BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Mixed Reality in Semi-Autonomous Systems

Habilitation Thesis

Ing. Vítězslav Beran, Ph.D.

Brno 2022

Abstract

This thesis presents research in the area of mixed reality application in natural human interaction with semi-autonomous systems. It explores two areas of applications: programming collaborative robots in particular and piloting UAVs. The goal of the research is to find an appropriate way of communication between human and machine so that both human and machine, given today's technology, do the activity in which they are most effective and complement each other. The work does not focus on automatic methods of data processing or autonomous decision-making but on human interaction with systems for which today's technologies allow a degree of autonomy. In the area of collaborative robot programming, it introduces a new concept of programming in a shared 3D workspace that aims to make this activity easier for less experienced programmers while reducing the mental load on the user during this activity. The research results demonstrate their application not only in academia but especially in industry, where the concept has already been successfully deployed. In the field of UAV piloting, the presented preliminary research focuses on using mixed reality to enhance pilot orientation in the environment and appropriate interaction when using the results of automatic sensor data analysis.

Keywords

Mixed Reality, Collaborative Robots, UAV, Spatial Programming, Spatial Augmented Reality, Augmented Reality, Augmented Virtuality, Spatial Awareness, Drone Piloting, Ergonomy, Safety

Acknowledgment

I would like to thank my advisor and mentor Pavel Zemčík, who started my academic career and who always supported me in my work, and together with Pavel Smrž, they constantly nudged me forward with their ideas and projects and supported me in building my research group. Also to Michal Španěl, with whom I shared the joys and sorrows of our scientific journey in the office, and Adam Herout, who patiently advised me on practical scientific issues. Many thanks to my colleagues and co-authors with whom we conduct our research in the presented area, especially Michal Kapinus, Daniel Bambušek, Zdeněk Materna, my former colleague Alfredo Chavez Plascencia and a number of students. I would like to thank my parents, who have always supported me on my life's journey, and especially my wife Eva, who has had to listen to dozens of hours of lamentations and complaints over the years, and despite this, has always been a loving and patient support in everything.

Over time, I was supported by a number of grant projects, in particular by the Ministry of the interior and Ministry of Industry and Trade of the Czech Republic, EU ECSEL and Technology Agency of the Czech Republic.

Contents

1 Introduction

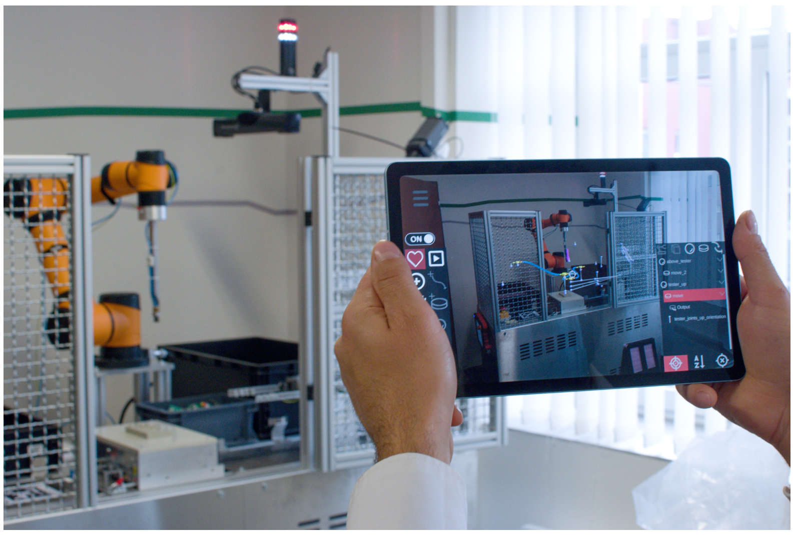
We live in a time when we are surrounded by machines – from mechanical to digital. These machines have a purpose, a function that makes our lives easier – from physical work to computational tasks. For humans to use machines, there has to be a way to operate them, control them, and give instructions. At the same time, we need to know the state the machine is in – to see the direction of movement, the temperature of the motor, and the result of the calculation. This interface between man and machine is an absolutely essential element for the effective use of machines by humans.

Nevertheless, what exactly does the term *efficient* mean in this context? Efficiency can be viewed from many angles. It can be *an amount of time* it takes to perform the desired task with a machine. For example, a machine may be able to perform its function quickly, but the number of control steps or the time to operate the machine may be high, and the speed advantage of the machine itself is reduced. There may also be *several errors* in operating or monitoring the machine. Controls may be imprecise, the order of their use unclear, and information from indicators and pointers challenging to interpret and complex. All of this can lead to errors in operating the machine and frustration for the user. It can also be a measurement of *the mental load* of an operator carrying out the machine control. The many controls, indicators and pointers can be very burdensome and exhausting for the machine user. This soon causes the user to lose focus and increases the risk of injury or damage to the machine.

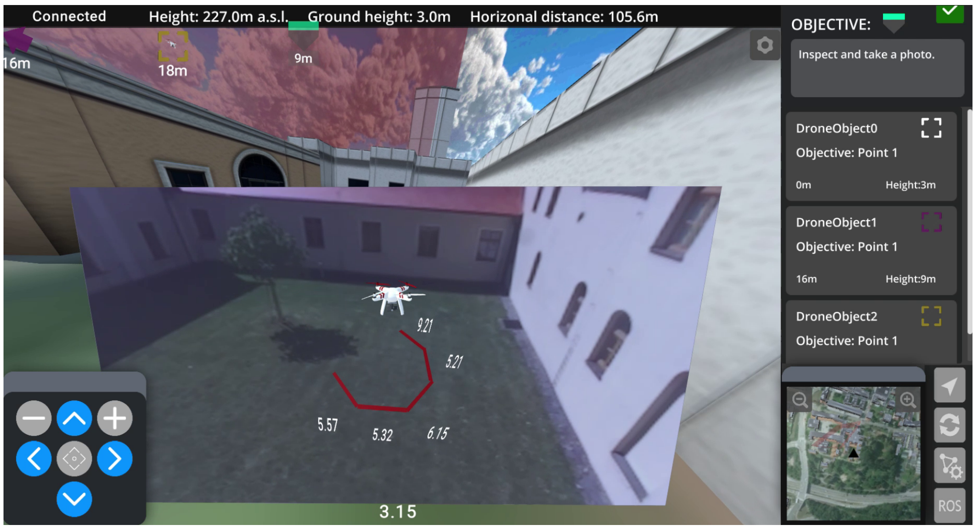
Today, the vast majority of mechanical machines are already equipped with measuring sensors and some computer system that allows the measured data from both the machine and its surroundings to be automatically processed and evaluated in a meaningful way. The measurement results can then be presented to the user or used directly by the machine for automatic self-control. The manufacturing industry today makes full use of automatic machines. Yet it still needs humans for many tasks. Unfortunately, these tasks are trivial in decision-making or creativity, primarily mechanical and repetitive. Creating single-purpose machines is inefficient in many cases; the degree of variability of conditions is higher and requires frequent changes. One solution may therefore be machines that can be more easily adapted to a new task and operate in close proximity to a human – something that most single-purpose automatic machines in the industry today do not allow for safety reasons. These collaborative robots bring the possibility to make production more accessible and cheaper by combining the capabilities of a human and a robot. The robot performs monotonous, repetitive tasks, and the human makes decisions in uncertain situations, modifies the robot's actions or performs complex manipulation tasks for the robot. To make this *human-robot collaboration* possible, it is necessary to have an interface that allows human-robot communication in *an efficient* way. Communication here means passing information about what exactly is to be done at what place in space, monitoring the current activity of each involved subject, informing about the activity just underway or planned, and indicating the system's internal state.

Both the human and the robot work in a real environment, i.e. in 3D space. Therefore, the ideal space for communication is their *shared workspace* where the activity is taking place. This space *is observable and measurable* by both the human and the robot, and it is the space *where both can reach and do their activity.* Current information technologies already offer partial solutions to enable human interaction with the digital (virtual) world in a 3D environment. *Mixed reality* is one such technology that brings the ability to correctly display digital data (virtual world) into the real world. There are also several technologies that allow the user's actions to be tracked so that the user can interact with the virtual world.

Both humans and robots, in their work, need to *perceive* and to some extent *understand the environment* in which they are working, i.e. create some interpretation of the shared environment (the human mental and the robot digital map of the environment), and then share this interpretation with each other. Further, they need to communicate the process of their or their joint work – the work process in the environment or the robot's program. It is essential that the human is then able to *control and manipulate* the knowledge and plans that the robot creates and according to which it will make further decisions.



(a) Robot-operator observing the program for the PDB testing task in augmented reality (Kapinus et al., 2022c).



(b) Drone pilot GUI application based on augmented virtuality (DroCo)[1].

Figure 1: Examples of UI using mixed reality for human interaction with semi-autonomous systems.

This work specifically explores the possibilities of using mixed reality for interaction in 3D environments between humans and semi-autonomous systems. These are systems

[1]https://www.fit.vut.cz/research/product/647/

that are able to perceive their environment to some extent and control their actions semi-autonomously based on predefined tasks. The essential thing is to find a natural and effective way of communication between the human and the semi-autonomous system (robot, UAV, etc.) so that the human can easily understand what the robot knows and does and can effectively intervene and control its actions. In addition to the main focus on human interaction with collaborative robots in mixed reality, the use of MR in the field of piloting UAVs (or drones) is also presented (see illustrative images on Fig. 1).

This habilitation thesis is conceived as an annotated collection of previously published articles of the Robo@FIT research group[2] under my leadership and/or students' work under my supervision. I have selected the articles according to their relevance to my research area concerning the use of mixed reality in semi-autonomous systems. The selection of specific articles is summarized in the introductory of Section 3 and Section 4. The specific contributions of the articles are then presented in the individual subchapters. All selected articles relevant to this thesis are then included in Appendix. In Section 2, where I put the reader more in context, I briefly introduce the broader range of technologies and scientific areas. The knowledgeable reader of the issues can skip this chapter.

[2] `https://www.fit.vut.cz/research/group/robo/`

2 Background Technologies and Scientific Areas

In this chapter, I will introduce the reader to a number of technologies and scientific areas that are key to research in the field of mixed reality and interaction with collaborative robots or UAVs. The knowledgeable reader of the subject matter can skip this brief introduction to the technological context and continue withSection 3. Most of the presented areas are multi-disciplinary and encompass a range of other technical and scientific disciplines. The summary is intended to provide the reader with a concise introduction to relevant approaches and technologies.

2.1 Mixed Reality

The digitisation of the real world has been going on since the first computers were built in the 1950s. From office documents, digitisation has now reached the point where we can digitise and simulate a large variety of different parts of the real world in near real-time. So we have a vast amount of heterogeneous data available to us in digital form. In today's modern world, we are also surrounded by computers[3] – from smart watches, cameras and regular computers to vehicles, manufacturing machines, smart infrastructure, etc. From the inception of computers to the present day, one research challenge still remains, **how to create an interface for humans to interact with this digital world** (both data and computing power) naturally and effectively. So that humans can easily and quickly submit tasks and queries to computers and so that the results of the computer's work can be presented to the user in a clear and understandable way. Without realising it, we are still using concept from the 70s (WIMP[4]) or 90s (post-WIMP[5]) today to communicate with the digital world. New concepts for human-machine interaction that are still finding their massive application include, among others, mixed reality.

Mixed reality technology[6] is based on the idea of linking the real and digital worlds by displaying virtual objects in such a way as to create the illusion of the existence of these objects in the real world. The ratio of real and virtual worlds may vary (see reality-virtuality continuum on Fig. 2). On the one hand, virtual reality (a completely artificial world) can be supplemented with data from the real world (augmented virtuality). On the other hand, the real world can be supplemented with virtual elements (augmented reality). The key is that physical and digital objects coexist and interact with the user in real-time. There is also a definition of mixed reality that extends it further and says that the synthetic content and the real-world content are able to react to each other in real-time, or other definitions like an extended reality (XR) that connects all AR, VR, MR or spatial computing[7].

Several technologies are available to implement mixed reality. The essential thing is to know the location and direction of the user's view (head pose or display device pose) in the real world. The image of the virtual world must then be generated in such a way that the user observing the scene of the real world, supplemented by the display or projection of the image of the virtual world, obtains the illusion of the real occurrence of virtual objects in

[3]`https://en.wikipedia.org/wiki/Ubiquitous_computing`

[4]WIMP, stands for "windows, icons, menus, pointer", was developed at Xerox PARC in 1973, see more at `https://en.wikipedia.org/wiki/WIMP_(computing)`

[5]Post-WIMP consider new conditions brought by mobile devices (small screen, gestures) (Nielsen, 1993)

[6]`https://en.wikipedia.org/wiki/Mixed_reality`

[7]`https://en.wikipedia.org/wiki/Spatial_computing`

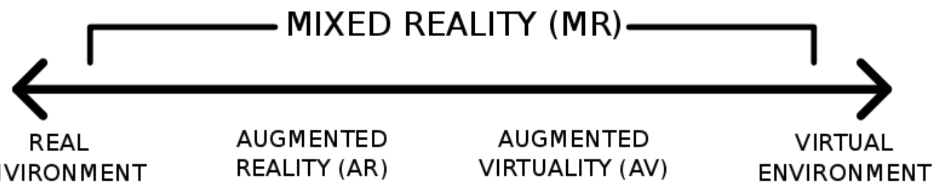


Figure 2: Reality-virtuality continuum (Milgram et al., 1994).

the real world. Among the main technologies providing the **visualisation**, we can name: head-mounted displays (e.g. Hololens glasses), mobile devices or projected augmented reality (see Fig. 3). A key aspect is a possibility of interacting with virtual objects in mixed reality. Leaving aside very specific approaches and devices, widely used technologies for **interaction** in MR are, e.g.: surface gestures (e.g. touch-screens, mobile devices, or any flat surface covered by touch-sensitive foil), free-space gestures (HoloLens, Leap Motion or similar), VR gloves, wearable sensors (e.g. Myo Armband) etc. (see Fig. 4).

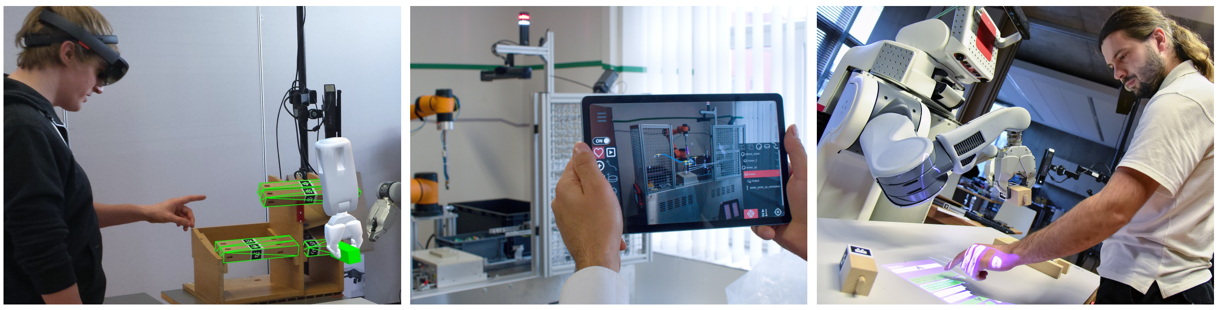


Figure 3: Examples of augmented reality visualisation technologies (from left to right): head-mounted display (Microsoft HoloLens), mobile device and projected AR.

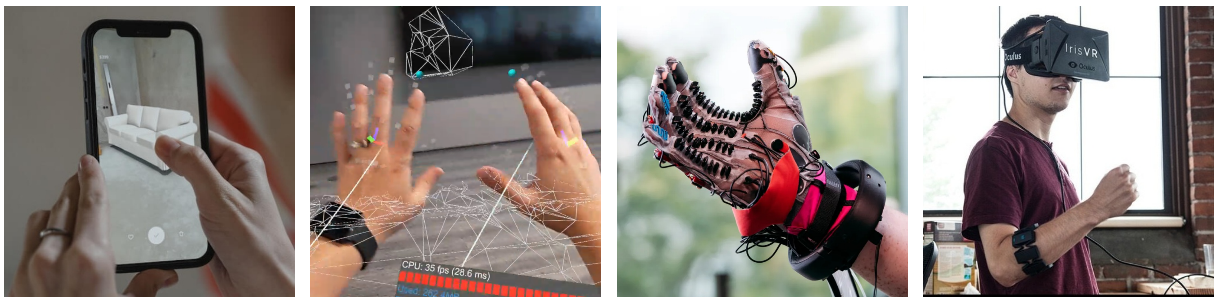


Figure 4: Examples of user-input technologies (from left to right): surface gestures e.g. on mobile devices[8], free-space gestures e.g. by HoloLens[9], VR gloves e.g. Meta prototype[10]and wearable sensors e.g. Myo Armband[11].

[8]Photo by Tima Miroshnichenko from Pexels, https://www.pexels.com/photo/person-holding-an-iphone-6474485/

[9]https://github.com/microsoft/MixedRealityToolkit-Unity/issues/4990

[10]Meta Reality Labs Research, Published on Nov 16, 2021 (e.g. https://www.protocol.com/meta-haptic-gloves)

[11]Posted by Jack Donovan on Sep 5, 2014, https://blog.irisvr.com/blog/uncategorized/vr-input-myo-armband

2.2 Collaborative robots and programming

The historical and current practice in production automation is to create a closed automated system where machines are more or less single-purpose. Such a solution is very efficient in terms of work speed and accuracy. The key disadvantage is its almost zero adaptability to other tasks. This approach is practical to large enterprises where the production of a given product is expected to run for several years, and the investment in the machinery will pay off. Small and medium-sized enterprises (SMEs) produce smaller batches and therefore need a more general functionality solution that can often be reconfigured to a new automated task. Collaborative robots (see examples on Fig. 5) are such machines that can perform more general handling tasks (see the forecast for collaborative robot applications on Fig. 6) and can also work closer to humans. Compared to conventional robotic manipulators, they generally have less force, operate at lower speeds, have rounded edges, and are equipped with technology that instantly stops the robot's motion in the event of an unplanned collision with a human or other objects.

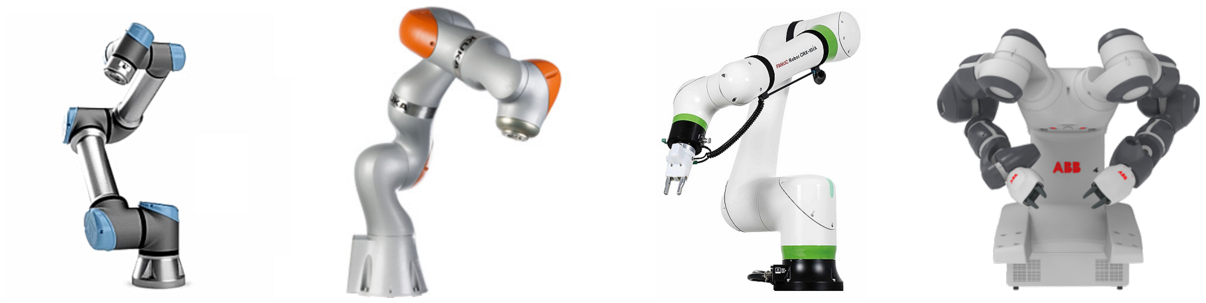


Figure 5: Examples of collaborative robots (from left to right): Universal Robots (UR), KUKA, FANUC a ABB Yumi.

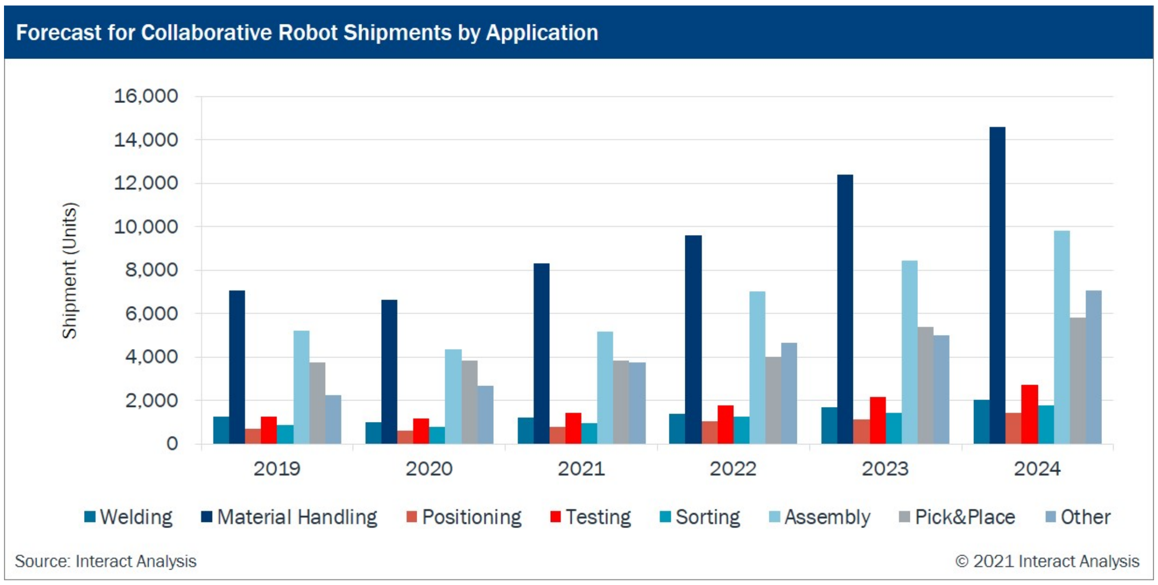


Figure 6: Forecast for collaborative robot shipments by application[12].

[12]`https://www.roboticstomorrow.com/article/2021/02/the-collaborative-robot-market-2021%E2%80%9328-grounds-for-optimism-after-a-turbulent-two-years/16190/`, published by Maya Xiao, Research Analyst, 2021

Although sales of collaborative robots are slowly increasing compared to conventional robots (see Fig. 7), the deployment of collaborative robots in small and medium-sized enterprises is still relatively slow. Indeed, the rapid reconfiguration of even such a system still has many obstacles. One of them is the programming of the robotic system, which requires professional programming skills. Enabling ordinary workers to program collaborative robots (or to program in general) is the goal of a number of research efforts. Besides traditional program source-code writing, some of the easier and more natural concepts are already being used in practice (see Fig. 8), such as visual programming, kinesthetic teaching or learning from demonstration (aka Programming by Demonstration).

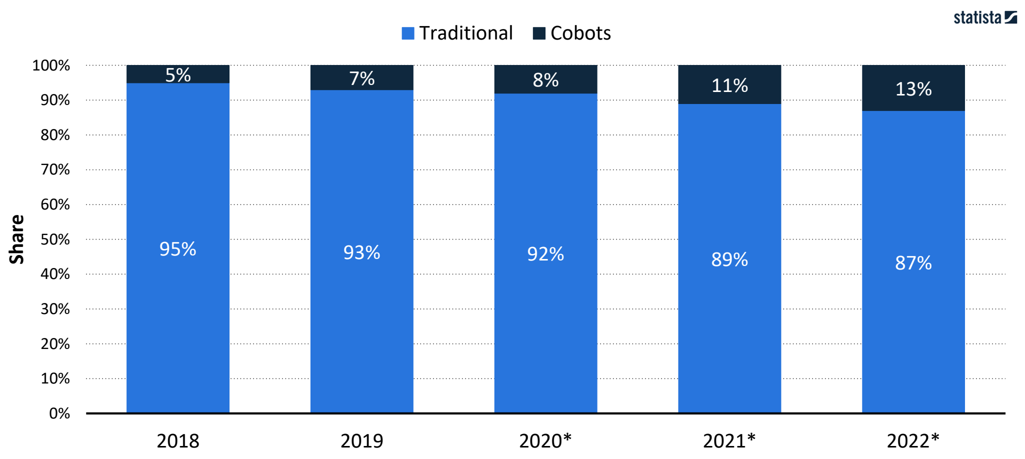
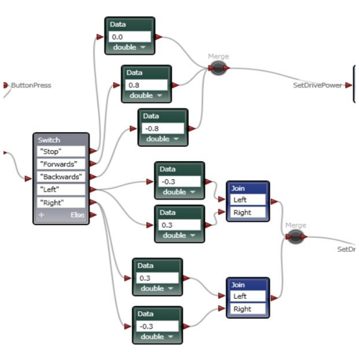
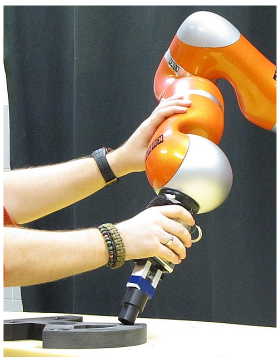


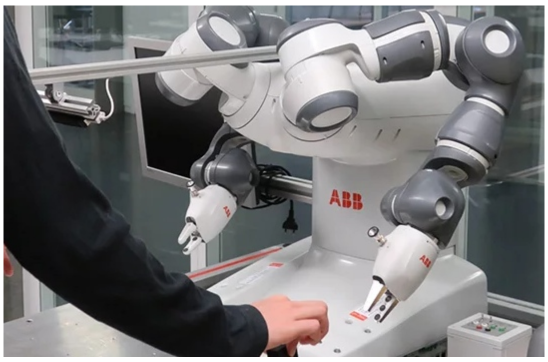
Figure 7: Share of traditional and collaborative robot unit sales worldwide from 2018 to 2022[13].



(a) Visual programming in Microsoft Robotic Developer Studio[14].



(b) Kinesthetic teaching example.



(c) Programming by Learning (or Demonstration)[15].

Figure 8: Examples of advanced approaches of robot programming.

Although research in the area of human-robot interaction in programming has produced some approaches that simplified and make programming more natural, such as

[13]https://www.statista.com/statistics/1018935/traditional-and-collaborative-robotics-share-worldwic published by Martin Placek, Feb 25, 2021

[14]https://en.wikipedia.org/wiki/Microsoft_Robotics_Developer_Studio

[15]https://www.hes-so.ch/en/recherche-innovation/research-projects/detail-projet-recherche/learning-from-demonstration-for-collaborative-robot

visual programming, kinesthetic teaching or learning from demonstration, there are still unresolved challenges on:

- enable programming of collaborative robots by the shop-floor worker,
- reducing the mental load of the robot operator,
- improving the orientation in the program,
- adapting the program to new conditions effectively,
- interacting naturally in real 3D space.

2.3 Unmanned Areal Vehicles (UAVs/drones)

Due to their high cost, unmanned areal vehicles (drones) were previously used mainly for military purposes. Technological advances, which have significantly reduced production costs and increased the safety of the operation of these machines (mainly by reducing weight), have contributed to their spread to other industrial spheres. Today, UAVs can be bought literally for a few bucks and operated even by children.

There are a number of different types of UAVs, which differ significantly in their design (fixed-wing or rotor-craft) and thus in their flight characteristics – flight time, manoeuvring limits, maximum payload, etc. (see Fig. 9). However, they also differ in the range of degree of autonomy – from remotely piloted or semi-autonomous (containing assistance functions) to fully autonomous.

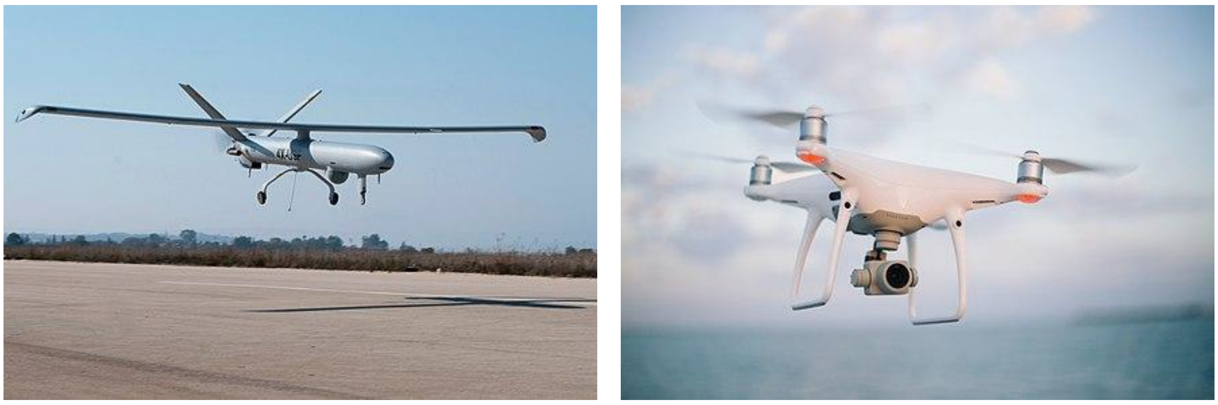
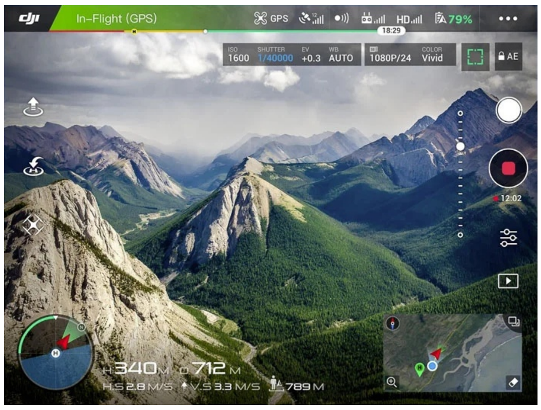
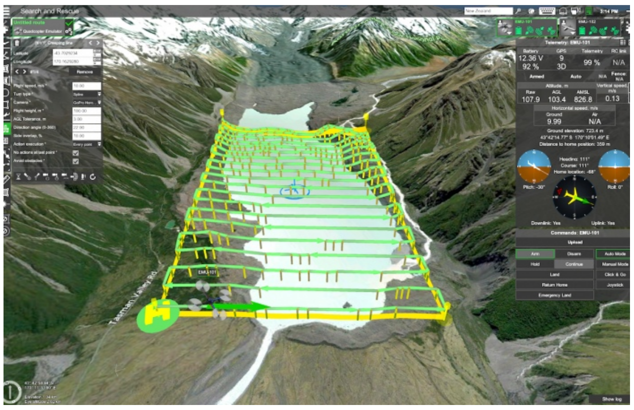


Figure 9: Examples of typical UAVs: fixed-wing UAV (left) and quadrotor UAV (right).

Remote piloting uses a radio controller (RC) or virtual joysticks on the mobile device, where the pilot uses direct sight of the drone for feedback and the now standard extension of video feed from the drone camera and flight information displayed on the mobile device display or controller (Fig. 10(a)). Semi-autonomous systems, thanks to their ability to analyse their immediate surroundings (impact prevention, object avoidance), can then fly along a specified trajectory (or checkpoints) autonomously (Fig. 10(b)), or use the results of automatic sensor data analysis (e.g. positioning based on object tracking) for interaction. Applications for the setup and control of monitoring missions of semi-autonomous systems allow to monitor flight information and video transmission from multiple drones and efficiently coordinate the whole squadron's activity. Interaction with these semi-autonomous systems consists of setting their planned path, setting synchronisation elements when multiple drones are used in a single mission, etc., and this interaction is done using a common GUI on mobile or desktop application.



(a) GUI with first-person view and sensory data from flying drone[16].



(b) GUI with virtual view based on off-line 3D maps, control points and trajectories, extended by flying data[17].

Figure 10: Examples of GUI applications for drone pilots and operators.

UAVs can be very useful in situations where there is a need to explore hard-to-reach locations quickly or deliver a package to such a location, monitor an activity from a safe distance, or carry out an action in a hazardous area directly. An example of this would be a rescue operation on a swollen river where drowning occurs. A drone can help to map remote parts of the river quickly, locate the positions of drowning people, explore access routes or even deliver rescue or medical aid. In these cases, an experienced pilot is needed who can control the drone safely, even in situations where there is no direct visual contact with the drone. However, such an operation can be mentally challenging even for an experienced pilot, who must control a drone in a complex environment, not endanger any persons and ideally not destroy the drone or close infrastructure.

Already in 2005, the Air Force Research Laboratory's Human Effectiveness Directorate (AFRL/HE) supported research addressing human factors associated with Unmanned Aerial Vehicle (UAV) operator control stations. One of the ways was to use a synthetic vision system that combined virtual vision data with live camera video presented on a UAV control station display, e.g. (Calhoun et al., 2005) (see Fig. 11). The solution is actually an augmented reality, where virtual information is constructed from databases (e.g., terrain, cultural features, pre-mission plan, etc.), as well as numerous information updates via networked communication with other sources and overlaid in real time onto the dynamic camera video image display presented to operators. Synthetic vision overlay technology is expected to improve operator situation awareness by highlighting key spatial information elements of interest directly onto the video image, such as threat locations, expected locations of targets, landmarks, emergency airfields, etc.

In addition to military purposes, drones are now used in a wide range of other applications, such as monitoring undergrowth fires, infrastructure inspection, surveillance, package delivery, aerial photography and, last but not least, entertainment. Requirements

[16]`https://www.dji.com/cz/downloads/djiapp/dji-go-4`

[17]`https://www.unmannedairspace.info/latest-news-and-information/sph-engineering-launches-centralised-drone-flight-management-software/`

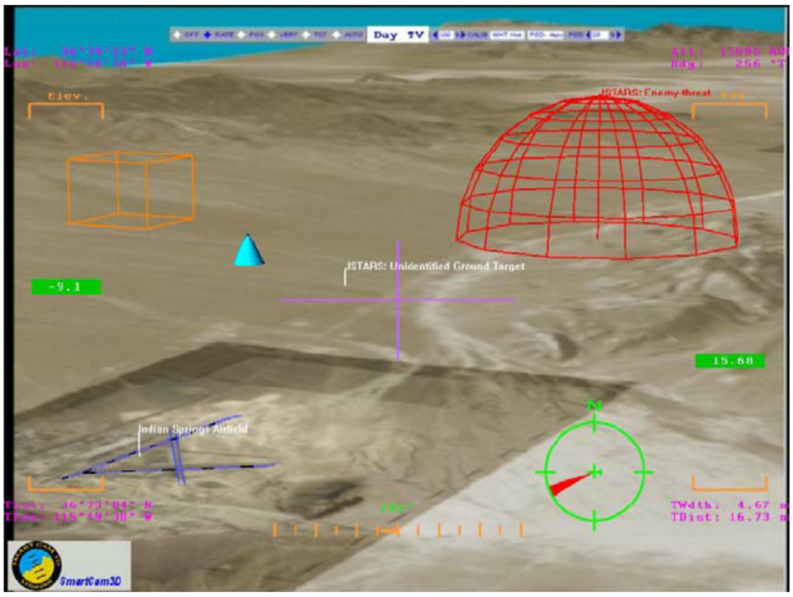


Figure 11: Synthetic vision symbology added to simulated UAV gimbal video imagery (symbology marking threat, landmarks, areas of interest and runway) (Calhoun et al., 2005).

for low-risk or high-precision operations would be addressed by one or two extra cameras monitoring the scene from different angles and distances. Inspiration in that direction might be, e.g. work (Temma et al., 2019), presenting a third-person view solution using a second drone, including innovative UI for spatial configuration of the second drone's position (see Fig. 12).

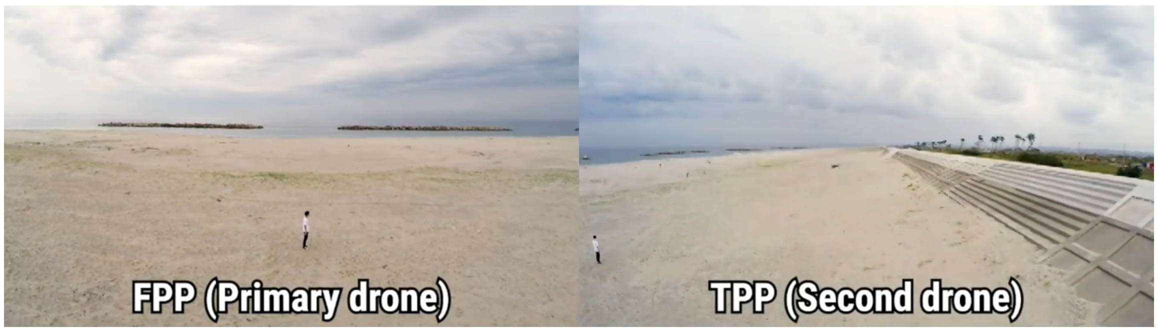


Figure 12: Demonstration of a third-person view solution using a second drone, including innovative UI for spatial configuration of the second drone's position (modified from (Temma et al., 2019)).

In many cases, autonomous systems cannot be used safely (or legally), and drones' remote control is required, often supplemented by semi-autonomous solutions. Besides the research challenges in data processing, there is again an excellent scope for improving the communication between the pilot and the machine (drone), and more precisely between the pilot and generally some computing machine that analyses and evaluates the data measured by the drone in real-time. Communication that will:

- facilitate the pilot's control and reduce the risk of an accident by improving his orientation in the real environment in which the mission takes place and in which the drone moves,

- increase the distance at which the drone can be operated safely, and
- reduce the overall mental burden on the drone pilot.

2.4 Human-Computer Interaction Research

Man invents and creates machines. They have a function that benefits people. To make good use of a given machine, man must control it. This is what the machine-human interface is for. One such machine is a computer, which works with data in digital form. The discipline of *human-computer interaction* is concerned with the design and use of computer technology by human. A sub-discipline of the field is *user-centred design*, which focuses on user needs and experience when designing interfaces. It offers tools and methods to identify user needs by observing and analysing user behaviour, to process these needs and derive user interface designs from them, create mock-ups (and then prototypes) of new interfaces, perform tests on these mock-ups including evaluation of observed and measured attributes, and make design modifications from the test findings.

Standard techniques then include *personas, scenarios and use-cases.* A *persona* is used to describe a fictitious person who is an archetype of the user of the proposed interface. It helps to create a clear and shared image of a typical user. It guides the selection of different design options and elements of the proposed interface. The *scenario* is a story describing the situation of the previously created persona, which must be specific to the problem the proposed user interface intends to solve. The context described in the story is more understandable and can be better followed for analysis and design. The *use-case* is a description of specific events composed of a series of step-by-step actions towards the desired goal.

In user testing of user interfaces, it depends on the goals that are required in a given context. An essential attribute of user interfaces is the user's mental workload when interacting with the interface (NASA TLX (Hart and Staveland, 1988)). When designing interfaces for new technologies or using new technologies for interaction, the level of user expectation and then the user's actual experience is often examined (SUXES (Turunen et al., 2009). This method is particularly useful when dealing with multimodal systems. In more general cases, it might sometimes be sufficient to find only the user experience (UEQ (Schrepp, 2015)). In the case of new interface concepts, where it is difficult to evaluate their properties by comparing them with another interface (which may not exist at all), it is still necessary to evaluate the usability of the new solution (SUS (Brooke, 1996)). There are several testing protocols and questionnaires, even in specific domains such as AR applications, e.g. HARUS (Santos et al., 2014). The methods mentioned above are probably the most commonly used in the research area presented here.

This brief introduction to the context of the research introduced, in addition to particular semi-autonomous systems (collaborative robots and UAVs) and mixed reality technologies, especially the research methods in the field of human-computer interaction used in the research presented further.

3 Spatial Robot Programming

If a robot is to be beneficial to man, man must somehow order it what action to perform. The definition of the activity must contain, first of all, information about *what* the robot is supposed to do, *where* it is supposed to do it, and *how* are the other specific parameters of the particular operation set. An example is the grasp operation, where it is necessary to specify where in space the robot is to grasp an object, but also what force to use to hold it or from what direction to grasp it. This research focuses on working with collaborative robots, so the following text reflects this context. Programming more extensive multi-robot solutions is a topic not addressed in this research.

Apart from the standard way of defining robot actions, such as programming textual source code, there are now various alternatives with their pros and cons, ranging from visual programming and kinesthetic learning (see Fig. 8) to a dialog audio system, for example. The resulting program for the robot is then a sequence of parameterised actions in space. A human must communicate this program to the robot. Define the individual actions and their locations in plain space, the parameters of each action, etc. However, the user also needs to see what such an entire created program looks like, revise the actions in the program, and edit all parts of the program.

This programming activity is often done on a desktop or laptop computer or a teach-pendant[18]. Both of these approaches are effective for different needs. The teach pendant is mainly used to manipulate the robot directly but also to create the actual program itself. However, navigating the program is quite difficult. Applications for programming robots on desktop computers are essential for programming larger complex units and contain a large number of specific tools and functions – from custom programming to modelling and simulation tools. The disadvantage is often the complexity of use for a non-professional programmer and then the need to mentally map the robot model and end-effector positions to a real 3D environment. This mental mapping of the program into real 3D space happens not only during programming but especially when controlling the robot while it is working or during the revision or adaptation of the program already in the workplace. Then, an additional mental workload for the user arises due to the constant context switching between the real 3D environment of the robot workplace and the interactive device (monitor, keyboard or touch screen of the teach-pendant).

One way to solve some of the problems described above is to transfer programming from the 2D GUI of the computer directly to the robot workspace. Working with the program – creating, revising or modifying it – will take place in the real 3D environment of the actual robot. This requires technology that allows digital data (the program) to be represented and displayed in specific locations in the real world and to interact with this representation. Augmented reality technology meets these requirements.

The following chapter is based on the results of our research and draws on the following articles:

- **Simplified Industrial Robot Programming: Effects of Errors on Multimodal Interaction in WoZ Experiment** (Materna et al., 2016), explores and extends the understanding of the impact of errors in 6 different modalities on user experience and expectations in human-robot interaction context (Section 3.1).
- **Using Persona, Scenario, and Use Case to Develop a Human-Robot Augmented Reality Collaborative Workspace** (Materna et al., 2017), introduces

[18]control panel for robot motion programming

the use of HCI methods in an HRI context (Section 3.2).

- **Interactive Spatial Augmented Reality in Collaborative Robot Programming: User Experience Evaluation** (Materna et al., 2018), presents a solution for displaying information in the context of a robot task, both in the programming and processing phases (Section 3.2). It combines interactive spatial AR (ISAR), kinesthetic teaching and object detection to relieve the user of the problem of context switching between the computer and the robot environment without a hand-held device (free-hand solution). The system usability score (SUS) of the new solution is 75.80 (SD 8.90) compared to the reference (Huang and Cakmak, 2017) 66.75 (SD 16.95).

- **Combining Interactive Spatial Augmented Reality with Head-Mounted Display for End-User Collaborative Robot Programming** (Bambušek et al., 2019), seeks to address the limitations of interactive spatial AR (ISAR) technology that cannot provide free-space 3D visualisation (Section 3.2). Knowing also the limitations of HMD-based AR (limited field of view, single-user visualisation), we design a new UI concept that overcomes the known limits of these technologies. Experimental validation in a robot programming task demonstrates an improvement in the usability of novel UI over stand-alone ISAR in qualitative measures by 33.84% and quantitative measures by 28.46%.

- **End-User Robot Programming Case Study: Augmented Reality vs. Teach Pendant** (Kapinus et al., 2020)[19], presents a preliminary 2-condition (traditional teach-pendant and ISAR) within-groups case study (Section 3.2). Preliminary results indicate the potential of our ISAR UI because all used standard metrics, objective (time to accomplish the task) and subjective (TLX, SUS and UEQ) metrics, were better for our new ISAR solution.

- **Spatially Situated End-User Robot Programming in Augmented Reality** (Kapinus et al., 2019), seeks solutions to UI limitations with SAR and HMDs using AR on mobile devices. This loses the hands-free advantage, but mobile AR brings many other benefits (especially speed and user experience) as shown by research results, e.g. (Dass et al., 2018; Stadler et al., 2016; Magnenat et al., 2015). It introduces a new concept of representing a robotic program in real 3D space (Section 3.3). The potential of the new solution is demonstrated by an experimental evaluation of the system usability score (SUS), where a score of 82.86 (SD=9.29) is in class A and is at the 90–95th percentile (Sauro and Lewis, 2012) and low mental load (NASA TLX) 27.38 (SD=9.41), which means that the workload was lower than in at least 80% of studies analyzed (Grier, 2015).

- **Improved Indirect Virtual Objects Selection Methods for Cluttered Augmented Reality Environments on Mobile Devices** (Kapinus et al., 2022a)[19], focuses on solving the problem of selecting virtual highly occluded objects with large spatial distribution variability and heterogeneous size and appearance. Preliminary results show a promising way of indirect selection that increases the selection accuracy (Section 3.3).

[19]Late-Breaking Report

- **Augmented Reality Spatial Programming Paradigm Applied to End-User Robot Programming** (Kapinus et al., 2022b)[20], addresses the problems of effective use of AR on mobile devices for user interaction in robot programming, GUI ergonomics and precise positioning of virtual 3D points in real space. In addition to the successful use of different interaction modes for improved GUI ergonomics, the concept of accurate positioning of virtual objects using a combination of kinesthetic teaching and relative positioning is a crucial attribute of the new prototype (Section 3.3). The experiment was designed as a within-subject user study, comparing the two different interfaces – new prototype and standard visual programming. The task also included precise positioning of robot actions in real 3D space. Both program orientation and new program-creating tasks were significantly faster using the new prototype. According to subjective metrics, the new prototype is better, but not significantly.

- **ARCOR2: Framework for Collaborative End-User Management of Industrial Robotic Workplaces using Augmented Reality** (Kapinus et al., 2022c)[20], explores the usability of the new prototype in an industrial environment, including the applicability of precise positioning, multi-user collaboration, the appropriate level of abstraction for end-users, and the usability of the handheld AR modality (Section 3.3). The evaluation on PCB testing use case was performed by three experts and indicated the high potential of the solution.

- **How Do I Get There? Overcoming Reachability Limitations of Constrained Robotic Workspaces in Augmented Reality Applications** (Bambušek et al., 2022)[20], the research responds to situations in real industrial environments where not all parts of the scene are accessible, and programming in AR is therefore limited. At the same time, it investigates the problem of user fatigue when programming more complex programs in AR only. The new method uses VR in addition to AR and the ability to switch freely between each *reality*. This switching negatively affected the mental workload of the user, but the physical workload was statistically significantly lower when using both AR and VR, instead of AR only (Section 3.4).

[20]publications submitted

3.1 Modalities and their error effect

When looking for a suitable way of direct human-robot interaction in a 3D working environment, it was first necessary to determine what modality is suitable for the user, especially regarding the effect of the error rate of the particular modality on the user experience and efficiency of use. This reasoning is based on the experience that, given the maturity of modern technologies suitable for advanced UI in space and especially the distractions in the real environment, UI cannot be expected to be error-free. Such errors can be understood as, e.g. a wrongly represented gesture, an incorrectly selected virtual element, etc. The experiment was conducted with three levels of error rate: 0% (ideal unreachable condition), 10% (realistic condition) and 30% (worst case).

SAR[21] technology was used for the experiment, complemented by kinesthetic teaching. The UI elements that enabled the program's manipulation and visualization elements presenting the *robot system's knowledge* of the scene were projected into the working environment. The user could interact with the system using different modalities: touch table-top, touch screen, gestures, 6D pointing device and direct robot-arm manipulation. The experiment was implemented using a Wizard-of-Oz (WoZ) approach, where a hidden operator implemented a simulated system response without the user's knowledge. The speech was not used due to the unsuitability of this modality in real robotic system environments where high levels of noise and interference can be expected. The simulated working environment for the implemented experiment can be seen in Fig. 13.

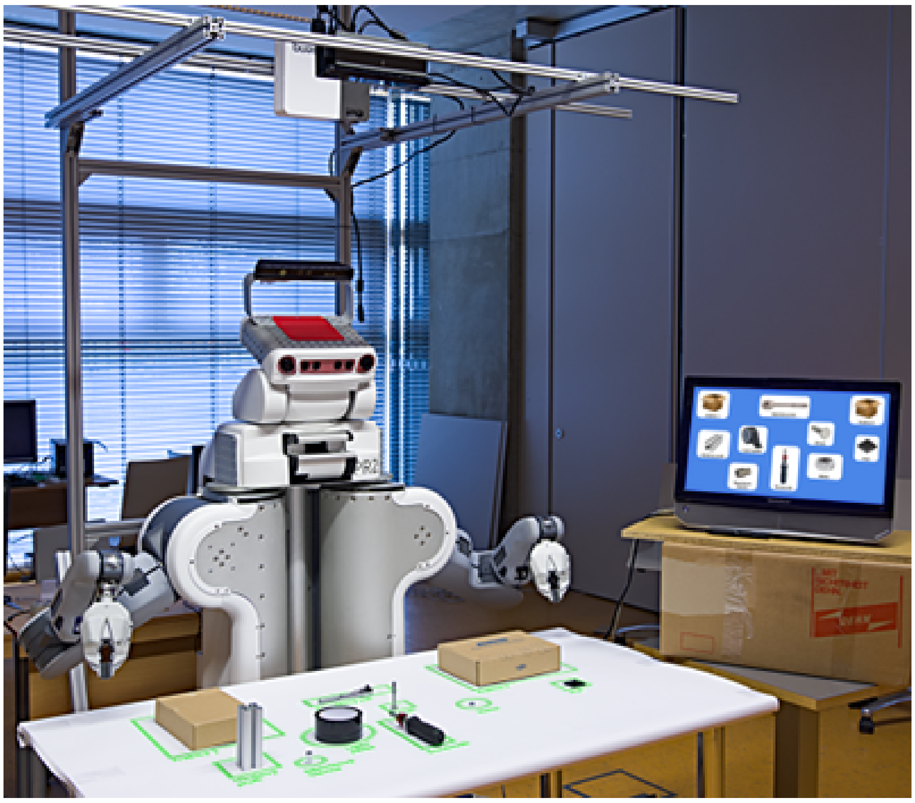


Figure 13: Prototype of the human-robot shared-space environment with augmented reality user interface (Materna et al., 2016).

