-through HMD adds one or more video cameras to a non-see-through HMD. These cameras serve as a replacement of human sight. This technology has some advantages over the OST HMD. Virtual content does not suffer from transparency and ghost-like appearance. In VST, pixel-accurate registration is easier to achieve. On the other hand, this technology is not suitable for dangerous work because when the display turns off due to some error, the user becomes practically blind. Latency could also cause problems.

Projected Displays

Projectors can be used to create spatial augmented reality without any explicit displays. With this approach, the projector casts images directly onto the real objects, altering their appearance to the naked eye. The projection cannot change the shape of the object, but adds surface details, texture, shadows, and shading, and even the impression of dynamic behavior, if animated content is projected. As long as the real world is static, spatial AR does not require any tracking. Only the relative position of the projector to the objects, and the geometry of the objects needs to be known. It is possible to use multiple projectors for better spatial coverage [21, Chapter 2].

Chapter 3

Microsoft HoloLens

Microsoft HoloLens is a headset for mixed reality, developed by Microsoft. It's wireless, runs Windows 10, has precise spatial mapping, and is capable of 3 ways of interaction – gaze, gestures and voice. Device draws computer generated holograms into the real world and has a potential to be used in many scenarios. Developer version was released on 30 March 2016 in United States and Canada. Later in October 2016, HoloLens hit markets in the United Kingdom, Australia, New Zealand, Ireland, France and Germany [11]. Headset is shown in Figure 3.1.

Section 3.1 outlines main advantages and disadvantages of the HoloLens. Sections 3.2 and 3.3 describes headset's hardware details and possible ways of interaction with it. Possible use of the HoloLens is discussed in section 3.4.

3.1 Key Advantages and Disadvantages

The main innovation of the Microsoft HoloLens over previously available augmented reality headsets is considered to be precise position tracking and spatial mapping of the surrounding environment. The inputs from sensors are combined in the Holographic Processing Unit to build and maintain a model of the surroundings. This allows to create immersive mixed reality experiences in which the position of virtual objects is fixed in space with centimeter scale precision, and they naturally react to collisions with the physical world or remain at their location over multiple sessions [22]. Thanks to that, user shouldn't get any nausea



Figure 3.1: **Left:** Microsoft HoloLens – headset for mixed reality. **Right:** Visualization of limited field of view of the headset.

Source: <https://www.microsoft.com/en-us/hololens>, <https://www.theverge.com/2015/6/18/8809323/microsoft-hololens-field-of-view-kudo-tsunoda>

sensation from virtual reality. Another plus could be considered excellent spatial sound system which can increase user's AR experience.

Another main feature is sharing holograms, where users can see same holograms on multiple devices. Microsoft's development toolkit for Unity (*MixedRealityToolkit-Unity*¹) provides support for such shared experiences in a form of world anchors. One of the devices establishes an anchor at a specific location and communicates its position within its model of the room to all other devices. The anchor is then used as a reference point for the coordinate system, in which the shared holographic objects are located at the same position to each user [22].

However, device is suffering with really small field of view – 30° x 17.5° degrees (aspect ratio 16:9) [13]. Which can lead to that the user will spend some time finding his holograms placed all over the room. Limited field of view can be seen in Figure 3.1. Next minus is high price. Currently, HoloLens commercial version costs \$5,000 and developer version is for \$3,000². Lastly, battery life (2-3 hours of active use) is not a miracle at all.

3.2 Interaction with Device

For interaction with device, you can use 3 ways – gaze, gestures or voice. Hand gestures allow users to take action in mixed reality. Interaction is built on gaze to target and gesture or voice to act upon whatever element has been targeted. Currently, there is support for two core component gestures - Air tap and Bloom. Air tap is a tapping gesture with the hand held upright, similar to a mouse click or select on a specific UI element after targeting it with gaze. Bloom is a gesture for invoking the Windows Start Menu. These two basic gestures are visualized in the figure 3.2. Air tap can be performed also as a gesture for tap and hold, so user can move with holograms, scroll page or zoom in. It is important to use the gestures in a range that the gesture-sensing cameras can see appropriately (roughly from nose to waist, and between the shoulders) [26].

Besides gestures, HoloLens supports voice commands. User can achieve same things with them, as with gestures, so there are commands such as “select” (equivalent of air tap), “place” (equivalent of air tapping to place a hologram), “face me” (to turn hologram your way), “bigger/smaller” (to resize hologram), etc. User can communicate with Cortana, which is virtual assistant created by Microsoft for Windows 10. In dictation mode, voice can be used for typing, which can greatly save time, instead of air tapping the virtual keyboard.

Optional way of interaction with holograms is the clicker. It's a small device separated from the headset itself, designed to fit into the human wrist between thumb and forefinger.

3.3 Hardware Details

Optical part of the device contains see-through holographic lenses, followed up with 2 HD 16:9 light engines, that can produce up to 2.3 million total light points of holographic resolution. Device has various types of sensors, four cameras for environment understanding, two on each side, one depth camera, 2 Mpx photographic video camera, four microphones, ambient light sensor, and lastly, inertial measurement unit (IMU) which includes

¹<https://github.com/Microsoft/MixedRealityToolkit-Unity>

²<https://www.microsoft.com/en-us/hololens/buy>

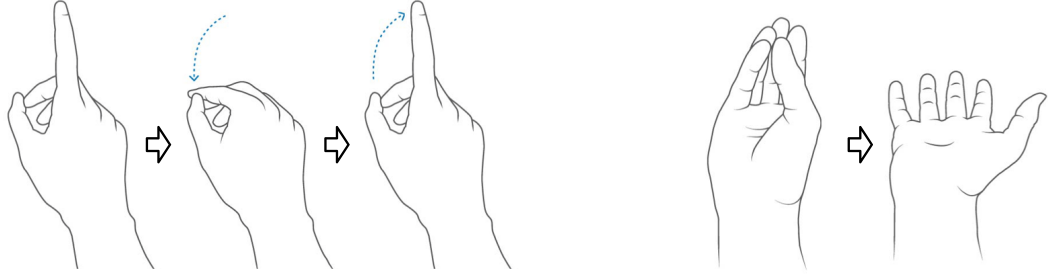


Figure 3.2: Two basic interaction gestures which Microsoft HoloLens recognizes. **Left:** *Air tap*. **Right:** *Bloom*.

Source: <https://support.microsoft.com/cs-cz/help/12644/hololens-use-gestures>



Figure 3.3: HoloLens hardware. **Left:** Optics. **Middle:** Sensors. **Right:** Motherboard.

Source: https://developer.microsoft.com/en-us/windows/mixed-reality/hololens_hardware_details

an accelerometer, gyroscope, and magnetometer [10] [27]. For optical and sensor part, see Figure 3.3.

Computing performance is ensured by an Intel 32-bit processor (1GHz) with TPM 2.0 support, followed up with custom-built Microsoft Holographic Processing Unit (HPU 1.0) and 2GB RAM. Internal storage can hold up to 64GB (flash memory). Battery lasts for 2-3 hours of active use or up to 2 weeks on standby mode. Device is fully operational when charging [27].

Device has standard features, such as 802.11ac Wi-Fi, Bluetooth 4.1 Low Energy (LE), micro USB 2.0 port, built-in audio speakers and 3.5mm jack. HoloLens runs on Windows 10 operating system [27].

3.4 Possible Future Usages

HoloLens has a potential to take place in many real-world scenarios, from teaching at universities to designing a new cars. Since its official release, HoloLens already has been adopted by industries keen to change the way they work [7]. Currently, many companies has already shown proof of concept, that use of the HoloLens in their industry might work and could ease their everyday tasks.

Main use case is visualizing holograms and specific information in 3D. User can observe models of some real scene, buildings or vehicles, from which he can get a way better perception and more information, than from 2D photography or computer. This use case could find its place in architecture, where architects could easily see how a new structure would fit into its environment; army, where soldiers would plan actions better with informations from 3D model, rather than from 2D photographs; and many others. Even in home design would headset find its usage. Customers could design their own apartments with virtual



Figure 3.4: Use cases for Microsoft HoloLens in real-world environments. **Top left:** Education purposes. **Top right:** Home design. **Bottom:** Game industry – the *Fragments*.

Source: <http://jewishbusinessnews.com/2015/07/09/hololens-may-be-used-in-research-on-human-testing-medication/>, <https://blogs.microsoft.com/transform/feature/lowes-innovates-with-sci-fi-to-expand-and-enhance-its-microsoft-hololens-mixed-reality-kitchen-design-experience-for-customers/>, <https://www.microsoft.com/en-us/hololens/apps/fragments>

furniture to see how it would fit into the apartment before they buy any real furniture (see Figure 3.4). Volvo is using HoloLens to change the experience of buying a car. They allow customers to visualize how safety features work with use of the HoloLens. They also allow to adjust color and fabric of a car and visualize it as 3D holograms, which is more prominent than flat computer images [7].

Another use case for the HoloLens, is to augment skilled workers and technicians in the field. Instructions and data can be overlaid on the screen and if the system has good object recognition software then the headset can recognize what needs to be done and assist the user. This is especially helpful when the user is working in an area with too many systems for them to effectively learn. Rather than memorizing a series of pictures and instructions, the headset can show how things need to be manipulated to achieve the desired goal [24].

In the field of medicine, surgeons can have something close to X-Ray vision that makes it possible to see inside a patient [7]. Detectives and police could find HoloLens useful for crime investigations when recreating crime scene or registering evidences. Universities could use HoloLens for a new and more fun way of education, where students could observe models of human organs, planets in space or wild animals (Fig. 3.4). The mixed reality with Microsoft HoloLens can be used for gaming and entertainment. Currently, there are already available games such as *RoboRaid*³ – a first-person shooter that changes user’s room into the battlefield of robot invasion or *Fragments*⁴ – a game where you become the detective solving a crime (Fig. 3.4).

³<https://www.microsoft.com/cs-cz/hololens/apps/roboraid>

⁴<https://www.microsoft.com/cs-cz/hololens/apps/fragments>

Chapter 4

The ARTable – Augmented Reality Collaborative Workspace

In the following chapter, the ARTable and its components will be introduced. Section 4.1 describes what the ARTable is, the motivation for it, introduces its supported programs and core components. These components will be further described in detail in sections 4.2 and 4.3. Last but not least, in section 4.4 will be described the software part of the ARTable system – ROS (Robot Operating System).

4.1 ARTable Description and Motivation

The term “ARTable” stands for a prototype of an augmented reality-based collaborative workspace (Fig. 4.1), created by the research group Robo@FIT¹ at Brno University of Technology. The prototype focuses on small and medium enterprises where it should enable ordinary-skilled workers to program a robot on a high level of abstraction and perform collaborative tasks effectively and safely [17]. It aims to show possibilities of the collaboration of a human and robot in the near future. The source code and technical documentation of this prototype is available at github².

Materna et al. [17] claims that with the emergence of affordable industrial collaborative robots it seems likely that small and medium enterprises will widely adopt such robots in order to achieve higher precision for specific tasks, free experienced employees from monotonous tasks, and increase productivity. He says that would be beneficial for such enterprises to enable ordinary-skilled workers to program robots easily, without robot-specific knowledge, due to the fact that such enterprises may produce various products for which they may want to customize their robots without a help of an expert. Thus the ARTable was created.

Core Components

The ARTable setup uses the PR2 robot (described in section 4.2), which is located behind the table (from user’s perspective) and serves as a demonstrator of a near-future collaborative robot. However, the setup is robot independent.

¹<http://www.fit.vutbr.cz/research/groups/robo/index.php.en>

²<https://github.com/robofit/artable>

The table itself has a touch-sensitive layer onto which an interface is projected using Acer P6600 projector mounted on the construction above the table. The projected interface contains various elements to visualize state of the robot and current task. Together with the touch-sensitive layer, the table serves as the main source of input for the system and feedback for the user. Another possible way of passing input into the system is by direct manipulation with robotic arm. The table is complemented with speakers on each sides, which serves as supplementary source of feedback for the user.