The 39 participants were divided into three groups. Each group did the same task but with a different level of error rate. The results show that the number of interaction errors

[21]spatial augmented reality

does not substantially impact the preferred modality. On the other hand, however, the results show (see Fig. 14) an expected trend of lower experience compared to expectations when the interface error rate is higher. Preferred modalities are gestures and a 6D device, such as a second touch table and touch screen. Objective measurements, e.g. speed of task completion, show the same order of modalities. With a growing amount of errors, the perceived intuitiveness of the modalities decreases, except for the touch display, where it grows. This could be caused by the fact that the touch screen is the only control commonly used by the participants.

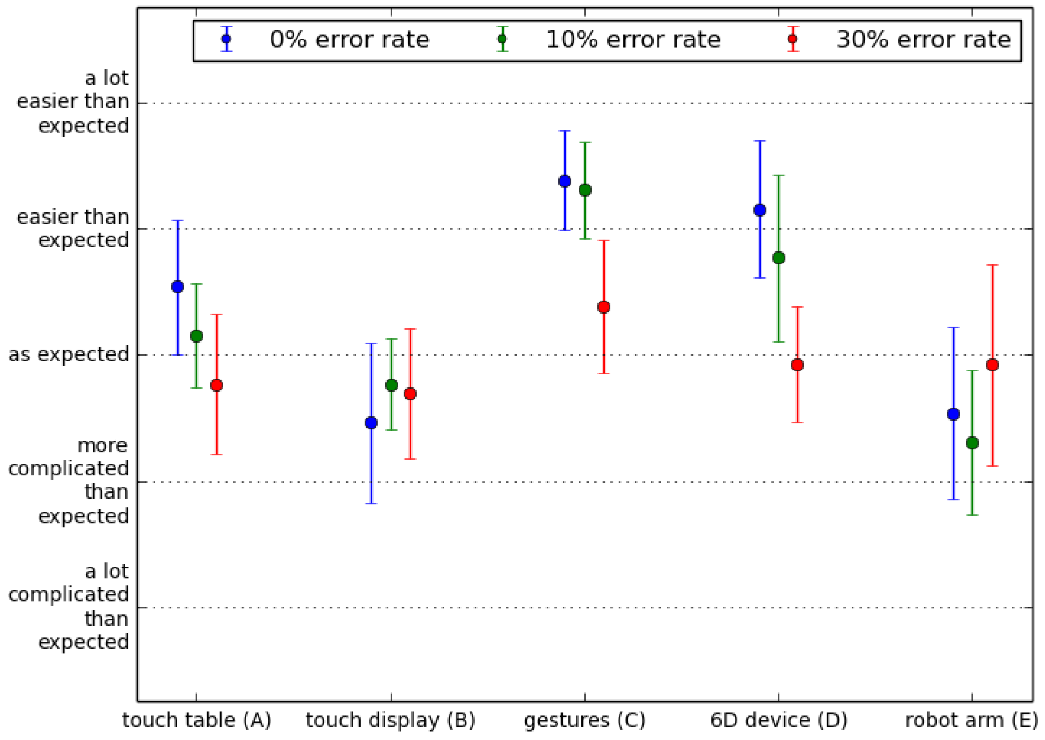


Figure 14: User's assessment on how the experience matched expectation (Materna et al., 2016).

The most preferred (subjective measurement) and fastest (objective measurement) modality was the 6D device. This is not to say that this modality is the best in general. Each modality is suitable for different situations and requirements for the resulting interactive system. For example, kinesthetic learning (robot manipulation) turned out to be the worst and slowest modality and yet it can be crucial for solving problems with accurate 3D position input in space (discussed more in Section 3.3). Finding out the effect of modality error rate on its usability by the user is essential in our further research, where we need to know the limitations of each modality when designing a new concept.

3.2 2D GUI in 3D task space

The first concept, transforming the programming environment from a desktop computer to a workspace, works with a scenario where the user needs to program a robotic manipulator for pick&place tasks and specific operations at a particular location (e.g. glue application). Thus, the user primarily needs to set locations in the workspace where to pick and place objects and where to perform some machining operation. At the same time, specific parameters need to be set for some operations. Furthermore, the example scenario works with the constraint that the user should have his hands free and not have to wear extra equipment.

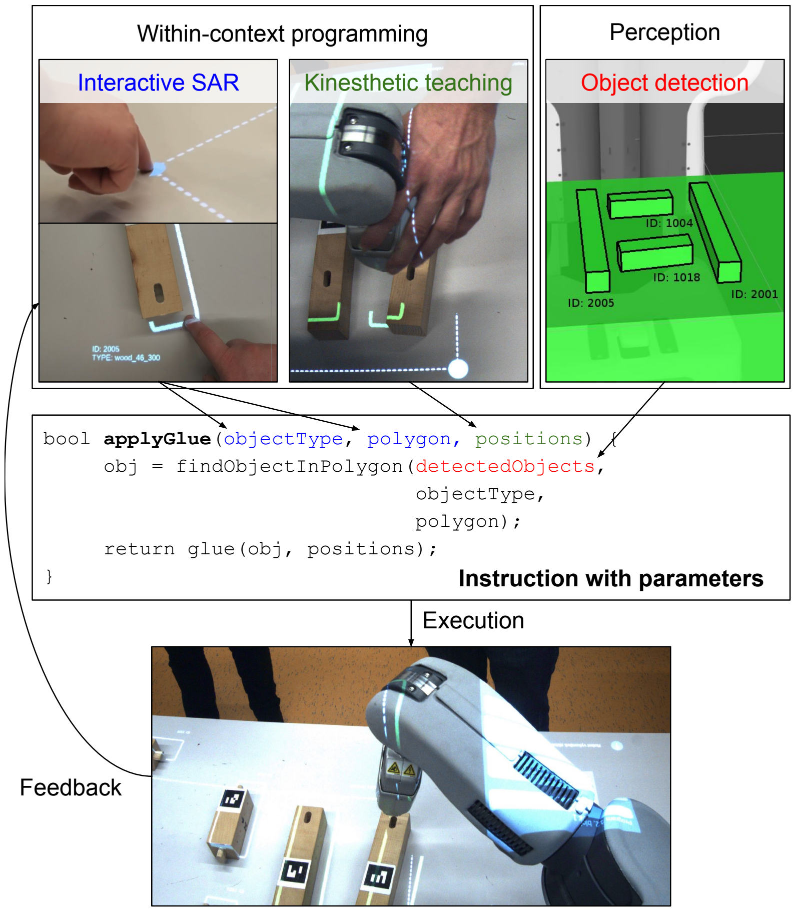
Classic HCI methods are used in the analysis and scenario definition and are based on common situations in the industry. The resulting scenario is then defined as follows (Materna et al., 2017): *The user will teach the collaborative robot to assist him in the task of assembling aircraft service trolleys. He needs to show to the robot which parts are needed in every step of assembling, where holes must be drilled, and what parts should be glued together.* At the same time, the persona was defined (Hartson and Pyla, 2012; Cooper and Saffo, 1999): *Jan, a 22 year old man, recently graduated at technical-based high school. He works as an assembly worker at Clever Aero, a company focused on aircraft equipment. He has no experience with robots, but he loves new technologies and he is really keen into working with robots.*

The constraints defined by the scenario lead to the use of SAR technology for communication from the robot to the user, i.e. projecting the GUI onto the workstation's desktop. Based on previous research about preferred modalities and free-hand constraint, a touch-table modality applied to the workbench is used to interact with the projected GUI. A second modality, gestures, is also used for this interaction for experimental purposes. The technology for tracking the position of the user's hands and fingers evaluates the pointing direction and the gesture stopping time. The detection results are mapped to the pointer position in the GUI on the table. Kinesthetic learning is used to define the spatial parameters of actions, where the user defines specific locations in space by placing the end of the robotic manipulator (end-effector) at the desired location (see Fig. 15). The concept assumes the system's ability to recognize objects in the scene as well as their type. It uses this information to define operations at a higher level of abstraction, where a command can be given to the robotic system, e.g. pick up an object of a specific type from a feeder. In experiments, this automatic object detection and classification capability have been replaced by QR codes.

The design of the projected GUI reflects the needs to display both the program as a sequence of actions and the details of the parameters of each action (see Fig. 16 (a-c)). Furthermore, the ability to manipulate the program, from program management (running, creating a copy, etc.) to action manipulation (reordering, creating duplicates, cancelling, etc.). A key element of the designed GUI is the visualization and interaction with 2D spatial information on the table in the working space. These elements (widgets) are used to 1) edit the areas for picking up and placing objects, and 2) visualize the detected objects, including their type (see Fig. 16 (d)).

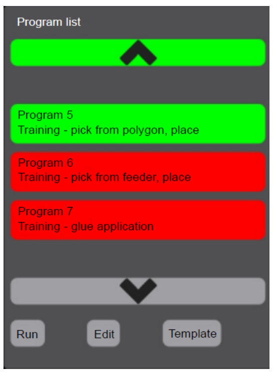
Proposed concept reflects the main objective of the research to reduce the mental demands and attention switches by centering all interaction in a shared workspace, combining various modalities and enabling interaction with the system without any external devices. The experiments focused both on the usability and user experience of this new concept (Materna et al., 2018) and on a comparison with the now widespread method of programming robots using the teach-pendant (Kapinus et al., 2020).

Non-expert users programmed a robot on a high level of abstraction and work within the task context, free of any additional external devices and with immediate visual feedback. The intended workflow of the main task was that the user did the assembly while the robot prepared the parts needed in the following steps itself in the background. The program was divided into three blocks. Blocks 1 and 2 had the same structure and served to prepare the parts for the sides of the stool (two legs, two connecting parts, application



```
bool applyGlue(objectType, polygon, positions) {
     obj = findObjectInPolygon(detectedObjects,
                                objectType,
                                polygon);
     return glue(obj, positions);
}
```

Figure 15: Illustration of program parameters definition (combination of manually set parameters by the user with perceived information by the system) and its execution with visual feedback (Materna et al., 2018).

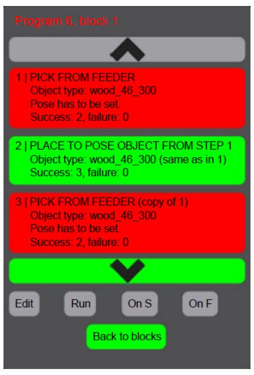

Program list

Program 5
Training - pick from polygon, place

Program 6
Training - pick from feeder, place

Program 7
Training - glue application

(a) List of programs. Green ones are ready to run, red ones need to set parameters.

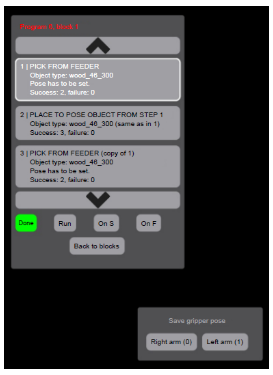

Program 6, block 1

1 | PICK FROM FEEDER
Object type: wood_46_300
Pose has to be set.
Success: 2, failure: 0

2 | PLACE TO POSE OBJECT FROM STEP 1
Object type: wood_46_300 (same as in 1)
Success: 3, failure: 0

3 | PICK FROM FEEDER (copy of 1)
Object type: wood_46_300
Pose has to be set.
Success: 2, failure: 0

(b) List of instructions. Green ones are ready to run, red ones need to set parameters.

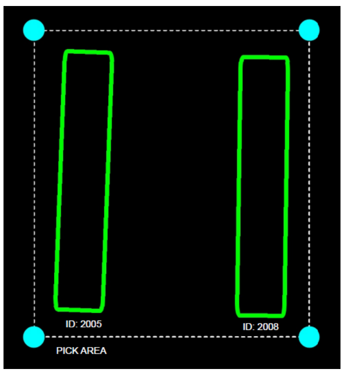

Program 6, block 1

1 | PICK FROM FEEDER
Object type: wood_46_300
Pose has to be set.
Success: 2, failure: 0

2 | PLACE TO POSE OBJECT FROM STEP 1
Object type: wood_46_300 (same as in 1)
Success: 3, failure: 0

3 | PICK FROM FEEDER (copy of 1)
Object type: wood_46_300
Pose has to be set.
Success: 2, failure: 0

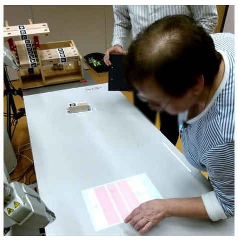
(c) A small dialog shows if the robot is able to detect an object in the feeder and allows the user to save the arm pose.

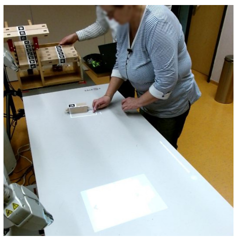


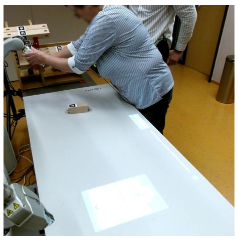
(d) Polygon defining the area on the table from which the objects will be picked up. The green outlines correspond to detected objects.

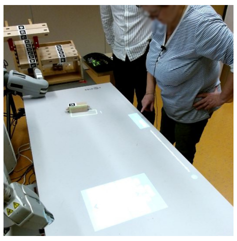
Figure 16: Examples of different widgets from the prototype system (Materna et al., 2018).

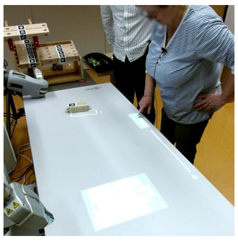
of glue). The purpose of two blocks was that the user might set parts within one block to be supplied from, e.g. the left feeder, and in the other block from the right feeder. Block 3 served to prepare the connecting parts for the final assembly of the sides of the stool. An example of human-robot interaction during the experiment is shown in Fig. 17.

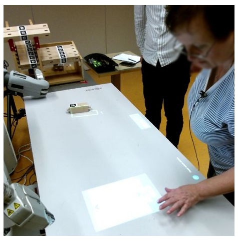

(a) User selects instruction to be set from list (pick).

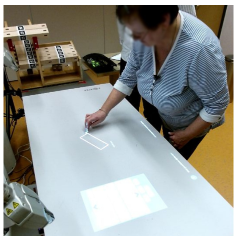

(b) Object type is set by touching its outline.

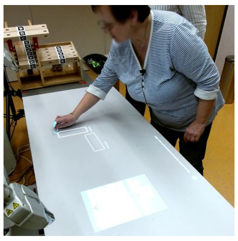

(c) Robot arm is used to teach detection position.

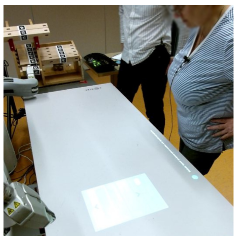

(d) Dialog shows if robot is able to detect object in feeder.


(e) User saves position (confirmation sound is played).


(f) User selects follow-up instruction (place).


(g) User adjusts place pose by dragging it on the table.


(h) Another pose, first one also shown for convenience.


(i) User tests *pick from feeder* instruction.

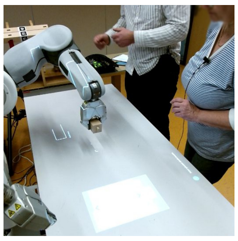

(j) Test of *place to pose* instruction.

Figure 17: An example of human-robot interaction during the experiment. In this case, the user sets parameters for two *pick from feeder* instructions (one shown) and consequent *place to pose* instructions (both shown). Then, instructions are tested. Two input modalities are used: touch table and robot arm (Materna et al., 2018).

The conducted user experience tests proved the potential of this concept when all six regular shop-floor workers were able to program the robot to prepare parts for a stool assembly, collaborate with the robot, and adapt the program for an alternative product within a reasonable time. The mean usability (SUS) rating was 75.80 (SD 8.90), while for comparison, the system from (Huang and Cakmak, 2017) scored 66.75 (SD 16.95). The mean mental load (TLX) was 33.3 (SD 8.8). No fundamental issues were found during the experiment forcing us to reconsider the approach. However, the task state awareness, in particular, has to be improved as well as support for the workspace layout. The participants rated the system positively despite some minor usability issues and system errors caused by its experimental nature.

Comparing a traditional method of programming an industrial collaborative robot using a teach pendant, with a novel method based on augmented reality and interaction on a high-level of abstraction brings necessary preparations to deal with different complexity, level of abstraction (high for AR, low for pendant) and specifics of each method. In the experiment, three participants programmed a visual inspection task (Kapinus et al., 2020). The robot was instructed to pick the bottle opener from the table, put it in front of the camera, trigger the inspection method and based on the inspection result, put the bottle opener into one of the boxes on the table (an illustrative example is in Fig. 18). To make

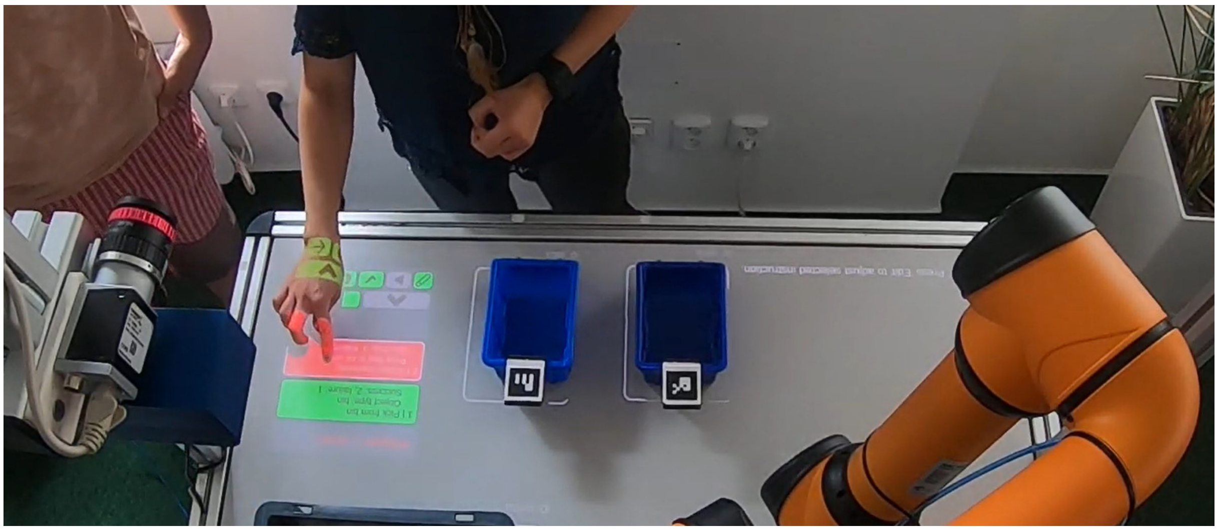


Figure 18: Participant programs a visual inspection task using the proposed spatial augmented reality interface (Kapinus et al., 2020).

Participant	$\mathbf{A_p}$	$\mathbf{A_a}$	$\mathbf{B_p}$	$\mathbf{B_a}$	$\mathbf{C_p}$	$\mathbf{C_a}$
Introduction [s]	359	179	449	311	185	174
Task [s]	562	189	749	309	510	146
TLX $[0, 100]$	72.22	36.11	44.44	27.78	33.33	19.44
SUS $[0, 100]$	52.50	82.50	42.50	80.00	70.00	90.00
$\mathrm{UEQ_{ATT}}$ $[-3, 3]$	−1.17	2.00	−0.17	1.83	1.83	2.50
$\mathrm{UEQ_{PRA}}$ $[-3, 3]$	0.25	2.08	−0.50	1.83	1.58	2.25
$\mathrm{UEQ_{HED}}$ $[-3, 3]$	−0.25	2.12	−1.25	1.62	0.25	2.00

Table 1: Durations of introduction and programming for both (**p**)endant and (**a**)rcor modality. Subjective metrics for each participant and both modalities. Higher means better for all subjective metrics except TLX (Kapinus et al., 2020).

the comparison fairer, a few high-level functions such as *pick, place or suction (on/off)* were prepared in advance in teach-pendant. The results indicated the potential of the proposed solution (see Table 1), which was preferred by the participants over the tech-pendant (based on system usability (SUS) and user experience (UEQ)), the task load was also better (TLX). Also, it required less time to train as well as to program the visual inspection task.

The results of the experiments led to further modifications of the proposed spatial interface in AR and its integration into an industrial application[22]. In addition to practical applications, new experiences regarding the benefits of different modalities and approaches led the research to use other AR technologies and attempt to transfer the task of programming the robot into 3D space.

An existing solution with SAR technology and a touchscreen on the workspace table was also extended with augmented reality glasses, namely Microsoft HoloLens (Bambušek et al., 2019). Main advantages of the proposed approach are the possibility to program the collaborative workspace without the presence of the robot, its speed in comparison to the kinesthetic teaching and an ability to quickly visualise learned program instructions

[22]`https://www.testitoff.cz/en/`

in the form of virtual objects in 3D space enhancing the users' orientation within those programs (illustrative screenshots in Fig. 19). Visualisation of a partial robot simulation directly in a 3D environment also significantly reduced the risk in testing the program.

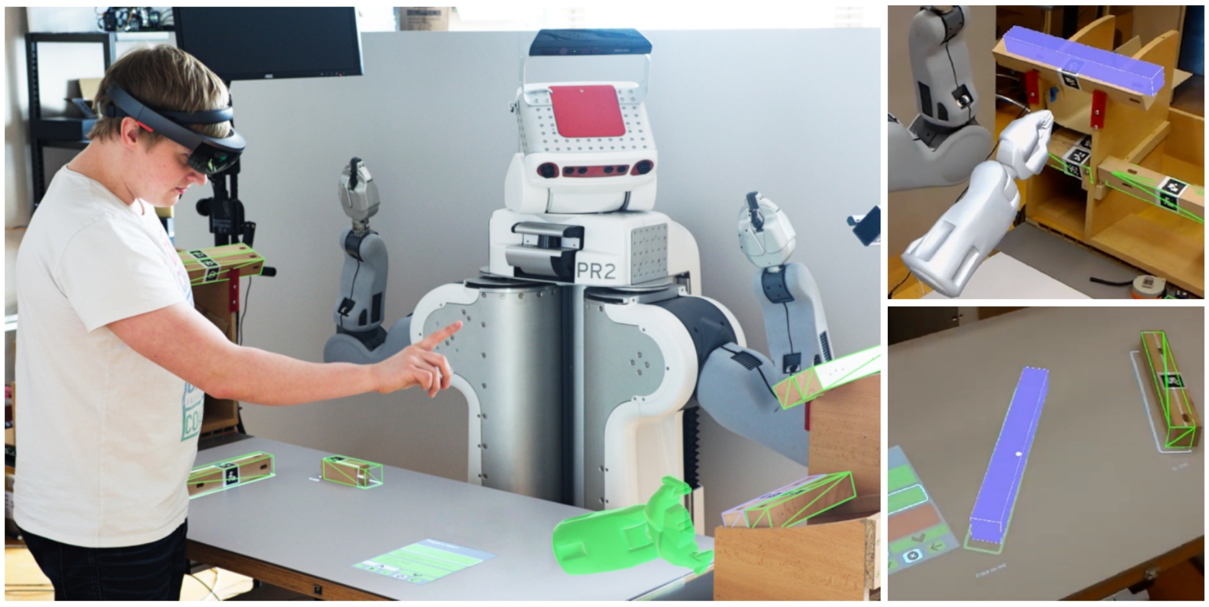


Figure 19: Spectator's view of the collaborative workspace, with projected user interface, extended by virtual objects seen through the HMD – an illustrative example of setting program parameters using the HMD gestures (left). Virtual gripper is rendered on a set detection position along with the virtual object of specified type (pick from feeder instruction) (right-top), and virtual object is rendered on a set place position (place to pose instruction) (right-bottom) (Bambušek et al., 2019).

The new solution, which extended the existing system by using HMDs, brought improvements in pick and place tasks in qualitative measures by 33.84% and by 28.46% in quantitative measures over the baseline SAR approach for both tasks (Bambušek et al., 2019). Ongoing research has already focused on programming the robot entirely in a 3D environment.

3.3 3D GUI in 3D task space

The main goal of our research was to fully transfer the interaction in robot programming to a real 3D robot manipulator workstation. After experiences with different modalities, program representation, user needs and expectations, key action parameters and kinesthetic learning, the new concept of a user interface in 3D space was proposed using augmented reality on mobile devices. The basic concept and representation of the program were presented on a PCB testing[23] task (see Fig. 20 right). The program is represented as a set of points in 3D space, where each can contain one or more specific actions to be performed by the robot at that point. The sequence of actions is then defined by logical connections between the actions (see Fig. 20 left).

Experiments were conducted with seven users with no or very limited knowledge of programming and augmented reality. A system usability scale (SUS) was in the range of 80.8–84.0, that is rated by grade A and is at the 90–95th percentile. This shows

[23]printed circuit board testing

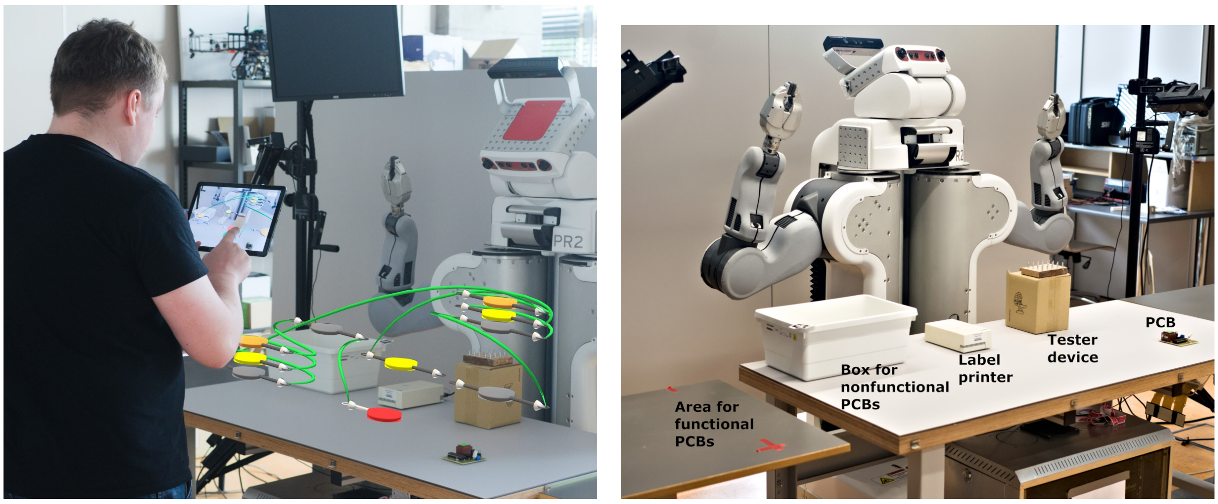


Figure 20: Illustrative representation of the robotic program as logical connections of actions in space (left) and layout of real objects for use-case PCB testing (right) (Kapinus et al., 2019).

promising potential for future research in this field and shows that the created prototype user interface is highly usable. Although the mental workload (based on NASA-TLX evaluation) in laboratory scenarios cannot be generalized directly to the workload in a real environment, it still can be helpful to reveal potential issues. The mean TLX in our experiment was 27.38 (SD=9.41), which means that the workload was lower than in at least 80% of studies analyzed by Grier (Grier, 2015). The UEQ benchmark measured the user experience. The system was overall rated as Excellent in all UEQ categories, i.e. Attractiveness (mean score 1.93, SD=0.58), Pragmatic attributes (mean score 2.26, SD=0.28) and Hedonic attributes (mean score 1.86, SD=0.72). We found no fundamental problem during the experiment forcing us to reconsider the proposed approach. Although participants observed or self-reported minor issues, all participants were able to complete the task.

In the further development of the new concept, several key objectives were strongly considered: *efficiency* of AR usage, *ergonomics* of the GUI on a mobile device, *fast* yet *accurate* positioning in 3D and object selection in cluttered scenes.

The effective use of AR reflects the shortcomings of UI design in 3D, which can lead to a situation where AR is used for minimal interaction (e.g. it only displays virtual objects and allows their selection), but most of the interaction then takes place using on-screen GUI elements (buttons, menus, form elements, etc.) in a head-up manner. The desire to minimize on-screen controls and bring the maximum amount of interaction into 3D space has led to solutions based on various *interface modes*. Thus, the amount of 2D GUI elements on the screen is minimized to only buttons for switching modes and then two context buttons implementing the execution of an action (select, manipulate, insert, remove, etc.). The key element is the *pointer*, which is fixed in the middle of the screen. This final solution is also influenced by the emphasis on **the ergonomics of the designed GUI**. The user should focus on working with the virtual data in 3D space, not on the GUI on the tablet. Therefore, the key GUI elements are designed to be operable only with the thumbs of the hands and have a fixed location, changing only the function of the two context buttons. The interaction then takes place mostly in 3D space with virtual objects using the pointer and context buttons according to the current interface mode. The schematic concept is illustrated in Fig. 21.

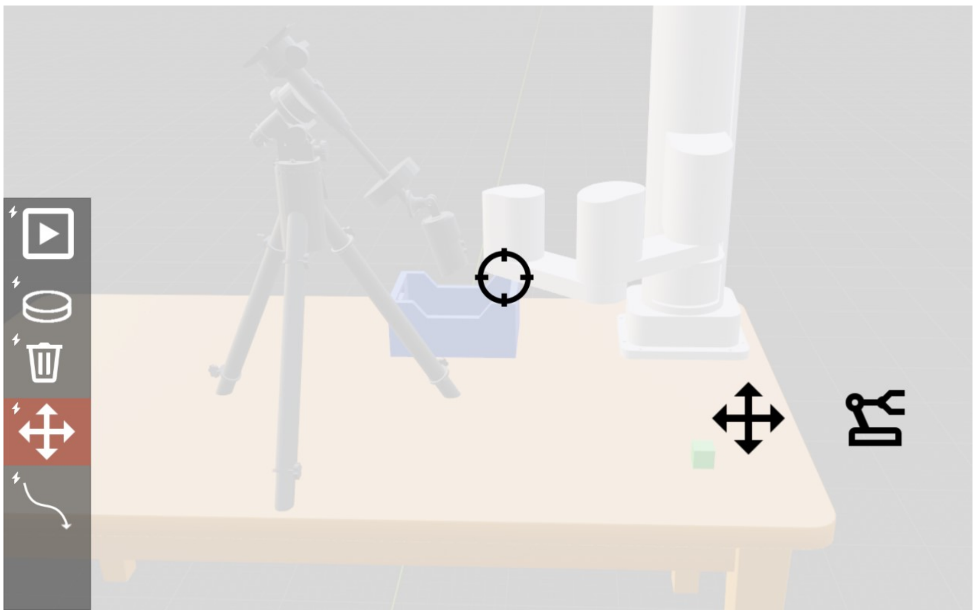


Figure 21: Schematic visualization of the user interface. The left side contains the main menu allowing the user to select the appropriate interaction mode. In the middle is a crosshair for indirect virtual object selection. On the right are two context-aware mode buttons, easily reachable by the user's thumb (Kapinus et al., 2022b).

Fast manipulation of virtual objects in 3D means such manipulation where the position is determined by the movement of the mobile device and positioned by the user based on the visualization of the object in AR. By this direct manipulation, objects can be manipulated quickly and naturally, and their location in 3D space can be directly observed. However, the accuracy of this manipulation is only indicative and can be used, e.g., for rough 3D annotation of the environment. This is caused by the two main coordinate systems of the whole concept:

1. the robot coordinate system, in which the locations of actions in space have to be defined, and

2. the AR scene coordinate system of the mobile device, in which we visualize the program in the 3D environment.

A straightforward way to register these two coordinate systems can be done, e.g. by some easily locatable reference mark (e.g. QR code) whose location is known in the robot coordinate system. However, this does not solve the problem of the inaccuracy of AR on mobile devices, which is based on estimating the position of the mobile device camera based on image analysis. All positions given by direct manipulation in AR are implicitly inaccurate.

Precise manipulation, which is crucial in most robotics scenarios, is enabled by the new concept under investigation in two ways. For both methods, kinesthetic learning is the basis, where the position of the robotic manipulator's end-effector is used to precisely position the virtual object in the 3D scene. This is because its position is precise in the robot's coordinate system to which the program relates in 3D space (i.e., as precise as the robotic manipulator's precision). The precisely specified positions of virtual objects or 3D spatial anchors in the scene then serve as **reference points**. The position of other virtual objects can then be set precisely relative to these reference points. For this purpose, the concept of a 3D manipulation gizmo (see Fig. 22) and a transformation menu (rotational selection) are used to adjust the displacement in the selected axis. This translational GUI element also includes a shift order switch to accelerate the shift by larger distances.

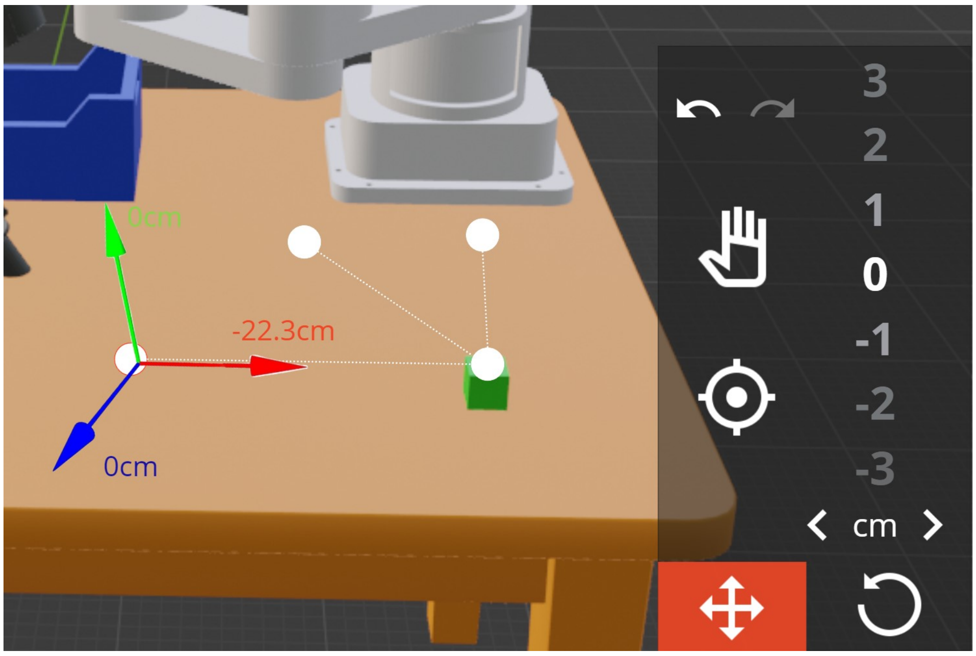


Figure 22: The schematic visualization of the precise manipulation tools available in the transform mode. 3D widget, so-called gizmo, rendered over the manipulated object (left) and the transform menu, with several interactive elements and rotary widget (Kapinus et al., 2022b).

This new concept of spatial programming was experimentally compared with the visual programming approach already frequently used today for collaborative robots (Dobot M1 Studio[24]) was chosen as a representative of this approach. The experiment was designed as a within-subjects design with two conditions: C_1 – a new concept of spatial programming, and C_2 – conventional visual programming approach with Blockly technology, which was tested on 12 subjects of various ages, self-reported genders, and technical backgrounds.

Both mental load (TLX score) and usability (SUS questionnaire) were better for the new concept (C_1) than for the conventional (C_2) (see Fig. 23(a)). However, differences are not significant for both metrics according to paired t-test, so the research hypothesis that the new interface (C_1) is more usable than conventional (C_2) and puts less task load on the user, can not be confirmed. C_1 usability was compared with a similar solution for virtual object manipulation in AR (SlidAR (Polvi et al., 2016)) using HARUS method (Santos et al., 2014), where it achieved better values, 82.9 vs. 76.3 (the higher, the better).

Objective measurements were performed on 3 types of tasks (see Fig. 23(b) and (c)):

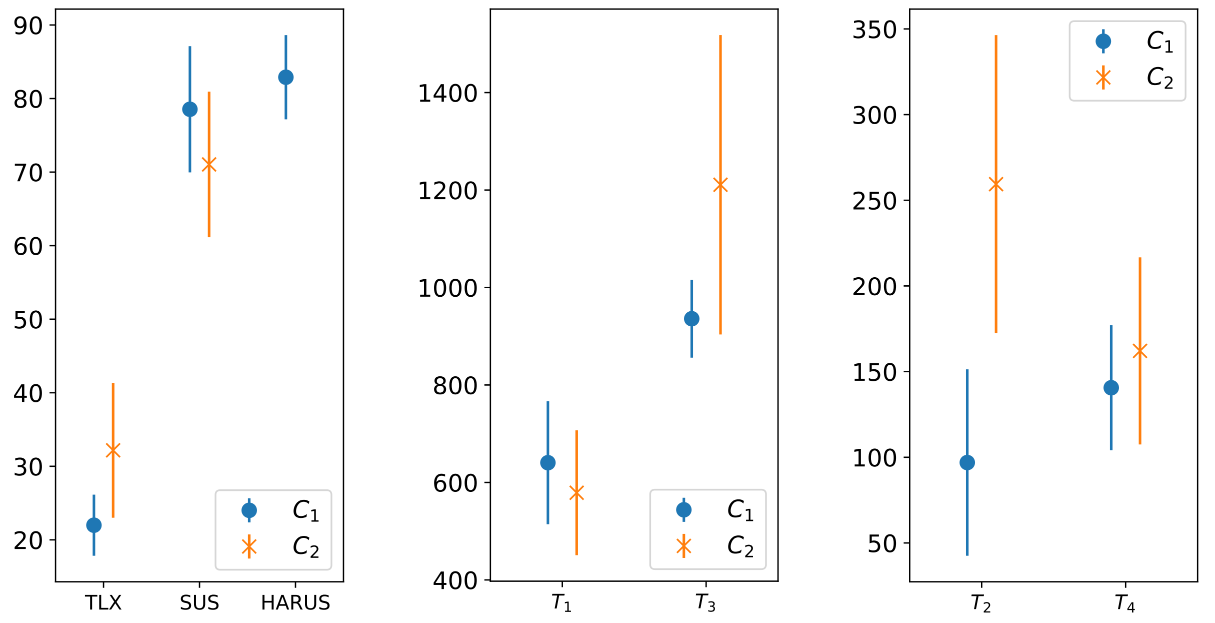
1. **visualization and program orientation (T_2)** – the measured data showed that the new concept (C_1) enables significantly faster program orientation and understanding,

[24]uses the Blockly technology, `https://en.wikipedia.org/wiki/Blockly`

2. **creating a new program (T_3)** – again, from the measured data it can be significantly demonstrated (according to the Wilcoxon test) that the new concept (C_1) allows creating a new program faster than using visual programming (C_2) on a desktop computer,

3. **program adaptation (T_4)** – the use of the new concept (C_1) was slightly faster, moreover, in the C_2 condition, participants did not fit the entire trajectory, and therefore the difference becomes even larger as the number of points on the trajectory increases.

The training phase for familiarization with both approaches was insignificantly worse in time for the new concept (C_1) than for conventional one (C_2) (Fig. 23(b), T_1). A somewhat surprising subjective finding was that for fast (i.e. inaccurate) manipulation of virtual objects in 3D space, users preferred kinesthetic learning (50%) over direct manipulation by moving the mobile device in space (33%) (the remaining 17% preferred the rotating widget). The new concept proved to be more time efficient in the given scenario compared to visual programming on a computer.



(a) The subjective measurements. For the TLX, the lower means better, for SUS and HARUS[25], the higher means better.

(b) Time (in seconds) needed to complete T_1 (training) and T_3 (main) tasks.

(c) Time (in seconds) needed to complete the T_2 (visualization) and T_4 (adaptation) tasks.

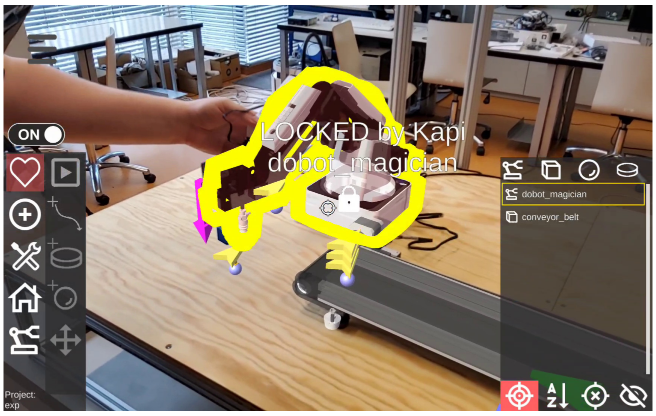
Figure 23: Comparison of subjective and objective measurements (mean values and corresponding 95 % confidence intervals) for conditions C_1 (new concept) and C_2 (conventional visual programming) (Kapinus et al., 2022b).

When testing the proposed concept, it was found that when creating more complex programs, the 3D scene in AR becomes cluttered with virtual objects representing the program: spatial anchors, actions and logical connections. The study of existing solutions

[25]evaluation of the usability of AR in mobile devices (HARUS method (Santos et al., 2014)) was evaluated only for the new concept (C_1)

showed that the problem of selecting virtual objects within augmented reality on handheld devices had been tackled multiple times. Still, evaluations were carried out on purely synthetic tasks with uniformly placed homogeneous objects, often located on a plane and with none or low occlusions. Conversely, the robotic program contains highly occluded objects with large spatial distribution variability and heterogeneous size and appearance. Two new methods were therefore designed to enable long-term usage with a tablet-like device (Kapinus et al., 2022a). A preliminary evaluation of proposed methods suggested that using **indirect virtual object selection in cluttered AR on mobile devices could help increase selection accuracy**. The increase could be achieved primarily in tasks with heavily cluttered environments, such as robot visual programming, AR-enabled visualization of robotic trajectories, editor of robotics work-cells, or any other situated visualization.

Finally, the research results were integrated into a single unit and supplemented with technical functions to enable the results to be used in industry. In addition to purely practical functionalities, such as the possibility to integrate various external industrial robotic and automation devices, there were also conceptual extensions, such as the possibility of multi-user programming collaboration (see Fig. 24) or the generation of program source code in Python for further modifications by experts.



(a) During hand teaching, the robot is locked by the user and therefore unavailable for others, which is indicated within the 3D scene.

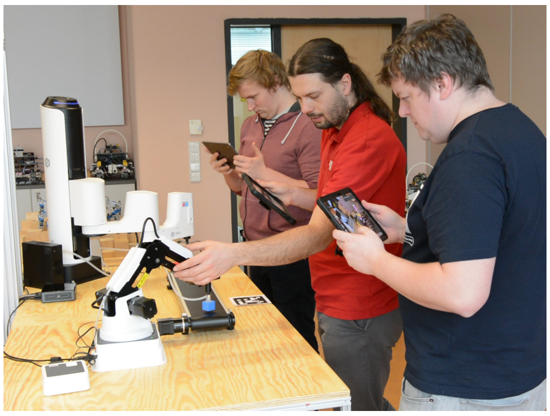
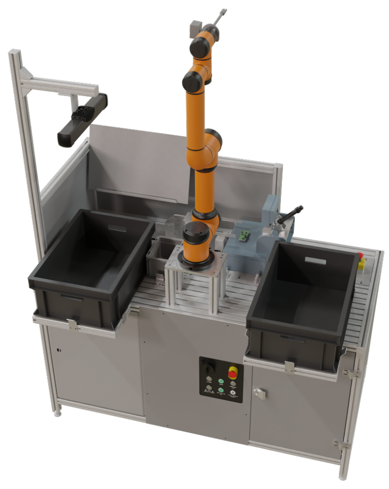

(b) Multiple users collaborating on the task of moving boxes between robots using a conveyor belt.

Figure 24: Technically oriented evaluation of the collaborative-related functionality (Kapinus et al., 2022c).

The latest evaluation of the complete system was performed by three experts and indicated the high potential of the solution. The framework, when adopted by a system integrator and its customer, allows efficient collaboration between professional (robot) programmers and end-users, who are domain experts (see Fig. 25). By allowing end-users to set up a workplace, create or adapt the program, high flexibility needed for SMEs is achieved. Moreover, as SMEs are able to perform program modifications/adjustments on their own, expenses are reduced.

The gained experience also helped to at least partially answer the research long-term research questions. Within the context of visual programming, it turned out that situated interaction could be realized using augmented reality and handheld devices, when supported by specifically designed interfaces. The proposed solution overcome imprecise registration of AR visualization by utilizing the robot's precision and then by moving


(a) Render of a PCB testing workplace with the Ensenso 3D camera for bin-picking, 6 DoF Aubo i5 robot, 2 DoF custom-build robot, functional tester, barcode reader, and printer, source, and target boxes.

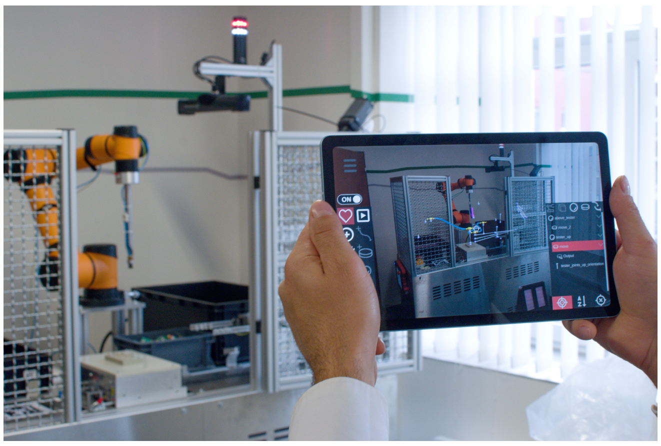

(b) A user observing the program for the PDB testing task on real 3D working space.

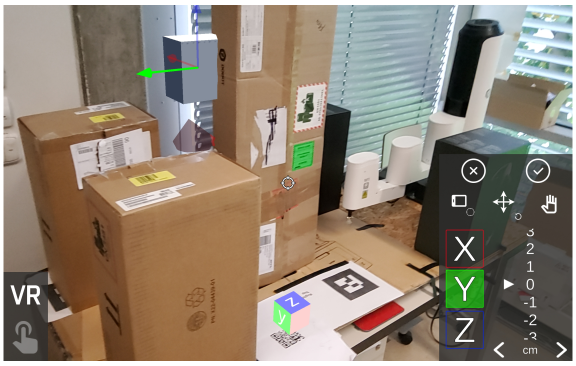
Figure 25: Experimental PCB testing workplace (Kapinus et al., 2022c).

virtual objects relatively to precisely obtained positions (even if the visualization might be slightly shifted or not fully stable), because the spatial relations between objects are kept and users can manipulate objects with arbitrary precision.

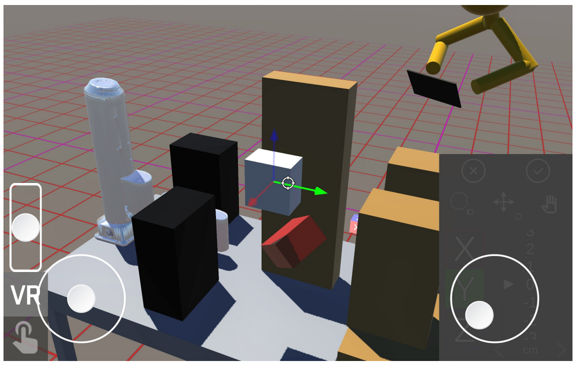
3.4 VR vs. AR in Robot Spatial Programming

Experimental experience has shown two general drawbacks when using AR for programming. One is fatigue when performing some of the more tedious operations, such as setting action parameters or precise positioning. This was expected and is a known problem when using AR. For these operations, it would be more comfortable for the user to sit down, put the tablet down and adjust parameters and precise positioning already outside of AR. The second problem was, less expected, that in some cases it was necessary to look at the scene from an angle that was not possible to implement in practice (e.g. due to space constraints of the machine, poorer accessibility of important locations in space due to density and overlapping of real objects, etc. (see example on Fig. 26(a)). And also one more aspect led to a new extension of the concept. Working in AR on a monoscopic display impairs the user's spatial perception and the user then needs to view the scene from two angles simultaneously in some situations.

This observation led to the extension of the concept to include the possibility of working only in virtual reality (VR). It was necessary to extend the interface with a new element of a virtual camera that the user could move freely either by moving the tablet but in a different coordinate system (without reference to the real 3D space) or by using GUI elements (see Fig. 26(b)). The rest of the interaction then remains the same. Thus, the user can explore and modify the scene without having to connect to the real space (e.g., sitting down without getting tired hands) or take advantage of the offset view of the scene and explore and modify the scene from different angles than the real world would allow in AR.