The system uses several sensors mounted on a stable tripods, which will be described in section 4.3. Each tripod has its own processing unit (Intel NUC) where the projector and sensors are connected. This units are connected to the central computer. AR codes, attached to objects, are used for better tracking. They are also used for calibration of the whole system. Regarding to software setup, the ARTable uses the Robot Operating System (ROS), further described in section 4.4.

Supported Programs and Instructions

Programs are sets of instructions which are collected into blocks. These instructions are linked together based on the result of specific instruction (success or failure). Currently, the system supports parametric instructions as *pick from polygon* (to pick an object from specified polygon on the table), *pick from feeder* (to pick an object from gravity feeder), *place to pose* (to place previously picked object to selected place on the table) and *apply glue* (to simulate gluing). User sets parameters in these instructions, such as object type to pick, place position of picked object or robots gripper position for picking objects from feeder.

Besides mentioned instructions, there are also few non-parametric instructions: *get ready* (which moves robot arms to a default position), *wait for user* (which pauses the program execution if user moves away from the table) and *wait until user finishes* (which pauses the program execution until user finishes current interaction with objects on the table).

4.2 PR2

PR2 (Personal Robot) is a robot developed by Willow Garage, it runs ROS and is mainly used for research purposes. Original intent was to use the robot as a collaborator for home environment. The robot itself consist of the omni directional base, telescopic torso, two arms with eight degrees of freedom (DOFs) and the head. PR2 can be seen in Figure 4.1.

On the robot base are attached four Casters wheels enabling omni directional movement and Hokuyo UTM-30LX laser scanner. The base is quipped with two Quad-Core i7 Xeon processors, 24 GB memory, externally removable hard drive with capacity of 1.5 TB and 500 GB internal hard drive. The robot can reach up to 1 m/s speed [29]. However in the ARTable setup, the robot is always stationary.

Height range of the torso from floor to top of head is 1330 mm to 1645 mm. Two arms, attached to torso, are composed of arm with four DOFs, wrist with two DOFs and gripper with one DOF. Arm payload is 1.8 kg. Forearm is equipped with camera and gripper is equipped with accelerometer and fingertip pressure sensor [29].

The head is equipped with 5-megapixel color camera, wide-angle color stereo camera, narrow-angle monochrome stereo camera and LED texture projector. Above the shoulders is placed tilting Hokuyo UTM-30LX laser scanner [29]. Head is complemented with Microsoft Kinect 1.0, which is used in ARTable setup for objects detection and tracking.



Figure 4.1: Setup of the human-robot collaborative workspace – the ARTable. In this example the user sets program parameters using robot’s arm and gestures. Courtesy of Michal Kapinus.

4.3 Used Sensors

The ARTable setup uses two Kinect 2.0 sensors fixed on a stable tripod on each side of the table and one Kinect 1.0 sensor fixed on the robot’s head. These sensors are used for object detection and tracking. Objects are detected with use of the sensors HD cameras and AR codes attached to the objects. All sensors try to find all AR codes in scene and calculate its position in the real world. Final position of AR code is determined with use of weights, which are assigned with respect to distance of the AR code from specific camera. This means, that Kinect to which are objects closer, has a higher decision weight.

The Kinect 1.0 uses a structured light principle, which is a method of projecting a known pattern onto the scene and inferring depth from the deformation of that pattern. So the depth map is constructed by analyzing a speckle pattern of infrared laser light [15]. The Kinect 2.0 uses a Time-of-Flight camera which constantly emits infrared light in order to measure the time this light takes to travel from the camera to the object and back. With this principle the depth map is constructed [25]. Main improvement of Kinect 2.0 over 1.0 is increased resolution of RGB camera (Kinect 1.0 has 640 x 480 pixels resolution, Kinect 2.0 has 1920 x 1080 pixels resolution) [25].

4.4 ROS - Robot Operating System

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms [30].

System was developed by Willow Garage, is open source and has a large community, which develops new tools, packages and libraries, which publishes and maintains. Currently, the system is under maintenance of the *Open Source Robotics Foundation*³. As the system is still being developed, new distributions are gradually released primarily targeted for Ubuntu operating system. The latest LTS⁴ version is *Kinetic Kame*. However the ARTable setup uses an older version – *Indigo Igloo*.

Architecture of ROS is based on the existence of nodes, which are mutually communicating processes doing specific task. Nodes can communicate by messages or services [32].

Nodes

A node is a process that performs computation and sub-action of a whole application. Nodes are combined together into a graph and communicate with one another using streaming topics or RPC⁵ services. These nodes are meant to operate at a fine-grained scale; a robot control system will usually comprise many nodes. For example, one node controls a laser range-finder, one node controls the robot's wheel motors, one node performs localization, one node performs path planning, and so on [32].

The use of nodes in ROS provides several benefits to the overall system. There is additional fault tolerance as crashes are isolated to individual nodes. Code complexity is reduced in comparison to monolithic systems. Implementation details are also well hidden as the nodes expose a minimal API to the rest of the graph and alternate implementations, even in other programming languages, can easily be substituted [32].

The ARTable has many such nodes. The main is *art_brain*, which is a “brain” of a whole system. Next, there are nodes for calibrating the cameras (*art_calibration*), AR codes detection (*art_arccode_detector*), projecting the user interface (*art_projected_gui*), and others.

Messages and Services

Nodes communicate with each other by publishing messages to topics [31]. Topics are named buses over which nodes exchange messages. Topics have its publishers and subscribers. This way of communication between nodes is only unidirectional. Example of such communication could be transition of data from camera, when a node – *publisher*, which implements camera's drivers publishes captured data by camera to specific topic from which another node – *subscriber*, subscribes the data for another processing.

For a bidirectional communication, there is a possibility to use a *Service*. Services are communication of a type request and reply, which is done by a pair of messages: one for the request and one for the reply. A providing ROS node offers a service under a string name, and a client calls the service by sending the request message and awaiting the reply [33].

The ARTable uses some basic ROS messages and defines its own messages and services.

³<https://www.osrfoundation.org/>

⁴Long Term Support

⁵Remote Procedure Call

Chapter 5

Proposed Augmented Reality-based User Interface

The goal of this thesis is to use the potential and possibilities of Microsoft HoloLens, create graphical user interface (projected through HoloLens) which will extend already existing one (projected by projector onto the table) and integrate it to the ARTable system. Firstly, background and related works to the ARTable and use of the HMD in industrial robotics is analyzed in section 5.1. The motivation for use of augmented reality in the ARTable system is outlined in section 5.2. In sections 5.3 and 5.4, I introduce a proposal of an interface for visualization and programming guidance of the ARTable programs. Section 5.5 describes a proposal of how could augmented reality help in learning robots and covers some other possible beneficial use cases that could enhance the work with the ARTable.

5.1 Existing Solutions

There are several approaches of end-user robot programming. One of the main techniques are Programming by Demonstration and Visual Programming. Both approaches are combined in a robot programming system – *Code3* [12], which enables a non-roboticist programmers to create a complex robot programs using landmarks detection, programming by demonstration and visual drag-and-drop programming interface that lets users define control flow logic of their programs.

Usage of AR in industrial robotics may lead to decreased workload throughout robot control by showing useful information to users [23]. A concept and architecture for programming industrial robots with use of AR is presented in [8]. In [16], authors developed an application that augments an industrial robot for shop floor tasks and evaluated different approaches of augmentation – using mobile phone or smart glasses (Epson Moverio). Glasses suffered from need of continuous marker tracking and limited field of view. However, usage of handheld devices might be limiting because it prevents usage of both hands.

Use of the mixed reality head-mounted displays frees user’s hands and as Rosen et al. [20] proved, 3D visualization of valuable information is more effective than 2D visualization. They used Microsoft HoloLens to visualize robot arm motion intent and asked participants to determine which robot motions are going to collide with obstacles on the table. They reached up to 16 % increase in accuracy and 62 % decrease in time it took participants to complete the task when using HoloLens.

5.2 Motivation for Use of the Microsoft HoloLens in the ARTable Setup

Programming of collaborative robots in the ARTable approach is based on setting up parameters for specific instructions which forms the whole program. These instructions are described only with text identifiers and as previous experiments showed up, users had difficulties in orienting in such program topology. There were uncertainties of how instructions follow, what exactly will specific program do or what is expected from user to program.

Use of the Microsoft HoloLens could clarify user's uncertainties in programming robots by providing program visualization system in spatial augmented reality. Being able to see what will happen when specific program runs, what specific instruction means and what is expected from the user to program should rapidly help in understanding of the whole ARTable system.

Besides detecting objects that are placed right on table, the ARTable sensors are able to detect objects that are out of the table, like in robot's gripper or in feeder. This functionality is used for the user to see what exactly robot sees. However, with current 2D projected user interface it can be visualized only for objects placed onto the table. HoloLens would surpass this limit and extend this functionality from 2D to 3D space by rendering 3D bounding boxes around detected objects.

Another reason why to use AR in ARTable system is because of the possibility of programming the robot through it. Or just to show places where the robot is learned to grab objects for example. It could also warn the user if anything goes wrong with the system. HoloLens would pop up error messages or warnings, which would direct the user to the source of the problem, e.g. jammed robot arm.

Overall, the headset could be used to visualize anything that current projected interface can't do.

5.3 User Interface for Programs Visualization

The HoloLens interface for robot programs visualization should contain a brief description of a program itself, possibility of controlling the visualization (stop button, pause button, play button, etc.) and a description of single steps of the program. This interface will get triggered after clicking on specific button (called for example *visualize*) of the projected interface.

Visualization of specific program will consists of specific steps, which will depend on a number of instructions of that program. It could be considered as an projected animation. However, the animation will be in 3D space, realized with holograms of robot's arm, objects for manipulation, etc. It is a question of user testing, if it would be better to have fluent animation, where user just observes what will happen if he runs current program on the real robot, or to have a stepped animation, where he can pause it, go step backwards and so on. Another approach to consider, whether it would be better to show animation, where one step is shown at exact time, or to have a possibility of showing all steps at once in static mode (without animating them). On the first thought, showing all steps at once could produce messy and unclear visualization. Thus, it seems that the first approach with controllable animation is the right way.

Proposal of such visualization interface can be seen in Figure 5.1. In the picture is shown visualization of the program *feeder-training*, where the robot picks an object from

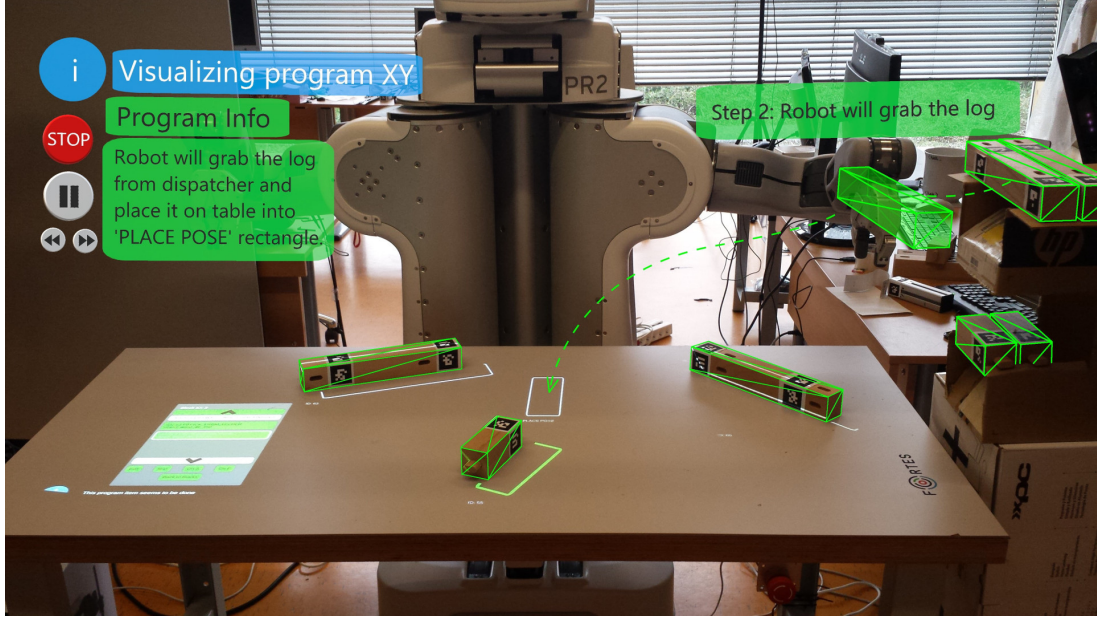


Figure 5.1: Augmented reality user interface proposal of the ARTable programs visualization for Microsoft HoloLens

the feeder and places it onto specified position of the table. Animation of this program could be done in three steps – robot grabs the object, robot places the object onto the table, robot gets into default position (retracts arms etc.). Further on you can see buttons for controlling the visualization, which would be controlled by clicking on them with an air tap gesture, and some text boxes with the information about current program.