(a) A user is trying to fit the grey object onto the red one, which is occluded by the real-world environment.



(b) Moved camera to the physically inaccessible position to get a better view of the scene in VR mode.

Figure 26: An example of a fitting task in AR and VR mode (Bambušek et al., 2022).

Experiments were then focused on the extent to which the ability to work in VR under limited spatial conditions was exploited. The task was the spatial annotation of objects that could not be bypassed and thus seen from the other side. Twenty subjects performed the task under 2 conditions – AR only and the possibility of using both AR and VR (within-subject experiment). Although some results were affected by the inappropriate implementation of GUI elements for controlling the virtual camera (which affected the mental workload), the physical workload was statistically significantly lower when using both AR and VR. Since the task execution time was the same in both conditions and the concept of using VR was positively accepted by the users, this extension can be evaluated as the right direction (Bambušek et al., 2022). The room for improvement then remains a more appropriate implementation of the virtual camera control to make this task more natural and not unnecessarily burden the user.

4 UAV Pilot Support by New UI Elements

UAVs are an area that has seen significant growth in recent years. Technological developments have brought about a considerable proliferation of these devices in both the private and industrial domains. One of the areas where UAVs are now beginning to be used successfully is in monitoring or rescue operations by police or emergency services. Research and development in this area focus on aspects ranging from legislation and air traffic management, to adverse weather conditions or technological challenges. My research focuses on UI and specifically on the possibilities of using mixed reality to reduce the drone pilot's mental workload and the risk of a drone collision with the environment.

The tasks in which rescue and police forces use drones today often require a situation-specific expert – an intervention commander, an operator of some other equipment (e.g., a crane), a medic, a structural engineer, etc., who needs to *see* some specific remote or inaccessible location. However, this expert cannot pilot the drone with sufficient safety and quality; therefore, another person needs to be involved- an experienced drone pilot. In some situations, even a professional pilot can have a problematic situation in various tasks where piloting a drone is a mentally challenging operation.

Of course, the difficulty of piloting depends on the specific situation – weather conditions, safety constraints, infrastructure complexity, etc. A common legal requirement is direct visual contact (the pilot must see the drone). In addition to the information about the exact position of the drone thanks to GPS and IMU sensors, the video transmission from the drone camera to the pilot's display and various distance measurements from additional sensors on the drone helps to control accuracy.

Present UAV systems (drones), both rotorcraft and fixed-wing, have the capability to carry a variety of sensors as well as additional handling elements. The computing power of embedded devices today offers processing and automatic analysis of signals directly on board the drone. The control of the UAV can then be implemented semi or fully autonomously. Legislation, varying by country, often does not allow autonomous control. However, in many use cases, the autonomous capabilities of these systems cannot even be used and require pilot control. Automatic methods of processing drone sensor data are primarily used offline; the information is aggregated from the stored data only after the flight is completed.

Many papers inspire my research in this area. Three specific problem areas of flight are already defined by Mouloua (Mouloua et al., 2001): workload, situational awareness and teamwork issues. They address these areas from the perspective of flight system designers, who must appropriately address these human error factors and provide recommendations for maximizing UAV operator efficiency, identifying key human factors, ergonomic recommendations, and implications for making piloting more effective. System monitoring, checking error messages or visually checking status indicators require the full attention of the pilot (or operator) and can become repetitive over time, leading to performance degradation and operational errors. This dynamic model of stress and attention is addressed in detail in paper (Hancock and Warm, 1989). The relationship between the degree of autonomy and UX is addressed, e.g. in paper (Christ et al., 2016), where it was shown, in addition to the case study, that the level of autonomy has various influences on UX, particularly in situations with the high perceived workload. Recent research (Agrawal et al., 2020) describes a procedure to involve the target users in the design process (here emergency services in tasks like fire surveillance and search and rescue). For my research, this work inspires designing for situational awareness (SA) using a scenario-driven, participatory design process.

Research in the use of mixed reality for UI in industrial applications could help answer two key questions.

1. What information or results of autonomous methods that could process sensory data in an online (real-time) manner would facilitate drone piloting and thus reduce the mental workload of the pilot without reducing the pilot's attention?

2. How to effectively communicate these results to the drone pilot?

Thus, the research goal is not to replace the pilot but to use the results of the automatic processing of sensor data from the drone to facilitate piloting, enable safe piloting even for less experienced pilots and reduce the mental burden on pilots in general. I do not aim to replace today's traditional controls by, e.g. gestures (Teixeira et al., 2014) and force pilots to control the drone in a different way, but to keep the procedures and skills learned today and only extend and improve the use of current technologies.

Mixed reality technology offers a visual combination of virtual and real data and interaction with them. Virtual data can be of different kinds: data acquired earlier (3D model of infrastructure, results of weather simulations, etc.), meta-data acquired in real-time (location, video data, 3D reconstruction of the environment, occupancy map, distance measurements, positions of external objects in the mission (vehicles, drones, people), etc.), but also user meta-data (control points, danger areas, planned mission trajectory or other spatial annotations). Thus, the research question is what data to use and how to use it to reduce the risks of drone piloting and the mental load of the pilot during the operation. This is done not only by visualizing this data but also by appropriately incorporating automatic assistant features during piloting. To ensure that the pilot is always the only central command and control element in the system but is sufficiently oriented to be able to focus more on the analysis and evaluation of the situation and less on the actual piloting.

The following chapter is based on the preliminary results of my research and draws on the following works:

- **Effective Remote Drone Control Using Augmented Virtuality**, Sedlmajer et al. (2019), presents preliminary experiments with a new UI of pilot application solving the limitations of pilot spatial awareness. The novelty is combining virtual and real data using augmented virtuality instead of augmented reality, which gives the pilot the freedom to move the virtual camera around the scene without losing awareness of the actual situation (video stream from the drone camera). Based on this original concept, both HW experimental device (Plascencia and Beran, 2018) was created, which is an open solution and allows the development and testing of automatic machine learning methods directly on the drone, and SW framework (Plascencia et al., 2019) and SW application with GUI for pilots (DroCo[26]), which serves as a basis for further experiments and research in cooperation with the rescue and police forces of the Czech Republic (Section 4.1).

- **Safe drone exploration with clever trajectory movement**, student's diploma thesis (Ferencz, 2022)[27], presenting novel idea to solve the problem with pilot focus when flying search and monitoring mission in a complex environment. The control

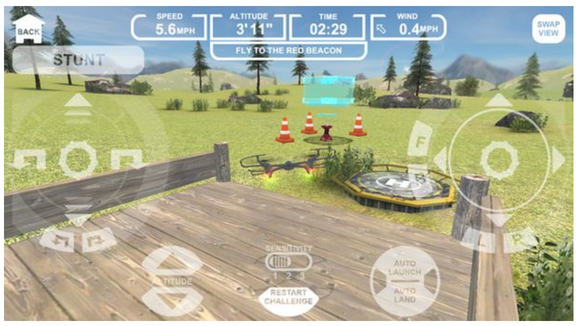
[26]`https://www.fit.vut.cz/research/product/647/`

[27]supervised by the author of this thesis

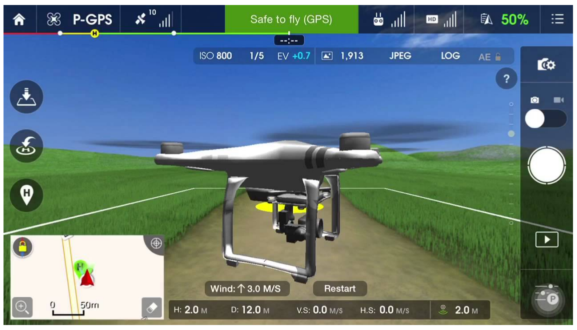
assistant pulls the drone back to a manually pre-set safe trajectory. The pilot thus does not have to concentrate on keeping the drone in a secure area but can entirely focus on the mission task, e.g. taking pictures of objects of interest, without losing manual control of the drone (Section 4.2).

4.1 Augmented Virtuality for Pilot Situational Awareness

When controlling a drone, the pilot has several options for situational awareness: direct visual contact with the drone, a map view of the drone's position, information from sensors (e.g. height above ground), collision reports, and live video feed from the drone's camera. Existing SW solutions are designed to manage multiple drones in search and rescue (SAR), police and security operations from anywhere in the world. The single command centre brings together all video streams, 3D maps, placemarks and other incoming drone data to provide real-time situational awareness (see examples in Fig. 10). The interfaces allow users to simultaneously view multiple video streams along with a 3D map of the mission terrain with the actual coordinates of the drones. Both the operator and the pilots can manually control the drones when needed, as well as add placemarks for particular areas with attached geolocation and inspection images.



(a) Sky Viper Flight Simulator[28].



(b) DJI GO simulator[29].

Figure 27: Example of third-person view in flight simulators.

Inspired by a study of the issue led to the design of a new user interface concept for the drone pilot, which would allow visualising the broader context of the situation by using the third-person view (see Fig. 27) and looking around the scene. The new concept, which aims to increase the pilot's situational awareness and thus reduce his mental load, is also based on a spatial combination of different data sources, but instead of augmented reality, it will use augmented virtuality. The scene that is visualised to the pilot primarily contains existing available (off-line) data from the location, e.g. 3D models of the infrastructure (buildings, maps, etc.). However, this data may not be up-to-date and does not contain objects occurring in the scene in real time. Therefore, the scene is supplemented with spatially registered (on-line) live data – video transmission from the camera on the drone, height information, etc. (see Fig. 28).

Some of this live data are processed in real-time to extract information about objects in the scene (3D occupancy map, object detection, object classification, etc.). Furthermore, existing objects that can locate themselves (rescuers, firefighters etc., wearing GPS

[28]https://apps.apple.com/us/app/sky-viper-flight-simulator/id1135441810
[29]https://www.dji.com/cz/simulator

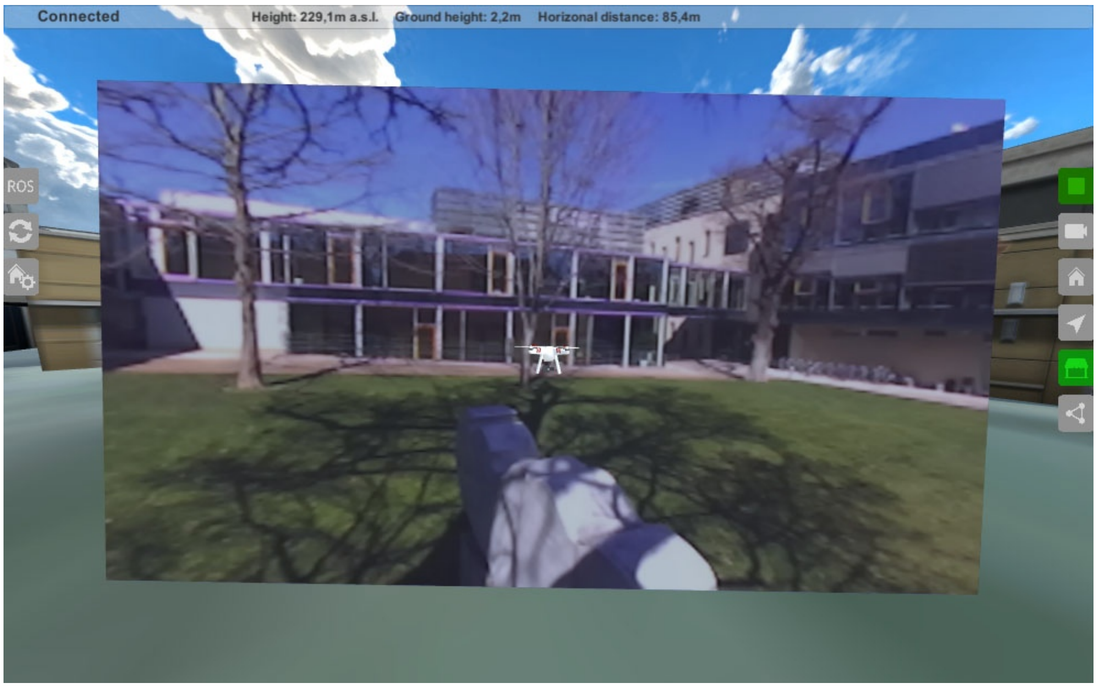


Figure 28: Concept of controlling the drone from the third person view. The virtual screen (with the video feed) moves in front of the drone at a certain distance and is synchronised with the physical gimbal configuration (Sedlmajer et al., 2019).

locator) and share this information with the visualisation system, might be integrated into the scene also. Finally, user spatial data such as mission control points, safe trajectories, danger zones in the area of operation, etc. are integrated into the scene in this way. Since the concept is based on a virtual data scene, where the drone is visualised as a virtual object based on its location information, and to which live data from the real scene is added, this system can be classified as augmented virtuality. Thus, this integrated system allows the creation of supporting navigation and orientation functions and widgets for the pilot, such as distance indicators to various nearby measured objects (see red octagon representing the close distance to neighbour objects in Fig. 29), navigation and distance indicators to external objects communicating with the system and locating themselves in the area (other drones, vehicles, persons) or spatial instructions from third parties (intervention commander, pilots of other drones, ground troops, etc.) (see Fig. 30). The new concept allows the pilot to move the camera freely around the scene and change the view of the piloted drone (without losing the live video feed from the drone camera), allowing him to gain better spatial situational awareness. Local distances to objects, navigation along the planned flight path, location of other units, instructions from the action commander, etc. then also increase the logical situational awareness and together reduce the mental load on the pilot.

Another way to increase situational awareness and, most importantly, to increase the accuracy of piloted operations is to use more drones. An example could be a situation where a drone, carrying e.g. a critical consignment on an overhead device, has to be very precisely guided to a specific location (e.g. without endangering a distant person or to safely insert the consignment into the target object). In these situations, the choice of a second drone as an additional source of information for the pilot (especially visual – video transmission) is suggested. When using multiple drones, it is essential to 1) find

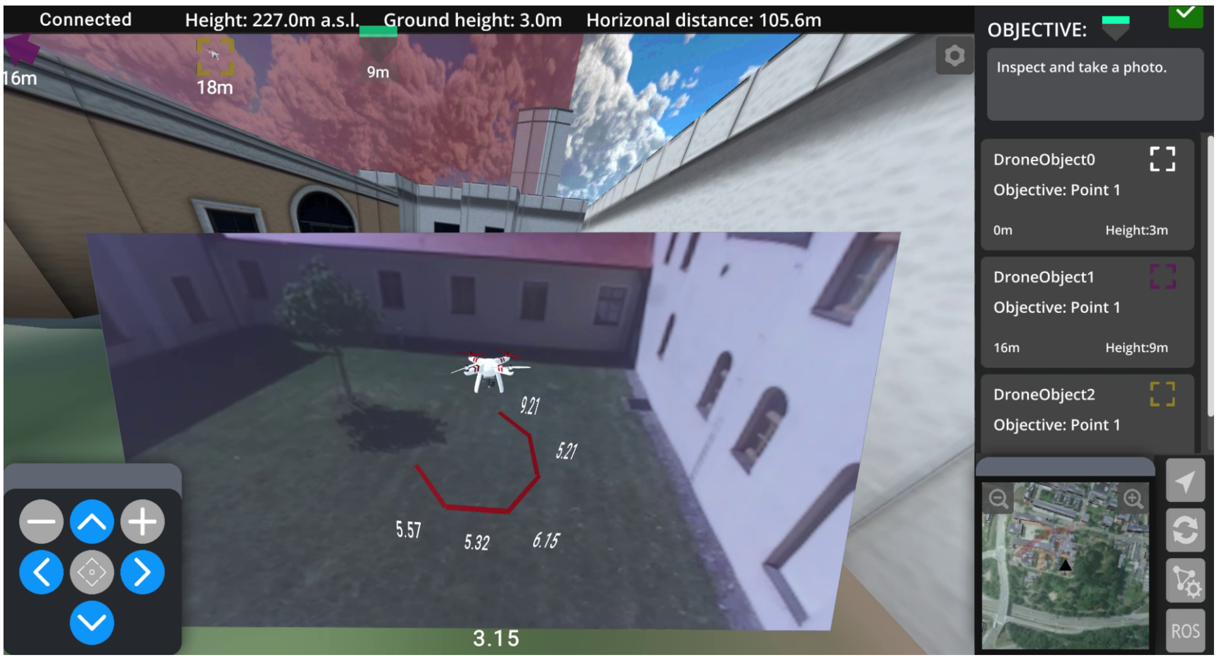


Figure 29: Screenshot from an application with GUI (DroCo) using augmented virtuality to implement a third-person view with live transmission of real video-data and GUI elements visualising estimated distances to virtual objects.

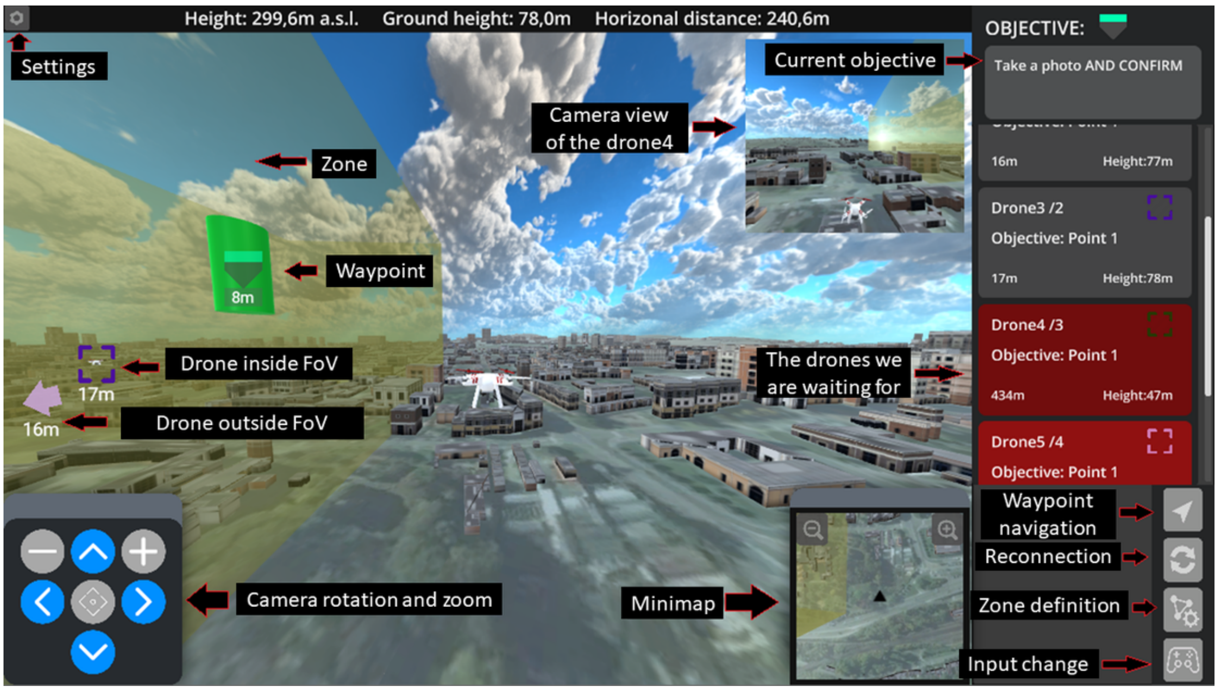
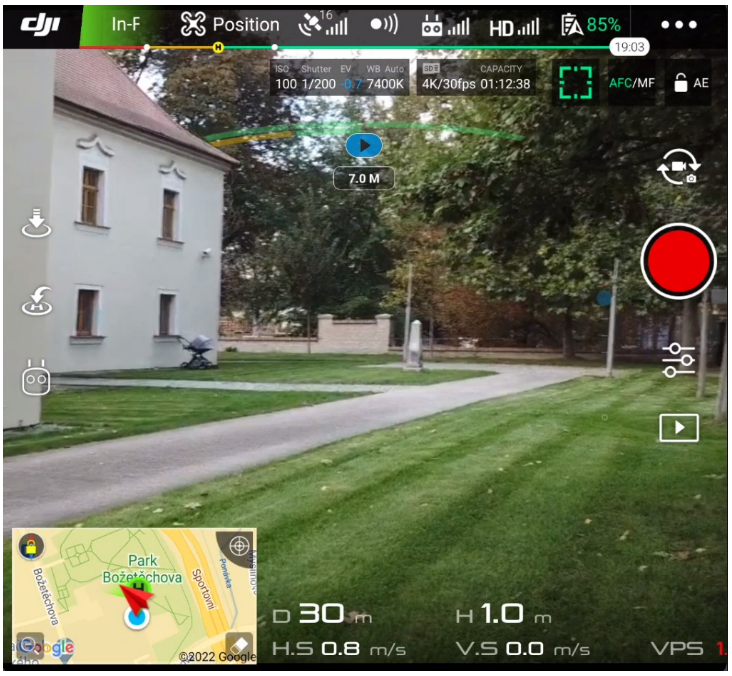


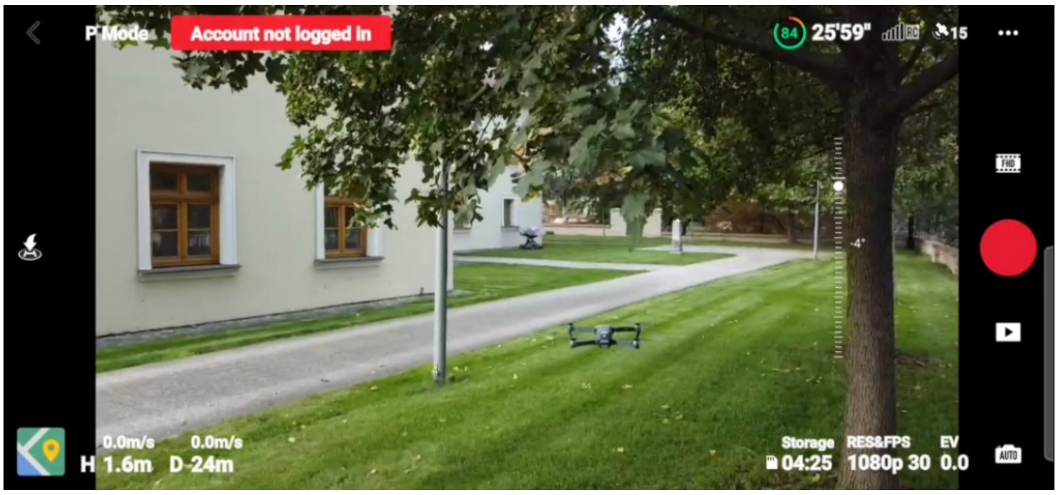
Figure 30: Screenshot of the DroCo application with descriptions of the GUI elements and functions of the new GUI concept for the drone pilot (here without the image of live data from the drone).

a suitable way to set up the spatial configuration of the drones and then 2) enable safe autonomous flight of the drones in the squad.

Our ongoing research explores situations where a third-person view of a piloted drone can be most beneficial (e.g., precision operations in a remote location, flying through complex environments (forest, industrial infrastructure), reversing, etc.). An example of a video-transmission for the pilot of ongoing experiments is shown in Fig. 31. The experiments are conducted in the form of Wizzard-of-Oz, i.e. the additional drones are manually piloted in the experiments.



(a) Video-transmission of the main piloted drone – missing visual information about objects on the left (street lamp) or right (tree-trunk).



(b) Video-transmission from the second drone realizing the third-person view.

Figure 31: Excerpt from the realization of experiments analyzing the benefit and applicability of the TPV concept in piloting drones in complex environments.

The goal of my further research is, therefore, to let the pilot control two (or more) drones as *one unit*, where the formation of drones behaves autonomously (keeps in a specified formation, maintains the direction of the camera's view regardless of its position, etc.). In addition to the UI for the pilot, which must efficiently allow to configure the behaviour of the whole unit, other aspects of such a solution are essential – monitor the space around the unit so that individual drones can navigate themselves to safe locations or automatically plan the movement of all drones. That is, everything to allow the pilot to monitor only his mission target or to let a less experienced pilot to handle such a mission. This is the direction in which my research is going.

4.2 Automatic Assistant for Safe Drone Control

Suppose the aim is to facilitate the pilot's activities during reconnaissance or surveillance actions, given the legislative restrictions on an autonomous flight. In that case, there is room for a solution that leaves the drone's control to the pilot but makes it as easy as possible. For example, when the pilot's goal is to explore a given area, but it contains locations with a high risk of drone movement (presence of people, dangerous infrastructure, or just an increased risk of collision with objects or infrastructure). It is then desirable that the pilot focuses on the critical goal of exploring the area instead of being highly focused on piloting the drone in dangerous locations. The proposed method is based on the assumption that the pilot or operator can manually specify areas or trajectories in advance for the safe movement of the drone. In this case, a method can be developed to revise the pilot's manual control in real-time based on the specified trajectories and the actual drone movement (direction, speed, acceleration). The pilot will thus control the drone's movement, for example, only along a specified trajectory (forward/backwards) or in an area that will not allow it to fly out of a given safe area. The safe areas can be specified manually (e.g. safe area away from crowds, trajectory over a road) or automatically (analysis of previously flown missions, e.g. in complex industrial infrastructure). According to the mission parameters (risk), deviating from the specified trajectory to a safe extent will be possible. Still, the drone will keep itself as close as possible to the pre-specified safe position (illustration in Fig. 32).

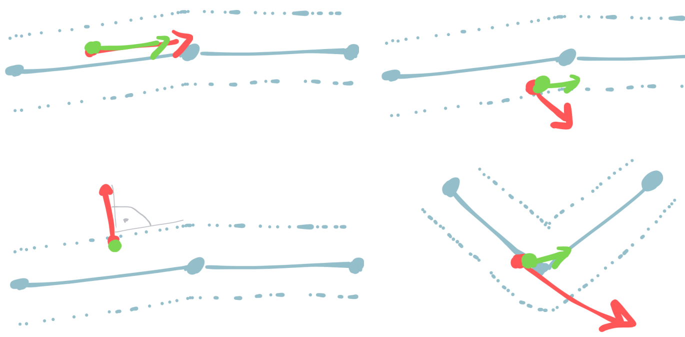


Figure 32: Illustration of automatic assistant functionality for drone control correction. The red arrow shows the direction and speed of the pilot manual command to the drone, and the green arrow shows the desired safe direction and speed of the drone after the correction when the assistant is active (Ferencz, 2022).

So far, this concept of an automatic assistant for correcting the drone control has been implemented in a fundamental version in a simulated environment (Ferencz, 2022)[25].

Keeping the drone in a safe area is based on a predefined safe flight trajectory. The model predicts the future position $x_{t+\Delta t}$ of the drone based on the position x_t in current time t, velocity v_t and pilot control commands to drone (acceleration a_t). Next, both a correction vector c_t and a predicted correction vector $c_{t+\Delta t}$ are computed. The correction vectors pull the drone back to a safe position on the trajectory.

The position and velocity of the drone are expressed by the equations for the change of particle position in time:

$$x_{t+\Delta t} = x_t + v_t \cdot \Delta t + \frac{1}{2} a_t \cdot \Delta t^2 \tag{1}$$

Correction vectors c_t, resp. $c_{t+\Delta t}$ are computed by a stiffness function S modelling the imaginary spring pulling the drone back to the safe trajectory based on the L_2 distance between the actual, respective predicted drone positions, and ideal positions on the trajectory p_t, resp. $p_{t+\Delta t}$:

$$c_t = S(d(x_t, p_t)) \cdot (x_t - p_t) \tag{2}$$

$$c_{t+\Delta t} = S(d(x_{t+\Delta t}, p_{t+\Delta t})) \cdot (x_{t+\Delta t} - p_{t+\Delta t}) \tag{3}$$

The resulting corrected acceleration $a_t^{correct}$ is then a linear combination:

$$a_t^{correct} = w_0 \cdot a_t + w_1 \cdot c_t + w_2 \cdot c_{t+\Delta t}, \tag{4}$$

where w_i are weights ($\sum_i w_i = 1$) controlling the assistant behaviour. The assistant finally remaps the corrected acceleration into drone control commands so that the drone does not fly out of the safe area and gets as close to the safe trajectory as possible. The desired behaviour of the assistant to keep the drone safe is shown in Fig. 33.

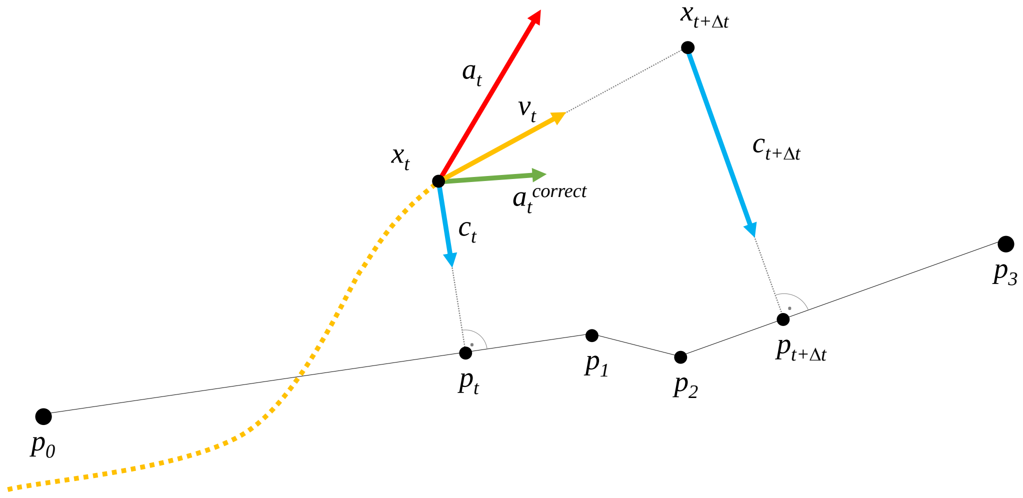


Figure 33: Simple implementation of control assistant keeping the drone close to the safe trajectory (modified (Ferencz, 2022)).

A prototype of the assistant was implemented in the AirSim[30] simulated environment for experimental purposes. The experimental task mimics the real exploration needs in an environment with limited drone motion. The pilot is tasked to take photos of particular objects (cars) in a predefined area while keeping the drone position only in the safe zone (above the road). A schematic view of the experiment is outlined in Fig. 34.

[30]`https://microsoft.github.io/AirSim/`

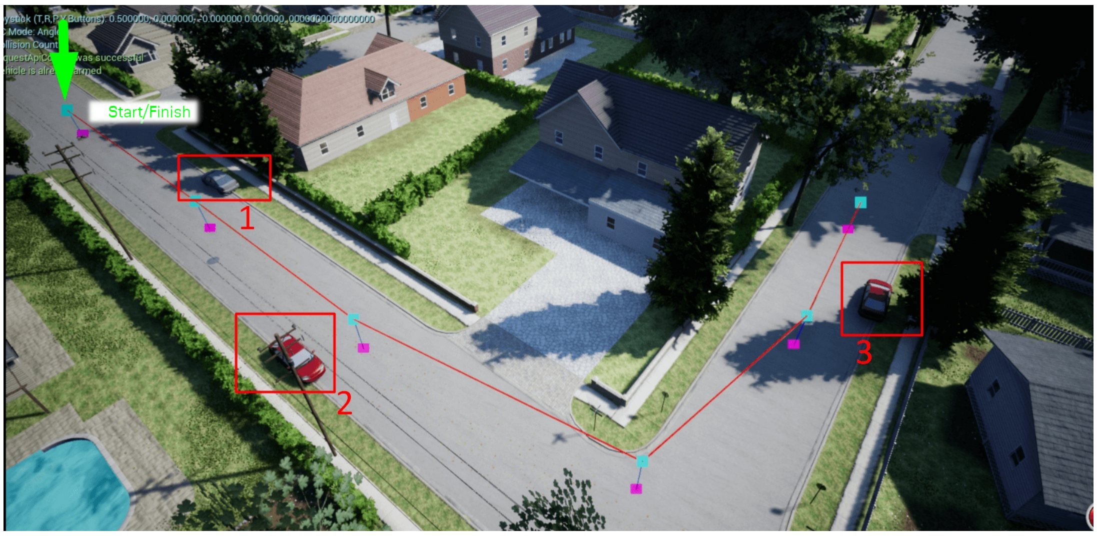


Figure 34: Schematic view of the experiment with the safe area for drone movement (trajectory) and target objects to be photographed indicated (Ferencz, 2022).

The pilot experiment compares not only objective measurements, such as the time of the entire mission or the time spent outside the safe zone but also subjective measurements, which is the quality of the photos taken of the desired objects for further use or analysis. The first, however, statistically insignificant results show the benefit of the concept with respect to keeping the drone in safe positions (time outside the safe zone was 55% of the total flight time without the assistant versus 5% with the assistant). However, subjective measurements of the photographs' quality show the concept's shortcomings. It is difficult (and sometimes impossible) for the pilot to take the necessary pictures of the desired object due to the restriction of the drone's movement when using the assistant, which does not release the drone out of the safe zone. Thus the pilot does not get to the position where he would take the photo from the desired angle. The solution is to extend the UI to temporarily disable the assistant or, which can be considered a safer solution, to adjust the assistant's severity according to the level of risk in a given location. Further research in this direction is currently underway.

5 Conclusions

In an era when we are increasingly surrounded by systems that can perceive our real world and perform various spatial operations in it (to some extent independently based on their observations and understanding), it is crucial to be able to communicate our needs with these systems appropriately, effectively and naturally our needs. We must know what these systems plan to do in our real world, we must know the extent to which these systems understand and navigate our real world, we must be able to easily and quickly correct and adjust their perception and plan, and in many situations, we must directly control these machines. Since this is a communication of spatial information, mixed reality seems to be one of the appropriate technologies for this communication. It offers the possibility to visually connect the digital world of machines with our real world.

Two domains are selected as offering or directly requiring direct human interaction with a semi-autonomous system: collaborative robots and drones. In both cases, the system user must communicate the desired mission plan. *What* is to be done and *where*. In both cases, the user needs to see how the semi-autonomous system perceives the real world, based on which it then decides what to do. In both cases, the real data and the results of the automatic processing and evaluation of the measured data need to be clearly visualized so that the user can orient himself and control the situation. In both cases, the user must be able to modify and refine the plan quickly and easily.

In the collaborative robot scenario, these requirements were addressed by creating a new concept that brings the programming of collaborative robots from a personal computer or teach-pendant directly into the robot's shared workspace. The new proposed 2D concept uses Interactive Spatial Augmented Reality (ISAR) to project the GUI on the working environment and detects hand gestures from the touch-sensitive table top. The solution relieves the user of the problem of context switching between the computer and the robot environment without a hand-held device (free-hand solution). The new ISAR concept proved on the System Usability Scale (SUS) better performance at 75.80 (SD 8.90) than the existing reference solution (66.75 (SD 16.95)). The extension of our ISAR system by HMD-based AR brought the free-space 3D visualization and improved the usability even more by 33.84% in qualitative measures, resp. 28.46% in quantitative measures. Comparison with the standard teach-pendant approach also confirmed better results in all used standard metrics; both objective (time to accomplish the task) and subjective (TLX, SUS and UEQ) metrics were better for our new ISAR solution.

Spatially Situated End-User Robot Programming, the new concept design fully in 3D based on hand-held mobile AR, overcomes the UI limitations of ISAR (and HMD) system. The program is visualized as a logical sequence of actions in the working space. From spatial positions and action parameters to logical connections between actions, the user can work directly in the robot's actual 3D working environment. In the design of the GUI in AR, the focus was on ergonomics when using mobile devices and especially on reducing the user's mental load when programming or adapting the program. Experimental results confirmed the potential of this new concept targeted to shop-floor workers with no special education in programming. The potential of the new solution is demonstrated by an experimental evaluation of the system usability score (SUS), where a score of 82.86 (SD=9.29) is in the class A (the 90–95th percentile) and low mental load (NASA TLX) 27.38 (SD=9.41), which means that the workload was lower then in at least 80% of studies analyzed. This concept has also been successfully deployed in the real application in the industry.

The second investigated scenario is the use of mixed reality in piloting UAVs. Here, the focus is on reducing the mental workload of the pilot by increasing his spatial orientation and making better use of automatic data analysis. The concept is based on prior knowledge of the scene (e.g. 3D infrastructure), where the pilot is actually moving in a virtual scene but supplemented with real data (camera image from the drone, 3D reconstruction of the environment in real-time, drone position and movement or flying data). The novelty is in a combination of virtual and real data using augmented virtuality instead augmented reality that brings the pilot the freedom of moving the virtual camera around the scene without losing awareness of the actual situation. This concept allows the pilot to simultaneously visualize not only the position and movement of other objects in the scene (other drones, ground units, restricted zones, etc.) but also manual annotations (planned mission, instructions from the chief commander or spatial notes from pilots). This information can also correct the drone pilot's flight commands, e.g., to more easily stay on the planned flight trajectory, avoid collisions with real objects that may be out of the flight direction, etc. In cooperation with the police and rescue services of the Czech Republic, this research is still ongoing, and although the results in this direction are still preliminary, it can be expected that mixed reality technologies will be beneficial here as well.

The results of our and other research on using mixed reality for human interaction with semi-autonomous systems confirm that this technology is beneficial in reducing the communication barrier between human and machine. When used appropriately, it leads in selected scenarios to a reduction of the user's mental load in controlling these systems and facilitates the orientation and revision of plans in the digital world of machines.

References

Agrawal, A., Abraham, S. J., Burger, B., Christine, C., Fraser, L., Hoeksema, J. M., Hwang, S., Travnik, E., Kumar, S., Scheirer, W., Cleland-Huang, J., Vierhauser, M., Bauer, R., and Cox, S. (2020). The next generation of human-drone partnerships: Co-designing an emergency response system. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA. Association for Computing Machinery.

Bambušek, D., Materna, Z., Kapinus, M., Beran, V., and Smrž, P. (2019). Combining interactive spatial augmented reality with head-mounted display for end-user collaborative robot programming. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–9. Institute of Electrical and Electronics Engineers.

Bambušek, D., Materna, Z., Kapinus, M., Beran, V., and Smrž, P. (2022). Handheld augmented reality: Overcoming reachability limitations by enabling temporal switch to virtual reality. In *HRI '22: Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*, pages 698–702. Association for Computing Machinery.

Brooke, J. (1996). *"SUS-A quick and dirty usability scale." Usability evaluation in industry.* CRC Press. ISBN: 9780748404605.

Calhoun, G., Draper, M., Abernathy, M., Delgado, F., and Patzek, M. (2005). Synthetic vision system for improving unmanned aerial vehicle operator situation awareness. *Proceedings of SPIE - The International Society for Optical Engineering*, 5802.

Christ, P. F., Lachner, F., Hösl, A., Menze, B., Diepold, K., and Butz, A. (2016). Human-drone-interaction: A case study to investigate the relation between autonomy and user experience. In Hua, G. and Jégou, H., editors, *Computer Vision – ECCV 2016 Workshops*, pages 238–253, Cham. Springer International Publishing.

Cooper, A. and Saffo, P. (1999). *The Inmates Are Running the Asylum.* Macmillan Publishing Co., Inc., USA.

Dass, N., Kim, J., Ford, S., Agarwal, S., and Chau, D. H. P. (2018). Augmenting coding: Augmented reality for learning programming. In *Proceedings of the Sixth International Symposium of Chinese CHI*, ChineseCHI '18, pages 156–159, New York, NY, USA. ACM.

Ferencz, A. (2022). Bezpečný průzkum dronem s využitím chytrého pohybu po trajektoriích. Master's thesis, Brno University of Technology, Faculty of Information Technology.

Grier, R. A. (2015). How high is high? a meta-analysis of nasa-tlx global workload scores. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 59(1):1727–1731.

Hancock, P. and Warm, J. (1989). A dynamic model of stress and sustained attention. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 31.

Hart, S. G. and Staveland, L. E. (1988). Development of nasa-tlx (task load index): Results of empirical and theoretical research. In Hancock, P. A. and Meshkati, N., editors, *Human Mental Workload*, volume 52 of *Advances in Psychology*, pages 139–183. North-Holland.

Hartson, R. and Pyla, P. (2012). *The UX Book: Process and Guidelines for Ensuring a Quality User Experience*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Huang, J. and Cakmak, M. (2017). Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts. HRI '17, page 453–462, New York, NY, USA. Association for Computing Machinery.

Kapinus, M., Bambušek, D., Materna, Z., Beran, V., and Smrž, P. (2022a). Improved indirect virtual objects selection methods for cluttered augmented reality environments on mobile devices. In *HRI '22: Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*, pages 834–838. Association for Computing Machinery.

Kapinus, M., Beran, V., Materna, Z., and Bambušek, D. (2019). Spatially situated end-user robot programming in augmented reality. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–9. Institute of Electrical and Electronics Engineers.

Kapinus, M., Beran, V., Materna, Z., and Bambušek, D. (2022b). Augmented reality spatial programming paradigm applied to end-user robot programming. *Virtual Reality*. Submitted.

Kapinus, M., Materna, Z., Bambušek, D., and Beran, V. (2020). End-user robot programming case study: Augmented reality vs. teach pendant. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, pages 281–283. Association for Computing Machinery.

Kapinus, M., Materna, Z., Beran, V., and Bambušek, D. (2022c). Arcor2: Framework for collaborative end-user management of industrial robotic workplaces using augmented reality. *Journal of Intelligent & Robotic Systems*. Submitted.

Magnenat, S., Ben-Ari, M., Klinger, S., and Sumner, R. W. (2015). Enhancing robot programming with visual feedback and augmented reality. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pages 153–158. ACM.

Materna, Z., Kapinus, M., Beran, V., and Smrž, P. (2017). Using persona, scenario, and use case to develop a human-robot augmented reality collaborative workspace. In *ACM/IEEE International Conference on Human-Robot Interaction*, pages 201–202. Association for Computing Machinery.

Materna, Z., Kapinus, M., Beran, V., Smrž, P., and Zemčík, P. (2018). Interactive spatial augmented reality in collaborative robot programming: User experience evaluation. In *RO-MAN 2018 - 27th IEEE International Symposium on Robot and Human Interactive Communication*, pages 80–87. Institute of Electrical and Electronics Engineers.

Materna, Z., Kapinus, M., Španěl, M., Beran, V., and Smrž, P. (2016). Simplified industrial robot programming: Effects of errors on multimodal interaction in woz experiment. In *25th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2016*, pages 200–205. Institute of Electrical and Electronics Engineers.

Milgram, P., Takemura, H., Utsumi, A., and Kishino, F. (1994). Augmented reality: A class of displays on the reality-virtuality continuum. *Telemanipulator and Telepresence Technologies*, 2351.

Mouloua, M., Gilson, R., Kring, J., and Hancock, P. (2001). Workload, situation awareness, and teaming issues for uav/ucav operations. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 45.

Nielsen, J. (1993). Noncommand user interfaces. *Commun. ACM*, 36(4):83–99.

Plascencia, C. A. and Beran, V. (2018). Experimental flying platform. Technical report.

Plascencia, C. A., Beran, V., and Sedlmajer, K. (2019). Drone sensory data processing for advanced drone control for augmented reality. Technical report.

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

Santos, M. E., Polvi, J., Taketomi, T., Yamamoto, G., Sandor, C., and Kato, H. (2014). A usability scale for handheld augmented reality.

Sauro, J. and Lewis, J. R. (2012). *Quantifying the User Experience: Practical Statistics for User Research*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Schrepp, M. (2015). User experience questionnaire handbook.

Sedlmajer, K., Bambušek, D., and Beran, V. (2019). Effective remote drone control using augmented virtuality. In *Proceedings of the 3rd International Conference on Computer-Human Interaction Research and Applications 2019*, pages 177–182. SciTePress - Science and Technology Publications.

Stadler, S., Kain, K., Giuliani, M., Mirnig, N., Stollnberger, G., and Tscheligi, M. (2016). Augmented reality for industrial robot programmers: Workload analysis for task-based, augmented reality-supported robot control. In *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*, pages 179–184. IEEE.

Teixeira, J. M., Ferreira, R., Santos, M., and Teichrieb, V. (2014). Teleoperation using google glass and ar, drone for structural inspection. In *2014 XVI Symposium on Virtual and Augmented Reality*, pages 28–36.

Temma, R., Takashima, K., Fujita, K., Sueda, K., and Kitamura, Y. (2019). Third-person piloting: Increasing situational awareness using a spatially coupled second drone. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, page 507–519, New York, NY, USA. Association for Computing Machinery.

Turunen, M., Hakulinen, J., Melto, A., Heimonen, T., Laivo, T., and Hella, J. (2009). SUXES - user experience evaluation method for spoken and multimodal interaction. In *Proc. Interspeech 2009*, pages 2567–2570.

Appendix: Selected publications

Simplified Industrial Robot Programming: Effects of Errors on Multimodal Interaction in WoZ experiment

Zdeněk Materna, Michal Kapinus, Michal Španěl, Vítězslav Beran, and Pavel Smrž

***Abstract*— This paper presents results of an exploratory study comparing various modalities employed in an industrial-like robot-human shared workplace. Experiments involved 39 participants who used a touch table, a touch display, hand gestures, a 6D pointing device, and a robot arm to show the robot how to assemble a simple product. To rule out a potential dependence of results on the number of misrecognized actions (resulting, e.g., from unreliable gesture recognition), a controlled amount of interaction errors was introduced. A Wizard-of-Oz setting with three user groups differing in the amount of simulated recognition errors helped us to show that hand gestures and 6D pointing are the fastest modalities that are also generally preferred by users for setting parameters of certain robot operations.**

I. Introduction

Industrial robots were traditionally used mainly in a large-scale production. This was primarily due to the large price of the automation and low flexibility requiring long and costly adaptation for new products. Recently, EU-supported projects as SMErobotics[1] and EuRoC[2] emerged to support development of easily reconfigurable cognitive robots able to achieve flexibility required for small to medium scale manufacturing. Such flexibility must be supported by easy to use and effective human-robot interaction substituting traditional ways of programming industrial robots requiring expert-level knowledge.

Our long-term goal is to create a shared-space environment similar to the experimental setup shown in Figure 1 where a human operator can cooperate with a semi-autonomous cognitive robot using multi-modal interaction and augmented reality: ARTable. The robot within the envisioned solution could be programmed once and then perform independently or it may continuously provide assistance to the operator. There was a research on what modalities are appropriate for what most common operations [1] in such a system. As a first step towards ARTable we were interested in how various modalities would perform in a similar experiment however under realistic conditions. Therefore we designed a WoZ experiment where input modalities were not always working perfectly and participants had to face interaction errors. The aim of the experiment was to uncover whether there is dependence between preference for using particular modality for setting particular parameter and amount of experienced interaction errors. Secondarily, we were interested in how task completion times will be influenced by used modality and amount of errors as a time-effective human-robot interaction will be of paramount importance for a practical usage of such system. Video summary of the experiment can be seen at https://youtu.be/LtiDc3pGjug.

All authors are affiliated with the Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Bozetechova 1/2, Brno, 612 66, Czech Republic. Contacts: *imaterna, ikapinus, spanel, beranv, smrz@fit.vutbr.cz*

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

[2]http://www.euroc-project.eu/index.php?id=challenge_1

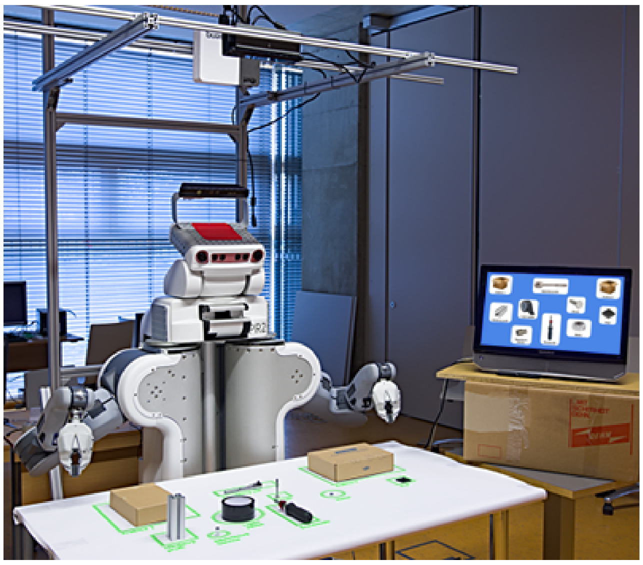


Fig. 1: Prototype of the human-robot shared-space environment with augmented reality user interface (image edited).

II. Related Work

Robot manipulators used to be programmed by experts at a low level making them less flexible to production changes. Recently, approaches allowing high-level programming by end users appeared. One of these approaches is programming by demonstration [2] also referred to as kinesthetic teaching [3], where an operator programs a robot by positioning its end-effector while learning poses [4] and/or forces [5]. Existing solutions can be divided into those allowing so called offline programming where a robot is programmed once [6], [7], those allowing a continuous human-robot collaboration [8] and those allowing both [9] modes. The interface may be for instance projected [10] or integrated into a hand-held device with augmented reality [6], [7]. Interaction also may happen in a virtual reality [9]. Alternatively to positioning a robot's end-effector, a human operator may demonstrate the task by actually performing it [11] or by giving high-level instructions using one [8] or more modalities [6].