5.4 User Interface for Robot Programming

Possibility of visualizing an existing program would be good to quickly see what that program does. However, it does not fully speeds up the programming process. This means the speed and overall understanding of setting parameters for specific programs by new and ordinary skilled users. Thus some kind of interactive guider should be provided by the headset. This guider would help the user with the robot programming by displaying animated virtual elements, such as human hand that would point to the point of interest. Additional text elements would be displayed, similarly as in the programs visualization system (see Figure 5.1).

There are two possible ways of implementing this interactive helper. One way would guide the user within individual instructions separately, where the helper would start when individual instructions are edited. The other way would guide the user within a context of a whole program. This means that the helper would start immediately when the program is edited. Second approach brings questions, like how to behave when the program is partially set or is completely set and user just wants it to reprogram. From this point of view it seems easier to use the first approach.

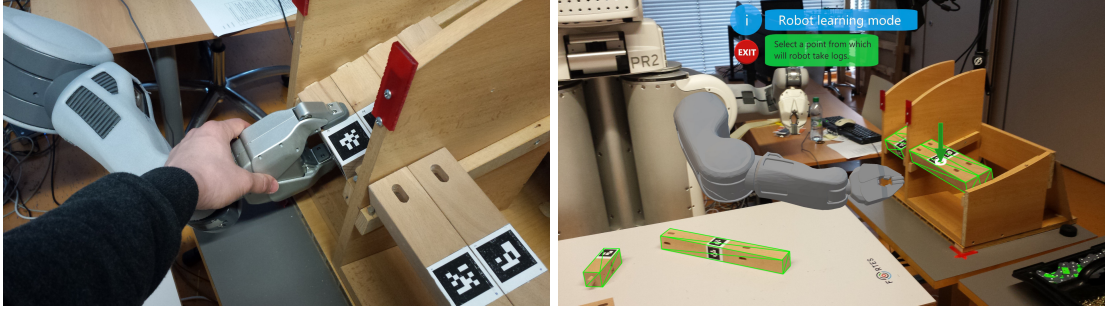


Figure 5.2: **Left:** Programming by Demonstration – user manually moves the robot arm directly to the feeder. **Right:** Learning with use of AR – user clicks on the exact position on the feeder through the headset.

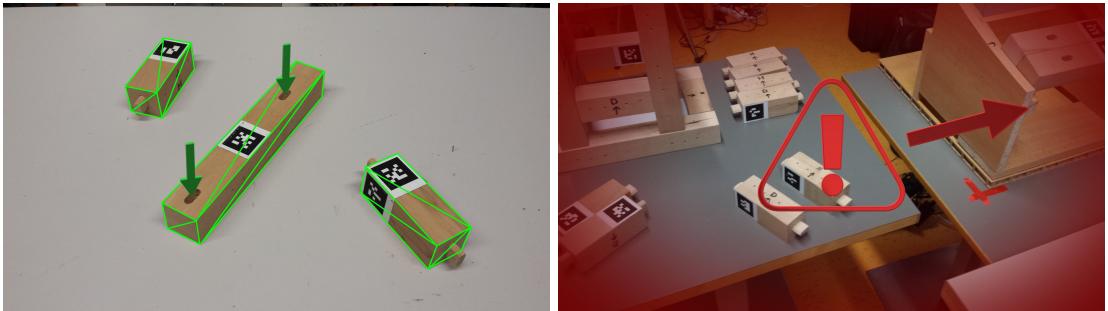


Figure 5.3: **Left:** Visualization of robot’s learned positions for *apply glue* instruction. **Right:** Situation when some error of the ARTable system occurs.

5.5 Robot Learning Using Augmented Reality and Other Possible Use Cases

One of the possible ways of programming robot is known as Programming by Demonstration. In this method, programmer manually moves the robot to desired positions, where the robot records the internal joint coordinates corresponding to that position. Afterwards, the robot is able to repeat such sequence of joint coordinates in order to achieve previously learned move [14]. Similar approach is used in the ARTable setup, where programmer manually navigates robot’s arm to desired position and the robot saves the end position of its gripper. It afterwards computes joint coordinates real-time. This manual programming is used for instance to show the robot from where exactly to grab the objects from the feeder (see Figure 5.2).

Use of AR in this field could eliminate the need of manual assistance of the programmer. User would just point to desired area and virtual robot would compute joint coordinates with use of inverse kinematics. This way the position of the virtual gripper would be transformed into the real robot’s coordinate system and saved. AR interface would then render virtual robot’s arm showing the animation of the learned move as can be seen in Figure 5.2.

When the *pick from feeder* or *apply glue* instruction is set, it is not possible to visualize exact position of the robot gripper (where exactly will be the robot grabbing the object or where exactly will be simulating “gluing”). With use of the HoloLens, those positions could be visualized for example by rendering hologram arrows pointing to them (see Fig. 5.3).

Last use case I propose is visualization of errors and warnings. When anything goes wrong with the ARTable system, user would get notified with an error or warning message popping right in front of him. As user could be confused of what happened, it is necessary to navigate him with use of arrows to location of occurred problem, e.g. to jammed robot arm or to robot gripper when it's unable to grab an object. Error message and navigation arrow would be pinned to a user's view until he reaches or looks at the location of occurred problem. Example of how this error occurrence could look can be seen in Figure 5.3.

Chapter 6

Implementation

This chapter describes implemented mixed reality extension with Microsoft HoloLens for the ARTable. Integration steps, which are needed to be done in order to integrate Microsoft HoloLens into the ARTable system, are described in section 6.1. When both systems are mutually calibrated, the headset can visualize spatial holograms, such as 3D bounding boxes representing robot's perception (see section 6.2). In order to help the user better understand robotic programs of the ARTable system, the program visualization system was implemented, which is further described in section 6.3. To help the user with programming the robot, the interactive helper is introduced in section 6.4. Lastly, in section 6.5, I will describe more in depth developed application details, its components and their behavior. Not all proposals from previous chapter were implemented. For example the warning system was rejected due to poor HoloLens field of view or an alternative way of programming the robot by demonstration is discussed later on as a possible future work.

For development, the Unity¹ was chosen, which is a cross-platform game engine used mainly for developing 3D and 2D video games or simulations. Unity has native build support for UWP² applications, which are used on HoloLens. Microsoft even released Unity package – *MixedRealityToolkit-Unity*³ – which is a collection of scripts and components intended to accelerate development of applications targeting Microsoft HoloLens and Windows Mixed Reality headsets. It contains various modules, such as gestures or voice input handling, spatial mapping or spatial sound, that I used in this work.

6.1 Integration of Microsoft HoloLens into the ARTable System

In order to successfully integrate the HoloLens into the ARTable system there were two major steps to take – make communication between these two systems work and calibrate the HoloLens with respect to the ARTable.

Communication via rosbridge

The ARTable system runs on ROS with Ubuntu but HoloLens runs on Windows. So there is a problem of how to transport data from one operating system to another. This problem

¹<https://unity3d.com/>

²Universal Windows Platform

³<https://github.com/Microsoft/MixedRealityToolkit-Unity>

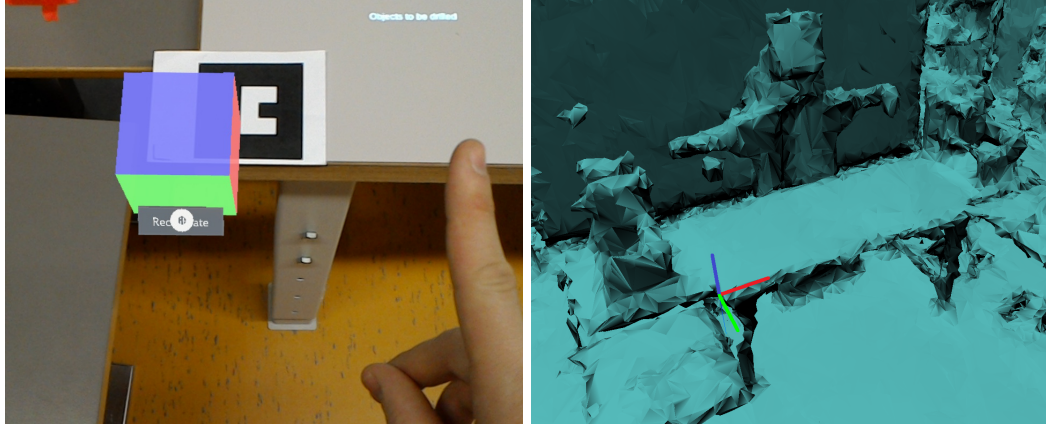


Figure 6.1: **Left:** Calibration of the HoloLens with respect to the ARTable. Colored cube visualizes how well the calibration was done and has an option button for recalibration. Red side of the cube represents positive x axis, green side represents positive y axis and blue positive z axis. **Right:** Mesh of the workplace created by HoloLens sensors that contains a spatial anchor which represents the origin of the ARTable's coordinate system.

is solved by ROS tool – *rosbridge*, which is a tool that provides a JSON API to ROS functionality for non-ROS programs.

For such communication I used and extended existing Unity library – *ROSBridgeLib*⁴ (author Michael Jenkin, edited by Mathias Ciarlo Thorstensen). This library uses *SimpleJSON*⁵ parser for parsing JSON messages. However, due to differences in scripting backends of Unity editor (Mono) and UWP applications (.NET), functionality of this library had to be extended to support the UWP format which is used by HoloLens. With inspiration from Unity library *holoROS*⁶, created by Gabriel Santos Solia, who successfully integrated Microsoft HoloLens with ROS via *rosbridge*, in order to simulate a holographic *turtlesim*⁷ environment, I was able to extend the *ROSBridgeLib* functionality to support communication between the ARTable (ROS) and Unity along with HoloLens (Windows).

Having this type of communication enables to connect multiple headsets with the ARTable, where all of them receive same data.

HoloLens Calibration with Respect to the ARTable

The ARTable is calibrated with use of three markers placed into corners of the table. Origin of coordinate system is estimated in bottom left corner of the table. The HoloLens also needs to know where that corner is. This is accomplished by detection of a marker placed in that spot with use of HoloLens world-facing camera (see Figure 6.1). For detection purposes, the *HoloLensARToolKit*⁸ was used, which is a Unity library that integrates ARToolKit⁹ functionality with UWP applications, created by Long Qian.

If marker detection is successful, colored cube is drawn on this marker spot which helps to visualize how well the calibration was done. This cube is afterwards wrapped with an

⁴<https://github.com/MathiasCiarlo/ROSBridgeLib>

⁵<https://github.com/Bunny83/SimpleJSON>

⁶<https://github.com/soliagabriel/holoROS>

⁷<http://wiki.ros.org/turtlesim>

⁸<https://github.com/qian256/HoloLensARToolKit>

⁹<https://github.com/artoolkit/artoolkit5>

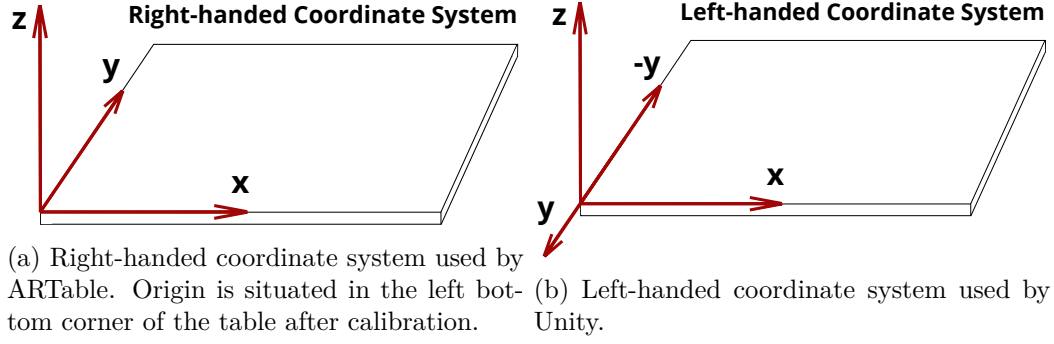


Figure 6.2: Difference of used coordinate systems. After HoloLens calibration, the y axis needs to be inverted and rotations adjusted in order to convert from one system to another.

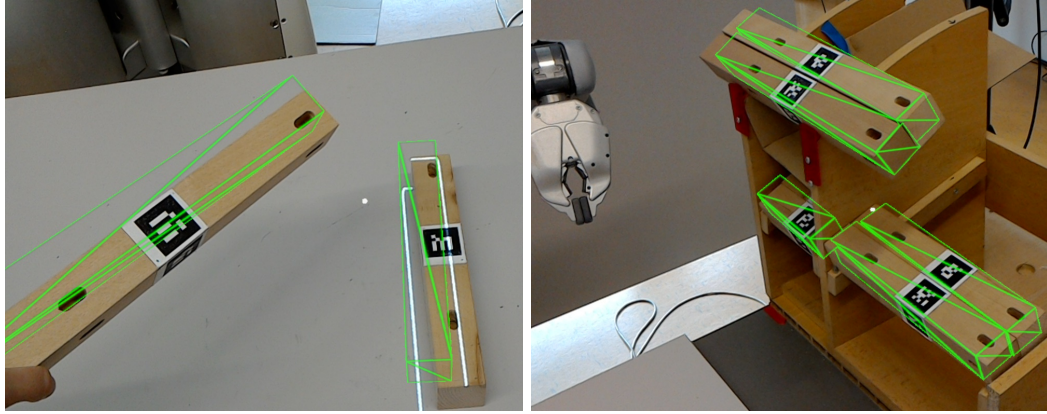


Figure 6.3: Visualization of robot's perception. **Left:** Detected objects on the table and above the table visualized by 3D bounding boxes. **Right:** Detected objects in the feeder.

empty Unity *GameObject* which is then stored as a spatial anchor and set as a parent to every other visual object in the scene. Thus it represents origin of the table. Spatial anchors are persistent during sessions, so calibration process needs to be done only once. Mesh of the workplace created by HoloLens sensors that contains this spatial anchor can be seen in Figure 6.1. There is also possibility of recalibration by clicking the *Recalibration* button as can be seen in Figure 6.1.

After calibration, one last step has to be made. There is a conflict in coordinate systems, the ARTable uses right-handed (see Figure 6.2a) while Unity and therefore HoloLens uses left-handed (see Figure 6.2b). To solve this problem, every object position and rotation that came from ARTable needs to invert its y axis and Euler x and z angle as shown in Figure 6.2.

6.2 Visualization of Robot's Perception

Current 2D projected user interface is limited in visualization of spatial information. For instance, it draws bounding boxes around detected objects placed on the table, that indicates what robot actually sees. However, sensors are able to detect these objects even if they are not placed directly on table, e.g. when they are in gravity feeder.

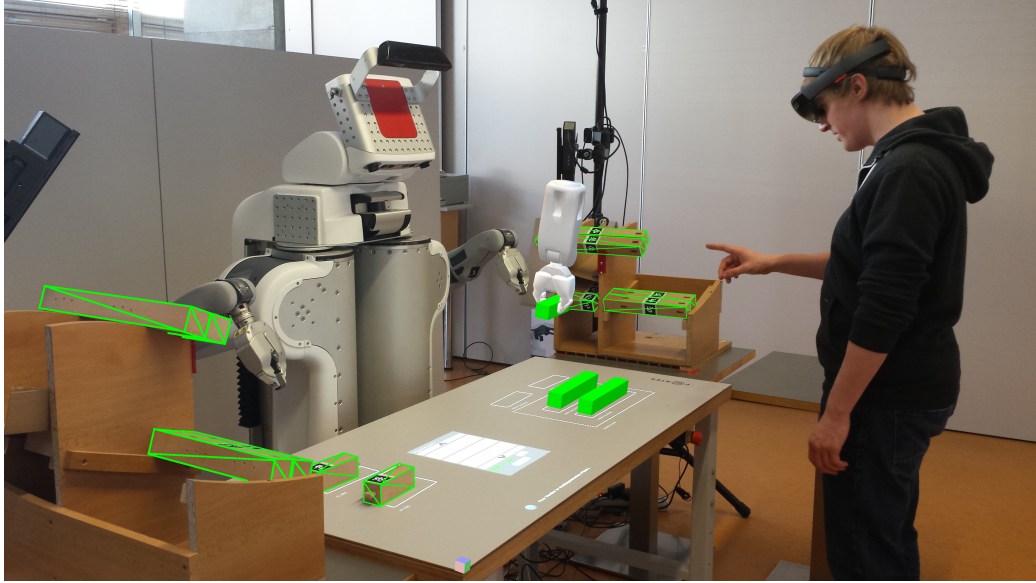


Figure 6.4: Spectator’s view of the workspace extended by virtual holograms seen through the headset. In this situation the user observes visualization of “Stool assembly” program. Image was created in external tools to demonstrate how it would look from spectator’s view and does not correspond precisely to reality (bounding boxes of detected objects would be slightly off the real objects due to imperfect calibration).

Provided solution eliminates limitations of 2D projected interface. With use of the headset, drawing of 3D spatial bounding boxes around detected objects is easily solved. The HoloLens just loads the data about position, size and type of currently detected objects from the ARTable system. Being able to see what the robot sees can be helpful in various situations, e.g. when user wants him to pick some object from the feeder that is physically present but the robot doesn’t see it.

This visualization can be seen in Figure 6.3 where green 3D bounding boxes are drawn around detected objects. There are notable inaccuracies in fitting the virtual bounding box onto real object. These inaccuracies result from small inaccuracies of partial calibrations of the whole system – calibration of Kinects, projector, touch-table and HoloLens. Summing these up creates a notable inaccuracy.

6.3 User Interface for Robot Programs Visualization

In order to eliminate user’s uncertainties in programming the robot, programs visualization mode was implemented. This mode extends current 2D projected interface by *Visualize* button, which starts the visualization itself. Only programs with set parameters are able to be visualized (see Figure 6.5). After hitting this button, the interface checks if HoloLens are running, otherwise visualization won’t start. Visualization is fully controllable with projected buttons – *Pause/Resume*, *Stop* and *Replay* (as shown in Figure 6.5).

As described in subsection 4.1, ARTable programs are compounded of various instructions that are linked together. Instruction set of currently visualized program is loaded from ROS environment into HoloLens which then dynamically builds this program from internally implemented classes representing these instructions. With this approach any pro-

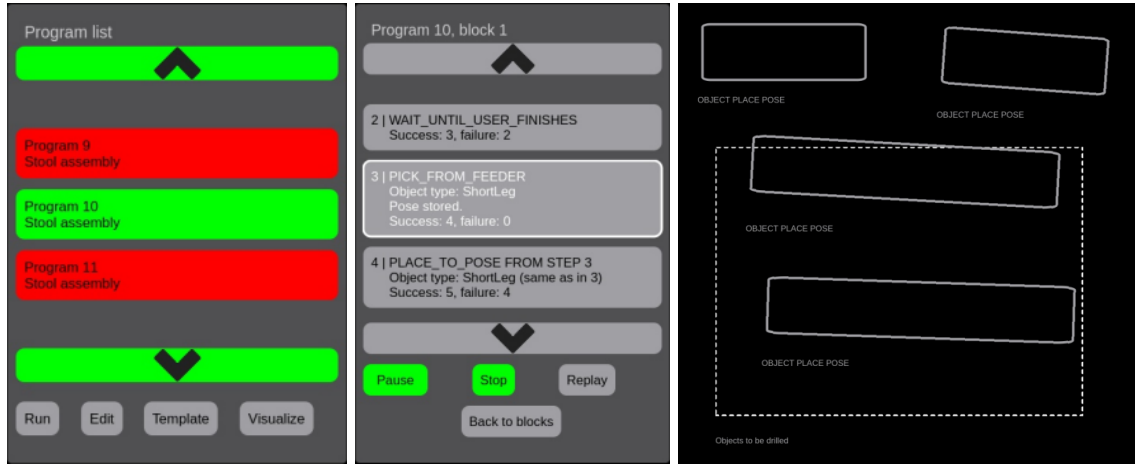


Figure 6.5: Widgets of projected ARTable GUI extended by visualization mode. **Left:** List of programs, where green programs are ready to visualize or run and red programs need to set parameters. **Middle:** List of instructions of currently visualized program. Visualization is fully controllable with provided buttons – “Pause”/“Resume”, “Stop”, “Replay”. Visualization can be also controlled with voice by saying same keywords as are names of those buttons. **Right:** Rectangles defining areas on the table onto which the grabbed objects of different type will be placed. Dashed polygon defines area in which the robot will apply glue to corresponding objects. All these supplementary elements of projected GUI are displayed at the same time when program visualization runs.

gram variations built from supported instructions should visualize correctly. After build, execution starts.

Example usage of the headset is shown in Figure 6.4, where the user observes visualization of “Stool assembly” program. Visualization of this program is further demonstrated in Figure 6.6. Whole process visualizes individual instructions which are connected together in a context of the program. For example, when visualizing *pick from feeder* instruction, virtual object is created, which is then picked by a virtual robot gripper from the feeder and placed on the preset position on the table. This object then stays active in the scene in case that another instruction could want to manipulate with it (e.g. to apply glue to it or to move it somewhere else). Whole process is commented by Microsoft Zira text-to-speech voice. In this case, it would say: “The robot is grabbing the object from feeder on your left/right side”, “The robot is placing the object to preset place pose” and so on. This voice commenting replaces the original idea of displaying text boxes that would describe current state of the visualization.