Errors in interaction can be according to [12] divided into following types: misunderstandings, non-understandings and misconceptions. For our experiment, we choose to simulate misunderstandings with third-turn repair of the errors. Dealing with errors is often limited to resolving problems during program execution [13]. The experiment with social robot programming [14] where gesture and speech-based interfaces and even the robot's software were not perfectly reliable has shown importance of the provided feedback. However, those errors were not simulated and thus their amount was not controllable. The framework to support WoZ studies from [15] allows to insert given amount of random misrecognition errors, however it is limited to the speech-based interfaces.

Misunderstandings may be caused by a non-perfect input. For instance the pointed object estimation from [16] is reported to have 83% success rate despite usage of a prior information about location of the objects. Another approach to detection of pointing directions [17] achieved $\pm 10^{\circ}$ angular and 93% distance error. The speech recognition system from [18] achieved 16% error in a noisy environment with background TV or radio. It can be speculated that amount of errors would be higher in an industrial environment.

III. User Study Design

The main goal of this study was to find out how errors affect user preference of input modality while programming a robot. We were interested in three industrial use cases: assembly, pick&place and welding of points and seams. These use cases were transformed into a simple product manufacturing scenario, better fitting our laboratory settings. A Wizard-of-Oz approach was utilized to avoid implementation specific errors. Without participant's knowledge, a man in a separated room (wizard) observed the scene through a set of cameras and simulated system responses and a feedback. Moreover, WoZ allowed us to simulate certain amount of errors in interaction.

The experimental setup consisted of a table with a top-mounted Kinect v2 sensor and a projector, a robotic platform (PR2) and a touch screen computer besides the table. All sensors were used only for surveillance purposes. During the experiment, the robot was immobile but it helped to create impression of a real robotic workspace.

A simple GUI was created to give users feedback through the projector mounted above the table. There was a bounding box around each object on the table and a label with its name. The selected object was highlighted and points and lines on the objects (selected by a user) were displayed in a different color. The user interface contained a back button used for stepping back, when the system made an error. The button was projected on the table as a red arrow for each modality except the touch screen (there was an on-screen one). Moreover, there was an area dedicated to projecting additional information, animations etc.

A. Input Modalities

Touch table (A) An object is selected by clicking on its projected description. Welding points and seams are selected on a projected image of the object. Assembly constraints are not set with this modality.

Touch screen (B) An object is selected by clicking on it on a screen. Welding points and seams are selected on a zoomed picture of the object. Assembly constraints are not set with this modality. Theoretically there should not be errors in determination of user intention (e.g. where user clicked), but in such a complex system, there could always raise an error, or a user can accidentally click on a wrong place.

Gesture (C) Objects and welding points are selected by pointing on them with the index finger. Welding seams are selected by hovering over a desired seam with the index finger. A gesture used to specify assembly constraint was up to the user. Hand gesture recognition and hand pointing direction recognition is widely studied problem [19], [20]. Recent research shows that 75 to 98% recognition rate is achievable [16], [17].

6D pointing device (D) Similar to C, but instead of the index finger a 6D pointing device was used. Although detection of pose and orientations of this device is more precise and robust than detection of a hand, there still may be errors caused by a user, who can point on a wrong object, or point imprecisely.

Direct robotic arm programming (E) Selecting of objects and welding points and seams was done by pointing on them with a robot's gripper. Just like the 6D pointing device, determining of pose and orientation of a robotic arm is very precise, due to reading arms actuators' internal state, but it can suffer from the same user errors.

Compared to [1], a direct robotic arm programming and a touch table were added. A speech was considered inappropriate as it is probably not sufficiently robust for noisy industrial environments. Our goal was to perform experiment under realistic conditions and we expected participants (mostly university students) to not believe speech programming without predefined vocabulary could work. Moreover, in [1] speech was the lowest rated modality.

Direct robot arm programming (kinesthetic teaching) is commonly used [3], [21], however we are using this technique in a different manner (e.g. selecting objects instead of teaching robot how to grasp them). Touch-sensitive table could be an advantageous alternative to a touchscreen in an industrial environment, as the feedback, system information and interaction with system is held in the user's working space and due to the fact, a user is not forced to divide attention between more places.

B. Tasks

Each participant was told to program the robot to make a simple assembly and packing in a scenario imitating the most common industrial tasks. The scenario was divided into four tasks, each consisting of ten steps (setting ten parameters) in total:

- Assembly: select two objects (e.g. plastic cover and aluminum profile) and set an assembly constraint(s) (e.g. cover orientation)
- Pick&place: select an object and select a place where to put it

- Welding point: select an object, select four points on its top side (to glue stickers in our scenario)
- Welding seam: select an object, select four edges on its top side (to seal boxes with tape)

Each task consisted of ten steps meaning that participant had to set ten parameters: i.e. five times select an object and place where to put it in the pick&place task or select and object and according of its type select one or two assembly constraints in assembly task (see Figure 2). According to participant's group, there were zero, one and three (i.e. 0, 10 and 30%) errors in each task. For instance, in 30% error-level group the system randomly misrecognized three parameters from ten during each of the four tasks. The errors were generated automatically by our WoZ application and were not influenced by the wizard. Order of tasks and steps was the same for all participants.

We see 0% error rate (used for experiment in [1]) as an ideal state however hardly achievable with most of the modalities. 10% seems to be a current realistic level. 30% was selected as the worst case scenario. We assume it to be the worst error ratio probably acceptable by users.

C. Methodology

The SUXES evaluation method for subjective evaluation of multimodal systems has been adopted [22]. It is based on collecting user's expectation and experience and provides means to analyze various interaction methods. The methodology divides experiment into following four phases:

1) Background Information: The experiment is briefly introduced to the subject by a conducter, who is with the subject during the whole experiment. Then, a background information about subject (i.e. age, technical knowledge etc.) is collected.

2) User Expectation: The conductor introduces the shared workspace, all input modalities and the feedback provided by the projector. The subject is allowed to ask questions and to try any modality. Then the subject fills in the questionnaire about his or her expectations based on the introduction.

3) Experiment and User Experience: The conducter guides the subject through four strictly defined tasks: the subject is told what is the current task and step and what to do when error occurs. The task itself is performed solely by the participant. Each subject performs those four tasks with all five modalities (with exception of assembly task, where modalities A and B are skipped). The order of modalities is random for each subject to prevent a learning effect. After that, the subject answers the same questions as in the previous step.

4) Feedback: The subject answers questions about the system using Likert scale rating (see Figures 3 and 4). Most of the subjects also filled valuable fulltext responses.

D. Participants

The experiment has been conducted with 39 participants assigned randomly into three groups. There were eleven males and two females in each group. Participants were mainly university students and researchers with mean age of 23.7 (CI: 22.5 to 24.9) years. Most of them (30) marked themselves as PC experts and at the same time beginners (23) or advanced (15) in robotics. Majority of participants knew what a touchless interface stands for but never used one (31), some indicated that they already used this kind of interface (7) and only one did not know something like this exists.

The whole experiment took approximately 45 minutes for each participant and the interaction itself was recorded by a video camera. Participants' answers have been collected into a spreadsheet.

IV. Results

Participants from all groups (0, 10 and 30% of interaction errors) ordered modalities according to their preference for setting a given parameter before (expectation) and after the experiment (experience). Mean of the order from expectation phase is denoted as r_B and from experience phase as r_A. Statistically significant differences between r_B and r_A within one group were tested using paired t-test (p_{tp}). Differences for a particular modality across the groups were tested using Kruskal–Wallis test with Dunn's multiple comparisons test p_{Wd}. The same test was also used to compare task completion times. Confidence level of 95% was used for all tests. Experience from all participants (all groups) is denoted as r_{Ao}.

A. Parameters

From the Table I showing users' self-reported data it can be seen for which modality and which parameter there were significant differences between r_B and r_A. Moreover, it can be seen which modality was the most preferred for a given task regardless the amount of errors (r_{Ao}). It should be noted that r_B of C differs between 0% and 30% groups ($p_{Wd} = 0.028$).

Considering the number of significant differences between r_B and r_A from all groups, C and D were ranked significantly better six times, B and E were both worse once and A was worse four times. There are no significant differences in r_B between groups meaning that participants from different groups had similar expectations (with one exception of C in 0% group, parameter *select an object*). Moreover, there are also no differences in r_A. From these results it seems that number of errors in interaction does not have strong impact on preferred modality. In other words, participants from different groups had similar expectations (r_B) as well as experience (r_A). Overall, it seems that participants mostly preferred modalities C, D, followed by A, B and the least preferred was E. Figure 3 shows how participants evaluated expectation and experience for all modalities overall (regardless task).

B. Task Completion Times

Before performing a task the participants were told all relevant information. During the task, only the next step was reminded by the conducter. When beginning the task a participant pressed the "Start" button and then the "Stop" one when finished. We use time between those presses as

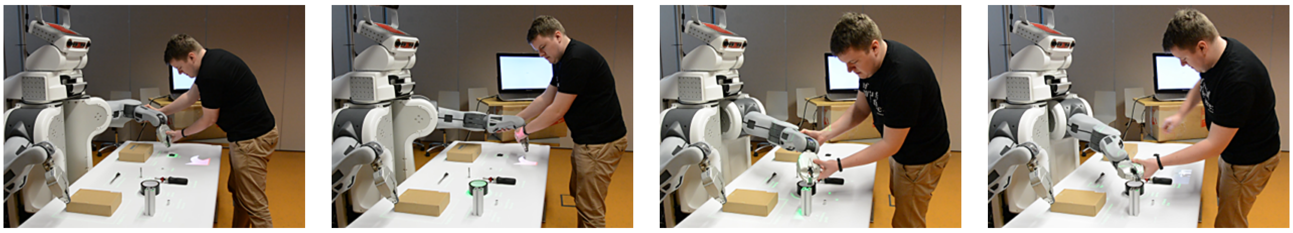

(a) User selects plastic cap to be assembled with aluminum profile.
(b) User performs step back as a tape was selected instead of the profile.
(c) Now the intended object (profile) was selected.
(d) Animation shows how the robot understood user's assembly demonstration.

Fig. 2: An example of a typical interaction for the assembly task using the robot arm as an input modality.

modality	group	select an object				select a place				select a point				select a line				assembly constraint			
		r_B	r_A	p_{tp}	r_{Ao}	r_B	r_A	p_{tp}	r_{Ao}	r_B	r_A	p_{tp}	r_{Ao}	r_B	r_A	p_{tp}	r_{Ao}	r_B	r_A	p_{tp}	r_{Ao}
	0%	3.7	3.3	-		4.3	3.2	0.015		3.0	2.9	-		3.2	3.1	-		2.8	NA	-	
A	10%	4.4	3.4	0.012	3.3	4.6	3.9	-	3.5	3.5	3.1	-	3.2	3.8	3.3	-	3.3	2.9	NA	-	NA
	30%	4.1	3.3	0.0024		4.6	3.5	<0.001		2.9	3.5	-		3.0	3.4	-		2.4	NA	-	
	0%	3.2	2.1	0.02		2.8	2.1	-		3.2	2.3	-		2.9	2.3	-		3.2	NA	-	
B	10%	3.7	3.1	-	2.9	3.1	2.9	-	2.7	3.5	2.9	-	2.8	3.7	3.2	-	2.9	3.0	NA	-	NA
	30%	4.2	3.5	-		3.3	3.2	-		3.5	3.2	-		3.2	3.2	-		2.9	NA	-	
	0%	4.2	4.2	-		3.7	3.7	-		2.7	3.7	0.021		3.0	3.8	-		3.9	4.0	-	
C	10%	2.9	3.6	-	3.8	2.9	3.5	-	3.6	2.4	3.9	0.0031	3.6	2.8	3.9	0.012	3.7	3.6	4.5	0.035	4.1
	30%	2.7	3.6	0.046		2.7	3.6	0.027		2.9	3.3	-		3.4	3.4	-		3.8	3.9	-	
	0%	2.3	3.5	<0.001		2.5	3.6	0.0045		3.7	4.0	-		3.6	3.7	-		2.3	3.5	0.011	
D	10%	2.1	3.4	0.0018	3.3	2.5	3.5	0.012	3.4	3.6	3.9	-	3.7	3.2	3.4	-	3.5	1.9	3.9	<0.001	3.4
	30%	2.7	3.1	-		2.9	3.0	-		3.8	3.2	-		3.5	3.5	-		2.2	2.9	-	
	0%	1.7	1.9	-		1.7	2.5	-		2.4	2.1	-		2.2	2.2	-		2.8	2.6	-	
E	10%	2.0	1.5	-	1.7	2.0	1.4	-	1.9	2.0	1.3	-	1.7	1.6	1.2	-	1.6	3.6	2.5	0.021	2.6
	30%	1.4	1.5	-		1.5	1.7	-		1.9	1.9	-		1.9	1.5	-		3.7	2.9	-	

TABLE I: Participants ordered modalities for each parameter separately from the most preferred (5) to the least (1) before (r_B) and after (r_A) the experiment. Where significant difference was found between r_B and r_A p-value is given. r_{Ao} stands for preference after the experiment regardless of the group (0, 10 or 30%).

an objective measure. The Table II shows those times as well as found significant differences between groups for each modality. Differences between modalities are noted below.

The *assembly* task (consisting of *select an object* and *assembly constraint* parameters) was performed only using C, D and E modalities. In all groups there are significant differences between C and E (0%: $p_{Wd} = 0.003$, 10%: $p_{Wd} < 0.001$, 30%: $p_{Wd} < 0.001$) and between D and E (0%: $p_{Wd} = 0.034$, 10%: $p_{Wd} < 0.001$, 30%: $p_{Wd} = 0.002$).

The *pick&place* task consisted of setting *select an object* and *select a place* parameters. In all groups there are significant differences between E and each of rest of the modalities (with max. $p_{Wd} = 0.049$).

The *welding point* task consisted of setting *select an object* and *select a point* parameters. In 0% group, time for B differs from C ($p_{Wd} = 0.0091$) and D ($p_{Wd} = 0.023$). E differs from C and D ($p_{Wd} < 0.001$). In 10% group, time for A, C and D differs from E ($p_{Wd} < 0.001$). The 30% group shows differences between E and A ($p_{Wd} = 0.0018$) and C, D ($p_{Wd} < 0.001$).

The *welding seam* task consisted of setting *select an object* and *select a line* parameters. In 0% group, there is significant difference only between C and E ($p_{Wd} = 0.0029$). 10% group shows difference between E and A, C, D ($p_{Wd} < 0.001$) and 30% group between E and A ($p_{Wd} = 0.0105$), B ($p_{Wd} = 0.014$), C, D ($p_{Wd} < 0.001$).

For most of the tasks C and D were the fastest modalities followed by A and B. E seems to be unsuitable to the sort of tasks as those in this experiment as even 10% of errors affects performance in three of four tasks. It seems that for other modalities a little amount of errors does not play crucial role.

C. System Opinion

The last phase of the SUXES evaluation contains opinion questions. We used the same questions as in [1], with addition of those related to the erroneous behavior (see Figure 4).

Regardless of the group, participants were satisfied with ease of completing the tasks and with time needed to do so. Participants also claimed it was not difficult to understand how to use different modalities. The results are highly similar to those of [1].

mod.	group	assembly		pick&place		welding point		welding seam	
		mean time [s]	significant differences	mean time [s]	significant differences	mean time [s]	significant differences	mean time [s]	significant differences
A	0%	NA	-	34.7 (27.9, 41.5)	0/30: 0.0017 10/30: 0.038	36.8 (31.2, 42.4)	0/30: 0.003 10/30: 0.0022	33.6 (26.1, 41.0)	0/30: <0.001 10/30: 0.0056
	10%	NA		37.4 (33.5, 41.3)		35.2 (31.6, 38.9)		37.0 (33.4, 40.5)	
	30%	NA		47.5 (42.3, 52.6)		49.1 (43.8, 54.4)		53.13 (47.6, 58.6)	
B	0%	NA	-	32.8 (28.7, 36.8)	0/30: <0.001 10/30: 0.04	38.4 (34.6, 42.1)	0/30: <0.001 10/30: 0.0047	36.9 (31.7, 42.1)	0/30: <0.001
	10%	NA		41.2 (36.9, 45.4)		41.6 (37.3, 45.9)		44.2 (41.2, 47.2)	
	30%	NA		52.3 (47.1, 57.4)		54.5 (50.1, 58.9)		53.6 (48.0, 59.3)	
C	0%	54.8 (46.2, 63.5)	0/30: 0.03	28.0 (25.7, 30.3)	0/30: <0.001	28.3 (24.8, 31.8)	0/30: <0.001 10/30: 0.033	27.6 (24.6, 30.6)	0/30: <0.001 10/30: 0.019
	10%	60.4 (47.9, 73.0)		33.7 (29.4, 38.0)		31.9 (27.6, 36.2)		34.8 (30.9, 38.7)	
	30%	70.5 (61.8, 79.2)		40.9 (37.0, 44.8)		41.2 (36.3, 46.1)		47.4 (41.1, 53.7)	
D	0%	61.5 (45.4, 77.6)	0/30: 0.03 10/30: 0.044	28.8 (25.1, 32.5)	0/30: <0.001 10/30: 0.0037	29.3 (25.1, 33.4)	0/30: <0.001 10/30: 0.002	31.0 (26.6, 35.4)	0/30: <0.001 10/30: 0.0023
	10%	61.0 (50.8, 71.2)		32.3 (29.8, 34.9)		31.9 (28.6, 35.3)		36.7 (32.2, 41.1)	
	30%	88.0 (69.3, 106.6)		43.9 (39.1, 48.6)		44.5 (40.6, 48.3)		52.8 (47.9, 57.7)	
E	0%	90.2 (71.2, 109.2)	0/10: 0.013 0/30: <0.001	43.7 (40.5, 46.9)	0/10: 0.0059 0/30: <0.001	42.9 (38.1, 47.7)	0/10: 0.014 0/30: <0.001	42.6 (35.7, 49.4)	0/30: <0.001 10/30: 0.044
	10%	129.6 (112.2, 146.9)		60.6 (54.9, 66.2)		58.9 (54.1, 63.6)		58.4 (52.7, 64.1)	
	30%	156.7 (127.6, 185.8)		75.3 (69.2, 81.4)		83.0 (68.2, 97.8)		85.1 (68.9, 101.3)	

TABLE II: Task completion mean times (with 95 % confidence intervals) for all modalities, groups and tasks. For each modality, significant differences between times are noted where found in form of $group_x/group_y : p_{Wd}$.

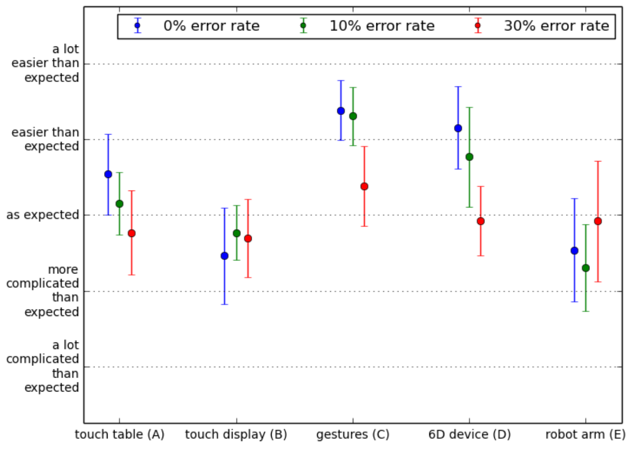


Fig. 3: User's assessment how experience matched expectation.

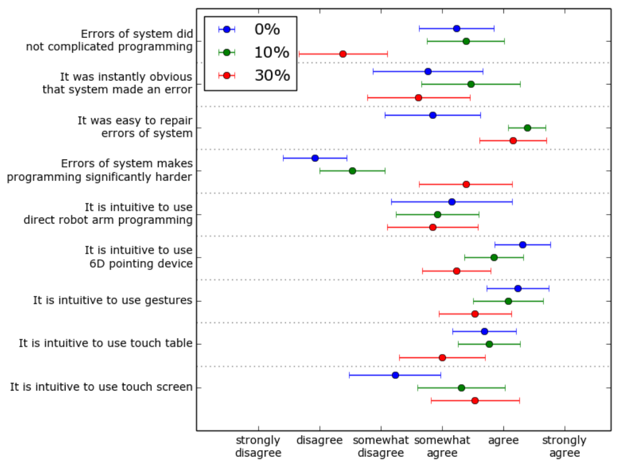


Fig. 4: System opinion

Most of the subjects rated modalities C and D similar, however had a stronger believe in 6D pointing device as they expect it to be more precise than gesture, despite there was the same amount of errors. Participants were also often distracted by the fact, that feedback was always projected on the real objects on the table and not on the place they were working with. Especially, for B most of them would prefer feedback (e.g. selected object) to be shown on the screen and not only on the table. This was however done by purpose, to ensure each modality has exactly the same feedback and participants were noticed about this in advance.

In questions related to erroneous behavior a difference can be seen between error groups. With a growing amount of the errors, perceived intuitiveness of the modalities decreases, except for the touch screen, where it grows (see Figure 4). This could be caused by the fact, that the touch screen is the only control commonly used by the participants. Moreover, the back button was on the screen, so the participants were not forced to think about how to press projected button as for other modalities. Modalities B and E were in general evaluated as the least intuitive. Participants stated that with growing amount of errors, programming was significantly harder and that errors in communication complicated programming.

A few of the participants found out that errors were made by purpose or that some parts of system were simulated. However, according to feedback and discussion with participants, none of them found out the experiment was WoZ.

V. Conclusions

The aim of the conducted experiment was to explore how different modalities used for setting common parameters when programming a robot cope with interaction errors. Participants were divided into three groups according to amount of simulated errors. Their ranking of the modalities before and after the experiment as well as answers from feedback phase were analyzed as subjective measures. Moreover, task completion times were recorded and analyzed as an objective measure.

The gesture and 6D pointing device modalities were the most preferred and fastest modalities in all groups. Touch-sensitive table and display were in general preferred similarly and similar task completion times were obtained. With respect to the task completion times as well as feedback from participants (system opinion) the robot arm seems to be inappropriate as a pointing device for tasks as those in this study and its usage should be reconsidered. It seems that order of preferred input modalities for a given task is not affected by amount of interaction errors. Obtained results support our prior speculation of 10% to be an acceptable level of errors and 30% to be a worst case scenario as especially task completion times grow dramatically.

According to the results, multi-modal interaction based on gestures with complementary usage of a 6D pointing device seems to be promising. We also see touch-sensitive table as a perspective modality however it will be necessary to improve interaction and solve setting more complicated parameters as the *assembly constraint*. The robot arm has advantage of no additional cost however, its usage is physically more demanding than other modalities and for our use-case with relatively simple tasks it had no added value. However, for different types of tasks, e.g. requiring high precision, it could be more useful.

It should be noted that our study simulated the same amount of errors for all modalities. In practice, it can be expected that for instance robot arm modality will be less error-prone than gesture recognition.

As a future work, we will extend the ARTable prototype. The projected interface will provide more information and be fully interactive in conjunction with a touch-sensitive table. Instead of a touch display, a hand-held device or a see-through video glasses with augmented reality will be used. We will also experiment further with robot arm as it could be useful for complex tasks.

Acknowledgments

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science - LQ1602.

References

[1] S. Profanter, A. Perzylo, N. Somani, M. Rickert, and A. Knoll, "Analysis and semantic modeling of modality preferences in industrial human-robot interaction," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ Int. Conference on*, Sept 2015, pp. 1812–1818.

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*. Springer, 2008, pp. 1371–1394.

[3] C. Schou, J. S. Damgaard, S. Bogh, and O. Madsen, "Human-robot interface for instructing industrial tasks using kinesthetic teaching," in *Robotics (ISR), 2013 44th Int. Symposium on*. IEEE, 2013, pp. 1–6.

[4] S. Alexandrova, Z. Tatlock, and M. Cakmak, "Roboflow: A flow-based visual programming language for mobile manipulation tasks," in *Robotics and Automation (ICRA), 2015 IEEE Int. Conference on*. IEEE, 2015, pp. 5537–5544.

[5] F. J. Abu-Dakka, B. Nemec, A. Kramberger, A. G. Buch, N. Krüger, and A. Ude, "Solving peg-in-hole tasks by human demonstration and exception strategies," *Industrial Robot: An Int. Journal*, vol. 41, no. 6, pp. 575–584, 2014.

[6] A. Perzylo, N. Somani, S. Profanter, M. Rickert, and A. Knoll, "Multimodal binding of parameters for task-based robot programming based on semantic descriptions of modalities and parameter types," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Workshop on Multimodal Semantics for Robotic Systems, Hamburg, Germany*, 2015.

[7] J. Lambrecht, M. Kleinsorge, M. Rosenstrauch, and J. Krüger, "Spatial programming for industrial robots through task demonstration," *Int J Adv Robotic Sy*, vol. 10, no. 254, 2013.

[8] J. Norberto Pires, J. Norberto Pires, G. Veiga, and R. Araújo, "Programming-by-demonstration in the coworker scenario for smes," *Industrial Robot: An Int. J*, vol. 36, no. 1, pp. 73–83, 2009.

[9] K. R. Guerin, S. D. Riedel, J. Bohren, and G. D. Hager, "Adjutant: A framework for flexible human-machine collaborative systems," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ Int. Conference on*. IEEE, 2014, pp. 1392–1399.

[10] A. Gaschler, M. Springer, M. Rickert, and A. Knoll, "Intuitive robot tasks with augmented reality and virtual obstacles," in *Robotics and Automation (ICRA), 2014 IEEE Int. Conference on*. IEEE, 2014, pp. 6026–6031.

[11] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *The Int. J of Robotics Research*, vol. 32, no. 8, pp. 951–970, 2013.

[12] G. Hirst, S. McRoy, P. Heeman, P. Edmonds, and D. Horton, "Repairing conversational misunderstandings and non-understandings," *Speech communication*, vol. 15, no. 3, pp. 213–229, 1994.

[13] A. B. Beck, A. D. Schwartz, A. R. Fugl, M. Naumann, and B. Kahl, "Skill-based exception handling and error recovery for collaborative industrial robots," in *Procs. FinE-R Workshop*, 2015, pp. 5–10.

[14] C. Breazeal, C. D. Kidd, A. L. Thomaz, G. Hoffman, and M. Berlin, "Effects of nonverbal communication on efficiency and robustness in human-robot teamwork," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ Int. Conference on*. IEEE, 2005, pp. 708–713.

[15] S. R. Klemmer, A. K. Sinha, J. Chen, J. A. Landay, N. Aboobaker, and A. Wang, "Suede: a wizard of oz prototyping tool for speech user interfaces," in *Procs. of the 13th annual ACM symposium on User interface software and technology*. ACM, 2000, pp. 1–10.

[16] M. Pateraki, H. Baltzakis, and P. Trahanias, "Visual estimation of pointed targets for robot guidance via fusion of face pose and hand orientation," *Computer Vision and Image Understanding*, vol. 120, pp. 1–13, 2014.

[17] D. Shukla, O. Erkent, and J. Piater, "Probabilistic detection of pointing directions for human-robot interaction," in *Digital Image Computing: Techniques and Applications (DICTA), 2015 Int. Conference on*. IEEE, 2015, pp. 1–8.

[18] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[19] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10462-012-9356-9

[20] J. Suarez and R. R. Murphy, "Hand gesture recognition with depth images: A review," in *RO-MAN, 2012 IEEE*, Sept 2012, pp. 411–417.

[21] S. Alexandrova, M. Cakmak, K. Hsiao, and L. Takayama, "Robot programming by demonstration with interactive action visualizations," in *Procs. of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[22] M. Turunen, J. Hakulinen, A. Melto, T. Heimonen, T. Laivo, and J. Hella, "Suxes-user experience evaluation method for spoken and multimodal interaction." in *INTERSPEECH*, 2009, pp. 2567–2570.

Using Persona, Scenario, and Use Case to Develop a Human-Robot Augmented Reality Collaborative Workspace

Zdeněk Materna, Michal Kapinus, Vítězslav Beran, Pavel Smrž
Brno University of Technology, Centre of Excellence IT4Innovations
Manuel Giuliani, Nicole Mirnig, Susanne Stadler, Gerald Stollnberger, Manfred Tscheligi
University of Salzburg, Center for Human-Computer Interaction

ABSTRACT

Up to date, methods from Human-Computer Interaction (HCI) have not been widely adopted in the development of Human-Robot Interaction systems (HRI). In this paper, we describe a system prototype and a use case. The prototype is an augmented reality-based collaborative workspace. The envisioned solution is focused on small and medium enterprises (SMEs) where it should enable ordinary-skilled workers to program a robot on a high level of abstraction and perform collaborative tasks effectively and safely. The use case consists of a scenario and a persona, two methods from the field of HCI. We outline how we are going to use these methods in the near future to refine the task of the collaborating robot and human and the interface elements of the collaborative workspace.

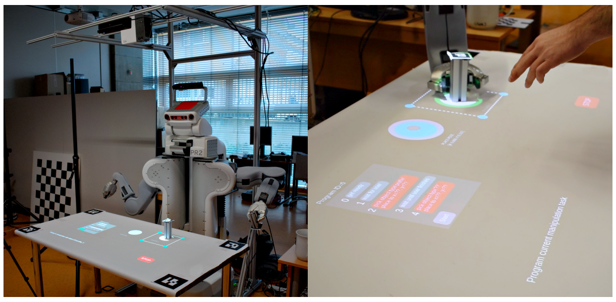
1. INTRODUCTION

With the emergence of affordable industrial collaborative robots it seems likely that SMEs soon will widely adopt such robots in order to achieve higher precision for specific tasks, free experienced employees from monotonous tasks, and increase productivity.

In a large-scale production, robots are usually programmed by an expert. For SMEs, batches are smaller and products may even be customized for a particular contract. Due to this, it would be beneficial to enable ordinary-skilled workers to program robots easily, without robot-specific knowledge. In this work, we present a new approach for simple robot reprogramming. The approach uses augmented reality (AR) to visualize the current program and the state of the robot's learning or execution, detected objects, instructions to a user etc. We describe an existing prototype[1], a use case of aircraft trolleys assembly and how we will apply HCI methods, in particular narrative scenarios and personas, in further development.

[1]The source code and technical documentation is available at https://github.com/robofit/ar-table-itable.

HRI '17 Companion March 06-09, 2017, Vienna, Austria
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4885-0/17/03.
DOI: http://dx.doi.org/10.1145/3029798.3038366



(a) (b)

Figure 1: Experimental setup (a): PR2 robot, top-mounted Kinect 2 and projector, table with AR markers. User interaction (b): adjusting place pose for the grasped object.

2. BACKGROUND

There exist various approaches to the problem of making robot programming viable for non-expert users, e.g., kinesthetic teaching [7] or visual programming [1]. Part of this problem is also the selection of suitable input modalities [6] and modalities for providing feedback to the user. One of the output modalities may be AR based on a hand-held device [8] or projected onto the workplace [4].

Scenarios are narrative stories about specific people and their activities in a specific work situation and context [5]. They describe key usage situations and they cover a multitude of aspects such as involved agents, user goals and background, work practices, system responses, tasks, context, and difficulties. Cooper et al. [3] developed the concept of personas to represent the hypothetical archetypes of users. Personas are not actual users but they represent specific users with their characteristics and work role [5]. They are given a name, a life, and a personality to make them concrete and appear *real*. Personas are an ideal instrument to design for the most relevant and common user classes.

Up to now, there are only few instances, where HCI methods were used in the field of HRI. For instance [2] uses scenarios and personas in the context of industrial robot programming.

3. AUGMENTED REALITY COLLABORATIVE WORKSPACE

The open source experimental setup uses the intrinsically safe PR2 robot as a demonstrator of a near-future collaborative robot and is centered around a table (see Figure 1a) where the HRI occurs. The interaction consists of programming a robot and collaboration on a programmed task. It happens through an interface projected onto the table using pointing gestures as an input modality (see Figure 1b or video[2]). The user is tracked by a Kinect sensor on the robot's head. Skeleton tracking is used to extract information about the user's position and pointing direction. Gestural control was chosen based on results of our previous experiment [6], where it was the fastest and highest ranked modality. We deal with uncertainty of pointing by highlighting pointed area on the table (circle of given radius) which serves as a visual feedback to the user. When this area visually collides with e.g. a highlight area of an object, the object is preselected. If the object is preselected for a certain time, it is selected. Objects in the scene are tracked using a top-mounted camera and AR codes on them. AR codes are also used for calibration of the whole system.

The interface contains various elements to visualize state of the robot and task as e.g. the currently loaded program. A robot's program is displayed to the user during both learning and task execution phases. Currently, the system supports basic instructions as *get ready* (move robot arms to a default pose) or *pick and place* (pick concrete object or object of given type from specified polygon and place it on given pose). The program structure is so far coded separately while program parameters (e.g. object type and place pose for *pick and place* instruction) are set by the user - the interface allows the user to select a program, set or adjust its parameters and then to collaborate on a programmed task with the robot. During program execution, the current program item is highlighted as well as e.g. objects to be manipulated by the robot.

4. USER-CENTERED DESIGN: USE CASE, SCENARIO AND PERSONAS

Based on our experiences from previous projects and discussions with industrial partners, we have defined our scenario as follows: *The user will teach the collaborative robot to assist him in the task of assembling aircraft service trolleys. He needs to show to the robot which parts are needed in every step of assembling, where holes must be drilled, and what parts should be glued together.*

We also defined a persona, who will act as a user in our use case: *Jan, a 22 year old man, recently graduated at technical-based high school. He works as an assembly worker at Clever Aero, a company focused on aircraft equipment. He has no experience with robots, but he loves new technologies and he is really keen into working with robots.*

These tools needs to be refined according to the demographic data, which has to be collected by observing and interviewing actual workers in real factories. Those data will then be transformed into well-defined persona(s), scenario and a use case, in order to update our current setup according to our personas' needs.

5. CONCLUSION AND FUTURE WORK

In our opinion, methods from HCI provide valuable tools to inform and improve HRI. With our paper, we recommend using methods such as scenarios, use cases and personas. Such instruments enable HRI solutions to better integrate user needs such as methods for simplified programming.

In the next step, we will include the results from using these methods (scenario, use case, persona) on our collaborative workspace.

In order to fulfill the defined use case and the corresponding scenario, it is now necessary to implement new robot instructions based on kinesthetic teaching as gluing and drilling. As the task is quite complex, it is inevitable to display the robot's program in addition to showing work instructions for users. The design elements as well as input methods of the user interface are adapted according to the needs of the refined personas. E.g. as our preliminary persona *Jan* often works with touch-based interfaces (phone, tablet) we will add a touch-sensitive layer on the worktable as an alternative input modality. We focus on making the system easily deployable, with multiple sensors and projectors. The user is enabled to switch between various interfaces based on the current task.

These system improvements result directly from our deployment of HCI methods in HRI. Having said this, we encourage other research groups to take a similar approach.

6. ACKNOWLEDGMENTS

This research was supported by the IT4IXS IT4Innovations Excellence in Science project (LQ1602).

7. REFERENCES

[1] Sonya Alexandrova, Zachary Tatlock, and Maya Cakmak. Roboflow: A flow-based visual programming language for mobile manipulation tasks. In *ICRA*, pages 5537–5544. IEEE, 2015.

[2] Petra Björndal, Mikko J Rissanen, and Steve Murphy. Lessons learned from using personas and scenarios for requirements specification of next-generation industrial robots. In *HCII*, pages 378–387. Springer, 2011.

[3] Alan Cooper et al. *The inmates are running the asylum:[Why high-tech products drive us crazy and how to restore the sanity]*. Sams Indianapolis, IN, USA:, 2004.

[4] Andre Gaschler, Maximilian Springer, Markus Rickert, and Alois Knoll. Intuitive robot tasks with augmented reality and virtual obstacles. In *ICRA*, pages 6026–6031. IEEE, 2014.

[5] Rex Hartson and Pardha S Pyla. *The UX Book: Process and guidelines for ensuring a quality user experience*. Elsevier, 2012.

[6] Zdeněk Materna, Michal Kapinus, Michal Španěl, Vítězslav Beran, and Pavel Smrž. Simplified industrial robot programming: Effects of errors on multimodal interaction in woz experiment. In *RO-MAN*, pages 200–205. IEEE, 2016.

[7] Casper Schou et al. Human-robot interface for instructing industrial tasks using kinesthetic teaching. In *Robotics (ISR), 2013 44th Int. Symposium on*, pages 1–6. IEEE, 2013.

[8] Susanne Stadler, Kevin Kain, Manuel Giuliani, Nicole Mirnig, Gerald Stollnberger, and Manfred Tscheligi. Augmented reality for industrial robot programmers: Workload analysis for task-based, augmented reality-supported robot control. In *RO-MAN*, New York, USA, August 2016. IEEE.

[2]https://youtu.be/yYNpKEClclA

Interactive Spatial Augmented Reality in Collaborative Robot Programming: User Experience Evaluation

Zdeněk Materna, Michal Kapinus, Vítězslav Beran, Pavel Smrž, and Pavel Zemčík

***Abstract*— This paper presents a novel approach to interaction between human workers and industrial collaborative robots. The proposed approach addresses problems introduced by existing solutions for robot programming. It aims to reduce the mental demands and attention switches by centering all interaction in a shared workspace, combining various modalities and enabling interaction with the system without any external devices. The concept allows simple programming in the form of setting program parameters using spatial augmented reality for visualization and a touch-enabled table and robotic arms as input devices. We evaluated the concept utilizing a user experience study with six participants (shop-floor workers). All participants were able to program the robot and to collaborate with it using the program they parametrized. The final goal is to create a distraction-free, usable and low-effort interface for effective human-robot collaboration, enabling any ordinary skilled worker to customize the robot's program to changes in production or to personal (e.g. ergonomic) needs.**

I. Introduction

Contemporary collaborative robots are collaborative in the sense that for human workers, it is safe to work alongside them. However, human-robot interaction is very limited if it exists at all: The behavior of the robot is pre-programmed without cognition of an environment, a user, tools, or the parts necessary for a given task. The robots are programmed by domain experts using specialized devices and an expert is needed even for small changes in the program. It is expected that, in the near future, collaborative robots will be cheaper and thus more affordable for small and medium-sized enterprises (SMEs). In such companies, all of the aforementioned issues will be even more prominent. As robots in SMEs will have to deal with higher product variability (smaller batches, customization) it would be beneficial to allow workers with no specific skills to make changes in a robot's program. At the same time, it will be necessary to support a close human-robot collaboration, as with rising cost of human labor, it might be expected that a trend will occur to offload non-ergonomic or repetitive parts of the workflow to robots. In order to allow this, robots will have to perceive and interact.

In this work, we present a novel approach to programming collaborative robots based on cognition, spatial augmented reality (SAR) and multimodal input and output. In order to make programming as simple as possible, programming takes place on a high level of abstraction where no robot-specific knowledge is necessary. Our intention was to make interaction with robots easy, fun, safe and effective.

All authors are affiliated with the Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Bozetechova 1/2, Brno, 612 66, Czech Republic. Contacts: *imaterna, ikapinus, beranv, smrz, zemcik@fit.vutbr.cz*

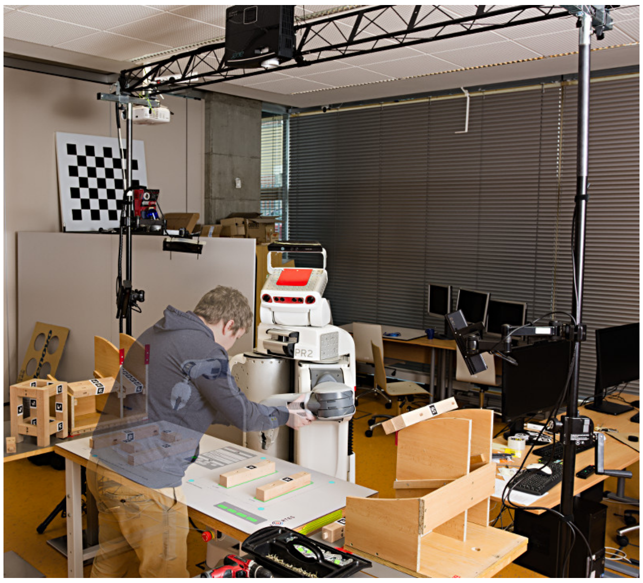


Fig. 1. Setup of the novel interactive system concept where all the interaction elements (visualization and control) are gathered in a shared workspace (example of setting program parameters using a robotic arm and gestures; image edited).

In order to evaluate the approach, we developed a proof of concept system (see Fig. 1)[1] and carried out initial user experience testing. The purpose of the testing was to discover whether there are some fundamental usability issues related to the approach as well as to find out issues related to the current implementation. In the experiment, the robot played the role of a worker's assistant, preparing parts for assembly in a fictional SME.

II. Related Work

Various approaches exist aimed at the simplification of robot programming or to support human-robot collaboration on a joint task. One of the techniques used to make programming robots more suitable for non-expert users is programming by demonstration. For instance, the approach proposed in [1] was rated by non-expert users as highly intuitive. However, the tasks are quite simple and there is

[1]The code is available at https://github.com/robofit/artable.

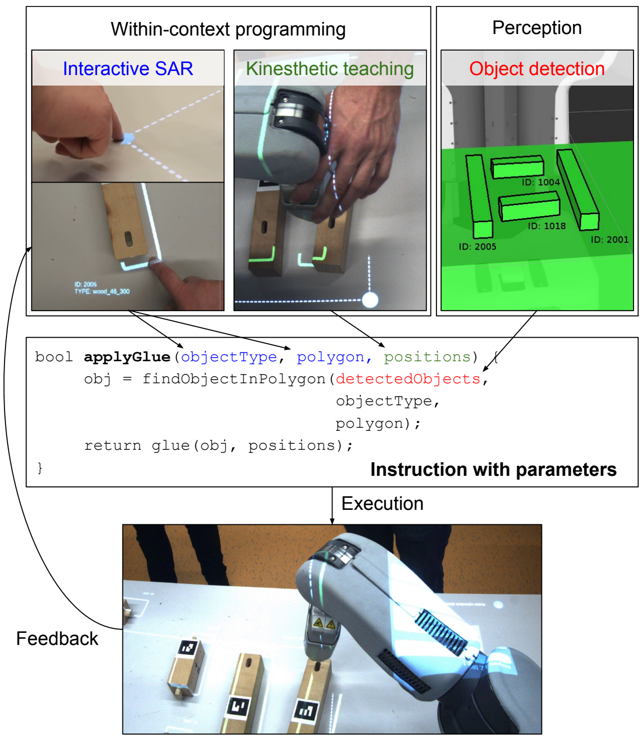
no feedback for the user. In [2], kinesthetic teaching is used in conjunction with an iconic based programming to enable users to create and edit non-trivial programs. While the usage of a graphical user interface (GUI) on a standard monitor adds more control over the program and provides feedback, it also leads to attention switches.

The system described in [3] uses behavior trees to represent the program and was successfully deployed at an SME. The program itself is created on the monitor. The parameters of the program could be set using GUI, object recognition or kinesthetic teaching. The usage of behavior trees leads to high flexibility and the creation of reusable pieces of programs; however, it also inevitably leads to a more complicated GUI. Similarly, the system described in [4] enables users to create complex programs using kinesthetic teaching and object recognition. However, three different GUIs and voice input are involved. Moreover, its target user group consists of general programmers.

The previous approaches share a common disadvantage: The inability to show information within a task context. On the other hand, [5] uses physical blocks to create a program which is highly intuitive (requires no training), although it is limited to trivial tasks. Recently, augmented reality (AR) has been used to show important information within a task context. Probably the most common approach is to use a hand-held device. In [6], the authors recruited robot programmers and evaluated a tablet-based AR interface for programming abstracted industrial tasks. From the results, it seems that the usage of an AR may lead to a decrease in the workload and higher motivation to perform accurately. However, the usage of a tablet prevents the usage of both hands. A head-mounted display frees the user's hands and according to [7] might lead to faster task completion times and higher accuracy. Unfortunately, the currently available devices have a limited field of view. Also, a head-mounted display probably would not be suitable for long-time usage. On the other hand, SAR is able to show information in context, does not require any hand-held devices, is suitable for long-term usage, and is visible to anyone. It was recently used to implement an interactive work desk [8], show instructions to workers [9], or to show robotic data and learn trajectories [10].

To the best of our knowledge, there is currently no existing interactive system targeting all of the following important issues:

- problems with attention switching when a monitor or a hand-held device is used to visualize the programming interface and system status during operation,
- too much information is presented to the user, leading to a higher mental workload,
- external devices are needed to fully interact with the robotic system (during both the programming and processing phases),
- low level of abstraction allowing only medium-expert users to program the robot.



```
bool applyGlue(objectType, polygon, positions) {
    obj = findObjectInPolygon(detectedObjects,
                              objectType,
                              polygon);
    return glue(obj, positions);
}
```

Fig. 2. Illustration of program parameters' definition (combination of manually set parameters by the user with perceived information by the system) and its execution with visual feedback.

III. Proposed Approach

We propose and initially evaluate a novel approach to collaborative robot programming with the following attributes (see also Fig. 2):

- avoiding switching of the user's attention during programming and cooperation by placing all the interaction elements in a shared workspace,
- decreasing the mental demands on the users by presenting the relevant information according to the current context,
- avoiding the usage of further external devices to interact with the system by making the shared workspace itself interactive,
- allowing non-expert users to work with the system by utilizing a high level of abstraction to program a robot.

Based on literature review and the current state of the technology, we see SAR as the most suitable instrument to visualize a user interface within a task context. While previous research has shown that gesture control is the preferred input modality for setting the parameters of common industrial tasks, we decided to use a touch-enabled table, which was also rated highly [11], and which is much more reliable. Moreover, together with SAR, it creates a similar user experience to tablets and smart phones, the usage of which is well-known to the general public. For tasks requiring 3D data input, the robot's arms could be used.

The user interface should be minimalistic, as the interface elements have to share space with real-world objects in the workspace: Tools, parts, etc. However, the design of the elements should allow convenient touch control. Depending on the state of the task, only the relevant information should be shown to lower the cognitive load [12]. The interface should clearly indicate the current state of the system, including an explicit representation of the robot's program and the context of the current program instruction (what happened before it and what is going to happen after it). Additional modalities, such as sound or light, could be used to for instance attract attention in special cases.

In order to make programming as well as the user interface as simple as possible, we decided to use complex instructions with a high amount of underlaying autonomy, at the price of lowering expressivity (see Fig. 2). While theoretically, with the system from [3] one can create complex instructions from basic ones, it also makes the user interface complex and the program representation complicated. For instance, one has to set several poses, specify open and close gripper commands, etc. We believe that, for the sake of simplicity, the user should be abstracted from such low-level commands and the robot should perform them automatically.

To achieve a high level of abstraction and effective collaboration, the robot needs to perceive its surroundings as well as track its human coworker(s) and plan motions according to the current situation.

IV. Proof of concept system

To evaluate the proposed approach, a proof of concept system has been developed. The system allows end-user programming of selected industrial tasks.

A. Setup

The experimental setup (see Fig. 1) was designed to be easy to deploy and modular. It is centered around a standard workshop table equipped with a capacitive touch foil. On the sides, two speaker stands are placed, connected by a truss. The truss is equipped with an Acer P6600 projector. There is a Microsoft Kinect V2 camera on each stand for object detection and calibration of the system. On one stand, there is an additional Kinect for user tracking. Each stand has its own processing unit (Intel NUC) where the projector and sensors are connected (in the study, only one projector was utilized). The unit is connected to the central computer using a wired network. The system is designed to be modular in a way so that it supports $1..n$ stands.

As a demonstrator of a near-future collaborative robot, we use the intrinsically safe PR2. The robot provides an additional set of sensors (Kinect and cameras on the head, cameras in the forearms). There is also a physical stop button under the table which shuts down the robot's motors.

B. System design

The system's state and behavior are defined and controlled by the central node and it can be manipulated by an arbitrary number of user interfaces. For instance, we currently use two interfaces: GUI projected on the table and a sound interface, providing audio feedback (e.g. confirmation of action, errors, etc.).

All parts of the system must be mutually calibrated first. Calibration of the Kinects utilizes an AR tracking library[2] to detect three markers placed on the table. One marker serves as an origin of the coordination system; the two others determine the X and Y axes. The PR2 robot is calibrated in the same way, using a head-mounted Kinect. To calibrate the projectors, a checkerboard pattern is displayed by each projector, and its corners are detected using already calibrated Kinects. In order to calibrate the touch-enabled surface, the points are projected on the table and the user has to click them. Then, homography is computed and used to convert the internal coordinates of the touch device into the common coordinate system.

Each of the objects used in our study has a set of two AR tags printed on the body, and multimarker detection is used to gain a unique ID of the object and its pose. Each object has an object type and a bounding box defined.

The manipulation pipeline is based on MoveIt! [13] and a library for grasp planning[3].