Besides provided buttons for controlling the visualization, which I had to implement, user can use his voice. Introduced system recognizes couple of basic keywords: “pause”/“resume”, “stop”, “replay”, “next” (to immediately move to next instruction) and “back” (to immediately move to previous instruction). These keywords are insured against unexpected behavior. For example if users says “stop” when visualization is already stopped, he gets notified by speech synthesizer. Proposal of this programs visualization described in section 5.3 suggests to control the visualization using holographic buttons. This idea was dropped due to low field of view of the HoloLens and replaced with the voice commands. The buttons would be rather disruptive than useful. With use of the voice controls, user

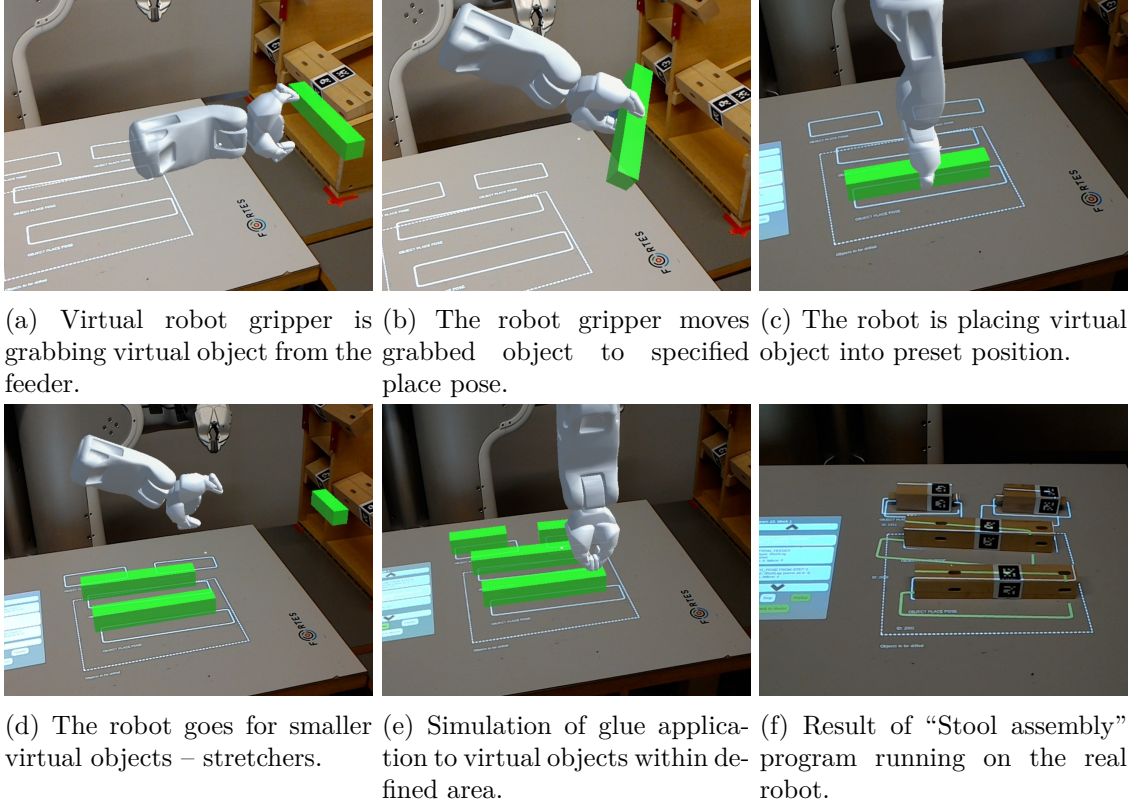


Figure 6.6: An example of program visualization. In this case – first block of “Stool assembly” program which consists of a total of three blocks. In this block the robot will pick the objects from the feeder, place them onto the table and apply glue into the holes in order to prepare these components for user to assembly. This block consists of eleven instructions – *wait until user finishes*, four *pick from feeder* and consequent *place to pose* instructions, *apply glue* and lastly *get ready*.

can fully focus on the visualization itself, without the need of coming closer to the table and looking on the projected UI in order to hit some button.

During whole visualization process, the HoloLens and the ARTable keeps communicating using aforementioned *rosbridge*. Both systems synchronizes their states about which instruction is visualizing, whether it’s playing or is stopped and so forth. This continuous communication ensures scrolling down in instruction list of projected 2D interface or enabling and disabling projected buttons. Lastly, supplementary elements of projected interface – such as object place pose rectangles, polygons defining areas from which to pick or in which to apply glue to objects – are displayed at once to help the user get an instant overview of an outcome of current program (see Figure 6.5).

It is important to mention that whole visualization should be rather understand as an animation, not simulation. It correctly visualizes initial and end state of every instruction. However the robot gripper movements are just animated to approximately match the movements of the real robot. Thus it cannot be considered as a simulation of the real robot movements. While simulation would allow proper verification of correctness of programmed program, chosen animation approach is sufficient enough for its purpose, namely to help the user faster understand robotic programs.

6.4 Interactive Helper for Robot Programming

In order to enhance the human-robot collaboration even more, the interactive helper for robot programming was implemented. This helper guides the user throughout whole programming process. With use of the HoloLens voice and 3D holograms, the user is instructed in what to do when he is programming the robot. He is also warned of occurred errors.

The helper mode was implemented for *pick from polygon* and *place to pose* instruction. It starts immediately after the user starts editing one of these two instructions. For *pick from polygon* instruction the object type and pick polygon needs to be set. Thus the user needs to do two main actions – click on the outline of some detected object placed on the table and drag the outline of pick polygon to specify the area from which the robot will be picking up objects. With the HoloLens put on, he is guided by Microsoft Zira’s voice, which tells him meaningful advices such as “Select object type to be picked up by tapping on its outline.” or “Adjust pick area as you want – or you can select another object type. When you are finished, click on done.” Zira’s voice is complemented by animated 3D holograms of hand with raised forefinger, which points to the object or polygon outline to visually help the user understand what needs to be done. In case of selecting the object type, the virtual hands appear above every detected object and moves up and down to indicate clicking movement. In case of adjusting the pick polygon, the virtual hand moves back and forward above the polygon outline to indicate dragging movement (see Figure 6.7). Only one major mistake can appear within programming this instruction, namely the absence of any object placed on the table. If so, the user will not be able to select the object type to be picked up. If this happens, the headset warns the user about absence of objects and asks him to put some of them on the table.

For *place to pose* instruction, the user needs to specify position on the table where the robot places picked object from previous *pick from polygon* instruction. He is guided by Zira’s voice and animated 3D hologram of hand, similarly as in the case of above discussed instruction (see Figure 6.7). The user gets warned if he tries to program *place to pose* before the corresponding picking instruction because the system doesn’t know which object type is going to be placed.

After the user successfully programs one of the instructions or partial tasks, he is commended with simple “Perfect!” or “Good job!” praise. In case that he didn’t understand everything what Zira said, he can say “repeat” keyword, to force Zira to immediately repeat lastly spoken advice.

6.5 Application Components

As mentioned at the beginning of this chapter, whole application was implemented in Unity, which is a multipurpose game engine that supports 2D and 3D graphics, drag-and-drop functionality within its editor and scripting using C# language. Implemented application, named – *ARTableHoloLens*, was divided into a couple of main components handling specific functionalities – communication with ARTable (ROS), headset calibration, program visualization, interactive helper for robot programming, handling detected objects or user voice input. These components are realized as singletons.

I tried to use existing tools and packages as much as possible, to stay focused on main goal of this work and not reinventing the wheel. Thus I used Microsoft’s Unity package

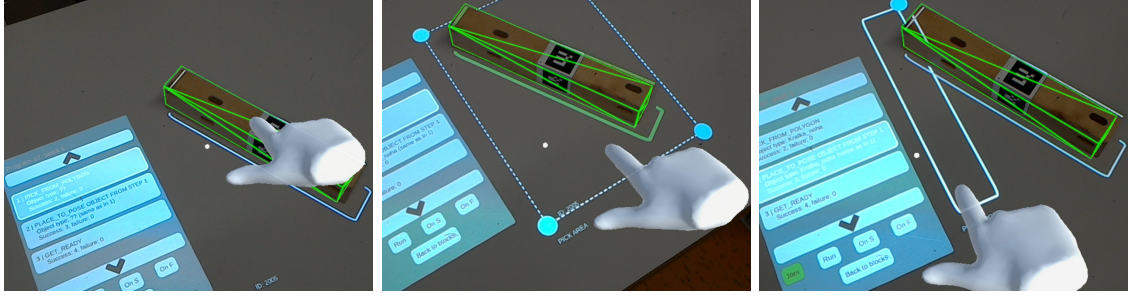


Figure 6.7: An example of interactive helper for robot programming. **Left:** User is setting parameters for *pick from polygon* instruction and is guided by device’s voice and 3D hologram of the hand to select some object type to be picked up by the robot. **Middle:** User is specifying polygon from which the robot will pick up previously specified object and is guided by device’s voice and animation of virtual hand. **Right:** User is adjusting place pose area for *place to pose* instruction.

*MixedRealityToolkit-Unity*¹⁰ that has a bunch of basic stuff for head-mounted displays already implemented, such as gestures or voice input handling, spatial mapping, spatial sound or hologram sharing modules.

Calibration and Communication Component

When the application starts, a *SystemStarter* script tries to load previously stored spatial anchor from the HoloLens local database. If spatial anchor doesn’t exist, script calls the *Calibration manager* that starts the calibration process as described in 6.1. This manager encapsulates the *HoloLensARToolKit*¹¹ library (integration of ARToolkit with UWP applications), which handles detection of preset marker. It takes images from HoloLens world-facing camera in order to detect black borders of potential markers with use of a threshold. By thresholding, connected groups of pixels are found, from which the contours are extracted. Those contours are surrounded by four straight lines, which form four corners, from which a homography matrix is calculated in order to remove perspective distortion and to transform potential marker to a canonical front view in order to compare it to a library of known markers by feature vectors [9]. With successfully detected marker, virtual cube, which is stored afterwards as a spatial anchor, is displayed above this marker. This spatial anchor will serve as a root *GameObject* for every other object displayed in scene.

The *Communication manager* uses extended *ROSBridgeLib*¹² Unity library, as described in 6.1. From the very beginning of application start, it creates a WebSocket connection with the ARTable’s *rosbridge* server and subscribes to ROS topic to receive messages containing information about current states of the ARTable system. Every received message is distributed among other components that somehow reacts to them (e.g. starts program visualization or starts functions of interactive helper). Besides this type of message, the *Communication manager* receives messages containing position, rotation and type of detected objects by ARTable’s sensors. Manager is even publishing its own message, about

¹⁰<https://github.com/Microsoft/MixedRealityToolkit-Unity>

¹¹<https://github.com/qian256/HoloLensARToolKit>

¹²<https://github.com/MathiasCiarlo/ROSBridgeLib>

the activity of the headset device, every 5 seconds; and is ready to inform the ARTable about changes in application states to keep coherency with both systems.

Program Visualization Component

When the ARTable system switches into a visualization state, meaning that user started visualization of an existing program, this component loads that specific program structure from the ARTable system over the *rosbridge*. It dynamically builds this program from pre-prepared C# classes representing individual program instructions. Those classes are linked together into a linear list that is afterwards sequentially traversed in a *coroutine* in order to execute the visualization of whole program. This approach has an advantage that the application doesn't need to have exact order of every program's instructions stored internally. It also allows to have multiple programs variations without the need of reimplementing application code.

Component supports all known instructions (described in 4.1). Only *pick from polygon* and *pick from feeder* are allowed to create *GameObjects* that represents real objects to be picked up by robot. Those are passed by reference to *place to pose* or *apply glue* instructions to enable the virtual robot gripper to manipulate with them. This gripper is created only once at the start of the application and is shared between the individual instructions.

During the visualization, the *Communication manager* keeps sending messages to the ARTable system about its progress (which instruction is currently visualizing or whether the user used some of the voice commands for controlling the visualization state). For recognizing voice inputs, the *Windows Speech Recognition API* that is a part of the Unity engine was used. When the user decides to stop the program's execution, every *GameObject* belonging to that program is marked as inactive. Thus all virtual objects and robotic gripper are hidden (not destroyed), which has an advantage of that the program and all related objects doesn't need to be created all over again, they are just activated and refreshed to their initial positions, if the user chooses to replay that program. They are destroyed only once the user decides to leave the visualization mode. "Pause" sets the Unity's *timeScale* to 0, which basically freezes motion of every object in scene. "Resume" sets this property back to 1. The *Windows Speech Synthesizer*, which converts text to speech, was used for commenting the visualization steps.