C. Program representation

The program in our system is a set of instructions, collected into blocks. Each program contains $1..n$ blocks; each block contains $1..n$ instructions. Every instruction execution can result in success (e.g. a successfully picked up object) or failure (e.g. failed to apply glue). Based on this result, the next instruction is determined. With this approach, simple branching and cycling of the program are possible (e.g. picking up objects from a feeder until the picking up failed, i.e. until there are no objects left). For an example of a program structure in the form of a graph, see Fig. 5.

Contrary to the conventional methods of programming robots, no precomputed joint configurations or arm paths are stored. By combining the perception capabilities of the system and on-the-fly motion planning, we do not rely on e.g. storing exact object positions.

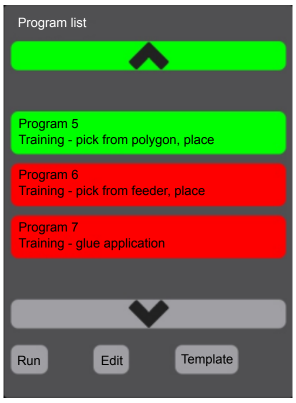
It can be expected that the parameters of the program will be changed more often than the structure of the program. For this reason, we have divided the programming process into two parts. First, an empty template is created offline. This template can be seen as a description of an industrial technological process. It contains a set of instructions with defined transitions; however, without parameters. Thus, the template can be created once and later be adapted to conform to different products by setting instruction parameters.

D. Supported instructions

The system currently supports the following parametric instructions: *pick from polygon* (to pick up an object from a table), *pick from feeder* (to pick up parts from a gravity feeder), *place to pose* (to place a previously picked-up object on a selected place on the table) and *apply glue* (simulated

[2]http://wiki.ros.org/ar_track_alvar

[3]https://github.com/davetcoleman/moveit_simple_grasps

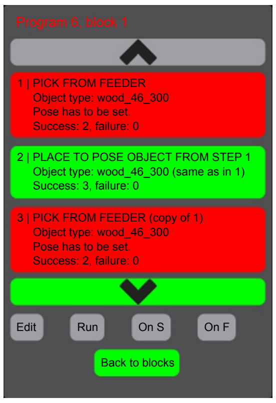

Program list

Program 5
Training - pick from polygon, place

Program 6
Training - pick from feeder, place

Program 7
Training - glue application

(a) List of programs. Green ones are ready to run, red ones need to set parameters.

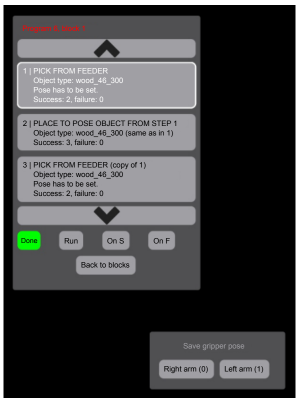

Program 6, block 1

1 | PICK FROM FEEDER
Object type: wood_46_300
Pose has to be set.
Success: 2, failure: 0

2 | PLACE TO POSE OBJECT FROM STEP 1
Object type: wood_46_300 (same as in 1)
Success: 3, failure: 0

3 | PICK FROM FEEDER (copy of 1)
Object type: wood_46_300
Pose has to be set.
Success: 2, failure: 0

(b) List of instructions. Green ones are ready to run, red ones need to set parameters.

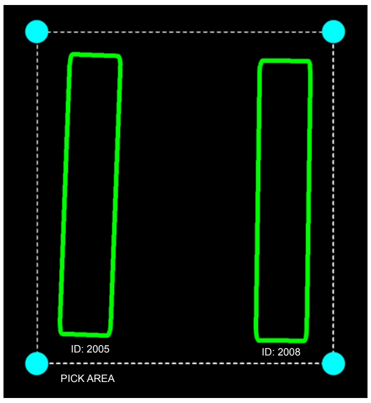

Program 6, block 1

1 | PICK FROM FEEDER
Object type: wood_46_300
Pose has to be set.
Success: 2, failure: 0

2 | PLACE TO POSE OBJECT FROM STEP 1
Object type: wood_46_300 (same as in 1)
Success: 3, failure: 0

3 | PICK FROM FEEDER (copy of 1)
Object type: wood_46_300
Pose has to be set.
Success: 2, failure: 0

(c) A small dialog shows if the robot is able to detect an object in the feeder and allows the user to save the arm pose.



(d) Polygon defining the area on the table from which the objects will be picked up. The green outlines correspond to detected objects.

Fig. 3. Examples of different widgets from a proof of concept system.

gluing). Each of these instructions has certain parameters to be set by the user.

The object type must be set for all of these instructions. For the *pick from polygon* and *apply glue*, a polygon defining the area of interest on the table has to be set, so that the user can limit objects of the given type affected by the instructions.

For the *pick from feeder*, a pre-picking pose (see Fig. 4(c)), used for object detection, has to be set using the robot's arm. While executing this instruction, the robot moves to the stored pose, observes the objects with its forearm camera and picks up the closest object in the direction of the gripper. For *apply glue*, the poses where the glue is supposed to be applied have to be set using an arbitrary arm of the robot.

There are also a couple of non-parametric instructions: *get ready*, *wait for user*, and *wait until user finishes*. The first one moves the robot's arms to their default position. The other instructions allow the synchronization of the system and the user. The *wait for user* instruction will pause the program execution until the user is in front of the table, while *wait until user finishes* will pause the program until the user finishes current interaction with the objects on the table. In our experiments, the behavior of these two instructions was simulated and controlled by the Wizard of Oz approach.

E. User Interaction

The interaction between the user and the system is currently achieved using three modalities: GUI projected on the touch-enabled surface (which serves as an input for the system and feedback for the user), kinesthetic teaching (input to the system only), and sound (feedback for the user only).

The GUI is composed of various widgets. The list of programs (see Fig. 3(a)) shows all the programs stored in the system. The color of each entry suggests whether the program has set all the parameters (green; only these can be started) or some of them are not set (red). Any program can be templated (it is duplicated as a new program, with no parameters set) or edited (the user may set or adjust its parameters). During the program editation, the user can see a list of blocks of the selected program and can edit a selected block or get back to the list of programs.

When editing a block of a program, the list of instructions is shown (see Fig. 3(b)). The selected instruction is always in the middle (with exception for the first and the last one) so the user can see its context. Similarly to the program list, each instruction has either a red or a green background, indicating whether it has all the parameters set. When all the parameters have been set, the selected instruction can be executed. Moreover, a gray instruction background suggests a non-parametric instruction. There are also buttons to navigate through the program, to select an instruction following either the successful or failed execution of the current instruction.

When a program has been executed, the list of instructions differs slightly. All the instructions are grayed out and are not interactive, and the buttons for pausing and stopping the program are displayed. The instruction detail shows: The type of the instruction (e.g. *pick from feeder*), the parameters (e.g. object type) and transitions for success and failure.

The user is notified about the state of the system and the errors, as well as the currently available actions, using a notification bar shown next to the front edge of the table.

It is important for the user to know the state of the system, so for every detected object an outline and ID are displayed (see Fig. 3(d)). The type of the object is displayed upon clicking on the outline. For the purpose of setting the parameters, more information is shown, such as a polygon defining the area on the table, the outline of the object showing the position for object placement, etc. The same is also shown during the program execution, so the user knows in advance what object the robot will work with.

Various dialogs exist which allows the user to specify additional information. For instance, while programming an *pick from feeder* instruction, the user has to specify a pre-pose for object detection by manipulating the robot's arm and then confirming the position using a dialog. The pose is saved after pressing a button corresponding to the arm used (see Fig. 3(c)). The whole procedure is shown in Fig. 4 (a-e).

F. Known Limitations

The main input modality – touch foil – is prone to false readings when metal objects are placed on it, which makes it unsuitable for certain industrial settings. In the future, it might be replaced with or complemented by a vision-based approach (e.g. one from [8]). 3D interaction is currently limited to the kinesthetic teaching of positions, with no means for their later visualization.

V. Evaluation

In order to evaluate the proposed approach and to discover the main usability issues of the early prototype, a user experience testing was carried out[4]. Prior to the experiment itself, a pilot experiment with three subjects (faculty staff) took place, which helped us to verify the functionality of the prototype and to create the final experiment design.

As measures, we choose a combination of qualitative and quantitative data. Self-reported data were obtained using a questionnaire consisting of the System Usability Scale (SUS) [14], NASA Task Load Index (TLX) [15] in its raw form (simplified, with a scale in the range [1..7]) and a custom questionnaire focusing on the specifics of the system. We recorded the task completion times and the corresponding number of moderator interventions as quantitative data.

A. Experiment protocol

The experiment protocol consisted of four phases. None of the phases of the experiment was time-limited. There were one moderator and one operator in a separate room in charge of system monitoring, data recording, and WoZ (used solely to simulate user activity recognition).

1) Introduction: At the beginning of the experiment, the participants signed an informed consent form. They were told a story about a fictional SME producing wooden furniture: *"The company cannot afford a dedicated robot programmer, so it bought a collaborative robot programmable by any ordinary skilled worker. The robot will serve as an assistant preparing the parts for the workers who will do the assembly."* They were given information about safety, the parts of the workspace (interactive table, robot, feeders with furniture parts), and basic usage of the interface.

2) Training: The training phase consisted of three simple programs demonstrating the supported instructions. No specific product was assembled in this phase. The parameters of each program were first set by the participant and then the program was executed. During the execution, errors (e.g. a missing object) were intentionally invoked in order to gain familiarity with the error resolution dialog. In this phase, the moderator proactively helped the participants to complete the tasks and answered all the questions. A short practice of the think-aloud protocol followed. After that, the participants were told to set the parameters of those three programs independently while thinking aloud.

3) Main task: The assembly process of a target product (a small stool) was explained and the participants assembled it manually. Next, the structure of the corresponding program and the expected workflow were explained.

After the questions were answered, the participants started working. When finished, they started the program and collaborated with the robot on the task of producing a stool. Two stools were produced and the participants were told that there was a demand to adapt a product - to produce a higher stool. After the parameters of the program had been adapted, they produced one more.

4) Feedback: After finishing the tasks, an open discussion took place. The participants were asked for their impressions, additional questions, etc. Then, they were asked to fill in the questionnaire.

B. Stool assembly

The intended workflow of the main task is that the user does the assembly while the robot prepares the parts needed in the next steps "on background". The program is divided into three blocks (see Fig. 5). Blocks 1 and 2 have the same structure and serve to prepare the parts for the sides of the stool (two legs, two connecting parts, application of glue). The purpose of two blocks is that the user might set parts within one block to be supplied from e.g. the left feeder and in the other block from the right feeder. Block 3 serves to prepare the connecting parts for the final assembly of the sides of the stool.

C. Participants

In cooperation with an industrial partner (ABB Brno), six regular shop-floor workers of various ages, genders and technical backgrounds were selected (out of 27 volunteers) to take part in our study. These participants will be labeled as Participants A, B, C, D, E and F. Five of them work in quality control; one (E) works as a mechanic. The demographic data of the participants can be seen in Table I.

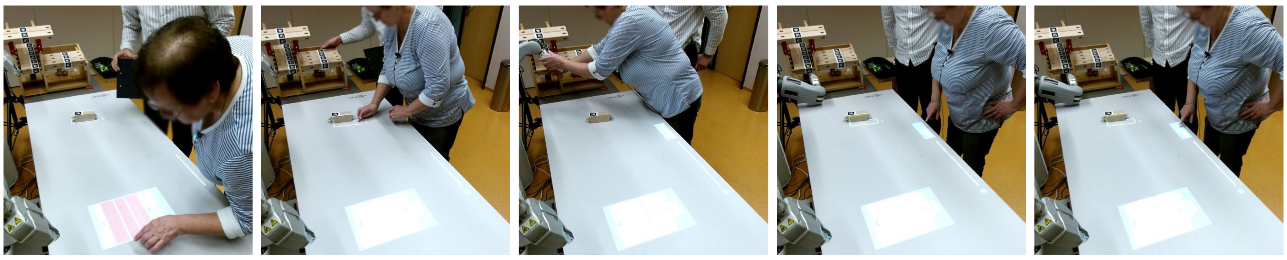
VI. Results

The section provides results of the experiment.

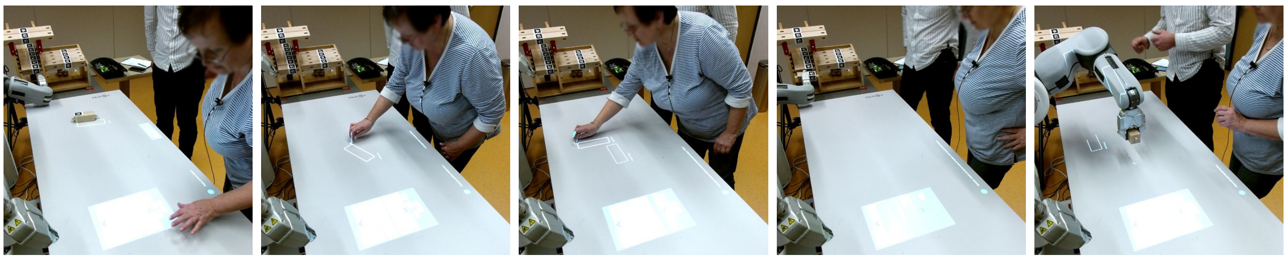
A. Qualitative and quantitative data

Table II shows the results per participant. The mean time to complete the main task was 2711 s (SD 620 s) with 11.7 (SD 6.7) moderator interventions. The main task consisted of setting the following instructions: 5x *pick from feeder* (2 parameters), 12x *place to pose* (1 parameter), 2x *apply glue* (4 parameters), resulting in settings of 30 parameters in total. The mean time for program adaptation task was 1053 s (SD 215 s). It consisted of setting: 2x *pick from feeder*, 2x *apply glue*, and optionally, adjustment of place poses (based on previously set poses), resulting in at least 12 parameters in total. These times include the delays caused by system errors (unreliable object detection, unstable manipulation pipeline, etc.). The mean SUS rating was 75.8 (SD 8.9), while for

[4]Overview of the experiment: https://youtu.be/cQqNLy6mE8w.



(a) User selects instruction to be set from list (pick). (b) Object type is set by touching its outline. (c) Robot arm is used to teach detection position. (d) Dialog shows if robot is able to detect object in feeder. (e) User saves position (confirmation sound is played).



(f) User selects follow-up instruction (place). (g) User adjusts place pose by dragging it on the table. (h) Another pose, first one also shown for convenience. (i) User tests *pick from feeder* instruction. (j) Test of *place to pose* instruction.

Fig. 4. An example of human-robot interaction during the experiment. In this case, the user sets parameters for two *pick from feeder* instructions (one shown) and consequent *place to pose* instructions (both shown). Then, instructions are tested. Two input modalities are used: touch table and robot arm.

part.	gender	age	education	experience with robots	attitude towards new technology
A	F	57	vocational (technical)	none	skeptical
B	M	46	secondary (technical)	seen robot at least once	neutral
C	F	27	secondary (economics)	none	neutral
D	M	33	secondary (technical)	seen robot at least once	early adopter
E	M	24	secondary (technical)	works on workplace with robots but not next to them	neutral
F	M	34	undergraduate (technical)	none	skeptical

TABLE I
DEMOGRAPHIC DATA OF THE PARTICIPANTS.

comparison, the system from [4] was scored 66.75 (SD 16.95). The mean TLX was 33.3 (SD 8.8).

From the custom questions (see Table III) it seems that the participants in general liked interacting with the system and felt safe; however, they were confused from time to time. However, during the experiment, in most cases it was enough to tell them to check the notification area and they were able to continue afterwards.

B. Programming

Observation of the users has shown that the current visualization of the robot program is probably not sufficient, as it often took considerable time to realize what was currently being programmed, especially for the case of repeating sequences of program items (e.g. *pick from feeder*, *place to pose*, *pick from feeder*, *place to pose*). Not fully consistent terminology (e.g. program instruction was sometimes referred to as item and sometimes as step) may have contributed to this. Probably because of the similar appearance, for some participants it was difficult at the beginning to distinguish between a program block and a program instruction.

Probably the most common issue during programming was the participant forgetting to press the *Edit* button in order to switch from the view-only mode to the parameter settings mode for the selected instruction. The participants often tried to adjust for example place pose and were confused as to why it was impossible. Also, it was often unclear that it is only possible to execute individual instructions. Initially, two participants thought that the instructions (displayed in the program visualization) were for them, so they should perform e.g. *pick from feeder*. One participant asked if there are also assembly instructions for the workers.

There have been cases where the user accidentally changed the selected object type. Despite the fact that this was covered during training, some of the participants thought that the object type is selected when they put an object of that type on the table. It seems that although the objects of a selected type were highlighted differently (with a green outline), most of the participants only guessed what type was selected, or rather, checked it in the program visualization where the information was in textual form.

C. Individual instructions

1) Pick from feeder: Participants were often confused, as it was required to select the object type on the table and then to use a robot arm to set the pose enabling the detection of parts in the feeder. We noticed cases where the

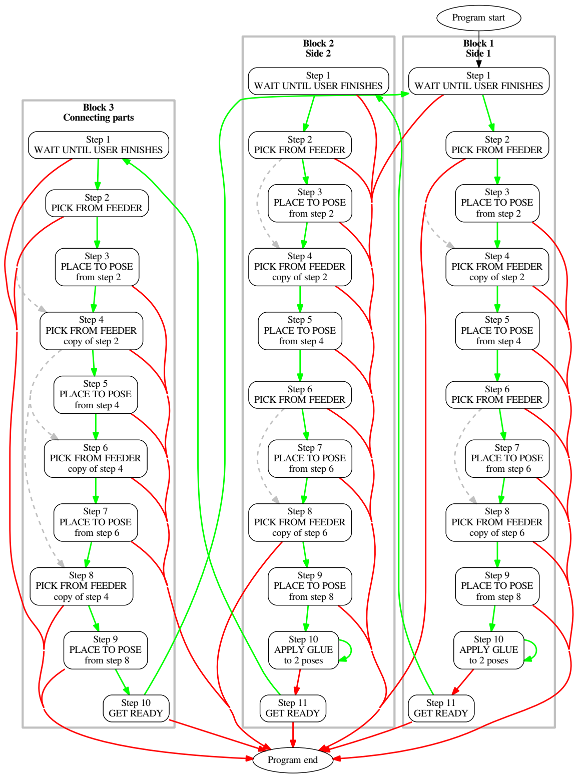


Fig. 5. Stool production program. The green edges represent *on_success* transition, while the red ones represent *on_failure*. The grey edges show dependencies. In the case of *apply glue*, there is a loop. The robot applies glue to one object in a specified area. If an object is found, the program flow continues to the *on_success* instruction - it tries to apply glue to another object. If there is no object without glue applied, the flow continues to *on_failure* (next instruction).

participant tried to select an object by knocking on it (instead of clicking on its outline), both on the object on the table and in the feeder. The participants commonly skipped the object selection, grabbed the robot arm and tried to set the pose, even above the object on the table, despite the fack that they were learning picking from feeder. After pressing *Edit*, dialog buttons for saving the arm pose (grayed-out at the time) were sometimes used "to select arm" before any other interaction. Most users took a new part from the feeder and put it on the table when they needed to select the object type even though there were already objects of that type that could have been used for this purpose. When adapting the program, it happened twice, that the participant by mistake set the position for the other feeder (e.g. the instruction originally used the left feeder, and they switched to the right one). This would mean that the robot would not be able later to place the object, as the following place pose (on the opposite side of the table) would be out of its reach.

2) Place to pose: Common sources of problems were unreachable place poses, or place poses too close to each other, which prevented the robot from placing parts successfully. The only possibility was to find out by trial and error. For all the participants, it was difficult initially to handle separated translation (by dragging) and rotation (using a pivot point). Some of them intuitively attempted to use multi-touch gestures (not supported by the interface thus far), including one participant who does not own any touch devices. Although the initial position of the place pose was in the middle of the table, some participants had trouble finding it, especially if there were many objects around. Some of them tried to drag the outline of a detected object or even placed an object into the outline of the place pose. Visualization of the place poses from other instructions (differentiated by a dotted line and a corresponding instruction number) were confused a few times with the current place pose and the users tried to move them.

For successful collaboration with the robot, it was necessary to organize the workspace so that the robot could prepare the parts for the next steps, while the user did the assembly. Only Participant B explicitly thought about organization of the workspace. The others had minor problems with it or required help. Participant C placed the parts in a very chaotic way. The participants were explicitly told during training that they may move widgets (e.g. program visualization) across the table; however, most of them did not use it and rather adjusted the place poses so that they did not collide with the widget.

3) Glue application: The most common issues were object type selection (attempts to select using the robot's arm) and difficulties with the number of actually stored poses (shown textually). The fact that it is necessary to store required poses only with regard to the one object and the fact that the robot will do it in the same way for other objects in a given area was also generally unclear.

D. *Program execution*

During the program execution, errors occurred relatively often, especially when the robot tried to place an object; erroneous detection prevented it from doing so. In the event of an error, a dialog appeared and sound was played. Most issues were solved just by pressing the *Try again* button. The participants were explicitly told to pay attention to errors. Some of the participants reacted immediately, others after some time and one seemed to ignore the errors and had to be told to solve them. Once in a while it was necessary to warn a participant that he or she was blocking the robot by occupying part of the table where the robot was meant to place parts.

E. *General findings*

No one complained about imperfections of the projection (shadows, inaccurate registration), low readability of the text, interface response times, etc. Each participant had an issue at least once with a non-touchable margin of the interactive table, which was not indicated by the projected interface. There were also issues with pressing the buttons twice, where user tried, for example, to select an instruction which was immediately unselected. While inactive buttons were grayed

Measure	A	B	C	D	E	F
System Usability Scale	87.5	67.5	77.5	75.0	85.0	62.5
Simplified TLX	25.0	33.3	30.6	22.2	41.7	47.2
time to set program (s)	3849	3025	2618	2217	2661	1897
interventions	21	7	20	12	6	4
time to adapt program (s)	1088	1447	1118	958	738	968
interventions	11	4	12	2	2	2

TABLE II

QUALITATIVE MEASURES, TASK COMPLETION TIMES (STOOL PROGRAM) AND NUMBER OF MODERATOR INTERVENTIONS (INCLUDING ANSWERING QUESTIONS).

Statement	A	B	C	D	E	F
Collaboration was effective.	5	4	5	5	4	4
I felt safe.	4	5	5	5	5	5
Robot motions were uncomfortable.	2	1	1	1	1	1
It was easy to see what the robot was about to do.	4	5	5	4	4	2
The robot hindered me at work.	1	2	1	1	1	1
I watched every movement of the robot.	3	1	2	3	4	2
Learning the robot using its arm was intuitive.	4	4	5	5	5	4
Learning the robot using the interactive table was intuitive.	4	4	5	5	5	3
Interactive table shows all necessary information.	5	2	5	5	5	4
Sometimes I did not know what to do.	5	5	4	2	4	4

TABLE III

CUSTOM QUESTIONNAIRE, 1 - TOTALLY DISAGREE, 5 - TOTALLY AGREE

out, most users tried to press them anyway when they thought they should work.

With many objects on the table or during the stool assembly, there was considerable visual clutter. Interestingly, no one mentioned it. Difficulties with moving interface elements (e.g. place pose) across longer distances were observed, especially if there were many objects on the table. Again, no one complained or asked if there was an alternative method to dragging.

As a complementary modality, there were sounds (confirmation, warning, error). Only Participant B explicitly appreciated it.

Regarding safety, only Participant A once noted that a particular movement was probably not safe. No one used the emergency stop button.

VII. CONCLUSIONS

In this work, we targeted problems of the existing solutions in the area of interaction between the human workers and the industrial collaborative robots, particularly in the context of programming robots in SMEs. The proposed and tested interaction system is an attempt to reduce the mental demands and attention switching by centering all interaction elements in the shared workspace. This is achieved by the interactive SAR (combination of projection and a touch-enabled table) and kinesthetic teaching. Non-expert users program a robot on a high level of abstraction, and work within the task context, free of any additional external devices and with immediate visual feedback.

The conducted user experience tests proved the potential of our concept when all six regular shop-floor workers were able to program the robot to prepare parts for a stool assembly, to collaborate with the robot, and to adapt the program for an alternative product within a reasonable time.

During the experiment, no fundamental issues forcing us to reconsider the approach were found. However, the task state awareness in particular has to be improved as well as support for the workspace layout. The participants rated the system positively despite a number of minor usability issues and system errors caused by its experimental nature.

In addition to the revision of the interface to solve the usability issues, we plan to investigate multi-touch support, group operations, intelligent placement of user interface elements, and visualization of robot reachability.

ACKNOWLEDGMENTS

The work was supported by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602".

REFERENCES

[1] E. M. Orendt, M. Fichtner, and D. Henrich, "Robot programming by non-experts: Intuitiveness and robustness of one-shot robot programming," in *RO-MAN*. IEEE, 2016, pp. 192–199.

[2] M. Stenmark, M. Haage, and E. A. Topp, "Simplified programming of re-usable skills on a safe industrial robot: Prototype and evaluation," in *HRI*. ACM, 2017, pp. 463–472.

[3] K. R. Guerin, C. Lea, C. Paxton, and G. D. Hager, "A framework for end-user instruction of a robot assistant for manufacturing," in *ICRA*. IEEE, 2015, pp. 6167–6174.

[4] J. Huang and M. Cakmak, "Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts," in *HRI*. ACM, 2017, pp. 453–462.

[5] Y. S. Sefidgar, P. Agarwal, and M. Cakmak, "Situated tangible robot programming," in *HRI*. ACM, 2017, pp. 473–482.

[6] S. Stadler, K. Kain *et al.*, "Augmented reality for industrial robot programmers: Workload analysis for task-based, augmented reality-supported robot control," in *RO-MAN*. IEEE, 2016, pp. 179–184.

[7] E. Rosen, D. Whitney *et al.*, "Communicating robot arm motion intent through mixed reality head-mounted displays," *arXiv preprint arXiv:1708.03655*, 2017.

[8] R. Xiao, S. Hudson, and C. Harrison, "Supporting responsive cohabitation between virtual interfaces and physical objects on everyday surfaces," *HCI*, vol. 1, no. 1, p. 12, 2017.

[9] M. Funk, "Augmented reality at the workplace: a context-aware assistive system using in-situ projection," 2016.

[10] F. Leutert, C. Herrmann, and K. Schilling, "A spatial augmented reality system for intuitive display of robotic data," in *HRI*. IEEE Press, 2013, pp. 179–180.

[11] Z. Materna, M. Kapinus *et al.*, "Simplified industrial robot programming: Effects of errors on multimodal interaction in woz experiment," in *RO-MAN*. IEEE, 2016, pp. 200–205.

[12] D. Wurhofer, V. Fuchsberger *et al.*, "Insights from user experience research in the factory: What to consider in interaction design," in *Human Work Interaction Design. Work Analysis and Interaction Design Methods for Pervasive and Smart Workplaces*. Springer, 2015, pp. 39–56.

[13] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[14] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[15] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," *Advances in psychology*, vol. 52, pp. 139–183, 1988.

Combining Interactive Spatial Augmented Reality with Head-Mounted Display for End-User Collaborative Robot Programming

Daniel Bambušek, Zdeněk Materna, Michal Kapinus, Vítězslav Beran and Pavel Smrž

***Abstract*— This paper proposes an intuitive approach for collaborative robot end-user programming using a combination of interactive spatial augmented reality (ISAR) and head-mounted display (HMD). It aims to reduce user's workload and to let the user program the robot faster than in classical approaches (e.g. kinesthetic teaching). The proposed approach, where user is using a mixed-reality HMD – Microsoft HoloLens – and touch-enabled table with SAR projected interface as input devices, is compared to a baseline approach, where robot's arms and a touch-enabled table are used as input devices. Main advantages of the proposed approach are the possibility to program the collaborative workspace without the presence of the robot, its speed in comparison to the kinesthetic teaching and an ability to quickly visualize learned program instructions, in form of virtual objects, to enhance the users' orientation within those programs. The approach was evaluated on a set of 20 users using the within-subject experiment design. Evaluation consisted of two pick and place tasks, where users had to start from the scratch as well as to update the existing program. Based on the experiment results, the proposed approach is better in qualitative measures by 33.84 % and by 28.46 % in quantitative measures over the baseline approach for both tasks.**

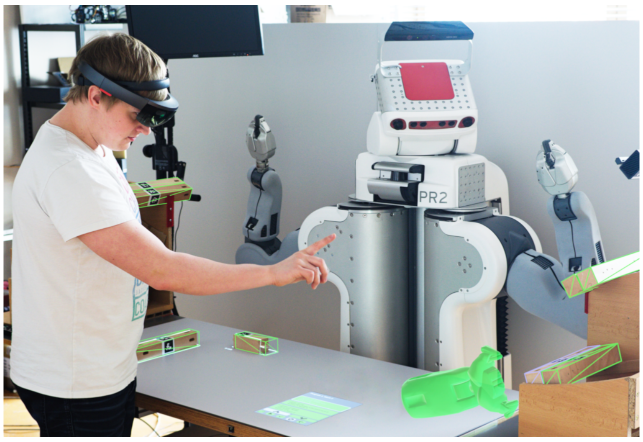


Fig. 1: Spectator's view of the collaborative workspace, with projected user interface, extended by virtual objects seen through the HMD. Example of setting program parameters using the HMD gestures.

I. Introduction

As industrial collaborative robots are getting more affordable, it is likely that small and medium enterprises (SMEs) 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, we created a prototype of a human-robot collaborative workspace – the ARCOR [1], which presents a novel approach to programming robots based on cognition, spatial augmented reality and multimodal input and output[1].

This work extends our previous solution that was dependent on a robot's presence when programming the workspace and was able to convey 2D visualization only. Integration of the head-mounted display (HMD) adds the possibility to quickly and easily visualize 3D information as e.g. pick and place positions[2] (see Figure 1). Moreover, it has a potential to at least partially eliminate the problem which occurred during the experiment from [1] where users had troubles with orientation in individual programs of the ARCOR system. Regarding the end-user programming, we focused mainly on simplifying the *pick&place* task, lowering its time to completion and the user task load.

The presented extension of the ARCOR also addresses the case of SMEs where there are more workspaces than robots. Robots are then moved between them in order to work on a workspace-specific task. To minimize enterprise losses inflicted by robot's idle time, it would be advantageous to enable workers to program the workspace even though the robot is currently working elsewhere.

All authors are affiliated with the Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Bozetechova 1/2, Brno, 612 66, Czech Republic. Contacts: $bambusekd, imaterna, ikapinus, beranv, smrz@fit.vutbr.cz$

[1]github.com/robofit/arcor

[2]github.com/xBambusekD/ar2cor

II. Related Work

Recently, variety of solutions allowing end users to program robots based on AR were published. Those were based on a handheld device [2]–[4], a HMD [5], [6] or a camera-projector solution [7]–[9]. When designing the AR interface, perceptual issues as e.g. a limited field of view, a depth ordering and occlusion introduced by the selected technology and used method has to be taken into account [10]. Despite the above-mentioned problems, the AR has potential to improve HRI. For instance, it could help to avoid context switches which are normally inevitable when the user has to observe the real environment and the robot as well as the video interface [5]. Another usage could be to convey the robot's intents, especially for appearance-constrained robots [6]–[8] not able to convey those by other means.

Nowadays, spatial augmented reality (SAR) seems to be a highly promising method enabling users to interact with the robot within the task-context. For instance, its use was investigated to program a mobile welding robot [11] or in a long-term study focused on projecting assembly instructions [9]. In contrast with handheld devices, SAR has following advantages: both hands are free, projection is visible by anyone, no physical load caused by need of holding the device. On the other hand, it cannot provide free-space 3D visualization.

For unconstrained 3D interaction, HMD with integrated gesture recognition and self-localization capabilities could be used, as e.g. Microsoft HoloLens, which was a first self-contained and un-tethered device of this type. The existing solutions based on HoloLens HMD include functionality as e.g. setting of trajectory waypoints [12], previewing robot motions [13], [14], or programming of a simple pick and place task [15]. In various experiments, interfaces based on HoloLens were in many aspects (task completion times, intuitiveness, physical effort) found superior to 2D interfaces [13], [16] or to kinesthetic teaching [12]. However, for robotic applications, HoloLens limited scanning accuracy of 1-2 mm and precision of 3-5 mm [15] has to be taken into account. Moreover, interfaces has to be designed with its narrow field of view (FoV) in mind. Although the usage of HMD similarly to SAR frees users' hands, there is question of its long-term use suitability: perceived discomfort, or possible health risks.

In our approach, the HMD is used as an extension to the existing ISAR-based (interactive SAR) user interface, where it aims to provide means for effective 3D interaction (instead of kinesthetic teaching) and visualization (which was previously not possible at all). Up to our knowledge, this unique combination of the two AR techniques was not so far described in the literature. It enables us to overcome shortcomings of particular modalities and provides seamless interactive environment for letting unskilled users to program complex robotic tasks.

III. Proposed Mixed Reality Interface

The baseline approach of the end-user robot programming uses the ISAR in a combination with kinesthetic teaching (ISAR-KT). We use the kinesthetic teaching only for setting the target pose, while the robot computes trajectory to it by itself according to the current state of the workspace. In order to fulfill outlined goals (remove robot dependency, reduce programming completion time and user's task load), we replaced the kinesthetic part with the HMD. Thus, we are proposing an approach that uses a combination of the ISAR with HMD (ISAR-HMD)[3].

A. Setup

The ARCOR setup, that we created, consists of a projector, which projects a user interface onto a touch-enabled table that forms the ISAR, two Microsoft Kinect sensors, two speakers placed beneath the table and a robot. As a demonstrator of a collaborative robot, the PR2 is used. For

[3]Video of the proposed approach: https://youtu.be/MNXhqpFBy9Y.

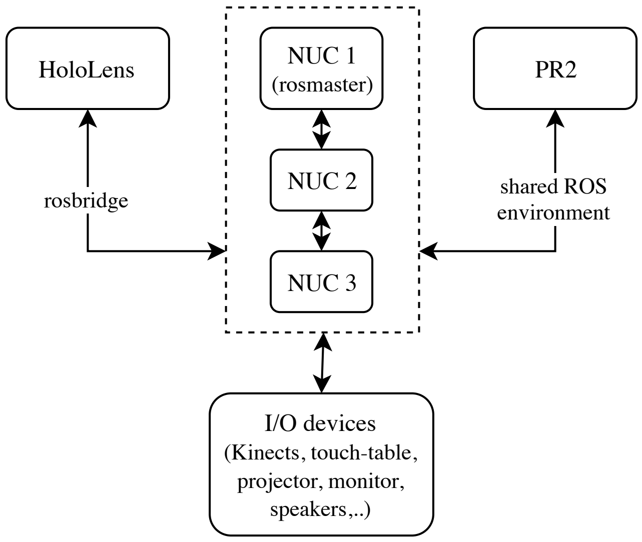


Fig. 2: ARCOR architecture. Intel NUCs are used as processing units, communicating together via local network. The whole system runs on Ubuntu along with ROS. Microsoft HoloLens communicates with the ROS environment via the rosbridge API.

a computational power, three processing units (Intel NUC) connected into a wired network, are used. The base setup is described in more detail in [1].

In order to overcome the inherent limitations of the ISAR-based solution, a HMD (Microsoft HoloLens 1) was integrated, which serves for visualization and interaction in 3D, whereas ISAR could still be used for tasks where 2D interaction is sufficient. The HoloLens communicates with the main processing unit, which runs on the Ubuntu 14.04 along with ROS Indigo, via the rosbridge API (see Figure 2). Since there are individual sensing units – Kinects, robot, HoloLens – they all need to be calibrated with respect to a common point (corner of the table). The calibration procedure is based on detection of a known AR marker.

Projected user interface offers various widgets. Most crucial is a program list, where all stored programs are displayed. User can either edit parameters for already set program or template selected program, which will create a new instance of it with empty parameters. Each program can be composed of multiple blocks, where each block has a set of instructions for which the parameters, like pick position, object type, place position, etc. needs to be set.

B. Problems to Solve in the Pick&Place Use Case

We focused mainly on simplifying the *pick&place* learning procedure. More specifically, on a scenario where user wants to set the robot to pick an object from the feeder and place it on the table.

In order to properly program this task in the ARCOR setup, the user has to set three main parameters – *detection position*, *object type* and *place position*. Due to the HoloLens limited scanning accuracy (1-2 mm), its precision (3-5 mm) [15] and a possible inaccurate user input, robot's cognitive abilities (attached forearm cameras) are used, which are able to find the object of specified type from the detection position in order to determine precise picking position.

Since the proposed solution (ISAR-HMD) is aiming on elimination of robot's presence, the user uses only touches of the touch-enabled table and gestures of the HMD for interaction with the system. Without the kinesthetic teaching and with the goal of keeping the ISAR-HMD as simple as possible, we needed to solve:

- How to efficiently select the object type to be picked up.
- How to set the detection position, from which the robot will be able to detect and pick up the object.
- How to set the place position and its rotation.

C. Solution to the Pick&Place Use Case

Based on the results from [14], a heading-based selection, where user is using his gaze for targeting (indicated with virtual cursor) and a hand for the selection gesture (HoloLens Air tap – equivalent of mouse click), is used.

For a sake of efficiency (lowering the number of actions the user has to take), setting the picking instruction (named *pick from feeder* in the ARCOR system) and setting the placing instruction (named *place to pose* in the ARCOR system) is tied together to form a fluent procedure.

All visible objects in the scene are detected and registered (Kinect sensors), making them interactive for the HMD. While the user is gazing at such detected object during setting the detection position, visual feedback – in a form of virtual robot gripper rendered with $0.3\,m$ offset from the HMD's cursor – is provided. This gripper, which is automatically positioned against the side of the object the user is looking at, is indicating current detection position directly in the scene. When colored green, the robot will be able to detect and pick up the object, when colored red, the robot will not be able to do so.

Final stage of setting the object type and the detection position is merged into one action – HoloLens Air tap gesture (equivalent of mouse click) on desired object in feeder. As the object is detected, the system automatically recognizes and saves the type of it, and as the HMD is calibrated with respect to the ARCOR system, the position and rotation of the virtual gripper is transformed to the ARCOR coordinate system and saved as the detection position.

Since the setting of the *pick from feeder* instruction is tied up with the setting of the *place to pose* instruction, a virtual object of the type the user selected in the previous step is created and attached to the end of user's gaze in order to create the illusion of naturally picking an object from the feeder and placing it on the table (we are benefiting from the HoloLens spatial mapping abilities, where the attached virtual object can collide with the real environment). For this purpose, we chose the *click-attach-click* approach (click on the object, attach the virtual one, click on the table to release it) rather than the *drag&drop*, because the virtual object could easily lost from the user's sight or the hand tracking of the HoloLens could easily lost (because of the limited FoV for hand recognition).

After placing the virtual object onto the table inside the reach zone of the robot (visualized by the SAR projection), virtual spheres, for setting the rotation, are displayed. By dragging them, the rotation is set.

The procedure of setting the *pick&place* program using the ISAR-HMD can be summarized into following steps:

1) Click on the *Edit button* of the *pick from feeder* instruction in the projected interface.
2) Look at desired object placed in the feeder, position the virtual robot gripper to desired detection pose and click on it (Air tap).
3) Position attached virtual object on the table and click when satisfied with the position.
4) Adjust the rotation by dragging displayed spheres around the virtual object.
5) Click on the tick button in the HoloLens or on the *Save button* of the projected interface to confirm and save the place position.

Whole procedure of setting the *pick&place* program using the ISAR-HMD is shown in the Figure 6.

Programming procedure, when using the ISAR-KT, is similar in the projected interface related steps – 1 and 5. In the step 2, user has to physically move the robot's gripper to desired detection position. As the arm is in interaction mode, its forearm cameras are on, seeking for any visible objects. If any are visible, the robot recognizes the type of the closest one. When the user is satisfied with the set object type and detection position, he saves it using the *Save button* of the projected interface. Thus *pick from feeder* instruction is set. Learning of the following *place to pose* instruction needs to be called manually. Robot's reach zone as well as interactive bounding box representing the place position are displayed. User drags the bounding box outline and blue point situated in its corner in order to set the place position and its orientation (steps 3-4). The procedure is shown in the Figure 5.

It has to be mentioned, that ISAR projections are synchronized with HMD's virtual objects and vice versa. Meaning that the user can whenever decide, if he wants to set the place positions using the touches on the table or gestures in HoloLens. It is also possible to put aside the HMD at anytime and continue the programming using the ISAR-KT approach.

D. Main Benefits

In a scenario, where company has multiple workspaces but limited number of collaborative robots that are moved between those workspaces, it would be time consuming to edit current programs at individual workspaces, because the need of robot's presence if kinesthetic teaching is applied. However, using our solution, the robot is not needed. Workers can effectively set programs in advance anytime, without the need of stopping the production of a current batch.

Thanks to the combination of the HMD with the ISAR, others are partially able to see directly in the scene in

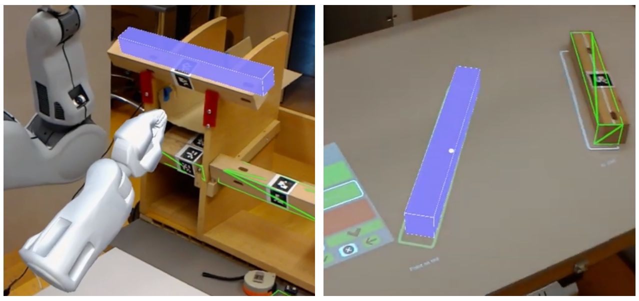


Fig. 3: Visualization of instructions with already set parameters. **Left:** Virtual gripper is rendered on a set detection position along with the virtual object of specified type (*pick from feeder* instruction). **Right:** Virtual object is rendered on a set place position (*place to pose* instruction).

realtime, what the user with the HMD put on is currently doing, which is not possible without any additional device (hand-held device, another pair of HMD, or HMD's stream). Moreover, the ISAR extends the HMD's limited FoV by 2D projections. As far as we know, no one ever combined those two augmented reality approaches.

If the program is set, the user can see a virtual gripper rendered directly on the detection position along with the virtual object, that is going to be picked, in case of *pick from feeder* instruction, or rendered virtual object on the place position in case of *place to pose* instruction (see Figure 3). This is beneficial if the user just wants to preview the program without running it. It could also positively impact the users' orientation within set programs.

We used a text-to-speech utility in order to play system related notifications, warnings and errors to users in their native language through the HoloLens embedded speakers. This could be beneficial for new users of our system.

IV. Experiment Design

In order to evaluate the proposed ISAR-HMD solution, an experiment was designed, where the solution was compared to the baseline ISAR-KT approach. Both approaches were tested on a set of 20 participants using the within-subject design methodology. Order of conditions was randomized to mitigate possible bias caused by a learning effect.

As measures, we chose a combination of three standardized questionnaires – the System Usability Scale (SUS) [17], NASA Task Load Index (TLX) [18] and the User Experience Questionnaire (UEQ) [19]. We also measured task completion times.

A. Hypotheses

As the main motivation for this work is to introduce a novel approach of teaching robots that could replace the kinesthetic teaching, we assume that our ISAR-HMD solution will be quicker, less demanding and more preferred by users than the ISAR-KT. Therefore, we set following three hypotheses:

(i) The ISAR-HMD approach is faster than the ISAR-KT approach.

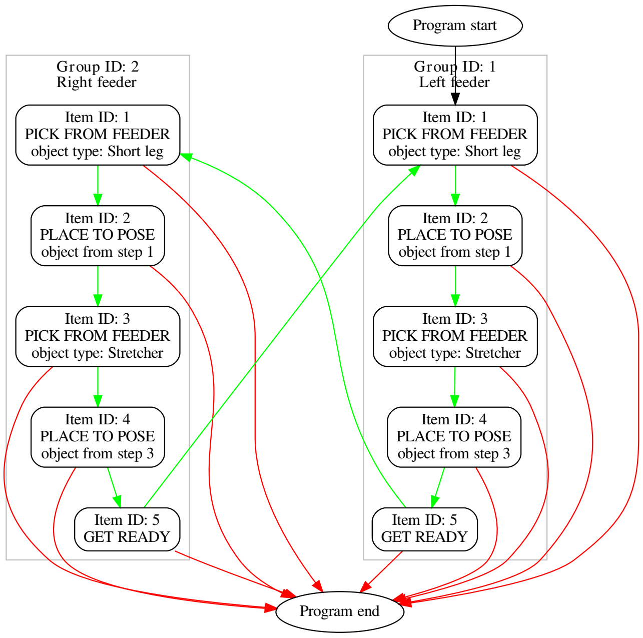


Fig. 4: Pick&Place program. The green edges represents transitions, that are triggered if current instruction was successfully executed, while the red ones represents transition of instruction's unsuccessful execution. The program is designed to run in loop, until user decides to stop it.

(ii) User task load of the ISAR-HMD approach is lower than the ISAR-KT approach.

(iii) In terms of UX, users will prefer ISAR-HMD over ISAR-KT.

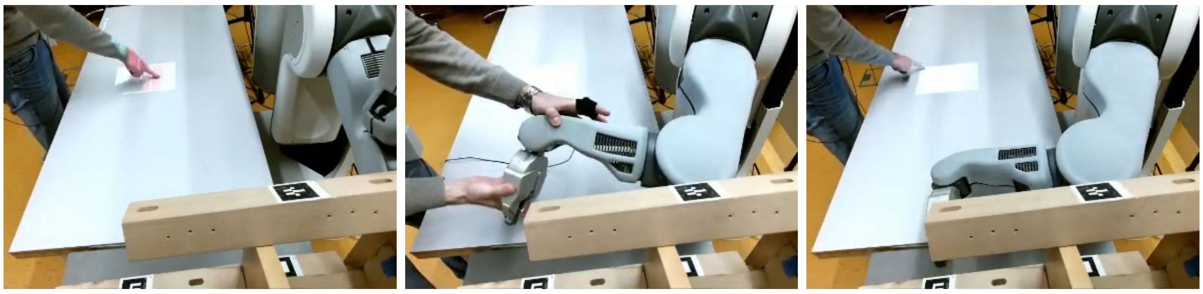
B. Tasks

Experiment workflow of tested methods consisted of following phases: *introduction*, *training*, *first task*, *second task* and *questionnaire* ended up with *discussion*. After the user finished the workflow using one of the methods, he/she repeated it using the other one.

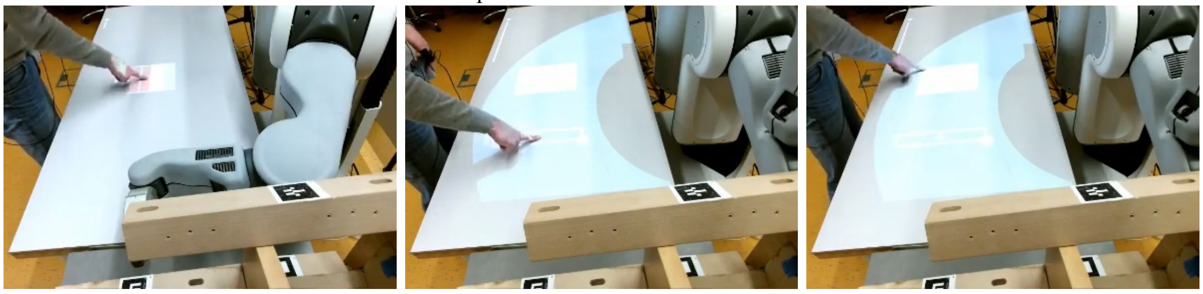
Within **the introduction phase**, participants got an overall idea of the collaborative robotics purpose, its related problems we are solving and a brief description of the upcoming experiment.

The training phase involved demonstrative and commented setting of one pair of the *pick from feeder* and the *place to pose* instruction, using currently tested method. In case of the ISAR-HMD method, the training phase involved getting familiar with the HMD (HoloLens). Participants went through the Microsoft's *Calibration* application – to calibrate their interpupillary distance, which can improve the quality of visuals – and the Microsoft's *Learn Gestures* application – to ensure they properly learn how to use the HoloLens gestures.

The first task consisted of setting parameters for an unset *pick&place* program. This program was composed of two blocks, the first one for picking from feeder on user's left side and the second one for picking from feeder on user's

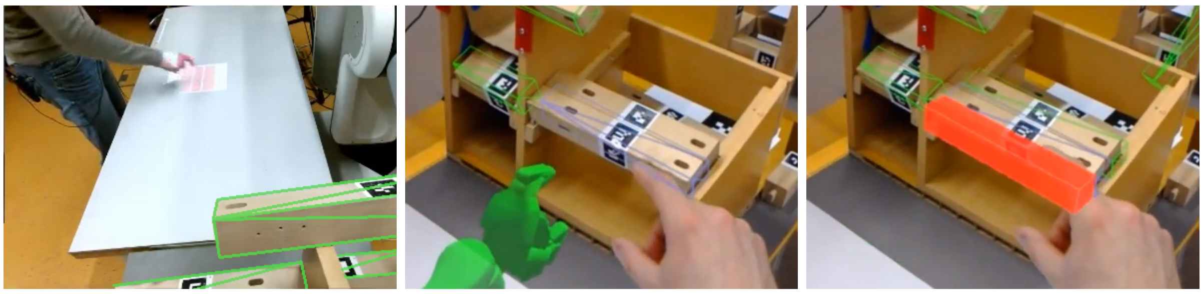


(a) User selects *pick from feeder* instruction to be set. (b) User moves the robot gripper to detection position. (c) Gripper's detection position is saved.