Interactive Helper Component

This component is comprised of two singleton C# classes representing program instructions – *pick from polygon* and *place to pose*. Both are waiting until the ARTable system switches into an edit state of one or another instruction. Both are using the *Windows Speech Synthesizer* and are allowed to create the *GameObject* of a virtual hand with pointing forefinger to guide the user throughout the edit process. The *place to pose* creates one virtual hand only once, when the user is asked to set the place pose, while *pick from polygon* creates virtual hand twice. Firstly, when the user is asked to pick desired object type (in this scenario, for every real object placed on the table is created virtual hand that points to that object) and secondly, when he is asked to set the pick polygon (only one virtual hand is created). When the user finishes current task, every such created hand is destroyed.

The component does not interact with the ARTable system. It only reacts to received messages about the system state within the headset (e.g. when the user tries to select the object type but no objects are placed on the table – which means that there are no detected objects at all and if so, *z* coordinate is checked to be approximately zero).

Objects Manager Component

The *Objects manager* parses received data from the *Communication manager* about detected objects ID, position, rotation, size and type. It creates list of *GameObjects* representing such detected objects. With every newly received data, it updates their position and rotation based on their ID. Those objects are rendered with custom wireframe shader that colors the object to black (this has an effect of transparency) and draws green lines on the object's outline representing its wireframe.

Besides the list of detected objects, it even holds the list of all virtual objects, created during visualization of some program. This is useful for a check of presence of any virtual object within given area or to easily destroy all virtual objects when the program visualization ends.

Chapter 7

Experiments

The aim of this work was to extend the ARTable workspace over the mixed reality headset – Microsoft HoloLens and to create user interface which will enhance the human-robot interaction. Thus the **visualizer of robotic programs** and the **interactive helper for robot programming** were implemented.

In order to validate proposed approach and to determine whether the use of the headset in the ARTable setup is valuable, a user experience testing has been carried out. Description of performed experiments is described in section 7.1, its results are described in 7.2 and in section 7.3 I discuss suggestions for further work.

7.1 Experiments Description

I prepared a set of three tests in which the participants tried to set parameters for a robot program, tried to estimate what a program with already set parameters would do and tried to use the Microsoft HoloLens in more complex visualization scenario. Participants were divided into two groups – **group X** and **group Y**. Participants in **group X** were performing designed tests without the use of the headset, while participants in **group Y** were performing same tests using the headset. Participants were tested one by one.

As measures, I chose the System Usability Scale (SUS) [4], and I created my own custom questionnaire for determining the usability of Microsoft HoloLens in the ARTable setup. I also recorded task completion times and corresponding number of needed interventions. Those values were afterwards averaged within the individual groups and compared to each other. Results indicated whether proposed solution in this work is useful or not. For detailed results, see section 7.2.

Testing of each participant was done in the following steps:

1. Introduction of the ARTable, involving its motivation, what is it good for and very basic description of projected GUI.
2. Assigning the first task – set parameters for *polygon-training* program.
3. Filling the SUS questionnaire.
4. Assigning the second task – estimate what exactly would already programmed *feeder-training* program would do.

5. Assigning the third task – try to visualize *stool assembly* program using the headset and use the voice commands and provided buttons for controlling the visualization (both groups were using the headset for this task).
6. Filling the custom questionnaire.

Before the proper testing, a pilot experiment with one participant from each group took place. This helped me to verify the functionality of proposed solution and to create the final experiment design.

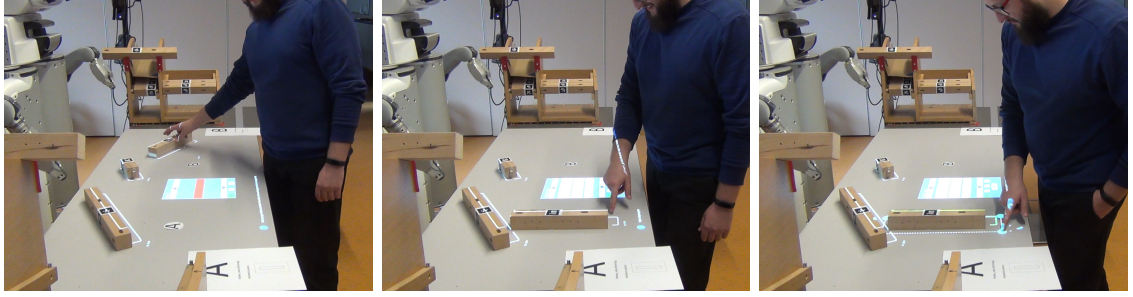
First Task – User Robot Programming

Within the first task, participants were asked to set parameters for *polygon-training* program, which consists of a total of four instructions, two programmable – *pick from polygon* and *place to pose* and two non-programmable – *get ready* and *wait until user finishes*. For the *pick from polygon*, the user is supposed to set the object type to be picked up by robot by clicking on the detected object’s outline of desired type and to set the pick area by adjusting projected polygon. For the *place to pose*, the user is supposed to set the place pose by dragging the projected outline of previously selected type.

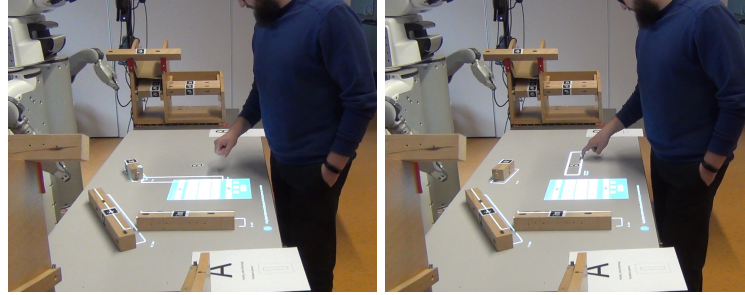
I designed a simple scenario in which the participants were told to set parameters for the robot to pick up the *ShortLeg* object from marked place on the table – marker **A**, and to place it on marked place on the table in specified rotation – marker **B**. Those two markers were placed on the table along with three objects of different type – *Stretcher*, *ShortLeg* and *LongLeg*. In order to correctly set parameters for this program, these steps needs to be reproduced:

1. Select the first instruction of the program – *pick from polygon*.
2. Find the *ShortLeg* object type by clicking on the objects outline.
3. Click on the *Edit* button.
4. Select the *ShortLeg* object type.
5. Move the projected pick area to the marker **A** and adjust it to match the assigned shape.
6. Click on the *Done* button.
7. Select the second instruction of the program – *place to pose*.
8. Click on the *Edit* button.
9. Move the projected place pose to the marker **B** and rotate it to match the assigned pose.
10. Click on the *Done* button.

Participants were trying to figure out these steps on their own, without any further interventions from my side (except the critical ones and those that were initiated by users). Users in **group X** were using just the projected user interface. The course of this first task can be seen in Figure 7.1. Users in **group Y** were using the projected user interface along with the headset’s interactive program helper (described in section 6.4), see Figure



(a) User finds the *ShortLeg* object type and moves it on marked position **A**. (b) User selects the *ShortLeg* object type by tapping on its outline. (c) User adjusts pick area by dragging it on the table to match the desired pose.



(d) User selects *place to pose* instruction from list. (e) User adjusts place pose to desired position.

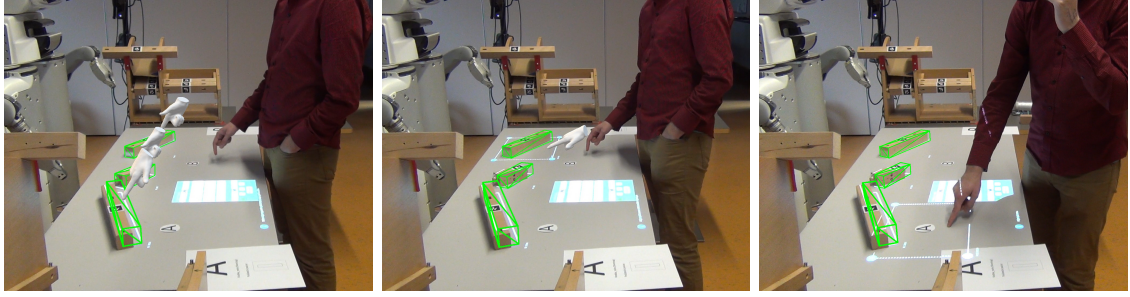
Figure 7.1: The first task of the experiment – the user from **group X** sets parameters for one *pick from polygon* and one *place to pose* instruction of the *polygon-training* program without the use of the headset.

7.2. Both groups had none previous experience with the ARTable nor Microsoft HoloLens and none demonstration example of how to set parameters for robotic program was showed to them. After completion of this task, the SUS questionnaire was filled in.

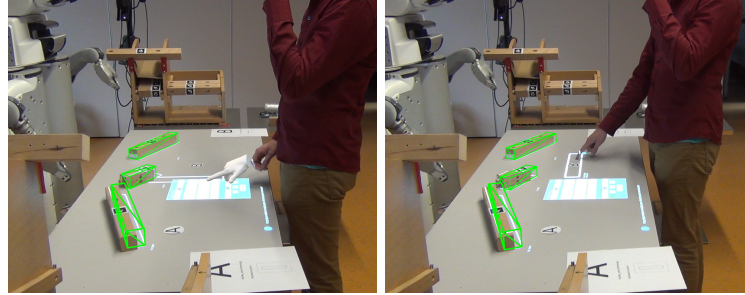
Second Task – User Understanding of Robotic Program

Withing the second task, participants were asked to estimate, what the program *feeder-training* with parameters already set would do if run on the real robot. In other words, how fast they can understand this program without any previous knowledge of it. The program consists of a total of six instructions, four programmable – two *pick from feeder* (second is a copy of the first instruction) and corresponding two *place to pose* and two non-programmable – *get ready* and *wait until user finishes*. The program was set as follows:

1. The robot grabs the *LongLeg* object from the feeder on user's left side (first *pick from feeder*).
2. The robot places grabbed object on marked position on the table, approximately in the middle of it (first *place to pose*).
3. The robot grabs another *LongLeg* object from the feeder on user's left side (second *pick from feeder* – copy of 1).
4. The robot places grabbed object on marked position on the table, approximately 10 cm next to the previously placed object from its left side (second *place to pose*).



(a) User is going to select the *ShortLeg* object type by tapping on its outline. (b) Animated virtual hand is suggesting the user to drag the pick area. (c) User adjusts pick area by dragging it on the table to match the desired pose.



(d) Animated virtual hand is suggesting the user to drag the desired position. (e) User adjusts place pose to match the desired position.

Figure 7.2: The first task of the experiment – the user from **group Y** sets parameters for one *pick from polygon* and one *place to pose* instruction of the *polygon-training* program using the headset. User is guided by the voice advices and 3D holograms of animated pointing hand from the application’s *Interactive Helper*.

5. The robot arm moves back to its default position (*get ready*).
6. The robot waits for the user to finish current task (*wait until user finishes* – just simulation).

Group X was allowed to use the *edit mode* only. Within this mode they could traverse through all of the listed instructions by selecting them in order to examine their text descriptions and visual elements displayed on the table, or by running them on the robot. This task was kind of tricky because currently it is not possible to clearly estimate (just from the text description) from which feeder will the robot grab the object (whether the one on the user’s left side or the other on his right side). User has to run the *pick from feeder* instruction to see where the robot goes for the object. Once he knows which feeder is used in this program, it should be easy to estimate the rest without the need of running it on the robot.

Group Y was allowed to use the *visualization mode* only (described in section 6.3). A commented animation of the program using spatial 3D holograms and speech synthesizer was played in the headset. User should get a clear overview of the program workflow thanks to the animation.

Third Task – HoloLens Usability Testing

Participants from both groups were asked to put on the headset and oversee the *stool assembly* program visualization. They were asked to try out all commands for controlling the visualization, both projected and voice, within this visualization and to focus on the headset-related attributes, like its field of view, its comfortness or a quality of the speakers.

Participants from **group X** tried out again the first task, but this time using the headset. All users filled the custom questionnaire afterwards. This questionnaire examines the usability of the HoloLens within all of those three tasks.

Participants

All volunteers are university students, in the age of 19 to 26 years old. They are labeled with letters A – F (six participants in the **group X**) and K – P (six participants in the **group Y**). None of them had previous experiences with the ARTable nor Microsoft HoloLens. Demographic data of the participants can be seen in the Table 7.1.

Participant	Gender	Age	Education	Group
A	M	26	higher	group X (without the headset)
B	M	24	higher	
C	M	24	higher	
D	M	24	higher	
E	M	19	secondary	
F	M	23	higher	
K	M	23	higher	group Y (with the headset)
L	M	21	secondary	
M	M	24	higher	
N	M	23	secondary	
O	F	23	secondary	
P	M	25	higher	

Table 7.1: Demographic data of participants.

7.2 Results

The Table 7.2 shows qualitative and quantitative data per participant of the **group X**. Mean time to complete the first task was 2 minutes and 48 seconds with 2.5 interventions. This task consisted of setting parameters for one *pick from polygon* and *place to pose* instruction. Mean time to complete the second task was 3 minutes and 4 seconds with 2.5 interventions. Second task involved understanding of the *feeder-training* program using only the *edit mode* of the projected interface. Mean SUS rating was 70.8.

Results of the **group Y** participants is showed in the Table 7.3. Mean time to complete the first task was 3 minutes and 41 seconds with 2.7 interventions and mean time to complete the second task was 1 minute and 32 seconds with 0.7 interventions using the *visualization mode* in the headset. Mean SUS rating was 67.5. This score is within the range of average usability, which, according to research, is around 68 [5]. Interestingly, to complete the first task using the headset took almost a minute longer than without the headset. Reasons

what could cause this delay are discussed below. However, the second task completion time was reduced by half when using the headset. Number of needed interventions also reduced significantly – from previous 2.5 down to 0.7 interventions. SUS ratings are nearly the same, although the SUS score when using the headset is slightly lower. This could be caused by targeting this SUS just for the first task where **group Y** participants didn’t fully appreciate added value of the headset and in some cases it rather confused them.

Measure	A	B	C	D	E	F	Average
System Usability Scale	60.0	72.5	82.5	80.0	37.5	92.5	70.8
Time to set <i>polygon-training</i> program (first task)	1:34	1:38	1:45	2:32	4:02	4:58	2:44
Interventions	1	1	1	1	5	6	2.5
Time to understand <i>feeder-training</i> program (second task)	4:20	3:43	2:58	1:43	3:08	2:37	3:04
Interventions	3	4	1	2	3	2	2.5

Table 7.2: Qualitative measures, task completion times and number of needed interventions for participants that weren’t using the headset (**group X**). Time is in the “minutes:seconds” format.

Measure	K	L	M	N	O	P	Average
System Usability Scale	47.5	90.0	70.0	60.0	57.5	80.0	67.5
Time to set <i>polygon-training</i> program (first task)	4:20	3:13	3:07	2:43	5:06	3:38	3:41
Interventions	4	2	1	2	5	2	2.7
Time to understand <i>feeder-training</i> program (second task)	2:27	1:53	1:09	1:00	1:05	1:41	1:32
Interventions	1	1	0	0	1	1	0.7

Table 7.3: Qualitative measures, task completion times and number of needed interventions for participants that were using the headset (**group Y**). Time is in the “minutes:seconds” format.

From the custom questionnaires (see Table 7.4 for **group X** participants and Table 7.5 for **group Y** participants) it seems that the main issue about the headset that participants didn’t liked is its low field of view. Interestingly, the users from the **group X** mostly complained about discomfort of wearing the headset but the users from the **group Y** mostly didn’t mind wearing it. Both groups agreed on easy understanding of the spoken word of the headset’s speech synthesizer and easy voice controlling. They also mostly preferred controlling the visualization by voice rather than projected buttons, however, there were two participants that preferred buttons over voice. After seeing the program visualization, all participants immediately knew what a real robot would do. Use of the headset’s helper for robot programming wasn’t that straightforward, participants appreciated it, but there were some users that didn’t use it or felt neutral about it.

Statement	A	B	C	D	E	F	Average	Mode
I didn't like low Field of View of the headset. I had to look for holograms for a while.	5	5	4	4	5	4	4.5	5
The headset was uncomfortable, and I would definitely mind wearing it for a long period.	5	3	4	5	5	4	4.3	5
The headset's speech synthesizer was easy to understand.	5	5	4	4	4	5	4.5	5
Controlling by voice was easy and intuitive.	4	5	5	5	5	5	4.8	5
I would rather control the system with buttons than by voice.	1	4	2	4	1	1	2.2	1
After seeing the program visualization, I knew what a real robot would do.	5	5	5	5	5	5	5.0	5
I used the headset helper when programming the robot. Thanks to it, I have been able to program the robot easily.	5	5	3	2	5	4	4.0	5
Sometimes I didn't know what to do.	1	3	4	4	4	2	3.0	4

Table 7.4: Custom questionnaire for participants that tried the headset after they completed all testing tasks without the use of the headset (**group X**). 1 – totally disagree, 5 – totally agree.

User Robot Programming

Regarding the first task, most common issues for both groups were mostly projected interface related, like double presses of buttons, where user tried to click on the *Edit* button or to select instruction which was immediately unselected. There also sometimes occurred other issues with the touch table, namely wrong touch detection due to imperfect calibration of the projector with the table and none touch detection on a non-touchable margins of the table. Almost all participants tried to run the program they programmed directly in *edit mode* that allows to run individual instructions separately, for fluent run, *edit mode* must be exited. This caused that five participants tried to run firstly *place to pose* instruction without previous picking instruction, which resolved in error because the robot didn't held any object.

Following issues occurred just at participants in **group X**. Participant B tried to select the object type by touching the object itself, not its outline. He also complained about user interface colors. He said that it took him some time before he got used to it (that grayed buttons are inactive, green ones are active, etc.). E had problems with object type selection. He wanted to select it by lifting it up with his hand. He most probably thought that he would teach the robot *polygon-training* program by demonstrating it by himself. After he was instructed that he needs to use the *Edit* button and other displayed elements on the table, he started setting parameters for the *place to pose* instruction without having the corresponding *pick from polygon* instruction set. This resulted in error because object

Statement	K	L	M	N	O	P	Average	Mode
I didn't like low Field of View of the headset. I had to look for holograms for a while.	5	2	5	2	2	5	3.5	5
The headset was uncomfortable, and I would definitely mind wearing it for a long period.	2	3	3	2	4	4	3.0	2
The headset's speech synthesizer was easy to understand.	5	5	4	5	4	5	4.7	5
Controlling by voice was easy and intuitive.	5	5	4	5	5	4	4.7	5
I would rather control the system with buttons than by voice.	1	3	2	2	2	3	2.2	2
After seeing the program visualization, I knew what a real robot would do.	5	5	5	5	4	5	4.8	5
I used the headset helper when programming the robot. Thanks to it, I have been able to program the robot easily.	4	4	5	4	3	3	3.8	4
Sometimes I didn't know what to do.	4	1	2	2	4	4	2.8	4

Table 7.5: Custom questionnaire for participants that completed all testing tasks using the headset (**group Y**). 1 – totally disagree, 5 – totally agree.

type wasn't set from previous picking instruction. This specific error is handled by text warning projected on the table, but in this case the text warning was insufficient and I had to interrupt. If E was using the headset, the speech synthesizer would warn him, which could be more effective. Participant F had troubles with setting the place pose. He forgot to hit the *Edit* button and tried to move the place pose, which was inactive. Once he figured out how to move with it, he set it to unreachable pose where the robot can't reach, which he afterwards tested himself by running it.

Participants in **group Y** had slightly different issues. Those were mostly caused by confusing holograms of pointing hand. Three participants did not understand that they must select the object type by clicking on its projected outline on the table. They tried several times to touch the virtual object's bounding box they saw in the headset. Thus they were basically reproducing the virtual pointing hand movement. This problem was basically design-related where the animation of the virtual hand wasn't clear enough (see the section 7.3 where I discuss how to solve this problem). Participant O also tried to select the object by lifting it. Two participants probably overlooked the place pose which displayed in the middle of the table and tried to set it by dragging the outline of selected object. M tried to program grayed non-parametric instructions before he realized that they are not programmable. Most of the participants had troubles with putting on the headset right.

User Understanding of Robotic Program

The second task which consisted of understanding the *feeder-training* program was tricky for participants that weren't using the headset (**group X**). All of them had to run the *pick from feeder* instruction on the robot to determine whether the robot will be grabbing the objects from the left or right feeder. Participant B was unsure about what the feeder actually is. When one of two *place to pose* instructions are selected, both place poses are displayed where outline of the corresponding place pose to selected instruction is filled while the other one is dashed. This mistaken three participants when they thought that dashed outline corresponds to selected instruction. E thought that the robot will pick up the object from the table instead of the feeder because the object's outline of the type to be grabbed by the robot from the feeder highlighted. F complained about bad readability of the selected instruction description (white text on green background).

Participants in **group Y** had this task quite easier. They just observed animation of the program. None of them had to run the robot. Only two users had to replay the animation because they missed the beginning of it. One of them tried to scroll up the list first before I instructed him that he needs to use the *Replay* button or voice command. M had constant complains about poor field of view of the headset. All participants understand the program without any major problems.

HoloLens Usability

All participants tried out the HoloLens. Biggest issue was its poor field of view and discomfort when wearing it. Participant A had problems wearing it along with eyeglasses. Two participants said that the headset is too heavy and it pushes the nose.

Other two participants appreciated voice controlling. They said that it has an advantage where they could afford to stand a few steps back from the table in order to focus on the visualization. One of them even suggested the extension of full voice controlling of the whole ARTable system.

Based on the quantitative results from presented tables where was reached up to 50 % decrease in time it took participants to understand the robotic program, it can be said that the Microsoft HoloLens is worthwhile using. Once the Microsoft improves the headset's field of view and lightens its weight, it will become much more price-acceptable.

7.3 Suggestions for Future Work

There are plenty of possible extensions of this work. I divided the major suggestions into categories listed below. Those are based on carried out user experience testing and my personal point of view.

Calibration Improvement

Current form of the HoloLens calibration is sufficient but not perfect. As described in section 6.1, it uses only one marker, which often results in imprecise rotation of the coordinate system. These inaccuracies could be eliminated with the use of two additional markers where the current marker is placed in the left bottom corner of the table, the first additional is placed in the left top corner of the table and the second additional is placed in the right bottom corner of the table. These markers would form perpendicular angle where the additional two markers would be used for rotation stabilization of the coordinate system.

Simulation of the Robot Trajectory

As mentioned in section 6.3, current robotic programs visualization is rather an animation than simulation of precise robot movements. Though the robot plans his trajectory real-time, it should be possible to extract the trajectory in advance in order to use it for visualization purposes. Mesh of the robotic virtual gripper could be replaced with the mesh of a whole arm (right and left arm) or even with complete virtual robot. Thanks to calibrated systems, it should not be problem to get exact positions of the robot arms and to render the virtual arms over the real ones.

Proper simulation of robot movement would open new use case for the headset. This would allow the user to verify the correctness of programmed program, which could prevent damaging the real robot, if any undetected obstacles occurred during program execution.

Improvement of the Interactive Helper

Current interactive helper supports only the *pick from polygon* and *place to pose* instructions. Reasonable extension would be the support of all parametric instructions. This helper works only within individual instructions. It would worth trying to connect them into a complex program helper, which would guide the user throughout whole programming process. This would eliminate for example forgotten *Edit* button press.