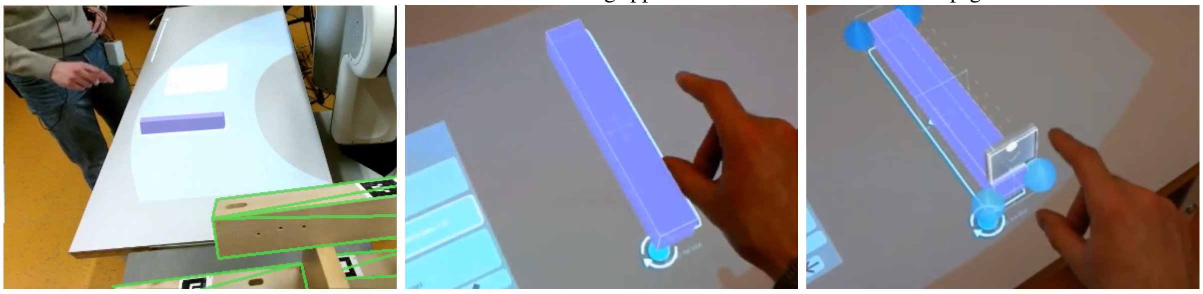


(d) User selects *place to pose* instruction to be set. (e) User adjusts place position by dragging it on the table. (f) The place position is saved.

Fig. 5: An example of setting the *pick&place* program using the ISAR-KT approach during the experiment.



(a) User selects *pick from feeder* instruction to be set. (b) User's first person view. While gazing, user sees the virtual gripper. (c) Virtual object snaps to user's gaze after the Air tap gesture.



(d) Place position is adjusted by user's head movements. (e) When user clicks (Air tap), virtual object snaps to the table and the rotation spheres are displayed. (f) User saves the place position by clicking on the tick button.

Fig. 6: An example of setting the *pick&place* program using the ISAR-HMD approach during the experiment.

right side. Both blocks contained four parametric instructions – two *pick from feeder* and two *place to pose*, and one non-parametric instruction – *get ready* (moves the robot's arms to their default position). The *pick from feeder* instruction takes two parameters – object type and robot's gripper position for detecting the objects in feeder followed up with picking the closest one. The *place to pose* instruction takes just object's place position as the parameter (object type is referenced from previous pick instruction). Structure of the program is shown in Figure 4. An example of completing part of this task using both tested methods is shown in Figure 5 (ISAR-KT approach) and Figure 6 (ISAR-HMD approach).

The second task consisted of editing preset *pick&place* program. Within made up backstory, we told participants that someone mistakenly set the program with wrong parameters (wrong object types, wrong place positions, overlapping place positions, etc.). Their task was to detect those instructions with wrong parameters and correct them to fulfill the assignment. The program had same structure as the program in the first task.

In order to be able to record participant's point of view and head tracking for both conditions (which was necessary for evaluation purposes), participants wore the HMD even in condition where it was not actually used by them. Moreover, this could prevent distortion of the results caused by potential discomfort from wearing the HMD, which would not be the case for forthcoming devices as e.g. HoloLens 2 (lighter, better balanced).

C. Participants

Prior the main study, the experiment design was tested out in pilot test with 2 participants. After that, 20 users participated in the main experiment. Most of the participants were IT students or faculty employees (18 male and 2 female, ages 20-31, $M = 25.00$). 13 participants never used VR/AR HMD. There were total of 12 participants reporting eye issues. 5 of them reported farsightedness, 1 reported nearsightedness and 5 reported wearing glasses or contact lenses without specifying exact eye issue. 1 reported color blindness. On a Likert scale from 1 to 5, most of participants expressed positive attitude towards new technologies ($M = 3.55, CI =< 3.19, 3.91 >$) and rather high IT skills ($M = 4.00, CI =< 3.41, 4.59 >$). On the other hand, experience with robots ($M = 2.00, CI =< 1.50, 2.50 >$) and experience with AR ($M = 2.20, CI =< 1.68, 2.72 >$) were self-assessed rather low, which could be expected to be close to the situation in the target user group (employees in SMEs).

V. Results

This section summarizes the experiment results and provides its analysis and interpretation. Regarding the task completion time measurement, intervals where participants were asking questions, technical problem occurred or when moderator had to intervene, were subtracted, in order to measure a pure task completion time. All statistical tests were done at the 5 % significance level. Data were first tested for normality (combination of D'Agostino and Pearson's tests) and based on the result, paired t-test (**T**) or Wilcoxon's signed-rank test (**W**) were used to test for the significant difference between conditions.

A. Quantitative and Qualitative Data

Both tasks were completed quicker when using the proposed solution (ISAR-HMD). Completing them both using proposed solution saved up to 153.94 seconds in average, which confirms the hypothesis (i).

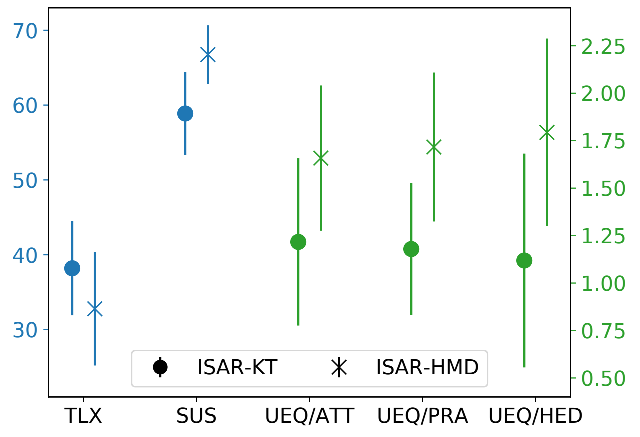


Fig. 7: Obtained qualitative measures (mean values and 95 % confidence intervals) for both evaluated conditions. The left y axis belongs to TLX and SUS, while the right one to the UEQ grouped scales.

Table I shows the results of measured metrics for both tested methods. There is a statistically significant difference for all metrics, except the TLX and majority of its subscales. The mean TLX score of the proposed ISAR-HMD solution was 32.78 which is less than 38.19 for the ISAR-KT; however, the difference is not statistically significant. Thus the hypothesis (ii) cannot be confirmed. For TLX subscales, there is a significant difference for *Temporal Demand*, corresponding to the objective measurement of task times, where ISAR-HMD was significantly faster then ISAR-KT for both tasks. Similarly to [12], the kinesthetic teaching required lower mental load and higher physical demand. However, the differences were not significant. We hypothesize, that higher mental demands for the ISAR-HMD are mainly caused by a hardware limitations of the used device, namely limited FoV for visualization (leading to increased demands on the user's spatial cognitive abilities) and for capturing gestures. Also the interface should cope with the given limitations better – e.g. it could indicate direction to the interactive elements which are currently out of the user's FoV.

The hypothesis (iii) is well supported by obtained UX-related ratings. The mean SUS rating of the ISAR-HMD approach was 66.75 which means improvement over the ISAR-KT (58.88). Figure 7 shows measured metrics – its mean values and confidence intervals – in a graph. UEQ consists of six categories, where some can be grouped together – **attractiveness** (ATT); **pragmatic quality** (PRA) that encapsulates perspicuity, efficiency and dependability; and **hedonic quality** (HED) that encapsulates stimulation and originality. According to the general benchmark [19], ratings of all three main categories of the ISAR-KT approach are ranked as *Above Average*. Ratings of the ATT and PRA of the ISAR-HMD approach are ranked as *Good*, which is one rank higher than the ISAR-KT and the HED score is ranked as *Excellent*, moving it into the top rank.

Further, we divided both quantitative and qualitative data

Measure	**ISAR-KT**	**ISAR-HMD**	**T/W Value**	**p**
SUS	$58.88; < 53.32, 64.43 >$	$66.75; < 62.84, 70.66 >$	$T(20) = 3.55$	$\mathbf{0.002}$
NASA TLX	$38.19; < 31.91, 44.48 >$	$32.78; < 25.20, 40.35 >$	$T(20) = 1.31$	0.206
NASA TLX / Mental Demand	$17.5; < 8.56, 26.44 >$	$25.83; < 13.06, 38.61 >$	$T(20) = 1.39$	0.180
NASA TLX / Physical Demand	$23.33; < 12.19, 34.48 >$	$16.67; < 7.20, 26.14 >$	$T(20) = 1.22$	0.237
NASA TLX / Temporal Demand	$41.67; < 28.64, 54.69 >$	$24.17; < 12.44, 35.89 >$	$T(20) = 2.27$	$\mathbf{0.035}$
NASA TLX / Overall Performance	$90.00; < 80.74, 99.26 >$	$86.67; < 77.68, 95.65 >$	$W(20) = 17.50$	0.546
NASA TLX / Effort	$30.83; < 19.16, 42.51 >$	$24.17; < 14.54, 33.80 >$	$T(20) = 0.94$	0.359
NASA TLX / Frustration Level	$25.83; < 16.54, 35.12 >$	$19.17; < 8.06, 30.28 >$	$T(20) = 1.51$	0.148
UEQ/ATT	$1.22; < 0.78, 1.66 >$	$1.66; < 0.78, 1.66 >$	$T(20) = 2.26$	$\mathbf{0.036}$
UEQ/PRA	$1.18; < 0.83, 1.53 >$	$1.72; < 1.32, 2.11 >$	$T(20) = 2.90$	$\mathbf{0.009}$
UEQ/HED	$1.12; < 0.56, 1.68 >$	$1.79; < 1.30, 2.29 >$	$W(20) = 16.50$	$\mathbf{0.001}$
1st task completion time (s)	$282.58; < 248.88, 316.28 >$	$196.18; < 152.84, 239.53 >$	$W(20) = 22.00$	$\mathbf{0.002}$
2nd task completion time (s)	$256.33; < 205.93, 306.73 >$	$188.78; < 145.85, 231.72 >$	$T(20) = 2.60$	$\mathbf{0.017}$

TABLE I: Qualitative measures (System Usability Scale, NASA Task Load Index and its subscales, User Experience Questionnaire which consists of three categories – Attractiveness, Pragmatic Quality and Hedonic Quality) and quantitative measures (task completion times). The data for both methods are in format "mean; and respective 95 % confidence interval". For a statistical comparison, we used paired t-test (**T**) and Wilcoxon (**W**) method.

into two parts according to following binary conditions: previous experience with HMD, presence of an eye-related health problem and order of evaluated method (whether user tested ISAR-HMD first). Differences in measures between aforementioned parts for both evaluated methods were tested using a t-test for independent samples or Kolmogorov-Smirnov's test based on normality test result. No statistical significant differences were found. Our interpretation is that HMD is suitable even for novice users without previous experience with HMD. Further, task completion times nor subjective assessment of the method are influenced by an existence of vision-related health problem or limitation, meaning that the used HMD device (HoloLens 1) does not posses problems for users wearing glasses, etc. Finally, in contrary to [16] where users rated 2D interface significantly lower after they interacted with the system using HMD, in our case no order effect was identified. This could mean that both methods (interfaces) are acceptable, likeable and roughly equally hard to learn and use.

B. General Findings

Biggest downside of our setup was probably unreliable touch-enabled table. False touches, double-clicks or undetected touches were source of frustration for most participants and probably caused the overall low ratings of the qualitative data.

Three participants struggled with positioning the robot's gripper. They were not able to rotate the arm links properly in order to find correct kinematic configuration.

On the other hand, 4 participants had troubles with learning and adopting the HoloLens Air tap gesture. Main source of such problems was caused by not having hands in HoloLens cameras detection zone. Three complained about the HMD's text-to-speech, claiming that they already know what to do and what is happening after the training phase and few set instructions from the first task. One participant reported headache after completing both tasks using the HMD.

One participant suggested that it would be better, if he could oversee all place poses at once. This suggestion needs to be further tested, because an overwhelming number of virtual objects displayed at once could cause user's confusion and inability to orientate within the program.

Interestingly, some participants preferred to use the touch table to adjust object's place position, even though they were supposed to use primarily the HMD's gestures. We also noticed few situations where participants were not able to distinguish SAR projections from HMD virtual objects. They tried to interact with those projections using the HoloLens gestures and not the touch-enabled table. This could be a good indicator that ISAR can be visually believable merged with the HMD's AR without direct distinction.

VI. CONCLUSIONS

In this paper, we presented a novel approach to the end-user robot programming using the unique combination of the interactive spatial augmented reality and the head-mounted display (ISAR-HMD). The purpose of this approach is to reduce the users' workload and to let them program the collaborative workspace faster than in kinesthetic approaches and without the need of the robot's presence.

We evaluated the proposed ISAR-HMD approach on a set of 20 participants using the within-subject experiment design, where we compared it to the baseline approach, that uses the interactive spatial augmented reality and the kinesthetic teaching only (ISAR-KT). We reached up to 33.84 % improvement in qualitative measures (SUS, NASA-TLX, UEQ) and saved up to 28.46 % of completion time of setting the *pick&place* program.

In the future work, we will focus on lowering the task load for HMD, which could be achieved by a constrained FoV-aware visualization. Another direction of the research will be further integration of ISAR and HMD. Additionally, we are going to develop a detector of UX-related events, based on combination of physiological data with data from external sensors (e.g. user pose tracking) and input data (e.g. clicks), which could be helpful for the system to automatically provide timely assistance to the user as an excessive amount of voice notifications was one of the most common complains from users in the current experiment.

ACKNOWLEDGMENT

The work was supported by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science – LQ1602".

REFERENCES

[1] Z. Materna, M. Kapinus, V. Beran, P. Smrž, and P. Zemčík, "Interactive spatial augmented reality in collaborative robot programming: User experience evaluation," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Aug 2018, pp. 80–87.

[2] G. Michalos, P. Karagiannis, S. Makris, Ö. Tokçalar, and G. Chryssolouris, "Augmented reality (ar) applications for supporting human-robot interactive cooperation," *Procedia CIRP*, vol. 41, pp. 370–375, 2016.

[3] S. Stadler, K. Kain, M. Giuliani, N. Mirnig, G. Stollnberger, and M. Tscheligi, "Augmented reality for industrial robot programmers: Workload analysis for task-based, augmented reality-supported robot control," in *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*. IEEE, 2016, pp. 179–184.

[4] S. Magnenat, M. Ben-Ari, S. Klinger, and R. W. Sumner, "Enhancing robot programming with visual feedback and augmented reality," in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2015, pp. 153–158.

[5] H. Hedayati, M. Walker, and D. Szafir, "Improving collocated robot teleoperation with augmented reality," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2018, pp. 78–86.

[6] M. Walker, H. Hedayati, J. Lee, and D. Szafir, "Communicating robot motion intent with augmented reality," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2018, pp. 316–324.

[7] E. Bunz, R. T. Chadalavada, H. Andreasson, R. Krug, M. Schindler, and A. Lilienthal, "Spatial augmented reality and eye tracking for evaluating human robot interaction," in *RO-MAN 2016 Workshop: Workshop on Communicating Intentions in Human-Robot Interaction, New York, USA, Aug 31, 2016*, 2016.

[8] M. D. Coovert, T. Lee, I. Shindev, and Y. Sun, "Spatial augmented reality as a method for a mobile robot to communicate intended movement," *Computers in Human Behavior*, vol. 34, pp. 241–248, 2014.

[9] M. Funk, A. Bächler, L. Bächler, T. Kosch, T. Heidenreich, and A. Schmidt, "Working with augmented reality?: a long-term analysis of in-situ instructions at the assembly workplace," in *Proceedings of the 10th International Conference on PErvasive Technologies Related to Assistive Environments*. ACM, 2017, pp. 222–229.

[10] E. Kruijff, J. E. Swan, and S. Feiner, "Perceptual issues in augmented reality revisited," in *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*. IEEE, 2010, pp. 3–12.

[11] R. S. Andersen, S. Bøgh, T. B. Moeslund, and O. Madsen, "Task space hri for cooperative mobile robots in fit-out operations inside ship superstructures," in *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*. IEEE, 2016, pp. 880–887.

[12] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. F. Machiel Van der Loos, and E. Croft, "Robot programming through augmented trajectories in augmented reality," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1838–1844.

[13] E. Rosen, D. Whitney, E. Phillips, G. Chien, J. Tompkin, G. Konidaris, and S. Tellex, "Communicating robot arm motion intent through mixed reality head-mounted displays," *CoRR*, vol. abs/1708.03655, 2017. [Online]. Available: http://arxiv.org/abs/1708.03655

[14] D. Krupke, F. Steinicke, P. Lubos, Y. Jonetzko, M. Görner, and J. Zhang, "Comparison of multimodal heading and pointing gestures for co-located mixed reality human-robot interaction," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1–9.

[15] S. Blankemeyer, R. Wiemann, L. Posniak, C. Pregizer, and A. Raatz, "Intuitive robot programming using augmented reality," *Procedia CIRP*, vol. 76, pp. 155–160, 01 2018.

[16] S. Yitzhak Gadre, E. Rosen, G. Chien, E. Phillips, S. Tellex, and G. Konidaris, "End-user robot programming using mixed reality," 10 2018.

[17] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[18] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," *Advances in psychology*, vol. 52, pp. 139–183, 1988.

[19] M. Schrepp, "User experience questionnaire handbook," 09 2015.

End-User Robot Programming Case Study: Augmented Reality vs. Teach Pendant

Michal Kapinus, Zdeněk Materna, Daniel Bambušek, Vítězslav Beran
Faculty of Information Technology, Brno University of Technology, Czech Republic

ABSTRACT

The work presents a preliminary experiment aimed for comparing a traditional method of programming an industrial collaborative robot using a teach pendant, with a novel method based on augmented reality and interaction on a high-level of abstraction. In the experiment, three participants programmed a visual inspection task. Subjective and objective metrics are reported as well as selected usability-related issues of both interfaces. The main purpose of the experiment was to get initial insight into the problematic of comparing highly different user interfaces and to provide a basis for a more rigorous comparison, that is going to be taken out.

ACM Reference Format:
Michal Kapinus, Zdeněk Materna, Daniel Bambušek, Vítězslav Beran. 2020. End-User Robot Programming Case Study: Augmented Reality vs. Teach Pendant . In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction (HRI '20 Companion), March 23–26, 2020, Cambridge, United Kingdom.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3371382.3378266

1 INTRODUCTION

With novel and unusual interfaces and interaction methods, a significant problem emerges: how to compare it with existing and well-known methods or interfaces. While comparing partial interfaces' features might be easy and intuitive, comparing two complex and highly different systems is challenging in terms of experiment design and evaluation of the results. Although experiments with novel interfaces could provide good insight into whether the interface is usable by measuring subjective data, a fair comparison with existing method is crucial for measuring improvements in e.g. efficiency, in order to provide justification that the new method offers added value over the existing one and could be successfully deployed in the real-world industrial settings.

Several experiments were conducted to evaluate usability of our augmented reality (AR) interface ARCOR for end-user robot programming [1, 5, 6]. This interface allows user to program the robot using highly abstracted instructions such as *PickFromTable*, *DrillHole*, etc., using a user-friendly graphical interface projected on a touch-enabled table. Although the interface was evaluated several times, no comparison with any existing method took place yet, as our system did not support any standard industrial robotic arm. Recently, support for Aubo i5 robotic arm was added. This paper presents preliminary experiment designed as a case study aimed to get the insight into comparing such different interfaces.

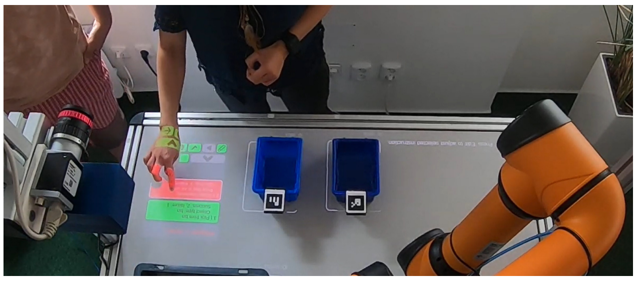


Figure 1: Participant programs a visual inspection task using the ARCOR spatial augmented reality interface.

2 BACKGROUND

The traditional method of programming industrial robots is through the teach pendant. There exist various pendant interfaces. Some of them, as ABB FlexPendant with its text-based programming, are targeted to expert users, while others are more suited for less skilled users as Universal Robots Polyscope with its tree-based program visualization and wizards. Emerging alternative methods aimed on simplification of the robot programming for non-experts were so far often not evaluated with a (user-friendly) pendant as a baseline method. There exist only few examples of evaluations, where such comparison have been carried out. However, the published experiments have various limitations. For instance, [7, 10] were carried out with only one pendant-expert user and [11] was carried out in a simulation. The experiment in [3] seems well designed, with sufficient number of participants, however only with a simple pick and place task. Existing experiments are usually designed ad hoc, as there is a lack of proven methodology. For instance, method to compare HRI approaches is proposed in [8], however extension beyond trajectory teaching task would be needed.

3 EXPERIMENT DESIGN

A preliminary 2-condition within-groups case study was conducted. The main goal of presented case study was to verify, that the proposed method of simplified robot programming is suitable for a visual inspection task and performs better than the teach pendant (which interface is similar to UR Polyscope). The robot is instructed to pick the bottle opener from the table, put it in front of the camera, trigger the inspection method and based on inspection result, put the bottle opener to one of the boxes on the table. In order to make the comparison more fair, a few high level functions as *pick*, *place* or *suction (on/off)* were prepared in advance in pendant. The experiment was conducted with 3 participants (2 males and 1 female), in a lab-like environment. All of the participants had little

Participant	A_p	A_a	B_p	B_a	C_p	C_a
Introduction [s]	359	179	449	311	185	174
Task [s]	562	189	749	309	510	146
TLX [0, 100]	72.22	36.11	44.44	27.78	33.33	19.44
SUS [0, 100]	52.50	82.50	42.50	80.00	70.00	90.00
UEQ_{ATT} [−3, 3]	−1.17	2.00	−0.17	1.83	1.83	2.50
UEQ_{PRA} [−3, 3]	0.25	2.08	−0.50	1.83	1.58	2.25
UEQ_{HED} [−3, 3]	−0.25	2.12	−1.25	1.62	0.25	2.00

Table 1: Durations of introduction and programming for both (p)endant and (a)rcor modality. Subjective metrics for each participant and both modalities. Higher means better for all subjective metrics except TLX.

or no prior experience with AR, participants A and B had little or no prior experience with teach pendant while participant C had moderate prior experience with pendant.

The experiment involved two sessions (first with pendant, second with ARCOR) consisting of training and programming the actual task. Each of the sessions was followed by filling in the standard questionnaires [2, 4, 9] and discussion. Participants were recorded using standard camera for future analysis. Moreover, several physiological data were recorded using the Empatica E4 wristband.

4 RESULTS

All participants were able to complete the task using both methods (teach pendant, ARCOR). For each participant, the time needed for both introduction and programming itself was lower for ARCOR interface (see Table 1). The ARCOR also performed better in terms of usability, UX and task load metrics. Detailed cases for each participant follows.

4.1 Participant A (25, male, programmer)

While using the pendant, the moderator had to intervene approximately 8 times, to help the participant to overcome the issues with the pendant interface, mainly finding the right buttons for desired task. Participant was a bit frustrated when he wanted to copy block of instructions, which was not possible.

With ARCOR, only one intervention of the moderator was necessary, when the participant overlooked the dialog for saving the robot position. Sometimes, the participant was unsure, what is the next required step, but he was always able to resolve this uncertainty using the notification area of the interface. The participant complained about the positioning of some GUI elements, which were sometimes hidden by real objects.

Overall, the participant considers the teach pendant too complicated, slow and cumbersome. He prefers the ARCOR interface more, because a lot of things are already prepared in advance and it allows him to focus on the programming itself.

4.2 Participant B (41, male, application tester)

With the pendant, the participant struggled with the complex GUI: there were difficulties in finding buttons, instructions and instruction lists. This was the main cause of frequent moderator’s interventions. Moreover, the participant asked the moderator several times, whether is he proceeding correctly in setting individual instructions and waypoints.

When the participant was using the ARCOR interface, there were significantly less moderator’s interventions, related only to the touch surface problems (e.g. non-registered touches). The participant was able to successfully use the notification area of the interface when felt lost or didn’t knew how to proceed further.

Although the participant preferred, based on the results, the ARCOR interface better, there were some complaints about setting the box location area, where the interface could be more automated and, for example, not allowing the user to move the UI elements off the touch-enabled surface.

The participant considered the pendant approach difficult, but admitted that it could be learned if there is no other option.

4.3 Participant C (23, female, programmer)

The prior experience with pendant of this participant is reflected by the lowest time needed for introduction to this modality and could explain better score in all measured metrics in compare to other participants. However, she still ranked the ARCOR modality better in all metrics. Despite the prior experience, the participant was insecure at the beginning and was using quite a big amount of help from the moderator. After few minutes however, she became more certain about various elements of the interface.

For this participant, setting the position of the robot was physically challenging, which could be one of the reasons why ARCOR interface was ranked better, as it requires less direct manipulation with the robot.

The participant had no fundamental problem with ARCOR interface, she only suffered from some design issues like ambiguous buttons, visualization of inactive buttons or slow response from the system, where she was uncertain whether e.g. some button was successfully pressed.

She felt good using both interfaces, but she considered the ARCOR interface simpler and faster.

5 CONCLUSIONS

The conducted preliminary experiment was focused on comparing two highly different methods of robot programming: spatial augmented reality and user-friendly teach pendant. It was necessary to deal with different complexity, level of abstraction (high for AR, low for pendant) and specifics of each method. The results indicate the potential of the ARCOR system, which was preferred by the participants over the pendant and also required less time to train as well as to program the visual inspection task. The upcoming experiment will involve more participants in order to enable statistical analysis of the results, contain various tasks in order to provide more generalizable results and will be performed out of the lab. Moreover, more high-level functions for the pendant will be prepared in advance, in order to improve the fairness of the comparison. Also, the pendant modality will require a more complex training procedure. Gained experience will allow us to formulate an exact methodology for this kind of experiments.

ACKNOWLEDGMENTS

The work was supported by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project “IT4Innovations excellence in science – LQ1602”.

REFERENCES

[1] Daniel Bambušek, Zdeněk Materna, Michal Kapinus, Vítězslav Beran, and Pavel Smrž. 2019. Combining Interactive Spatial Augmented Reality with Head-Mounted Display for End-User Collaborative Robot Programming. In *2019 28th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*.

[2] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[3] Yuxiang Gao and Chien-Ming Huang. 2019. PATI: a projection-based augmented table-top interface for robot programming. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. ACM, 345–355.

[4] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Advances in psychology* 52 (1988), 139–183.

[5] Zdeněk Materna, Michal Kapinus, Vítězslav Beran, Pavel Smrž, Manuel Giuliani, Nicole Mirnig, Susanne Stadler, Gerald Stollnberger, and Manfred Tscheligi. 2017. Using Persona, Scenario, and Use Case to Develop a Human-Robot Augmented Reality Collaborative Workspace. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction (HRI '17)*. Association for Computing Machinery, New York, NY, USA, 201–202. https://doi.org/10.1145/3029798.3038366

[6] Zdeněk Materna, Michal Kapinus, Vítězslav Beran, Pavel Smrž, and Pavel Zemčík. 2018. Interactive Spatial Augmented Reality in Collaborative Robot Programming: User Experience Evaluation. In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 80–87.

[7] Alexander Perzylo, Nikhil Somani, Stefan Profanter, Ingmar Kessler, Markus Rickert, and Alois Knoll. 2016. Intuitive instruction of industrial robots: Semantic process descriptions for small lot production. In *2016 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE, 2293–2300.

[8] Guilherme Boulhosa Rodamilans, Emília Villani, Luís Gonzaga Trabasso, Wesley Rodrigues de Oliveira, and Ricardo Suterio. 2016. A comparison of industrial robots interface: force guidance system and teach pendant operation. *Industrial Robot: An International Journal* 43, 5 (2016), 552–562.

[9] Martin Schrepp. 2015. User Experience Questionnaire Handbook. (09 2015). https://doi.org/10.13140/RG.2.1.2815.0245

[10] Maj Stenmark, Mathias Haage, and Elin Anna Topp. 2017. Simplified programming of re-usable skills on a safe industrial robot: Prototype and evaluation. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 463–472.

[11] David Weintrop, Afsoon Afzal, Jean Salac, Patrick Francis, Boyang Li, David C Shepherd, and Diana Franklin. 2018. Evaluating coblox: A comparative study of robotics programming environments for adult novices. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 366.

Spatially Situated End-User Robot Programming in Augmented Reality

Michal Kapinus, Vítězslav Beran, Zdeněk Materna and Daniel Bambušek

***Abstract*— Nowadays, industrial robots are being programmed using proprietary tools developed by robot manufacturer. A skilled robot programmer is needed to create even as simple task as pick a well-known object and put it somewhere else. Contrary, in every-day life people are using end-user programming to make different electronic devices work in expected manner, without even noticing they are actually programming. We propose augmented reality-enabled end-user programming system allowing regular shop-floor workers to program industrial robotic tasks. The user interface prototype for this system was evaluated in the user study with 7 participants with respect to usability, mental workload and user experience.**

I. INTRODUCTION

For decades, robots have been deployed in automated manufacturing. Their use is mainly in large-scale production, because the design and construction of such automated production operation is very time-consuming and expensive. This solution pays off for a type of production that runs for at least several years. Although it is known that robotic systems are unsuitable for some operations, when it is sometimes almost impossible to replace humans, neither the method of production nor the technological advancements have made it possible to exploit the potential of humans in the production process together with robots.

Today, however, the rapid development of technology brings the ability to produce robots who are already able to work in the vicinity of humans, can to some extent perceive the world around and to some extent cooperate with human. There are new possibilities of making automated production more efficient by integrating human into the automated process. This is especially important for smaller productions. Nowadays, many manufacturing activities can be replaced by a cheaper robot or robotized production line, which is not able to handle all the tasks of the production process itself, but thanks to the possibility to cooperate with human, these tasks can be effectively solved. Smaller production in smaller manufacturing companies, however, brings a new problem: how to program these robots effectively for a new production process?

Many robot manufacturers now offer a variety of robot programming solutions, from desktop programming tools, where drag&drop and intuitive icons can be used to quickly program a manufacturing process (such as ABB RobotStudio or RoboDK), to approaches where a user directly defines the process by manipulating a robot (such as Baxter or

All authors are affiliated with the Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Bozetechova 1/2, Brno, 612 66, Czech Republic. Contacts: $ikapinus, beranv, imaterna, bambusekd@fit.vutbr.cz$

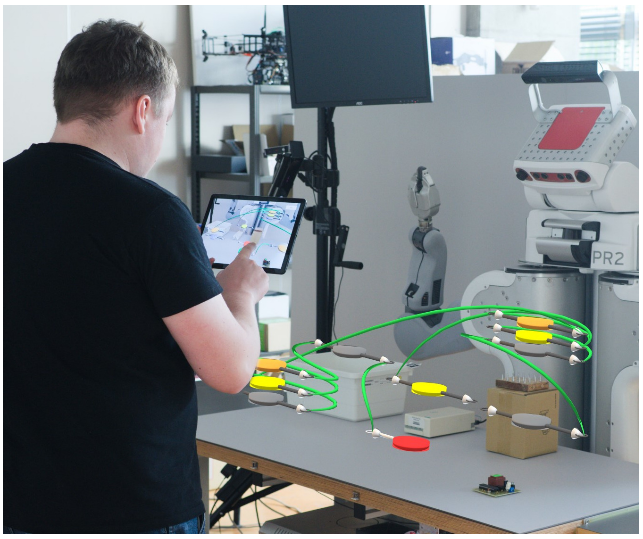


Fig. 1. ARCORO - Augment Reality for Connections and Operations of Real Objects.

YuMi). Today, many aspects affecting user workload or user experience when using UI devices to interact with the robots are known: the user often has to switch the context between workstation and computer, poor robot feedback worsens human-robot communication when visual or aural feedback is limited (light or sound signal, on-screen information, etc.). Although today's advanced technologies allow more natural human-machine interaction, most of the solutions are still based on a 50-year-old GUI concept. It implements a GUI using WIMP and 2D display on a desktop computer using real world metaphors.

The goal of our research is to return from a 2D desktop user interface back to a real 3D environment. In this work we present the application of this principle in the design of a new tool for robot programming. We define the requirements for a new type of user interface: integrate the programming tool into a real 3D environment, use scene and object knowledge to reduce user mental load, visualization of a program stage and robot's knowledge of the environment to improve user's feedback. We decided to address these challenges for the natural interaction of human with machine in real 3D environment. We design and introduce an innovative way of programming a spatial robotic task with high levels of abstraction using Augmented Reality (AR) technology (ARCORO[1]).

Based on our prior experience with simplified robot pro-

[1]Augment Reality for Connections and Operations of Real Objects

gramming in AR [1], an experimental ARCORO system, using a new concept of robot programming in 3D space using AR on a mobile device, was developed. Part of the work is also the definition of use-case, which is designed according to real demand from industry. This use-case was used to test the new ARCORO interface. The experiment was conducted with 7 participants and evaluated with respect to usability, mental workload and user experience.

II. RELATED WORK

An increasing number of collaborative robots in SMEs (small and medium enterprises) requires searching for new methods for end-user robot programming. Various techniques incorporating the AR were proposed, mostly based on visual programming [2], [3], programming by demonstration [4], [5] or combination of both [6], [1]. These methods may differ in both input and output modalities and utilizes the AR for both programming and giving visual feedback to the user.

Gadre et al. [3] proposed Mixed Reality (MR) system for robot programming using Head-Mounted Display (HMD). They compared this system against a 2D keyboard and mouse system for programming pick & place task. Gadre et al. [3] found that users were significantly faster and better able to successfully program the robot using the MR interface than the 2D interface.

Quintero et al. [7] designed AR system using Microsoft HoloLens HMD capable of 3D robot trajectory specification, virtual previews of robot motion and visualization of robot parameters. Blankemeyer et al. [8] presents another HMD-based system using Microsoft HoloLens for simple pick & place task programming.

Stadler et al. [9] discussed possibility of lowering mental demand of the robot programmer, by using tablet-based AR approach in simplified industrial tasks.

Magnenat et al. [10] have shown, that overall performance of the operator could be increased by incorporating AR and visual feedback into tablet-based system for robot programming system.

Recently, several solutions based on tabletop projections emerged. Materna et al. [1] have developed Spatial Augmented Reality (SAR) system using table with touch-enabled surface and projector above the table, projecting both User Interface to program collaborative robot and showing contextual information of objects on the table and the state of the system. Gao et al. [11] provided another tabletop SAR solution for industrial end-user robot programming of manipulation tasks, using common hand gestures detected by computer vision techniques.

To investigate effects of presenting robots intentions to the human, Bunz et al. [12] conducted experiment involving mobile robot with projector mounted on top of it, projecting various patterns indicating its intended movement.

Head-up displays and projected user interfaces benefit from freeing operator's hands, which enables direct manipulation with real objects. On the other hand, contemporary head-up displays such as Microsoft HoloLens and others, suffer from narrow field of view and potential user's discomfort in long-term usage. Moreover, end-user programming systems based on hand-held AR overcomes head-up based systems in terms of speed and user experience [13]. While projected interfaces does not suffer from these issues, they are not currently able to present information in free 3D space, and therefore only suitable for tabletop scenarios [1].

The AR systems often benefit from knowledge of the environment and therefore offer new possibilities in end-user programming. The spatial situated programming incorporates real objects into programming process. For instance, Ivy [14] enables user to link different smart devices, create automated behaviour based on readings from smart sensors and visualize data flows between those devices. Reality editor [15] is another example of spatial situated programming, enabling programming of behaviour and interactions of smart objects, using hand-held AR device.

In our proposed approach, the tablet-based AR is combined with semantic information of the objects on the table, to enable regular shop-floor workers to create robotic programs. Contrary to some aforementioned solutions [11], [7], [9], [1], our system aims to both defining the flow of the program and setting its parameters. By using relatively cheap mobile device, the cost of the solution can be significantly lowered comparing to approaches using high-end HMD devices [3], [8] while still remain more flexible than projection based solutions [1], [11].

III. PROPOSED APPROACH

When designing a novel user interface concept for robot programming, we first defined the following issues of current solutions:

- Mental mapping of robot instructions to the physical place in the environment
- Context switching between programming device (e.g. computer) and the workspace
- Low abstraction of the robot instructions, relations between the instructions, conditions and parallel execution.

A. Process-based vs. Object-based approach

The goal of the programmer is to prepare a list of steps that describe: in what order the robot should perform various actions, with what objects the action should be performed, how and under what conditions the action should be performed. The result is a sequence of actions - a program. In principle, this programming task can be implemented in two ways.

Process-based method of robot programming takes advantage of the so-called top-down approach. It describes the whole process with inputs and outputs and then it continues in dividing the process into several sub-processes until it gets to the low-level problems. On the other hand, object-based method describes functionality of different low-level objects and allows to use them to build a working system piece by piece. This is also known as a bottom-up approach.

We used the real world metaphor, when we describe the manufacturing process, we usually:

- first we describe the environment: components, devices, tools and objects that are in the scene and what they do or how to use them for the task,
- then we begin to describe the process step by step, including the links between the environment objects, their specific settings, and the expected outputs,
- Finally, we summarize the expected outputs and risk parts

Based on this observation, we decided to follow an object-based approach proposing concept using spatially-aware augmented reality on mobile device.

B. Program representation

Most of the basic operations of a robotic task are related to a particular place in 3D space, either by relation to a real or virtual object, or directly to an absolute position in the scene. The program representation in our concept was inspired by flowcharts in 3D. Discrete operations (e.g. pick the object, execute operation, etc.) are represented by nodes. Each node is spatially adjacent to the position, where the action takes place. For example node representing operation *Place object to the box* is located above the intended box. This adjacency helps the user with mental mapping of the instructions to the physical space. Nodes hold information needed for their execution, e.g. type of the object which should be manipulated, position on the table, where object should be placed, etc. In addition to setting individual operation parameters, linking these nodes to create a program flow is a key challenge for usable user interface.

Each node has inputs and outputs. By connecting the inputs and outputs of various nodes, user can define the flow of the program. By connecting one output to multiple inputs, the user can specify conditional transition or parallel execution. Based on the parameters of connected nodes, the actual executed path is derived. On the Fig. 2, parallel execution of the program is defined. Workpieces of all the *nodes* are the same, i.e. once the *Execute testing* operations is done, both *Execute printing* and *Pick from tester* operations will be executed in parallel. This approach is valid only when these parallel operations don't physically manipulate the workpiece. In this example, the workpiece will be picked by robot using the *Pick from tester* operation and corresponding label will be printed at the same time. This label will be stuck to the workpiece later in the program.

Conditional execution can be seen on the Fig. 3. The *Pick from table* node has set two different workpieces, e.g. red ball and green cube, meaning one of them will be picked. From this step, two different *Place* operation can occur, each with one of the aforementioned object set as a workpiece. Based on the picked object from step *Pick from table*, corresponding *Place* operation is selected.

C. Spatially situated programming in AR

Spatially situated programming is useful in scenarios, where spatial context is important, like: robot manipulating

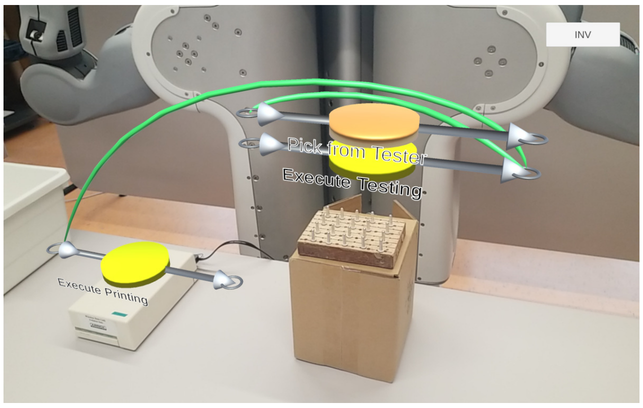


Fig. 2. Parallel execution of the program. The *Execute testing* node is connected with *Pick from tester* node and *Execute printing* node. All of the *nodes* have set the same workpiece, so during the runtime, both paths will be executed at once.

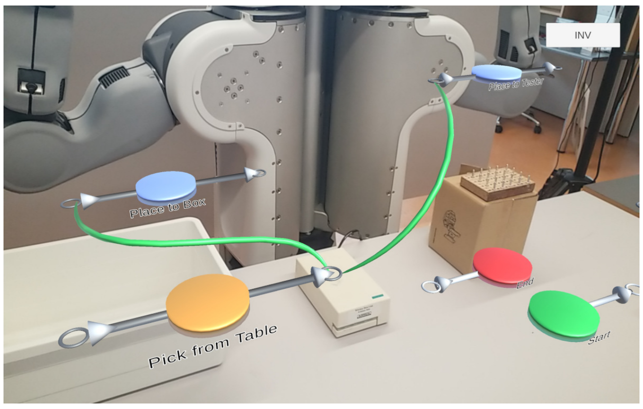


Fig. 3. Conditional execution of the program. There can be seen *Pick from Table* node, connected with two *Place* nodes. The left and right *nodes* have set different workpiece. The actual flow of the program is decided during the runtime, based on type of the workpiece picked in the *Pick from Table* node.

workpieces, picking them from conveyor belt, putting them inside the pressing machine, etc. We propose the system, which is aware of semantic properties of the objects in the environment: knowledge that some object can be picked up, that a box offers inserting of some object, etc. The user can benefit from that shared knowledge of the environment and by using these information, the user can define desired actions more effectively.

The visual elements of the system are presented to the operator using the augmented reality, either in head-up display or using hand-held mobile device.

IV. Prototype of the User Interface

To evaluate the proposed approach, we developed the prototype of the user interface, using hand-held mobile device. In cooperation with our industrial partner, we have selected a specific industrial use-case.

A. Use-case

The selected use-case represents the process of testing the printed circuit board (PCB). The PCB has to be inserted into

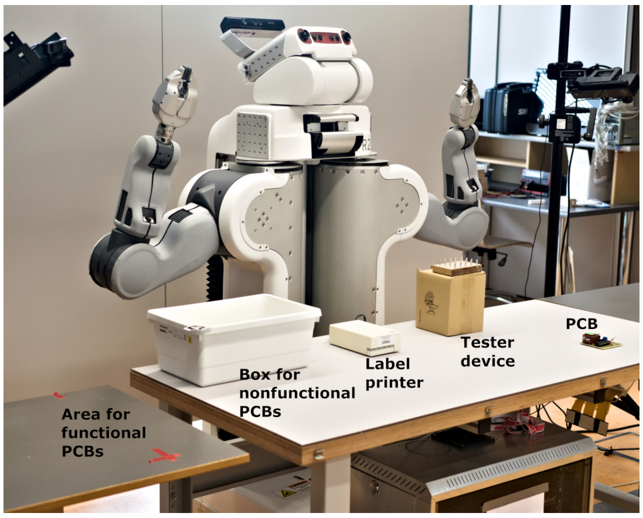


Fig. 4. Testbed used during the experiment. On the table, you can see, from right to left, example PCB, a mockup of the tester device, a printer of the labels, a box for disposing nonfunctional PCBs and another table for functional PCBs.

testing device (a.k.a. tester) and based on the test result either disposed or forward to the next stage of processing. Besides, the corresponding label should be printed and stuck to both functional and nonfunctional PCB. A mockup of the testing facility was prepared, as can be seen on Fig. 4.

The mockup environment consists of the table with the PCB, the testing device, the printer and the box for nonfunctional PCBs. Next to the main table, the other table intended for functional PCBs is placed. To improve the feeling of near future robotic facility, the PR2 robot was placed behind the table. The whole procedure of the use case looks like this:

1) Pick the PCB from the table
2) Place the PCB inside the tester device
3) Execute testing
4) Do in parallel ...
    a) Pick the PCB from the tester device
    b) Print corresponding label
5) Place the PCB on the table
6) Stick the label to the PCB
7) Pick the PCB
8) Place the PCB to ...
    a) the box OR
    b) the other table

Steps 4 represents parallel execution of two operations at the same time, as the robot is picking the PCB from the tester and simultaneously the printer is printing the label. The step 8 represents conditional transition, as the PCB is placed either to the box or to the other table based on the result of the testing process.

B. System

The prototype of the user interface was created using Unity3D game engine. To register motion of the mobile device and track its position in the real world, the ARCore

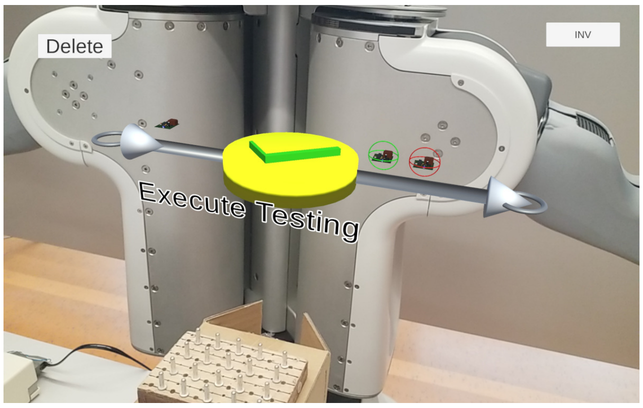


Fig. 5. The *puck* consists of central disc, two circles representing the input and the output and two pipes connecting input/output with the disc. In the first prototype, the type of the *puck* is represented by its color and the text placed in front of the disk. Above the pipes, small 3D models of input and output workpieces are placed.

framework was utilized. The prototype was developed for an Android-driven handheld mobile device. Display of the device shows the video stream from the back-facing camera with superimposed user interface.

Using the Unity3D, the virtual scene was created (see Fig. 7), spatially identical to the real scene described above. The virtual and real scenes are mutually calibrated using the AR marker placed in the lower left corner of the table. This calibration needs to be done once during the application startup.

The system simulates knowledge of the environment and context of all objects and devices on the table. We placed invisible virtual bounding box around each physical object on the table, so user can interact with them by touching them on the screen.

C. User interface elements

Several UI elements were designed for the prototype to allow user to interact with the system. These elements are either 2D or 3D. In this prototype, all elements representing different operations, their connections etc. are static and prepared for selected use-case, as can be seen on Fig. 7.

1) Operations: In our prototype, each operation is represented by so-called *puck* (see Fig. 5). The *puck* consists of central disc, two circles representing the input and the output and two pipes connecting input/output with the disc. The input is placed on the left of the *puck* (with inside the puck aiming arrowhead), output is placed on the right of the puck (with arrowhead aiming outside of the *puck*).

The *puck* serves as a visualization of operation and its parameters, and at the same time, as a main input point for the operator. To change any operation's parameter, the user has to select desired operation first. To enable this, the so-called edit mode was designed. User can switch between the normal and edit mode by clicking on the *puck*. While in edit mode, only the edited and directly connected *pucks* are visible to the user and the others are hidden to lower

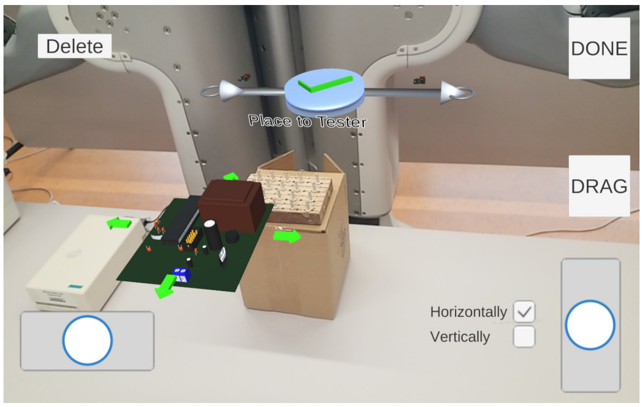


Fig. 6. Teleoperating user interface for navigating the 3D model of the PCB to the tester device. There are two joysticks on the bottom left and bottom right side of the tablet. Next to right joystick, there are two buttons for controlling whether the PCB should move in horizontal or vertical plane. Above the right joystick, there is a *DRAG* button. When it is held, the 3D model moves in the same direction and speed as the tablet.

visual clutter. In the edit mode, parameters of the operation are visible.

Most of the operations only manipulates the workpiece without changing it, i.e. the workpiece on the input is the same as the workpiece on the output of the *puck*. There are two exceptions in our use-case: *Pick from table* and *Execute testing*. The former has no workpiece on the input, because it is the first operation in our program. The picked object is automatically set as a workpiece for the output and it is added to the inventory (will be discussed later). The *Execute testing* operation works as follows. When the PCB is set as a input workpiece, it automatically creates two new workpiece types: *PCB_OK* and *PCB_NOK*. The former means *tested and OK* (functional) and the latter means *tested and not OK* (nonfunctional). These two workpiece types are also added to the virtual inventory.

2) Connections: The *pucks* themselves are not sufficient to define flow of the program, as they only define operations, but not the order in which they shall be executed. To define the flow, the operator can create connections between the *pucks*, by connecting the output of one *puck* and input of other *puck*. This connection is represented by a green spline between these two *pucks*. To make it easier for the user, once he clicks on the output of one *puck*, a big blue plus appears on the input of all other *pucks* and vice versa. By clicking on this plus, the connection is created.