As the user testing showed up, virtual animated hand was rather confusing than helpful. It should probably display some additional animation when the hand reaches the table (for example some ripple effect to indicate direct touch on the table) and it should be definitely pointing on the object outline, not the object itself.

Other Various Enhancements

Following suggestion are mostly minor patches. This involves support of voice commands for all projected buttons. Object type selection by clicking on the virtual 3D bounding box would be also possible. Or when the user just looks on the virtual 3D bounding box, additional information about it would pop up (its ID, object type, etc.).

As described in section 5.5, the HoloLens could be used as an alternative way of programming the robot. When the robot looks on the feeder in order to detect objects in it, it could be possible to mark a position on such detected object. This position could be transformed into the robot's coordinate system and saved as an initial point for the robot's gripper.

Chapter 8

Conclusions

In this thesis, usage of mixed reality head-mounted display was integrated into a human-robot collaborative workspace – the ARTable. Current 2D projected interface was extended over the visualization mode, which adds new buttons and functionality in order to successfully communicate with the headset. A visualization system that enhances user understanding of robotic programs in the ARTable workspace was implemented. This system enables the user to visualize learned programs by a form of animation, allows him to control this animation by his voice or buttons of the projected interface and guides him by speech synthesizer through whole visualization process. With possibility of controlling the visualization, even an experienced user might find it usable, when he can quickly see program's outcome without the necessity of running the real robot. Use of the headset allows to visualize valuable spatial information, such as robot perception, which was realized by drawing 3D spatial bounding boxes around detected objects.

In order to enhance the human-robot collaboration even more, the interactive helper for robot programming was implemented. This helper guides the user throughout the programming process by speech synthesizer and by drawing animated virtual elements that suggests moves for the user.

Provided solution enables to use multiple headsets, where all of them are synchronized through the ARTable. Thanks to that, they use same information and render same holograms. Solution of the ARTable's part is headset independent.

In order to evaluate provided solution, user experience testing was carried out. It was reached up to 50 % decrease in time it took participants to understand the robotic program when using the headset. However, using the headset for robot programming increased nearly up to 35 % in time it took users to set the parameters for it. Mean SUS score resolved in 67.5, which is within the range of average usability. Biggest downsides of the Microsoft HoloLens appears to be its poor field of view and discomfort of wearing it. However, based on the achieved results, it can be said that the headset is worthwhile using in the ARTable system.

Bibliography

- [1] Beran, V.: *Augmented Reality in Video-Conference System*. Master's Thesis. Brno University of Technology, Faculty of Information Technology. 2003.
Retrieved from: <http://www.fit.vutbr.cz/study/DP/DP.php?id=1202>
- [2] Bishop, G.; Bricken, W.; Durlach, N.; et al.: Research Directions in Virtual Environments: Report of an NSF Invitational Workshop, March 23-24, 1992, University of North Carolina at Chapel Hill. *SIGGRAPH Comput. Graph.* vol. 26, no. 3. August 1992: pp. 153–177. ISSN 0097-8930.
- [3] Bostanci, E.; Kanwal, N.; Ehsan, S.; et al.: User Tracking Methods for Augmented Reality. *International Journal of Human-Computer Studies*. vol. 5, no. 1. February 2013: pp. 93–98.
- [4] Brooke, J.: SUS-A quick and dirty usability scale. *Usability evaluation in industry*. vol. 189. June 1996: pp. 4–7.
- [5] Brooke, J.: SUS: A Retrospective. *Journal of Usability Studies*. vol. 8. February 2013: pp. 29–40.
- [6] Carmigniani, J.; Furht, B.: Augmented Reality: An Overview. In *Handbook of Augmented Reality*, edited by B. Furht. Springer, New York, NY. 2011. pp. 3–46.
- [7] Christian, B.: HoloLens trial gives doctors 'X-ray vision' to allow them to peer inside patients during surgery. WIRED. March 2017. [Online; cit 09.01.2018].
Retrieved from: <http://www.wired.co.uk/article/industries-using-microsoft-hololens>
- [8] Guhl, J.; Tung, S.; Kruger, J.: Concept and Architecture for Programming Industrial Robots using Augmented Reality with Mobile Devices like Microsoft HoloLens. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Sept 2017. pp. 1–4. doi:10.1109/ETFA.2017.8247749.
- [9] Hirzer, M.: Marker Detection for Augmented Reality Applications. Technical report. Computer Graphics and Vision; Graz University of Technology, Austria. October 2008.
- [10] Holmdahl, T.: BUILD 2015: A closer look at the Microsoft HoloLens hardware. [Online; cit 04.01.2018].
Retrieved from: <https://blogs.windows.com/devices/2015/04/30/build-2015-a-closer-look-at-the-microsoft-hololens-hardware/>

- [11] Hopping, C.: Microsoft HoloLens release date, rumours, specs & pricing: Microsoft launches HoloLens in 29 more countries. IT Pro. November 2017. [Online; cit 06.01.2018].
Retrieved from: <http://www.itpro.co.uk/mobile/24780/microsoft-hololens-release-date-rumours-specs-pricing-microsoft-is-creating-ai-chips>
- [12] Huang, J.; Cakmak, M.: Code3: A System for End-to-End Programming of Mobile Manipulator Robots for Novices and Experts. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. HRI '17. New York, NY, USA: ACM. 2017. ISBN 978-1-4503-4336-7. pp. 453–462.
- [13] Kreylos, O.: HoloLens and Field of View in Augmented Reality. August 2015. [Online; cit 09.01.2018].
Retrieved from: <http://doc-ok.org/?p=1274>
- [14] Lozano-Pérez, T.: Robot programming. *Proceedings of the IEEE*. vol. 71, no. 7. 1983: pp. 821–841.
- [15] MacCormick, J.: How does the Kinect work?
Retrieved from: <http://users.dickinson.edu/~jmac/selected-talks/kinect.pdf>
- [16] Malý, I.; Sedláček, D.; Leitão, P.: Augmented Reality Experiments with Industrial Robot in Industry 4.0 Environment. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*. July 2016. pp. 176–181.
doi:10.1109/INDIN.2016.7819154.
- [17] Materna, Z.; Kapinus, M.; Beran, V.; et al.: Using Persona, Scenario, and Use Case to Develop a Human-Robot Augmented Reality Collaborative Workspace. In *HRI 2017*. Association for Computing Machinery. 2017. ISBN 978-1-4503-4885-0. pp. 1–2.
- [18] Milgram, P.; Takemura, H.; Utsumi, A.; et al.: Augmented Reality: A class of displays on the reality-virtuality continuum. *SPIE Proceedings: Telemanipulator and Telepresence Technologies*. vol. 2351. December 1995: pp. 282 – 292.
- [19] Odom, J.: What's the Difference Between HoloLens, Meta 2 & Magic Leap? Next Reality. December 2017. [Online; cit 22.04.2018].
Retrieved from: <https://next.reality.news/news/whats-difference-between-hololens-meta-2-magic-leap-0181804/>
- [20] Rosen, E.; Whitney, D.; Phillips, E.; et al.: Communicating Robot Arm Motion Intent Through Mixed Reality Head-mounted Displays. *CoRR*. vol. abs/1708.03655. 2017.
- [21] Schmalstieg, D.; Höllerer, T.: *Augmented Reality: Principles and Practice*. Addison-Wesley. 2016. ISBN 03-218-8357-8.
- [22] Sluganovic, I.; Serbec, M.; Derek, A.; et al.: HoloPair: Securing Shared Augmented Reality Using Microsoft HoloLens. In *ACSAC*. December 2017. pp. 250–261.
- [23] Stadler, S.; Kain, K.; Giuliani, M.; et al.: Augmented Reality for Industrial Robot Programmers: Workload Analysis for Task-based, Augmented Reality-supported Robot Control. In *2016 25th IEEE International Symposium on Robot and Human*

Interactive Communication (RO-MAN). Aug 2016. pp. 179–184.
doi:10.1109/ROMAN.2016.7745108.

- [24] Toler, L.: Holographic Rovers: Augmented Reality and the Microsoft HoloLens. Technical report. NASA Kennedy Space Center; Cocoa Beach, FL United States. April 2017.
- [25] Wasenmüller, O.; Stricker, D.: Comparison of Kinect V1 and V2 Depth Images in Terms of Accuracy and Precision. In *Computer Vision – ACCV 2016 Workshops: ACCV 2016 International Workshops, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part II*, edited by C.-S. Chen; J. Lu; K.-K. Ma. Cham: Springer International Publishing. 2017. ISBN 978-3-319-54427-4. pp. 34–45.
- [26] Microsoft: Gestures. [Online; cit 07.01.2018].
Retrieved from:
<https://developer.microsoft.com/en-us/windows/mixed-reality/gestures>
- [27] Microsoft: HoloLens hardware details. [Online; cit 04.01.2018].
Retrieved from: https://developer.microsoft.com/en-us/windows/mixed-reality/hololens_hardware_details
- [28] Microsoft: Mixed reality. [Online; cit 08.01.2018].
Retrieved from: https://developer.microsoft.com/en-us/windows/mixed-reality/mixed_reality
- [29] Willow Garage: Hardware Specs. [Online; cit 13.01.2018].
Retrieved from: <http://www.willowgarage.com/pages/pr2/specs>
- [30] WWW page: About ROS. [Online; cit 14.01.2018].
Retrieved from: <http://www.ros.org/about-ros/>
- [31] WWW page: Messages. [Online; cit 14.01.2018].
Retrieved from: <http://wiki.ros.org/Messages>
- [32] WWW page: Nodes. [Online; cit 14.01.2018].
Retrieved from: <http://wiki.ros.org/Nodes>
- [33] WWW page: Services. [Online; cit 14.01.2018].
Retrieved from: <http://wiki.ros.org/Services>

Appendices

List of Appendices

A	Contents of the DVD	50
B	System Usability Scale and Custom Questionnaires	51

Appendix A

Contents of the DVD

/	
thesis.pdf	Text of the thesis.
tex/	L ^A T _E X source files.
src/	Application source files.
ARTableHoloLens/	Unity source files of the HoloLens application.
artable/	ROS packages of the ARTable system.
poster.pdf	Poster of the thesis.
video.mp4	Video of the thesis.
video_long.mp4	Longer version of the video.
README.txt	Text file with instructions of how to install and run the application.

Appendix B

System Usability Scale and Custom Questionnaires

Group: without headset / with headset

Name: _____

Age: _____

System Usability Scale (SUS)

This is a standard questionnaire that measures the overall usability of a system. Please select the answer that best expresses how you feel about each statement.

	Strongly Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Strongly Agree
1. I think I would like to use this tool frequently.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2. I found the tool unnecessarily complex.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. I thought the tool was easy to use.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4. I think that I would need the support of a technical person to be able to use this system.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5. I found the various functions in this tool were well integrated.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6. I thought there was too much inconsistency in this tool.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7. I would imagine that most people would learn to use this tool very quickly.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8. I found the tool very cumbersome to use.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9. I felt very confident using the tool.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
10. I needed to learn a lot of things before I could get going with this tool.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Would you like to have any additional comments?

Figure B.1: System Usability Scale questionnaire.

	Strongly Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Strongly Agree
1. I didn't like low Field of View of the headset. I had to look for holograms for a while.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2. The headset was uncomfortable, and I would definitely mind wearing it for a long period.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. The headset's speech synthesizer was easy to understand.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4. Controlling by voice was easy and intuitive.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5. I would rather control the system with buttons than by voice.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6. After seeing the program visualization, I knew what a real robot would do.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7. I used the headset helper when programming the robot. Thanks to it, I have been able to program the robot easily.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8. Sometimes I didn't know what to do.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Would you like to have any additional comments? (what bothered you, suggestions for improvement,...)

Figure B.2: Custom questionnaire.