In case of incorrectly created connection, the operator can use a big red cross to remove said connection. This cross is visible only for connections adjacent to currently edited *puck*. There can be several connection attached to one output or input allowing user to define conditions and parallel execution.

3) Interactive objects and context menus: To define an operation for any physical object on the table (e.g. printer, tester, etc.), the operator has to create appropriate *puck* by the object. As the system benefits from the semantic information about objects in the scene, context menu with each possible operations for the objects could be generated. By clicking on any object, this menu emerges, allowing user to define desired operation. This was enabled by creating a clickable invisible bounding box around each object in the virtual scene (semi-transparent boxes on the Fig. 7).

4) Inventory and teleoperating UI: While user composes the program, each workpiece he use in the program (e.g. PCB which shall be picked) appears in the inventory list. By clicking on the workpiece image in the inventory while in edit mode of some *puck*, user can set this object as a workpiece for this *puck*.

The operation "Place to tester" needs to specify 3D position of the workpiece while placing inside the testing device. To do so, a teleoperating user interface is prepared, allowing user to move with 3D model of the workpiece. There are two different approaches to control the position of the desk. The user can adjust the position in vertical or horizontal plane using two joysticks, placed on both side of the screen (see Fig. 6). The other way to set the position is by using so called *DRAG* button. When pressed, the desk moves in the same direction and speed as the tablet, so the operator can literally drag the desk by moving with the tablet (see Fig. 6).

D. User interaction

The screen of the mobile device is used for both visualization of the process and as a main input for the operator. The application on the device knows position and semantic information of all objects in scene (hard-coded for the prototype). Using the ARCore framework, the mobile device knows its position and orientation in the space, which enables the operator to interact with real objects by clicking on their 2D image on the screen.

V. EVALUATION

We provide qualitative results obtained from the user study with 7 participants. To evaluate our proposed approach, we created the prototype of the user interface using ARCore-enabled mobile device.

A. Experimental Procedure

Experimental protocol consisted of 4 phases: orientation, training, programming, discussion.

1) Orientation: During the orientation, the moderator introduced the evaluated system to the participant. He or she then signed an informed consent form.

2) Training: In the second phase, the participant learned how to use the mobile device to create robot instructions (a.k.a. *pucks*), how to set parameters of the instructions and how to connect them to create intended program. During the second phase, the moderator proactively helped the participant to complete the tasks and answered all questions.

3) Main task: The main task was presented to the participant. He or she was asked to program the robot to pick the PCB from the table, place it to the testing device, execute the testing process, print and stick correct label based on the result of the testing process and then place the PCB either

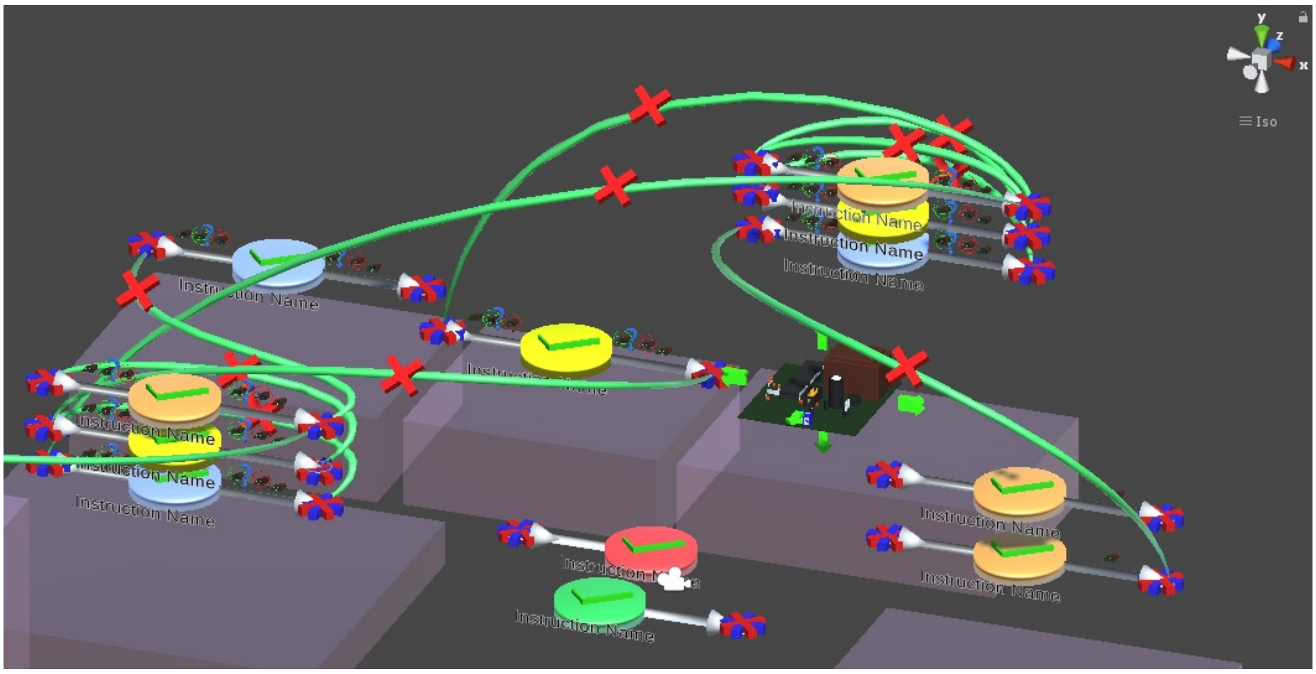


Fig. 7. Unity scene of the prototype UI. Semitransparent boxes define interactive places, which user can use to define intended operation. Above each of these boxes, there are *pucks* of various colors, representing different operations. They are connected with green splines, representing flow of the program (for the sake of clarity, only subset of possible connections are displayed in this figure).

to the box or on the other table (again, based on the result of the test).

After the task was presented to the participant, he or she began to work on the task by him or her self. The moderator was available to answer additional question or to help in case of problems with the prototype, but did not actively step into the programming process. Each participant worked until he or she claimed that the task is done. Moderator than reviewed the created program and either confirmed the correctness or suggested to the participant what should be altered.

4) Discussion: After completing the main task, the participant filled out the questionnaire. Besides, participants were asked for their thoughts of the system, additional questions, etc.

B. Sensors and collected data

The whole process of the experiment was recorded on several cameras. One of them was placed on participant's forehead, aiming to mobile device in participant's hands, other one was aiming towards the participant and two more cameras were recording the workspace. The screen of the mobile device was also recorded, together with indication of participant's input. To record voice of both moderator and participant, lavalier microphones were used.

C. Participants

There were 7 participants of various ages and genders, all of them with none or very limited knowledge of programming and augmented reality. These participants will be labeled as Participant A, B, C, D, E, F and G. Table I shows the demographic data of the participants.

VI. Results and Findings

The section provides measured results and observed findings of the experiment. The main goal of the presented experiment was to prove, that non-expert users are able to program the selected use-case, using the ARCORO system. We focused mainly on usability issues, mental workload of the participants and the user experience.

A. Qualitative and quantitative data

As a metric for the system usability, the SUS[2] [17] method was chosen. To evaluate SUS score for our system, each participant had to score 10 items with one of five responses that range from Strongly Agree to Strongly disagree. Table II shows the SUS score for each participant individually, the mean SUS score from all participants was 82.86 (SD=9.29). According to Sauro-Lewis curved grading scale [17], SUS score in range of 80.8–84.0 is rated by grade **A**, and is at the 90–95th percentile. This shows promising potential for future research in this field, and shows, that the created prototype user interface is highly usable.

To measure the mental workload of the participants, simplified NASA-TLX[3] method was utilized. The mental workload can negatively affect the performance of the operator, therefore is important to measure this attribute from the earliest phases of prototyping. Although the mental workload in laboratory scenarios cannot be generalized directly to the workload in real environment, it still can be useful to reveal potential issues. The mean TLX in our experiment was 27.38

[2]System Usability Scale
[3]NASA Task Load Index

Participant	**Age**	**Gender**	**Education**	**Experience with augmented reality**	**Experience with programming**	**Attitude towards new technology**
A	24	F	bachelor degree	little	little	late majority
B	24	F	bachelor degree	some	little	early majority
C	34	M	master degree	none	none	early adopter
D	22	M	secondary	little	little	late majority
E	21	M	secondary	little	little	early adopter
F	24	F	bachelor degree	little	none	early adopter
G	33	M	secondary	little	little	early majority

TABLE I
DEMOGRAPHIC DATA OF THE PARTICIPANTS. THE SCALE FOR BOTH EXPERIENCE-RELATED QUESTIONS WERE NONE, LITTLE, SOME, QUITE A LOT, MANY. THE ATTITUDE TOWARDS NEW TECHNOLOGY SCALE IS BASED ON ROGERS [16] DIFFUSION OF INNOVATIONS.

Participant	SUS	NASA TLX	UEQ ATT	UEQ PRA	UEQ HED	time to set (s)
A	95.00	25.00	2.67	2.50	2.12	535
B	80.00	25.00	2.00	2.42	0.75	427
C	67.50	47.22	1.17	2.00	1.75	460
D	85.00	27.78	1.67	2.25	2.25	507
E	92.50	27.78	2.67	2.75	2.88	431
F	82.50	19.44	2.00	2.08	2.38	521
G	77.50	19.44	1.33	1.83	0.88	806

TABLE II
DETAILED RESULTS OF ALL MEASURED RESULTS FOR EACH PARTICIPANT.

(SD=9.41), which means that the workload was lower then in at least 80% of studies analyzed by Grier [18].

For any interactive system to be successful, a high-quality user experience is the key. Among several methods to measure the user experience, we selected the UEQ[4], because of its simplicity for both participant and evaluator and reliable results. The system was overall rated as *Excellent* in all UEQ categories, i.e. *Attractiveness* (mean score 1.93, SD=0.58), *Pragmatic* attributes (mean score 2.26, SD=0.28) and *Hedonic* attributes (mean score 1.86, SD=0.72). All categories were evaluated using the standard UEQ benchmark [19].

The mean time for the main task completion was 527 seconds (SD=130s). The main task consisted of settings following operations and their parameters and of creating connections between them: 3x *pick object*, 4x *place object*, 3x *execute (testing, printing and sticking)*. For each operation, workpiece had to be set. Moreover, for one of the *place object* operations, an exact position of the PCB inserted to the tester had to be set. The completion time excludes delays caused by prototype errors.

B. General findings

During the experiment, we found no fundamental problem forcing us to reconsider the proposed approach. Although minor issues were observed or self-reported by participants, all participants were able to complete the task.

[4]User Experience Questionnaire

All participants reported, that the *pucks* (representing operations) were unnecessary large. In cases when there were more *pucks* above the same object, for instance *place object to tester, execute testing and pick object from tester*, the state and parameters of those *pucks* were unclear and it was hard to recognize mutual connections. To avoid this, design of *pucks* needs to be refined and better strategy of *pucks* placement should be adopted in further versions.

The participants were instructed to inform the moderator once they though they have successfully finished the programming. Most of the created programs contained one or more errors, which would lead to failure during execution. The participant C explicitly reported, that he is unable to check if the program is correct. The participant A in the end went through all created *pucks* to check whether all parameters are correctly set and connections between *pucks* are as intended.

After the errors were pointed out by the moderator, each participant was able to correct the error and to successfully finish the task. This has shown, that debugging system has to be improved and better system state indicators should be involved. To support users awareness of the program correctness, the program flow visualization needs to be improved.

Only two of the participants found out, that they can benefit from active movement of the mobile device inside the scene, to achieve higher accuracy when clicking on interface components. Most of them were just standing in certain distance from the table and using only vertical rotation in cases when FOV of the tablet was too narrow. The participant B stated, that it was more comfortable for her to just stand at one place to observe the whole situation and that she would appreciate the possibility of zooming the scene on the screen to avoid miss-clicks.

The usual procedure for most of the participants consisted of creating the *puck*, followed by creating the connection between said *puck* and previously created *puck*, repeated until the whole program was created. The participant A followed a different approach. At first she created most of the *pucks* to label all desired operations and once she was satisfied with *pucks*, she started to create connections between them.

Participants A, B, C and E were using only one hand to control both joysticks (placed on different side of the screen) while the rest of the participants were using both hands, as was intended when designing the user interface. The participant A was the only one to use a *DRAG* button, to set the initial position of the desk, followed by refining the final position using the joysticks.

Although minor issues were observed during the experiment, all of the participants rated the system positively. The participants agreed that the system is easy to use and requires no special knowledge from the operator.

VII. CONCLUSIONS

The aim of this work is to reflect current needs in the area of programming robots in low and medium complex tasks in a shared collaborative environment. We have designed a new concept of robot programming using augmented reality on a mobile device. The main goals pursued in the design of the new concept were: eliminating the need to switch user context between desktop and work environment by mapping instructions directly into a real 3D environment, reducing user mental stress by using semantic information about real objects and increasing the abstraction of instructions and their relations.

We have defined a simple use-case that is inspired by the real demands from the industry. In the experiment, we observed mainly usability of designed UI, workload of user and user experience with designed spatial programming concept. We have evaluated with 7 users which has shown that, despite some shortcomings discussed, this is the direction that can be taken. All participants were able to perform all the tasks independently after a short training. All participants evaluated the usability of the interface mostly positively.

Positive adoption of the new concept can also be attributed to the use of equipment that most users are used to working with. In the future, we want to verify this unambiguity and compare the usability of the concept with other, yet less common devices, such as HoloLens glasses. In the next research, we will also focus on improving the orientation in the programmed task, solving the UX deficiencies found in this study, and integrating the UI into a real robotic system.

ACKNOWLEDGMENT

The work was supported by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science – LQ1602".

REFERENCES

[1] Z. Materna, M. Kapinus, V. Beran, P. Smrž, and P. Zemčík, "Interactive spatial augmented reality in collaborative robot programming: User experience evaluation," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Aug 2018, pp. 80–87.

[2] C. Mateo, A. Brunete, E. Gambao, and M. Hernando, "Hammer: An android based application for end-user industrial robot programming," in *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, Sep. 2014, pp. 1–6.

[3] S. Yitzhak Gadre, E. Rosen, G. Chien, E. Phillips, S. Tellex, and G. Konidaris, "End-user robot programming using mixed reality," 10 2018.

[4] J. Aleotti, G. Micconi, and S. Caselli, "Object interaction and task programming by demonstration in visuo-haptic augmented reality," *Multimedia Systems*, vol. 22, no. 6, pp. 675–691, Nov 2016. [Online]. Available: https://doi.org/10.1007/s00530-015-0488-z

[5] P.-C. Li and C.-H. Chu, "Augmented reality based robot path planning for programming by demonstration," 12 2016.

[6] J. Huang and M. Cakmak, "Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts," in *HRI*. ACM, 2017, pp. 453–462.

[7] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. F. Machiel Van der Loos, and E. Croft, "Robot programming through augmented trajectories in augmented reality," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1838–1844.

[8] S. Blankemeyer, R. Wiemann, L. Posniak, C. Pregizer, and A. Raatz, "Intuitive robot programming using augmented reality," *Procedia CIRP*, vol. 76, pp. 155–160, 01 2018.

[9] S. Stadler, K. Kain, M. Giuliani, N. Mirnig, G. Stollnberger, and M. Tscheligi, "Augmented reality for industrial robot programmers: Workload analysis for task-based, augmented reality-supported robot control," in *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*. IEEE, 2016, pp. 179–184.

[10] S. Magnenat, M. Ben-Ari, S. Klinger, and R. W. Sumner, "Enhancing robot programming with visual feedback and augmented reality," in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2015, pp. 153–158.

[11] Y. Gao and C.-M. Huang, "Pati: A projection-based augmented table-top interface for robot programming," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, ser. IUI '19. New York, NY, USA: ACM, 2019, pp. 345–355. [Online]. Available: http://doi.acm.org/10.1145/3301275.3302326

[12] E. Bunz, R. T. Chadalavada, H. Andreasson, R. Krug, M. Schindler, and A. Lilienthal, "Spatial augmented reality and eye tracking for evaluating human robot interaction," in *RO-MAN 2016 Workshop: Workshop on Communicating Intentions in Human-Robot Interaction, New York, USA, Aug 31, 2016*, 2016.

[13] N. Dass, J. Kim, S. Ford, S. Agarwal, and D. H. P. Chau, "Augmenting coding: Augmented reality for learning programming," in *Proceedings of the Sixth International Symposium of Chinese CHI*, ser. ChineseCHI '18. New York, NY, USA: ACM, 2018, pp. 156–159. [Online]. Available: http://doi.acm.org/10.1145/3202667.3202695

[14] B. Ens, F. Anderson, T. Grossman, M. Annett, P. Irani, and G. Fitzmaurice, "Ivy: Exploring spatially situated visual programming for authoring and understanding intelligent environments," in *Proceedings of the 43rd Graphics Interface Conference*, ser. GI '17. School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society, 2017, pp. 156–162. [Online]. Available: https://doi.org/10.20380/GI2017.20

[15] V. Heun, J. Hobin, and P. Maes, "Reality editor: Programming smarter objects," in *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, ser. UbiComp '13 Adjunct. New York, NY, USA: ACM, 2013, pp. 307–310. [Online]. Available: http://doi.acm.org/10.1145/2494091.2494185

[16] E. Rogers, *Diffusion of innovations*. Free Press of Glencoe, 1962. [Online]. Available: https://books.google.cz/books?id=zw0-AAAAIAAJ

[17] J. Sauro and J. R. Lewis, *Quantifying the User Experience: Practical Statistics for User Research*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012.

[18] R. A. Grier, "How high is high? a meta-analysis of nasa-tlx global workload scores," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 59, no. 1, pp. 1727–1731, 2015. [Online]. Available: https://doi.org/10.1177/1541931215591373

[19] M. Schrepp, A. Hinderks, and J. Thomaschewski, "Construction of a benchmark for the user experience questionnaire (ueq)," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, pp. 40–44, 06 2017.

Improved Indirect Virtual Objects Selection Methods for Cluttered Augmented Reality Environments on Mobile Devices

Michal Kapinus, Daniel Bambušek, Zdeněk Materna, Vítězslav Beran, Pavel Smrž
Faculty of Information Technology, Brno University of Technology
Brno, Czech Republic
{ikapinus,bambusekd,imaterna,beranv,smrz}@fit.vut.cz

***Abstract*—The problem of selecting virtual objects within augmented reality on handheld devices has been tackled multiple times. However, evaluations were carried out on purely synthetic tasks with uniformly placed homogeneous objects, often located on a plane and with none or low occlusions. This paper presents two novel approaches to indirect object selection dealing with highly occluded objects with large spatial distribution variability and heterogeneous size and appearance. The methods are designed to enable long-term usage with a tablet-like device. One method is based on a spatially anchored hierarchy menu, and the other utilizes a crosshair and a side menu that shows candidate objects according to a custom-developed metric. The proposed approaches are compared with direct touch in the context of spatial visual programming of collaborative robots problem, on a realistic workplace and a common robotic task. The preliminary evaluation indicates that the main benefit of the proposed indirect methods could be their higher precision and higher selection confidence for the user.**

***Index Terms*—virtual object selection; augmented reality; spatially situated visual programming**

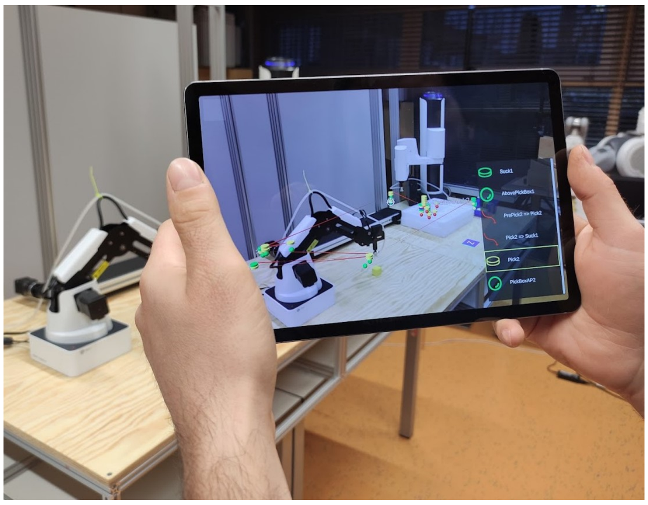


Fig. 1. A user observes the scene with robotic program visualized in AR. There is a crosshair and side menu for indirect object selection.

I. Introduction

The need for selecting objects on a 2D screen arose with the onset of graphical user interfaces and pointing devices. Traditional approaches for objects selections might be divided into two categories: direct and indirect. For computers, the most widespread method for object selection is indirect control of the graphical cursor, either by mouse, keyboard, or touchpad. The direct method is usually the most common for touchscreen devices – using either the user's fingers or stylus.

When it comes to a 3D user interface, such as augmented reality (AR) on mobile devices, Bowman et al. [1] state that the quality of interaction with 3D objects has a profound effect on the quality of the entire 3D user interface. They also state that selection of and manipulation with virtual objects is one of the most crucial features of such an interface, because *"if the user cannot manipulate virtual objects effectively, many application-specific tasks simply cannot be performed"* [1].

In the case of the cluttered scenes, with partially or fully occluded objects, traditional methods may begin to lose their breath in terms of precision or speed [2,3]. The problem becomes even worse on mobile devices, where the primary selection tool is usually a human finger. In the case of large displays (typically with tablets), the ergonomics of the whole process needs to be taken into account due to the weight of the device, especially when it comes to long-lasting operations, such as robot programming.

We have selected spatial visual programming in AR as a representative task, similar to the one presented in our previous work [4], where a relatively high amount of virtual objects is presented inside a small area, and these are partially occluded (depending on a view angle).

We have prepared an experiment to compare two indirect and one direct methods for object selection in AR:

1) Combination of the crosshair and a head-up side menu, containing a set of nearby objects.
2) Combination of the crosshair and in-space hierarchical menu, and 3) touch.

For this Late-Breaking Report, a pilot study ($n = 3$) was conducted to get a first impression of both developed methods and verify both experiment design and prototype application. This evaluation is a part of our ongoing research on simplified robot programming.

This project was implemented with financial support from the state budget through the Ministry of Industry and Trade in the TRIO program (FV40052).

II. Related work

The problem of selecting objects on a screen is well studied. However, there are many specifics when it comes to AR on handheld devices. First of all, objects on the screen are not stable during interaction due to hand tremors or visual tracking instability. Also, techniques intended initially for virtual reality, as Go-Go [5], for instance, are less usable on handheld devices [2].

In the literature, Fitt's Law [6], a technique to quantify the difficulty of a target selection task, is commonly used to compare selection methods. The original one-dimensional version was later adapted to 2D and even to 3D selection problems [7–9]. However, it implies several unrealistic limitations: participants have to be seated and locate homogeneous un-occluded targets on a plane in front of them. There exist several attempts to create more realistic conditions, where targets were: displayed at mixed depth [9], of different sizes with variable level of occlusion [10], highly occluded and with similar appearance [11]. Although those publications made some aspects of evaluation more realistic, they were carried out on purely synthetic tasks in an empty environment. The evaluation in a realistic industrial environment exists as an isolated example [12].

There is also a lack of experiments carried out on larger than phone form factors (device's physical size and shape); there are just indications that indirect methods might be more suitable for them [13,14]. Moreover, there are signs that indirect methods could be preferable for long-lasting tasks [3].

An example of a selection technique specifically designed for handheld devices could be DrillSample [11], intended to provide accurate selection in dense AR environments, optimized for one-hand operation on the phone. It is a direct method, using ray-casting (touch) and an optional refinement step. A set of selection methods for phone-sized devices and dense AR environments that outperformed ray-casting and Go-Go were proposed and evaluated in [2]. A screen-centered crosshair was compared with a relative one, bound to the physical object's frame in [14] for both phone and tablet. The relative one was more accurate and less sensitive to the registration jitter and the device's form factor. The list-based selection, with icons displayed on the side of the screen for objects nearest to the crosshair, was compared with a touch-based selection in [3]. The list-based method was designed to minimize the number of touches by taking advantage of device motion and is recommended for crowded scenes to select multiple objects during longer-lasting tasks.

In this paper, we deal with the use case of robot visual programming. Even a relatively simple pick and place task results in a dense AR environment, with a high amount of virtual objects of different appearance and semantic meaning, typically clustered nearby spatially important points. The task requires a large screen, and therefore, we were interested in tablet-like devices. To allow long-term usage, we have developed methods enabling users to hold the device with both hands, control the interface using their thumbs, and evaluated them on a realistic task.

III. Proposed Methods

The necessity to select virtual objects in dense AR environments arose during the development of robot programming tools in AR using tablet devices where specific virtual objects represent spatial anchors, robot actions, and process flow. Such approaches usually have a solid connection to the natural environment and thus show strong potential in AR [4,15]. Even relatively simple tasks usually involve many virtual objects in the scene with some degree of occlusion. In such an environment, the selection becomes problematic, so there is a need for fast, accurate, and easy-to-use methods, specific for large screens.

We have proposed two indirect methods for precise virtual object selection in heavily cluttered environments in AR. Both of them work with spatially clustered objects and use the following algorithm to obtain the cluster from the scene:

1) Cast a ray from the center of the screen (crosshair) and add the first hit object to the cluster (see Fig. 2, middle).
2) If the cluster is empty (i.e., the ray hit no object), cast a thick, square ray with a side size 1 cm from the same origin, allowing to select even tiny objects, and add the first hit object to the cluster.
3) If the cluster is still empty (i.e., no ray hit any object), return an empty cluster.
4) Search for any object colliding with a virtual sphere with a radius of 3 cm, with a center in the position where the cast ray hit the first object (see Fig. 2 right). Add all these objects to the cluster and return it ordered by their distance to the first hit object.

Proposed methods are meant to help the user select spatially clustered objects in a cluttered environment. Each method provides different access to hard-to-reach or occluded objects by either hierarchical or flat representation. One of the proposed methods uses the head-up menu, and the other the in-space menu, which allows us to observe whether the attention switches between the scene and the head-up menu will be problematic or annoying for users.

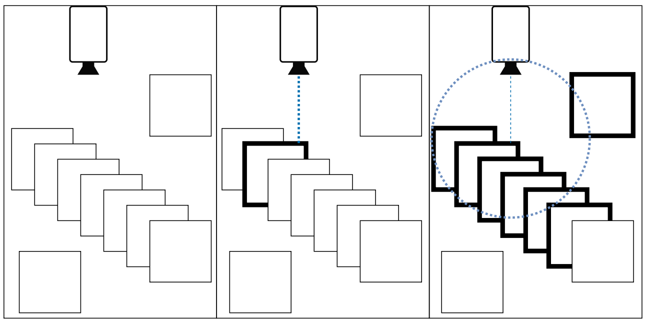
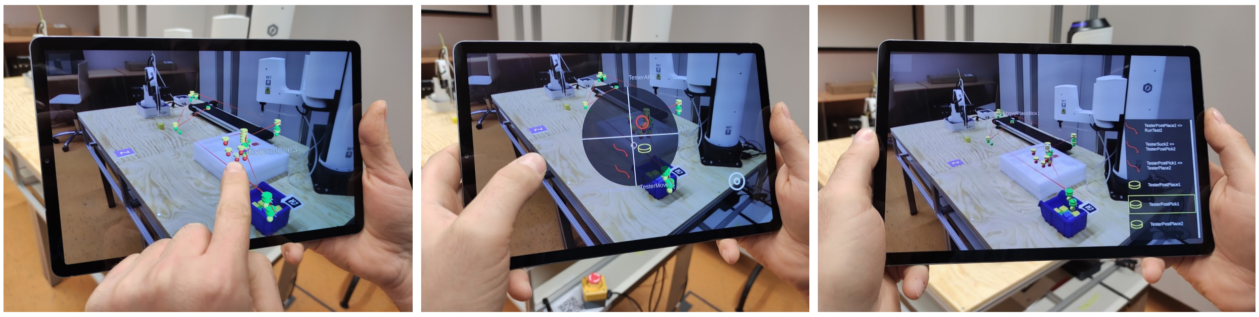


Fig. 2. Principle of our cluster selection in two dimensions. Left: the camera observes a scene. Middle: a ray is cast from the camera's center and hits an object (square with thick borders). Right: in the position of the hit, a circle (sphere in 3D) is generated and all objects located inside or colliding with are included into the cluster (squares with thick borders).



(a) Direct touch method. The user selects the object by touching its visual representation on the screen. (b) Spatial hierarchical menu. The user selects a cluster of objects by device movement, followed by selecting quadrants of the menu. (c) Selector menu. The user selects the object by combining the physical movement of the device and the selection of the object using a side menu.

Fig. 3. User interface of three selection methods studied in this work.

A. Spatial hierarchical menu

The first proposed method deals with the clustering of objects not only by distance but also by the class or other parameters of the object. Upon selecting a spatial cluster, the objects are divided into four categories, each represented by one quadrant of a full circle. If there are more than four classes, one of the quadrants is used as a container for *other* classes, through which the user could reach the remaining ones.

This menu is recursive, meaning that after selecting one of the quadrants, it further divides contained objects into the quadrants based on selected parameters (a color in our case). We have based this approach on SQUAD [16] technique, used for selection by progressive refinement of a cluster of objects based on spatial, visual, and other parameters.

This hierarchical menu is rendered in the 3D space at a certain distance in front of the device. The user selects desired quadrant by the same crosshair used to select the initial cluster and confirms the selection by clicking on the circular button in the lower right corner of the screen. This limits the user's attention switches between the scene and a head-up display.

B. Selector menu

The second proposed selection method is similar to the Icon-based selection presented by [3]. It shows the obtained cluster in the form of a list on the side of the screen (head-up like), easily reachable by the user's right thumb when holding the tablet device using both hands. Each item in the menu contains an icon representing the object's class and the object's name. The directly hit object is pre-selected, which is indicated by highlighting the object in both scene and the menu, using an outline. The user makes the selection by touching one of the items in the list, regardless of whether the item is pre-selected or not.

The list of the objects is continuously updated as the user hovers the tablet over the environment. A simple hysteresis supported the stability of the objects in the list, and updates were limited to 2 Hz.

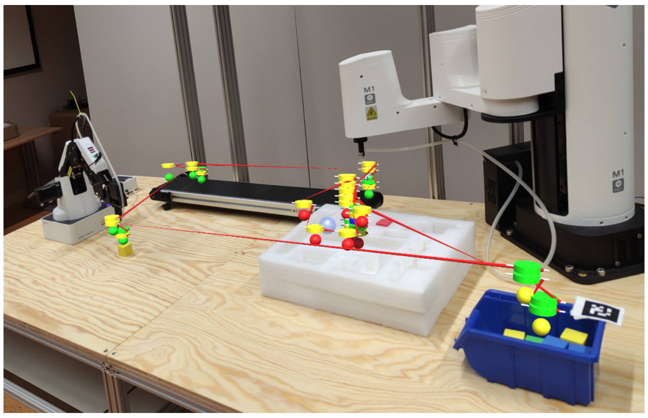


Fig. 4. The virtual scene rendered over the real environment, used in the conducted experiment. The virtual objects represent a simple program, where the small robot should pick a cube, move it to the conveyor belt and pass it to the second robot. The second robot should pick the cube again, touch the simulated device (represented by white foam box) on several places with the cube attached, and drop the cube to the blue box.

IV. Pilot Experiment

We organized the experiment as a within-subject study with three methods in randomized order: direct touch – baseline (A), spatial hierarchical menu (B), and selector menu (C). It was carried out in a laboratory, on a demonstration workspace for a visual programming framework, equipped with two robots (Dobot Magician and M1) and a conveyor belt (see Fig. 4). Robots were switched on but remained stationary during the study.

The objects to be selected represent a simple pick and place program. A total of 91 virtual objects were displayed above the workplace, spread across the area of $1.3m^2$, grouped into 5 clusters (see Fig. 4). There were four categories of objects: scene objects (e.g., a box or a robot), action points (spheres representing important 3D space anchors), actions above action points (cylinders representing program steps with the purpose indicated by different colors), and lines (connections between actions). For each tested selection method, the task was to select 30 objects (3 scene objects, 6 action points,

TABLE I
DEMOGRAPHIC INFORMATION ABOUT PARTICIPANTS (AR_x DENOTES EXPERIENCE WITH AR ON THE SCALE OF 1-5.) AND OBTAINED DATA, WHERE OBJECTIVE MEASURES ARE TASK TIME (TOTAL TIME TO SELECT 30 OBJECTS), SUCCESS RATE (IN THE RANGE OF $[0, 1]$), TRAJECTORY (TOTAL DEVICE DISPLACEMENT AS MEASURED BY VISUAL TRACKING) AND SUBJECTIVE MEASURE IS TLX (RANGE $[0, 100]$, THE LOWER THE BETTER).

	age	AR_x	task time [s]			success rate [–]			trajectory [m]			TLX [–]		
			A	B	C	A	B	C	A	B	C	A	B	C
p_1	35	4	114.56	258.42	187.39	0.77	1.00	0.93	15.40	26.60	20.10	19.44	66.67	41.67
p_2	25	2	135.93	380.55	240.26	0.73	0.90	0.93	16.44	24.11	15.80	2.78	11.11	13.89
p_3	30	2	142.46	319.42	311.92	0.83	0.90	0.80	39.78	40.96	32.66	2.78	25.00	16.67
mean	30	2.67	*130.98*	319.46	246.52	0.78	*0.93*	0.89	23.87	30.56	*22.85*	*8.33*	34.26	24.08

21 actions) – those were the same for all methods, however in a different order (same across the participants). The task inevitably contained a search phase (which Fitt's Law tries to avoid), as objects were spread across a large area, could not fit into the field of view, and users could move freely. However, the search phase was present for all methods and should not affect the results.

We chose *task time*, *success rate* (both representing task performance) and *trajectory* (distance reported by the tablet's visual tracking, therefore having limited precision; could be related to necessary physical effort) as objective metrics and NASA Task Load Index [17] (TLX) as an subjective metric.

Participants were recruited from faculty staff (3 males, one Ph.D. student, two postdocs), further denoted as p_{1-3}. They were first informed about the study purpose and signed informed consent. The moderator explained the usage of each method, and participants then tried to select four objects. After that, they performed 30 selections and filled in the TLX questionnaire for the method. In the end, the moderator carried out a debriefing with the participant.

The order of methods assigned to participants was $p_1 := \{A, B, C\}$, $p_2 := \{B, C, A\}$ and $p_3 := \{C, A, B\}$.

V. RESULTS AND DISCUSSION

From the results, it may be seen that both indirect methods provide better success rates than the baseline, however, at the expense of notably longer times. Those results seem to be consistent between participants. The big difference in the *trajectory* metric for all methods for the participant p_3 was probably caused by the interaction strategy he adopted. Most of the time, the first two participants stood in one place, while the last one walked around the workplace to acquire the best pose for selection.

Regarding the TLX metric, all participants ranked the **A** method as the one with the lowest task load, which could be influenced by the general acquaintance of the baseline method. We will extend and improve the training session for the final experiment, as we observed that most of the participants' problems occurred during the first few selection attempts of the main task.

During a debriefing, the p_3 stated that he felt confident when using the **C** and especially the **B** method, as he was informed about which objects were about to be selected. In the **A** method, he was sometimes unsure because of the small size and occlusion of the target. Participants p_1 and p_2 both complained about the instability of the objects in the list. The participant p_2 suggested a freeze button for the left thumb, which will pause any changes in the selector menu and help him comfortably select the desired object. Alternatively, the adoption of some temporally stable labeling methods, such as the one presented by Bobak et al. [18], could significantly improve the stability of objects in the list. The p_2 generally liked the method **A**, but he complained about the need to move close to the object to be able to achieve a certain degree of accuracy.

The participants p_1 and p_3 held the device with one hand on the short side while using the other hand for object selections. From our experience, such holding of a tablet device causes arm fatigue in the case of longer sessions. Also, we identified a few usability problems in the indirect methods' design, which might impact results. For **B**, participants p_2 and p_3 sometimes had problems with stepping back in the hierarchy menu, and they accidentally selected the wrong object instead, causing a worse success ratio. Moreover, the labels of the quadrants were too small and thus hard to read. In the case of **C**, the instability of the order of menu items probably caused some wrong selections and could lead to a higher task load. According to observations, users tend to precisely aim an object with a crosshair to get it pre-selected in the side menu, although they could select any object listed there. We speculate that removing the pre-selection mechanism and highlighting the whole cluster (the content of the menu) in the scene will make the method faster and improve its usability.

VI. CONCLUSIONS

The conducted experiment initially compared three different methods for virtual object selection, one direct and two indirect. The purpose of this pilot experiment was mainly to obtain first impressions and validate the study design.

A preliminary evaluation of our new methods suggested that using indirect methods in AR on mobile devices could help increase selection accuracy. The increase could be achieved primarily in tasks with heavily cluttered environments, such as robot visual programming, AR-enabled visualization of robotic trajectories, editor of robotics workcells, or any other situated visualization. We have collected valuable feedback for our prototype, which will be addressed in a refined version of our methods, thoroughly evaluated later.

A significantly higher number of participants will be involved to observe statistical differences between methods for the final experiment. The experiment design will be slightly altered, as the training session will be extended, and the methods will be explained more thoroughly to the participants.

References

[1] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev, *3D User Interfaces: Theory and Practice*. USA: Addison Wesley Longman Publishing Co., Inc., 2004.

[2] J. Yin, C. Fu, X. Zhang, and T. Liu, “Precise target selection techniques in handheld augmented reality interfaces,” *IEEE Access*, vol. 7, pp. 17 663–17 674, 2019.

[3] A. Samini and K. Lundin Palmerius, “Wand-like interaction with a hand-held tablet device—a study on selection and pose manipulation techniques,” *Information*, vol. 10, no. 4, p. 152, 2019.

[4] M. Kapinus, V. Beran, Z. Materna, and D. Bambušek, “Spatially situated end-user robot programming in augmented reality,” in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2019, pp. 1–8.

[5] I. Poupyrev, M. Billinghurst, S. Weghorst, and T. Ichikawa, “The go-go interaction technique: non-linear mapping for direct manipulation in vr,” in *Proceedings of the 9th annual ACM symposium on User interface software and technology*, 1996, pp. 79–80.

[6] P. M. Fitts, “The information capacity of the human motor system in controlling the amplitude of movement.” *Journal of experimental psychology*, vol. 47, no. 6, p. 381, 1954.

[7] R. J. Teather and W. Stuerzlinger, “Pointing at 3d targets in a stereo head-tracked virtual environment,” in *2011 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 2011, pp. 87–94.

[8] Y. Cha and R. Myung, “Extended fitts’ law for 3d pointing tasks using 3d target arrangements,” *International Journal of Industrial Ergonomics*, vol. 43, no. 4, pp. 350–355, 2013.

[9] Y. Y. Qian and R. J. Teather, “The eyes don’t have it: an empirical comparison of head-based and eye-based selection in virtual reality,” in *Proceedings of the 5th Symposium on Spatial User Interaction*, 2017, pp. 91–98.

[10] F. Argelaguet and C. Andujar, “Efficient 3d pointing selection in cluttered virtual environments,” *IEEE Computer Graphics and Applications*, vol. 29, no. 6, pp. 34–43, 2009.

[11] A. Mossel, B. Venditti, and H. Kaufmann, “Drillsample: precise selection in dense handheld augmented reality environments,” in *Proceedings of the Virtual Reality International Conference: Laval Virtual*, 2013, pp. 1–10.

[12] P. Perea, D. Morand, and L. Nigay, “Target expansion in context: the case of menu in handheld augmented reality,” in *Proceedings of the International Conference on Advanced Visual Interfaces*, 2020, pp. 1–9.

[13] I. Radu, B. MacIntyre, and S. Lourenco, “Comparing children’s crosshair and finger interactions in handheld augmented reality: Relationships between usability and child development,” in *Proceedings of the The 15th International Conference on Interaction Design and Children*, 2016, pp. 288–298.

[14] T. Vincent, L. Nigay, and T. Kurata, “Handheld augmented reality: Effect of registration jitter on cursor-based pointing techniques,” in *Proceedings of the 25th Conference on l’Interaction Homme-Machine*, 2013, pp. 1–6.

[15] S. M. Chacko and V. Kapila, “An augmented reality interface for human-robot interaction in unconstrained environments,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3222–3228.

[16] R. Kopper, F. Bacim, and D. A. Bowman, “Rapid and accurate 3d selection by progressive refinement,” in *2011 IEEE Symposium on 3D User Interfaces (3DUI)*, 2011, pp. 67–74.

[17] S. G. Hart, “Nasa-task load index (nasa-tlx); 20 years later,” in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 50, no. 9. Sage publications Sage CA: Los Angeles, CA, 2006, pp. 904–908.

[18] P. Bobák, L. Cmolik, and M. Cadik, “Temporally stable boundary labeling for interactive and non-interactive dynamic scenes,” *Computers & Graphics*, vol. 91, 08 2020.

Augmented Reality Spatial Programming Paradigm Applied to End-User Robot Programming

Michal Kapinus[1*], Vítězslav Beran[1], Zdeněk Materna[1] and Daniel Bambušek[1]

[1]Faculty of Information Technology, Brno University of Technology, Božetěchova 1/2, Brno, 612 00, Czech Republic.

*Corresponding author(s). E-mail(s): ikapinus@fit.vut.cz; Contributing authors: beranv@fit.vut.cz; imaterna@fit.vut.cz; bambusekd@fit.vut.cz;

Abstract

The market of collaborative robots is thriving due to their increasing affordability. The ability to program such a robot without requiring a highly skilled robot programmer would increase the spread of collaborative robots even more. Visual programming is a common method for end-user programming. It enables the end-users to quickly and easily define the program logic, but it usually struggles with defining spatial features, which is crucial for robot programs. The solution based on augmented reality (AR) would allow end-users to work in the robot's task space and easily and understandably define the spatial parameters. We identified two problems of contemporary solutions – program comprehension and spatial information setting. We have proposed a method based on Spatially Anchored Actions to address these issues and evaluated the method using a mobile AR-based user interface. The proposed solution was compared against a commercially available desktop-based visual programming solution in a user study with 12 participants. According to the results, the novel method significantly improves comprehension of pick and place-like programs and is more preferred by users than the standard end-user robot programming tool based on visual programming. Moreover, it greatly supports the user with the tools for spatial information settings and deals with the ergonomics of usage of AR interface on tablet devices.

Keywords: Robot programming, augmented reality, SME future

1 Introduction

In SMEs, small batches and frequent changes in production are typical. Many tasks involve pick & place operations, machine tending, quality inspections, packaging, and palletizing. The pick & place is one of the most common operations in the industry, and, at the same time, it is often a very repetitive and straightforward operation. Many industrial tasks involve some machinery, e.g., press, lathe, or CNC, which implies the need to tend the machine by the operator, which is a repetitive operation.

Although all of the tasks mentioned above are relatively simple for the human operator (and the cobot), the programming of such a task could be challenging, especially for a non-experienced person. The end-user programming methods could make the automation of such tasks affordable and profitable for SMEs. Moreover, it may help to free skilled workers from monotonous work and therefore contribute to resolving a lack of workforce. These factors make cobots, in conjunction with end-user programming methods, appealing to SMEs. Moreover, the share of cobots in the industry rises each year (see Fig. 1). Therefore, the need for simplified programming methods rises as well [1].

As robotic programs involve many operations within the real environment, the programmer must easily understand individual actions' spatial parameters. At the same time, most contemporary programming solutions have a weak connection between the program representation and the actual space where the program takes place. Visualization of spatial parameters is usually done using textual description (e.g., *coordinates: X=0.5, Y=0.9, Z=1.8*) or drawn into a virtual workplace or robot model. In both cases, the user needs to map the spatial representation to the actual environment mentally.

In addition, in the case of online programming, the programming often takes place on the screen of some additional device, such as a teach pendant. As robotic tasks take place in the real environment, the programmer needs to constantly switch their attention between the robot and the screen (e.g., when defining the spatial parameters of the program), which implies a high cognitive and attention-related workload [2].

Another challenge that every robot programmer faces is a precise definition of spatial information in the 3D world. The operator must be able to define spatial parameters for robotic programs precisely. At the same time, it is often quite challenging because the robot usually works with its coordinate system, which is not always aligned with the world and not apparent by just observing the robot. Many programming tools, especially those working with programming by demonstration, use the robotic arm to define spatial coordinates. The operator could use direct manipulation with the arm (if the robot supports it) and manually drag the arm into the desired position. It is pretty fast, but the

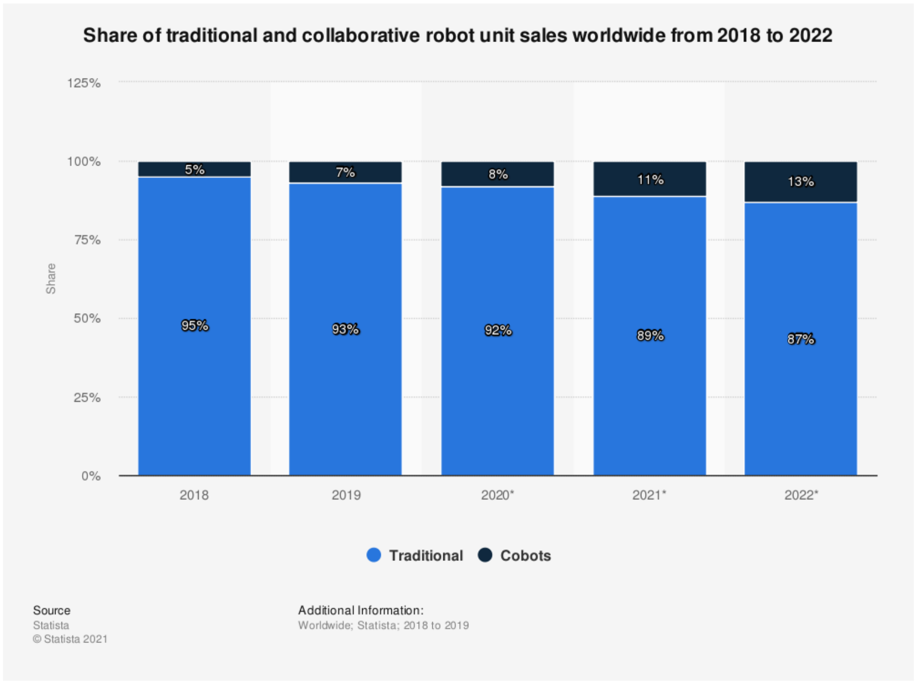


Fig. 1: Share of traditional and collaborative robot unit sales worldwide from 2018 to 2022[1]

precision could be limited, and it could be tiresome for the operator's hands. The other possibility is to use the robot's teach pendant manipulation methods, such as jogging or joint manipulation. It would relieve the human's arms, and the precision is virtually unlimited. On the other hand, using this method could be slow. The third possibility is the combination of both approaches.

Last but not least, the program must be represented understandably. The program is typically represented as a sequence of actions of just a few types, e.g., move line instruction, move joints instruction, or end-effector control instruction. These could be visualized in the form of diagrams or just as lines of code. By just looking at the program representation, it is hard to match individual action to the actual step in the program, i.e.: "*Which of the 20 `MOVE` instructions is the one I am looking for?*".

We address the challenges mentioned above in this paper and propose a method based on augmented reality (AR) for environment annotation and end-user robot programming. This method provides in-situ creation and visualization of the program and interaction with virtual and real elements of the scene. The method benefits from the semantic knowledge of the environment, i.e., of the present objects and their arrangement. Only relevant actions are available for the programmer based on the present objects. The pose and shape of present objects could be used for the program's spatial parameters settings. Therefore, the environment must be annotated (i.e., objects and their poses specified) before the programming.

[1] statista.com/statistics/1018935/traditional-and-collaborative-robotics-share-worldwide/

AR has been previously shown to increase users' understanding of spatial-related problems [3]. On the other hand, current mobile AR setups suffer from pose estimation imperfections [4–6]. It could affect the precision of spatial parameters, which is crucial for robotic programs. Moreover, different AR devices bring their problems; for example, the contemporary head-mount displays suffer from a narrow field of view and may not be convenient for some users, mobile AR utilizes one or both user's hands, and so on.

The main research objectives, based on the problems defined above, are:

RO_1 – Improve the robotic program comprehension over the standard method.

RO_2 – The creation and adaptation of programs in AR should be at least as simple as in current end-user programming tools.

RO_3 – Lower the perceived task load compared to the standard method.

RO_4 – Provide good ergonomics of the mobile AR user interface.

RO_5 – Allow precise specification of spatial information using mobile AR.

The main contribution of this paper lies in two domains: enhancement of robotic program comprehension using AR and precise definition of spatial parameters in AR.

The remainder of this paper is organized as follows: Section 2 presents the theoretical background of our method, which is explained and discussed in Section 3. Section 4 contains details of conducted user study and discovered observations, results of the user study are further discussed in Section 5, and everything is concluded in Section 6.

2 Related Work

The increasing spread of cobots in the industry raises the demand for allowing end-users to program them. Naturally, robots can be programmed using a vendor-specific language, such as ABB's RAPID [7], Fanuc's Karel, or Universal Robots' URScript [8]. Although these languages offer relatively simple syntax and programming commands, they still require programmers with expertise in programming and robotics [1].

A certain form of simplified programming is offered by some teach pendants. However, they often possess high mental and physical demands, lack the ability to use common syntax structures, and have no option for visualization [9]; therefore, their usability seems to be rather low [10]. Offline programming tools, such as ABB RobotStudio[2] [11], Fanuc RoboGuide[3] or RoboDK[4], offer more functionalities and allow to program the robot in a simulated environment, which reduces the robot downtime, but on the other hand, still requires extensive training. Additionally, these desktop and pendant user interfaces imply a high cognitive and attention-related workload for the user

[2] new.abb.com/products/robotics/en/robotstudio
[3] fanucamerica.com/products/robots/robot-simulation-software-FANUC-ROBOGUIDE
[4] robodk.com

due to a continuous switching of visual attention between the robot and the user interface [2].

Many approaches for simplified robot programming have been proposed throughout the past years. To allow end-users to program robots, some used variations of visual programming [12–15], programming by demonstration (PbD) [16], tangible programming [17, 18], natural language interface [19], and some explored programming directly in the robot's space using AR [20–26]. The published works usually differ in the type of device used for the interaction and the level of robot programming. Some of them used a head-mounted display (HMD) to program the robot by setting trajectory waypoints [22–24]; others used projected spatial AR [21], visual programming in combination with visualization of spatial waypoints in the workplace [26], or an HMD in combination with a handheld pointer [25]. Apart from robot programming, AR has been found useful for visualizations of robot programs and motions [27], inspection and maintenance [28], or training [29, 30]. Moreover, AR can display the visual content directly in the working space, in one's line of sight, which reduces the user's cognitive load when switching the context and attention between the robot and an external device [31].

Recently, frameworks such as Google ARCore[5] or Apple ARKit[6] enabled fast and easy development of AR applications for smartphones and tablets, which are in general significantly more affordable than HMD devices, and well known by users. Both deliver mandatory functionalities for AR using their closed-source implementation of Visual-Inertial Simultaneous Localization and Mapping [32–34], and both have their strengths and weaknesses [35]. However, with their current implementation, they are usable for simple, small-scale environments [36] and non-complicated use-cases only, as hologram drifting can often rise to above 30 cm in challenging scenarios [4]. The use of these AR frameworks is suitable for visualization and interaction tasks but not for the precise input of spatial information per se. If there is a need to input spatial information with high accuracy, AR should be used in combination with another technology, e.g., kinesthetic teaching.

3 Spatial Programming Paradigm and UI

The two crucial parts of typical robot programming are the specification of individual program steps, i.e., what should happen, and the precise definition of spatial information, i.e., where it should happen. Depending on the programming method and selected level of abstraction, the first or latter could be derived automatically by the system (e.g., in imitation learning) or hidden from the user (e.g., when computer vision and robot motion planning are involved).

Both these parts are naturally related because most robotic actions use predefined or calculated coordinates. Many contemporary robot programming

[5]developers.google.com/ar
[6]developer.apple.com/augmented-reality/arkit/

tools represent spatial data in a way that is not natural for non-experts, such as textual coordinates. For non-expert to understand the spatial dimension of a robotic program, more than just source code is required. When a 3D environment model is available, a visualization of important spatial parameters (points in space or robot trajectories) could be made. Unfortunately, the quality of the environment model heavily influences the immersion of the visualization (low-quality models could be ambiguous or vague). Moreover, the visual representation of spatial information is usually separated from the action definition in the above-described example, as the visualization of waypoints occurs in a 3D scene in one window, and the source code is presented in another window. To understand the program and its spatial meaning, the programmer needs to merge these two pieces of information mentally.

In the case of robotic programs, not even the source code could provide insight into the program's logic. Many robotic programs consist of just three types of instructions: move instructions, end-effector manipulation (open/close gripper, turn on/off suction), and IO control. The code could be tough to read and understand without properly naming methods or thorough comments. Some simulations using the 3D model of the environment could be utilized to overcome this problem. However, it suffers from the same challenges which had been already discussed, and preparation of such a simulation environment could be costly and time demanding.

We define Spatial Programming Paradigm for mobile AR devices. The concept namely defines:

- The program as a sequence of the actions in the 3D space.
- The effective usage of AR for visualization and interaction with virtual objects in the 3D space.
- The interaction modes for seamless program editing in the 3D space on mobile devices.
- Methods for direct and indirect virtual object fast, resp. precise manipulation in the 3D space.

The defined spatial programming paradigm is applied for robot programming task that is a suitable scenario for the proposed paradigm, and we use it to explain and test the paradigm. The paradigm introduces the *Spatially Anchored Actions* (SAA), which utilizes specific 3D elements to visualize spatial information in the task space for development and program execution. These elements serve as anchors for actions (program steps), meaning that users can directly see where the individual actions of the program take place during the execution. The effective usage of AR is designed with respect of existing guidelines focusing on mobile device GUI ergonomy.

3.1 Spatially Anchored Actions

The proposed approach is based on flow diagrams and represents the robotic program as a sequence of individual actions connected to the program flow. *Anchored actions* represent the individual program steps (see Fig. 2). The *anchored actions* are connected using the *connections*, and in terms of flow diagrams, the *anchored actions* are nodes of the graph, representing the program, while the *connections* are the edges of the directed acyclic graph.

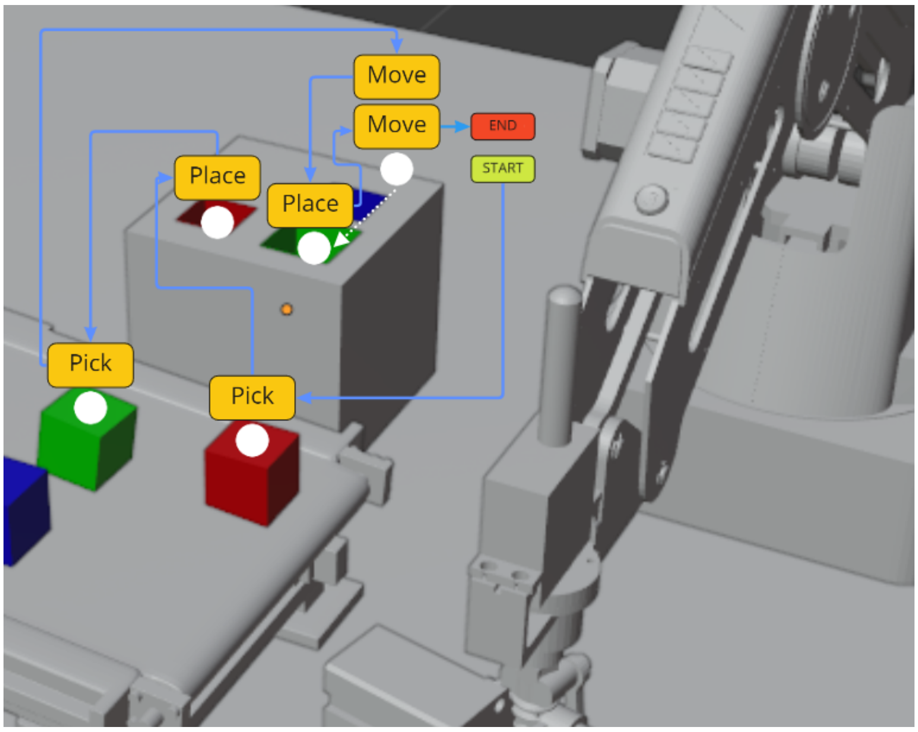


Fig. 2: The visualization of the *Spatially Anchored Actions* (SAA) concept. The white circles denote the *spatial anchors*, which serve for both the definition and visualization of spatial information. Above each *spatial anchor* is located one or more *actions*, represented by the yellow rectangle. The individual *actions* are connected by the blue lines, defining the program flow. Two *anchors* are connected by the white dotted line representing that the upper *anchor* is positioned relatively to the lower *anchor*.

Each *action* is anchored to one of the *spatial anchors*, representing the spatial information, as stated above. Using the AR, the *spatial anchors* are rendered on the exact place where the *anchored action* will take place, i.e., the *action* intended to pick a cube is located above said cube. This concept combines the spatial meaning of programmed action with its spatial parameters, which is crucial for robotic programs. Moreover, a single *spatial anchor* could serve for more *actions*, simplifying modification of joint *actions* (such as objects picking and placing on the same spot) and potentially enhancing the program comprehension. The *spatial anchors* could be attached to so called *scene objects*, which are virtual counterparts of real objects in the scene. This enable the user to define some spatial parameter relatively to the real objects.

The *spatial anchors* represent either specific points or poses in space. To visualize a specific point, a simple sphere that is natural for the observer is sufficient. To visualize a pose, the model of the end-effector, with a specific orientation applied, could be used.

3.2 Interaction modes

To enable fluent interaction with minimal interface overhead, the proposed user interface introduces so-called Interaction modes. Based on the current interaction mode, only relevant tools are available for the user so that they can focus on the current task and are not disturbed by an unnecessary on-screen interface. We propose five principal interaction modes.

The **execution mode** enables the user to execute selected *action*. The **transform mode** opens the transform menu over the selected *scene object* or *spatial anchor*. The **remove mode** enables the user to remove the selected *connection*, *action*, or *spatial anchor*. The **connection mode** allows the user to create arbitrary *connections* between two *actions*.

The **programming mode** allows the user to create program *actions* and *spatial anchors*. Its effects vary based on the selected object. When triggered, a context menu within the task space is opened, and the user can select desired *action* to be created. Once the *action* is selected, a new *spatial anchor* is created at a certain distance from the tablet in the forward direction and the *action* is attached to this *anchor*. Moreover, a *connection* is created automatically from the previous *action*. The *transform mode* is triggered afterward so that the user can specify the position of the new *spatial anchor*. The procedure differs slightly based on the currently selected object:

- Existing spatial anchor: the new *action* is created and attached to the existing *spatial anchor*, and the *transform mode* is not triggered.
- Existing action: the new *action* is created and attached to the existing *spatial anchor* to which the selected *action* is attached, and the *transform mode* is not triggered.
- Scene object: the newly created *spatial anchor* is set relatively to the *scene object*, so when the user moves with the *scene object* (using the *transform mode*), the *spatial anchor* moves the same way.
- Connection: the newly created *action* is inserted in the program flow between the two *actions*, connected by the selected *connection*.

3.3 Ergonomy of the user interface

Most applications nowadays (including some AR/VR apps) use WIMP[7] to interact with the user. In AR applications, it usually means that most of the interaction is made using some "head-up" displays, which causes constant context switching, where the user observes the scene for some time, then looks at the head-up menu to interact, then looks into the scene again and so on. To avoid this, we followed the design guidelines for UI elements in AR applications,

[7]Windows, Icons, Menus, Pointer

as defined by the authors of ARCore framework[8]. The main outcomes for our user interface are:

- Move most of the interactive actions and feedback information directly in the scene to minimize the head-up interaction.
- Make the necessary interactive elements (buttons, sliders, etc.), which would be inconvenient to have in the scene, large enough and place them in fixed, foreseeable places, so they could be easily remembered and quickly reached without the need to look at them.
- Help user to recover from missteps end errors by utilization of notifications displayed in the scene in front of the camera, so the user sees it comfortably.

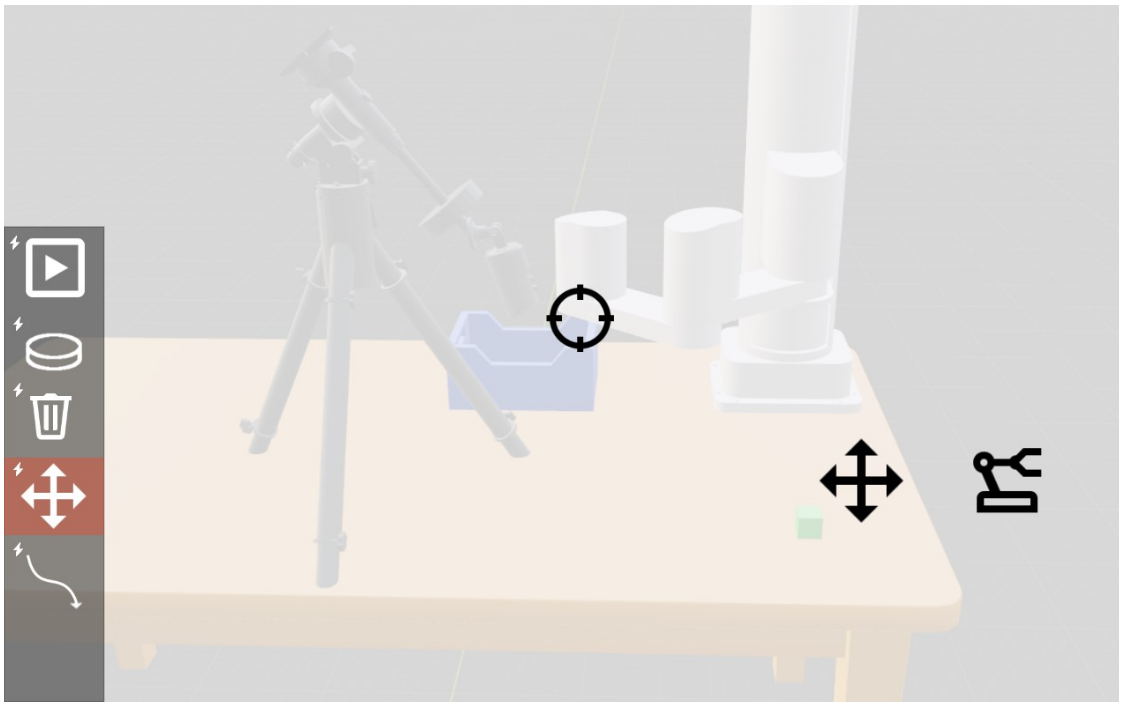


Fig. 3: Schematic visualization of the user interface. The left side contains the main menu allowing the user to select the appropriate interaction mode. In the middle is a crosshair for indirect virtual object selection. On the right side are two context-aware fixed *mode buttons*, easily reachable by the user's thumb.

The proposed user interface's layout is presented on Fig. 3. It consists of three parts. The left part contains the main menu, allowing the user to select one of the five interaction modes. The central part of the interface shows the scene image obtained from the camera. Additionally, a crosshair is placed in the middle of the screen, serving as a main virtual object selection tool. The right side contains two fixed buttons. The left one is the so-called *mode button*, whose appearance and function differ based on the currently selected interaction mode. The right one serves to relax the robot joints in order to allow

[8]https://developers.google.com/ar/design/interaction/ui

the operator to manipulate the robot arm. Both buttons are large enough and placed in the foreseeable place, according to the guidelines mentioned above.

The buttons have no textual labels to save space and make the interface as minimal as possible. The help for each icon is shown upon the long button press, and a training session is expected prior to usage of the interface.

3.4 Precise programming in AR

The main drawback of using AR is the low accuracy of camera tracking when using standard devices (such as cell phones or tablets). In other words, using just an AR device to specify an exact point in space is virtually impossible, as the tracking error might reach tens of centimeters [4]. On the other hand, when it comes to robot programming, there usually is a very precise device available for point specification – the robot itself. The robot could be used for the exact definition of points in space. The problem with this approach lies in the visualization of the created program and the synchronization of the robot with other devices used in the program.

Our approach utilizes the robot's precision to specify certain places in the environment, which serve as **reference points**. Interaction widgets could be used to precisely define several **relative points** using the imprecise AR visualization using these reference points (see Fig. 2). The "parent" *anchor* is set using a precise method (i.e., manual guiding of robot or using computer vision techniques). Other *anchors* are set using a combination of 2D and 3D widgets with selectable precision (see Fig. 4). We assume that, for understanding the program using its visualization in AR, the absolute precision (the correlation between the rendered virtual element and its actual position in the real environment) is not as important as the mutual relative precise position of virtual elements defining the program.

3.5 Transforming Spatial Anchors

The crucial interaction task is a manipulation with the *spatial anchor* in a real 3D task space. The proposed concept introduces direct (fast, but low precision) and indirect (slower, but precise) manipulation with the objects, i.e., *spatial anchors* or *scene objects*. Direct manipulation utilizes the physical movement of the handheld device. The *transform menu*, displayed on Fig. 4, contains a palm-shaped button for direct manipulation – when pressed, the object moves with the device's movement, allowing fast movement over large distances.

We propose an indirect manipulation for higher precision in setting the spatial parameters. The *rotary transform element* is placed on the right side of the *transform menu*, which allows moving the virtual object by scrolling the element. The numbers represent the number of steps by which the object will be moved. The *magnitude selector* under the *rotary element* selects the length of the step. Together, it allows to move the object by the exact length. On the bottom are two buttons to change between the translation and rotation.

The user needs to see and select the direction in which the virtual object will be moved. We propose a 3D *gizmo* (see Fig. 4) for both cases. The *gizmo* consists of three perpendicular arrows representing the direction of the desired movement, and it is attached to the virtual object selected for manipulation. Close to the tip of each arrow, a current displacement from the original position is visualized. The desired direction of movement is indicated by selecting one of the arrows using the cross-hair.

In the left part of the *transform menu* are several buttons with an additional functionality. The arrows in the top serve for undo and redo operation. Bellow the palm-shaped button is the so-called *pivot* button. This button causes the object to move on the position of another object selected using the cross-hair. Using this button, the user can, for example, move a *spatial anchor* on the position of aforementioned **reference point** and subsequently define a **relative point** using the *rotary element*.

4 Experimental Evaluation

The primary motivation for this work is to introduce a novel approach to end-user robot programming. The method was implemented into a functional prototype, and a user study was carried out to compare it with a traditional approach for end-user programming on a 2D screen. The experiment was

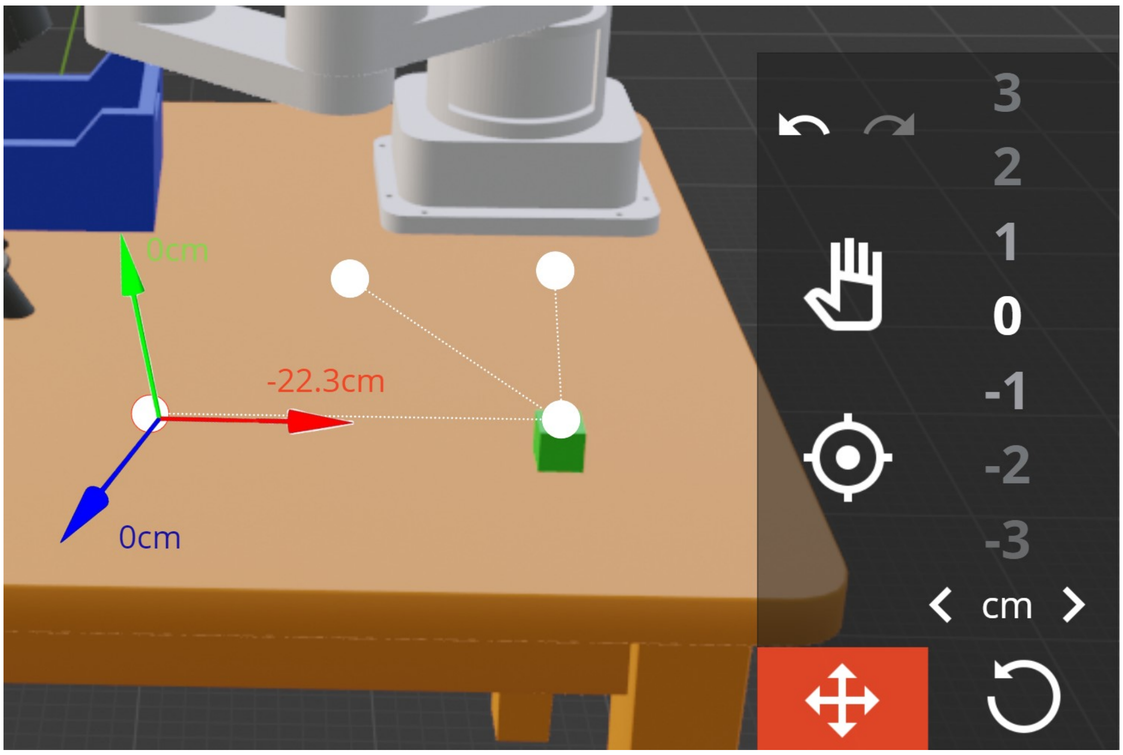


Fig. 4: The schematic visualization of the tools available in the *transform mode*. The left side contains the 3D widget, so-called gizmo, rendered over the manipulated object. The right side contains the *transform menu*, with several interactive elements.

designed as a within-subject, with two conditions, where C_1 is the proposed prototype, and C_2 is a Blockly-based tool in the Dobot M1 Studio environment. We have stated four hypotheses related to the objectives above:

- H_1 – The user is faster acquainted with the program, seen for the first time, using the C_1 interface.
- H_2 – The C_1 interface is more usable than C_2 and puts less task load on the user.
- H_3 – The user can create a new program faster using the C_1 interface than the C_2 interface.
- H_4 – The C_1 interface provides similar precision for selected task as the C_2 interface.

The following chapter presents a user study we have prepared and conducted, which will help us to support or reject the stated hypotheses.

4.1 Prototype

A functional prototype[9] was prepared for the experimental evaluation, containing basic functionalities for programming of pick & place-like tasks. The prototype application was developed in the Unity3D game engine, using the AR Foundation framework[10], which encapsulates the Google's ARCore[11], for AR-related parts. The application is designed to run on Samsung Galaxy Tab S6 or S7, a 10" Android tablet device compatible with the ARCore.

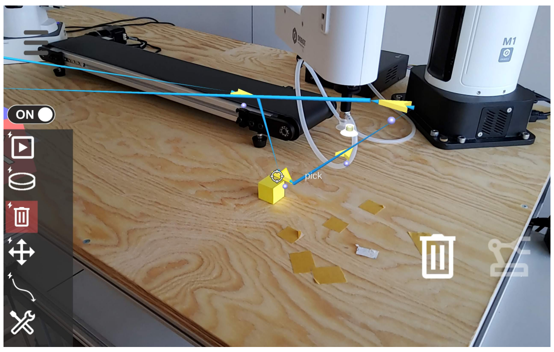
The prototype is designed as a non-immersive AR-enabled application, following the guidelines described in the Section 3.3. The SAA (see Fig. 5) are visualized as yellow arrows located above blue spheres. The spheres represent spatial anchors, anchoring the actions for visualization and execution.

The prototype is fully functional, except for the object aiming procedure, which allows the user to set a precise object's position and orientation by navigating the robot's end-effector into several specific points on the object's body. In the experiment, this procedure was utilized to define the position of the workpiece. However, it was done using the wizard of oz approach for the sake of the experiment, which was unknown to the participants. Besides that, the participants interacted with a real, functional robot and created a robotic program from scratch.

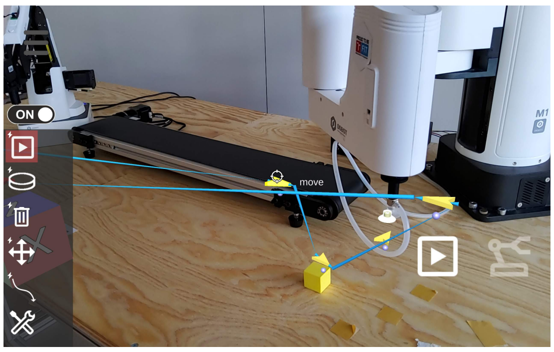
4.2 Experiment design

The experiment was designed as a within-subject user study, comparing the two different interfaces – our prototype interface based on presented SAA (C_1) and the standard programming tool for the Dobot M1 robot – M1 Studio (C_2) with the Blockly tool. Both selected interfaces utilize visual programming and contain specialized elements for robot manipulation.

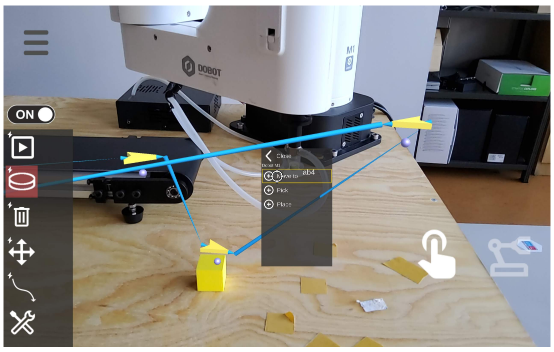
[9]Source code is available at github.com/robofit/arcor2_areditor.
[10]docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/manual
[11]developers.google.com/ar



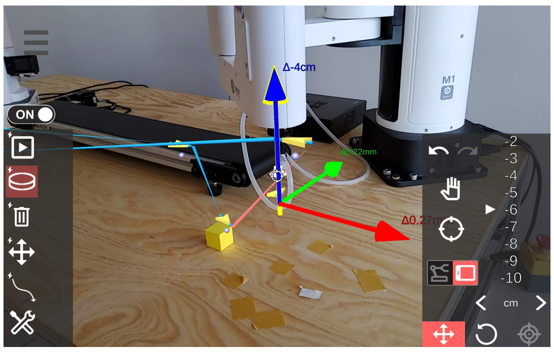
(a) The remove mode. The user could delete any virtual element by pressing the *mode button* when an object is selected.



(b) The execution mode. The user could execute any Action by pressing the *mode button* when an Action is selected.



(c) The programming mode. After pressing the *mode button*, an Action selector menu appears in front of the user, and they can select the Action to be created by selecting it with the crosshair and pressing the *mode button*.

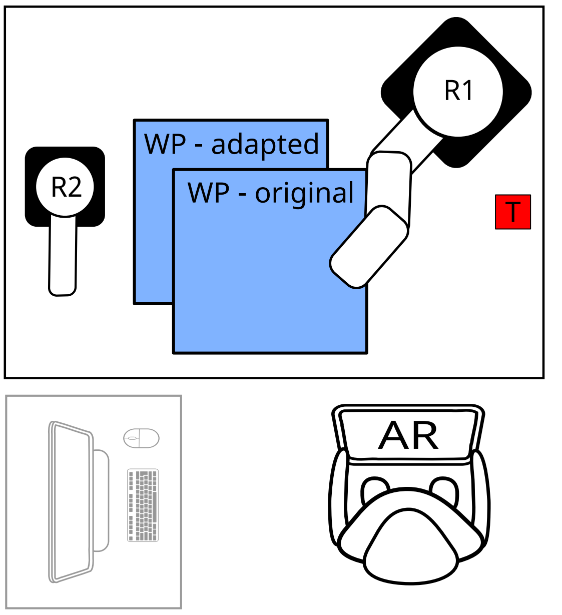


(d) The transform mode. The transform menu is placed on the right side, allowing the user to manipulate the selected virtual object using the scrollable rotary element. The transform axis is selected using the crosshair on the transform gizmo (in the center of the screen). The gizmo shows the offset from the object's original position.

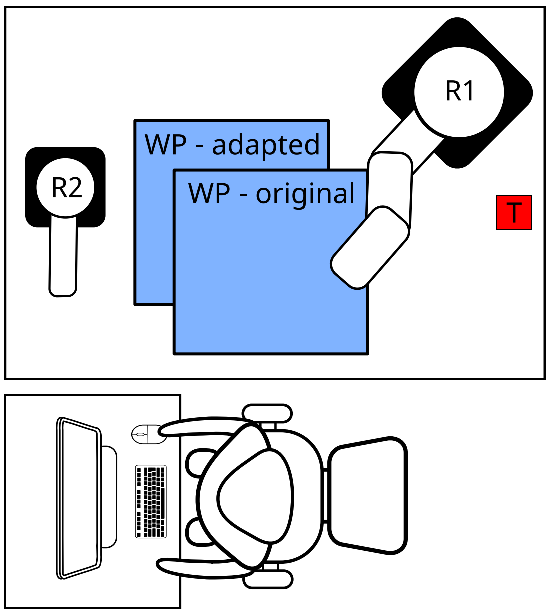
Fig. 5: Graphical user interface of the prototype application. The left side contains the main menu for mode selection. On the right side is placed either *mode button* (a-c), depending on the selected mode, or the transform menu (d) in case the object is being moved. The central part serves for viewing the scene with the superimposed interface.

C_1 utilizes a custom mobile AR application for visual programming in task space, based on the presented method of SAA described in the previous chapter. The participant was standing in front of the table and could interact with the workplace from the front and right side of the table (see Fig. 6a).

C_2 uses an application for desktop computers with the Google Blockly framework for visual programming, where the user combines special puzzle-shaped boxes into a functional program. These blocks represent instructions such as `MoveJoints`, `SetArmOrientation`, etc. The parameters for each block are defined using either the keyboard or, in the case of move-blocks, by physical movement of the robot into the desired position. The participant was sitting on a chair by the table equipped with a computer screen, mouse, and keyboard in front of the workplace (see Fig. 6b). They could reach the robot from the



(a) Scheme for C_1. The person stands in front of the workplace and holds the tablet. The workplace is accessible from the front and the right side. The computer is present but not utilized in this condition.



(b) Scheme for C_2. The person sits in front of the computer, which is located in front of the workplace. They can reach the robot from the chair as well.

Fig. 6: Workplace scheme for both conditions. The *R1* is the main robot, the Dobot M1. The *R2* is an additional robot, Dobot Magician, which was utilized only in the *visualization* task. The red square is the original position of the object the *R1* should pick and manipulate. The blue squares represent the *workpiece* in two positions, the original and the adapted.

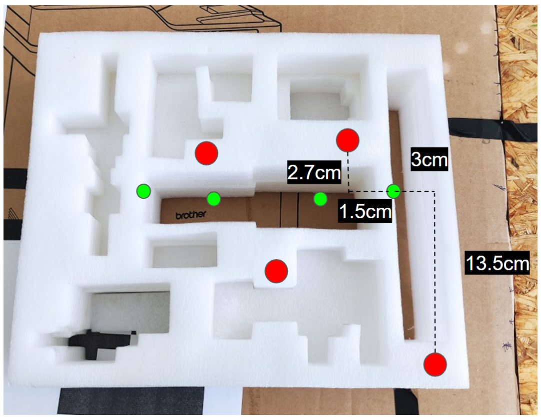
chair as well. They were allowed to stand up if they required better robot handling. The workplace was accessible from the front and right sides.

To minimize learning and transfer bias caused by the study being designed as a within-subject, the order of both conditions is randomized for each participant. For the safety purposes of both robot and subjects, each participant was thoughtfully instructed on how to control the robots safely, the maximal velocity and acceleration of the robots were lowered to safe levels, and robots without sharp edges were selected for the study. The manipulated objects were small cubes made of foam to minimize the potential risk of injury.

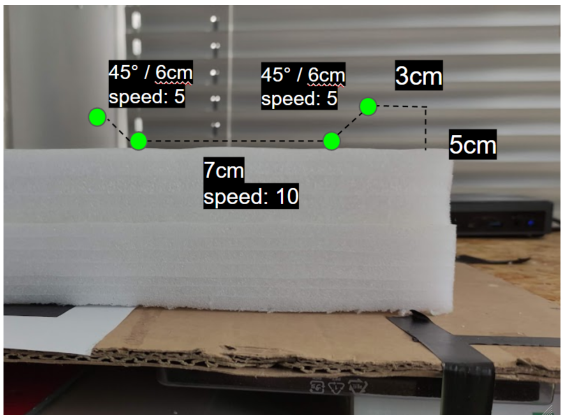
4.3 Experiment protocol

Each experimental run was organized as follows. At first, the moderator welcomed the participant and asked them to sign an informed consent and fill in a demographic questionnaire. After that, a brief workplace introduction took place.

The moderator randomly assigned the first condition to the participant and introduced them briefly the programming tool, and after that, the participant began with the training task (T_1). The participant was told to program the robot to pick a foam cube from the table and put it inside the box. The



(a) The position of reference points (the red circles) and the trajectory points (the green circles). The first trajectory point's position is referenced with respect to the reference points.



(b) The trajectory points (green circles) and their mutual positions.

Fig. 7: The drawing of the intended trajectory for the main task, superimposed over the workpiece used for the experiment.

created program was to be subsequently modified, so the robot followed a specified path before the cube releasing (the path was defined as a 10 cm line under the 45° angle, ending at a specific point on the bottom of the box). During this phase, the moderator proactively helped the participant with the programming, explained the required functionality, and answered all questions.

Following the training, the visualization task (T_2) took place. An existing program was presented to the participant. Their task was to identify some program steps according to the moderator's questions. The participant was explicitly informed that they could use anything the user interface offers, namely, the ability to run the program, program steps, or move the robotic arm. The presented program differs for both conditions, so the participants were not influenced by previous knowledge of the presented program. Both programs involved the pick & place task with various objects and the usage of the conveyor belt. All questions for both conditions are to be found below.

Questions for C_1: Find the action, which causes...

1. the bigger robot to pick the box from the conveyor belt.
2. the smaller robot to pick the cube from the table.
3. the conveyor belt to shift from the bigger robot to the smaller one.
4. the bigger robot to pick the box from the table.

Questions for C_2: Find the action, which causes the bigger robot to...

1. pick the yellow cube from the table.
2. place the red cube on the table.
3. move the green cube above the conveyor belt.
4. pick the blue cube from the table.

Next, the main task (T_3) was presented to the participant. It simulates precise robotic manipulation with workpieces in a structured environment.

Specifically, the robot should pick a cube and perform simulated grinding by following a specific trajectory defined by a technical scheme (see Fig. 7), which was available to the participant during the whole session. The scheme contains the position of each waypoint and the speed of the end-effector's movement between two consecutive waypoints. Lastly, the robot should put the cube back in the original spot on the table. The experiment task was the same for both conditions. For the C_1 condition, the participant had to annotate the position of the workpiece first as a part of the T_3 so that they could utilize its reference points afterward. The procedure consisted of setting the position of four reference points on the workpiece (the red circles at Fig. 7a) using the hand movement of the robot. Once the annotation was done, the reference points were automatically added to the scene as spatial anchors. The participants were told to define other anchors relative to the reference points.

After the moderator answered the questions, the participant started to work. The participant was allowed to ask questions during this phase, and they were noted and categorized by the context of the question (i.e., if they were related to the task or the programming tool).

When the T_3 was successfully programmed by the participant, the moderator moved the simulated workpiece to the new place, and the participant had to adapt the program (T_4). In the case of C_1, it meant only annotating the position of the workpiece again, as all related spatial anchors were defined relative to the workpiece's reference points. For the C_2, setting a new position for all the waypoints needed to be done again. For simplicity, the participants were told only to set the first waypoint.

During all the tasks, the participant was allowed to test the execution of both individual actions or the whole program. When the participant claimed that they thought the program was completed, the moderator observed and executed the program to check its functionality. In the case of problems, the moderator suggested what needed to be altered, and the participant was supposed to correct the program.

Once all four tasks were done with the first condition, the participant was supposed to fill in questionnaires regarding the current condition. After that, the same procedure was conducted using the other condition. In the end, an open discussion took place. The moderator asked the participant for their impressions, additional questions, and opinions.

4.4 Dependent Measures

As an objective measurement, the completion time was selected. This time is computed for each task separately so that we can compare the duration of each task individually for both conditions. As a subjective metric, standard questionnaires were selected. Namely, the NASA Task Load Index [37] for measuring mental and physical load, and the System Usability Scale [38] to rate the usability of the prototype interface. Besides these standard questionnaires, evaluated independently for each interface, another one containing specific questions regarding the prototype interface was utilized. Moreover, for the C_1

condition, the HARUS questionnaire [39], which is explicitly designed for the usability of handheld AR interfaces, was incorporated.

5 Results

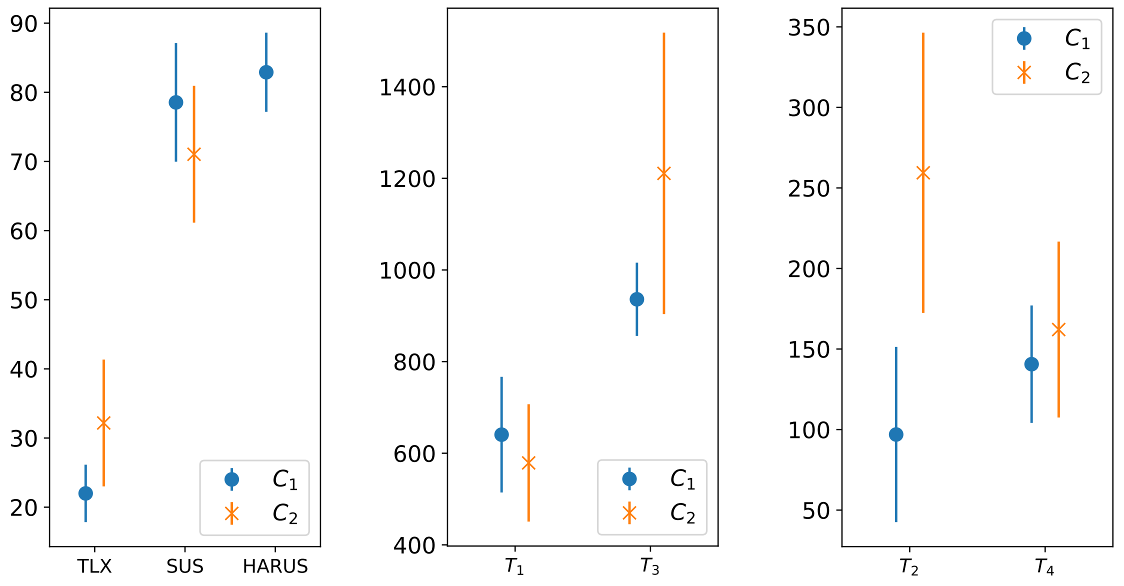
This section summarizes the user-study results and provides its analysis and interpretation. Regarding the task completion time measurement, intervals where participants asked questions, a technical problem occurred, or when the moderator had to intervene, were subtracted to measure a pure task completion time. All statistical tests were done at the 5 % significance level. Data were first tested for normality (combination of D'Agostino and Pearson's tests), and based on the result, paired t-test or Wilcoxon's signed-rank test were used to test for the significant difference between conditions.

The user study was conducted with 12 subjects of various ages, self-reported genders, and technical backgrounds. Eleven participants identified themself as males; one identified themselves as female. Most subjects are ordinary shop-floor workers, students, or graduates from humanities colleges with little or no prior experience in programming. One participant works as a programmer, and one works as a robot operator. They reported their experiences with robots on average 2.17 (on the scale of [1 .. 5], where higher means more experienced), experiences with AR on average 2.25, and experiences with programming on average 2.08. Each participant signed informed consent to data recording and its usage for evaluation and eventually propagation in anonymized form. Some participants reported eye defects, such as myopia or amblyopia, but none reported that they affected them during the experiment. The user study took place in a lab-like environment in a dedicated room, where no external factors could influence the process of the experiment. All participants were able to finish all the tasks using both conditions.

5.1 Quantitative and Qualitative Data

Results from SUS and NASA-TLX questionnaires (shown in Fig. 8a for both conditions) show that, on average, the participants perceived a lower task load using the C_1 interface and ranked it as more usable. The mean TLX score for the proposed SAA interface (C_1) was 21.99, which is less than 32.18 for the C_2. Regarding the usability of the interfaces for both conditions, the SUS questionnaire results show that participants consider the interface from C_1 more useful, scoring 78.54, compared to the C_2, scoring 71.04. However, differences are not significant for both metrics according to paired t-test ($p = 0.074$ for TLX, $p = 0.312$ for SUS); therefore, the H_2 can not be confirmed. Besides, the C_1 was scored 82.90 using the HARUS method, which is specifically designed to measure the usability of handheld AR systems. The score is higher than that of comparable interface SlidAR [40], which is aimed at virtual object manipulation and scored 76.3 (SD=10.83).

The training time (T_1) was comparable for both interfaces, although slightly longer with the C_1 interface (see Fig. 8b). Contrary, the main task



(a) The subjective measurements. For the TLX, the lower means better, for SUS and HARUS, the higher means better.

(b) Time (in seconds) needed to complete T_1 (training) and T_3 (main) tasks.

(c) Time (in seconds) needed to complete the T_2 (visualization) and T_4 (adaptation) tasks.

Fig. 8: Comparison of subjective and objective measurements (mean values and corresponding 95 % confidence intervals) for conditions C_1 (proposed method) and C_2 (standard method).

(T_3) was significantly faster with the C_1 interface according to the Wilcoxon test ($p = 0.042$); therefore, the H_3 was confirmed.

In the adaptation phase, the users were told to:

- complete the aiming procedure for the workpiece in the new position for C_1 condition,
- set the position of the first point of the trajectory for C_2 condition.

The completion times in Fig. 8c show that the adaptation using the C_1 condition was significantly faster even when the participants did not adapt the whole trajectory in the C_2 condition, showing that the gap will get even wider with the increasing amount of points in the trajectory.

Analyzing the completion times for the T_2 (visualization task), it was shown that for the C_1 condition, the participants required significantly less time to answer the questions (see Fig. 8c). This suggests that the AR interface greatly supports the user in program comprehension, especially for the actions with the spatial information, which are crucial for robotic program understandability; therefore, the H_1 is confirmed. The discussion with the participants showed that they felt more certain when they had to identify the program steps using the SAA presented in AR. Most of them could identify each step quickly by just looking over the scene and benefit from the fact that most of the program steps are represented by 3D objects placed on the spot where the action should take place. The only problem occurred when they had to identify the step causing the shift of the conveyor belt (third question within the C_1 condition), which has no clear spatial information. Most of the participants could identify it after a short time, which shows that the users can identify even actions without clear spatial information using the proposed interface. When using the M1 studio interface (C_2), most users did not utilize the ability to run the program (although they were explicitly remarked that they might run it). Instead, they used the robotic arm to estimate the spatial coordinates of each program step to identify them. This strategy was highly successful but very time-consuming. The participants, on average, needed 1.17 attempts (SD: 0.38) to identify the correct action for the C_2 interface and 1.2 attempts (SD: 0.45) for the C_1 interface, but over a significantly longer period of time.

5.2 Preferences

According to the questionnaire of the C_1 interface, the vast majority, specifically 64.3 % of participants, preferred the rotary control element for the precise movement of virtual objects. Both setting using the robot manipulation and the free-form setting using the tablet motion were preferred by 16.67 % of participants. On the other hand, for setting the coordinates where the approximate position is sufficient, 50 % preferred using the robot manipulator, 33 % preferred the free-form setting using the tablet motion, and 16.7 % preferred the rotary control element.

According to the questionnaire regarding the C_1 interface, the participants considered the rotary control element more useful than the free-form movement of the virtual objects (see Fig. 9). We argue that it is primarily because of the selected task, as it required settings of several precise positions. In contrast, the setting of non-precise positions was unnecessary, and the participants utilized it only for a couple of intermediate movements.

The participants also liked the inserting of new spatial anchors on the current position of the robot's end-effector (see Fig. 9) than freely to the space (in front of the tablet). We argue that this is because of a higher level of certainty, as the participants knew precisely where the spatial anchor would be placed and that the robot would be able to reach that position. As the participants had to set a path for the robot based on specification, they usually followed this pattern:

1. Set a waypoint.
2. Create a new waypoint on the position of the previous waypoint.
3. Move the new waypoint in a certain direction.

To achieve this pattern, the user had to create a new waypoint freely in the space (or at the position of the robot) and then use the *pivot* functionality (described in Section 3.5), which sets the position of the waypoint to another virtual object (previous waypoint in this case). Most of the participants struggled a bit on this at the beginning, and they would appreciate, according to our observations and discussion with them, the possibility of adding a new

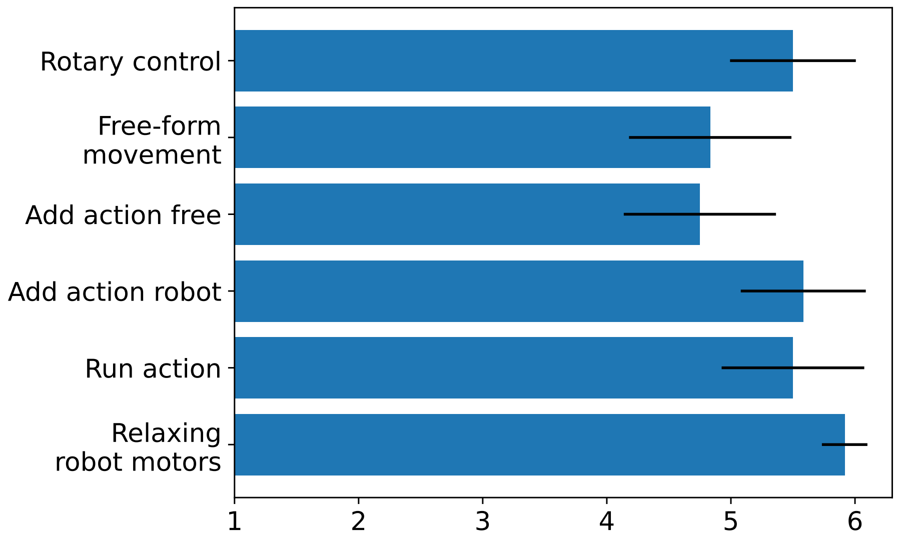


Fig. 9: Usefulness of selected features of C_1 interface, rated by the participants on a scale from 1 (useless) to 6 (very useful).

spatial anchor to an existing one, similar to adding it to the position of the robot's end effector.

Most users considered the robot motor's unlock button very useful. However, some did not like the dead-man-trigger concept, as they reported that it is hard to press that button while holding the tablet with one hand. Moreover, it was difficult for some of them to move the robot with one hand only.

5.3 Observations

The participants generally liked the possibility of quickly executing *actions*, using the C_1 during the T_3, as it enabled them to check the reachability of the spatial anchors. With the C_2 interface, the participants were using the execution of individual actions more often, as they were using it also for identification of the actions in the programming tool, which was not needed in the C_1 interface because they saw the position of the spatial anchor in the AR.

Several participants reported that at the beginning, they were stressed out by the C_1 interface, mainly because of a rich set of functions, compared to the C_2 interface. Moreover, they claimed that the 3D interface elements were entirely new to them, and it took some time to get used to them. Nevertheless, most of them agreed that after a short time, they got used to the controls and the programming was easier than with the C_2 interface, despite their initial concerns.

For the task T_2, all but one participant preferred the C_1 interface. They claimed the spatial distribution of individual actions in task space helped to distinguish the *anchored actions*. One participant stated that they could quickly orient themselves because of the spatial visualization in C_1. The other claimed that spatial visualization hugely helps them to identify which "pick" action is the one they are looking for, although they look the same.

Only two participants preferred the programming using the C_2 interface over the C_1. Both are rather technically oriented people; one works as a junior robot programmer (using RoboDK software), and the other has a background in CNC programming. The latter claimed that the visualization task was also easier for him using the C_2 interface. Both of them stated that the C_2 interface was more straightforward for them and reminded them of the tools they are or were using at their jobs.

All participants struggled with the visualization and control of the gizmo element in C_1. They were often unsure which axis was selected or accidentally selected the wrong one. Many participants struggled with the magnitude of the transform step selection, causing them to either move the object at the wrong length or wonder why the object is not moving because it only moved by several millimeters instead of centimeters. The transform widgets must be enhanced to provide better feedback for the operator of both the magnitude and direction of the desired movement.

Most participants considered the blue lines between the individual actions in the C_1 interface to be the robot's trajectory, although they were explicitly informed during the training that the blue line only indicates the order of the actions.

In the C_2 the users can modify the coordinates in textual form with virtually unlimited precision. The C_1 preserves the possibility to set the position with a selectable degree of precision in a graphical way, utilizing the 2D and 3D widgets with user-defined coordinate systems. The participants finished all tasks using both interfaces, which required setting several precise spatial parameters. Therefore, we consider the H_4 to be confirmed.

6 Conclusions and Future Work

This paper presents a novel paradigm for spatial programming in AR on mobile devices. The paradigm defines Spatially Anchored Actions for program visualization, their manipulation in real 3D space, and UI elements and rules for interaction in AR on mobile devices. The new concept was introduced and tested on a robot programming task. A fully-functional prototype was created using a tablet-like handheld device, which was evaluated with 12 potential users and compared to the existing visual programming method. The study revealed that the SAA concept significantly helped the participant's comprehension and understandability of the robotic programs, which correlates with the research objective RO_1. All participants successfully finished all tasks using both interfaces at a similar time; therefore, it was shown that the simplicity of program creation is similar to the standard tool (RO_2). We also aimed to lower the users' task load (RO_3). The study did not reveal any significant task load reduction, which was relatively low for both tested conditions. A higher number of participants could show significant differences. One of our objectives was to provide good ergonomics for the mobile AR interface (RO_4). To do so, we have designed the user interface to be controlled by users' thumbs, enabling them to hold the tablet in an ergonomic position.

Moreover, we moved most of the interaction elements from the on-screen menus to the 3D scene, allowing for lower context switching between the user interface and the visualization of the scene. We have also proposed several 2D and 3D widgets, allowing precise specification of spatial information using the AR (RO_5). The users could finish the task with similar precision in both conditions.

In future work, we plan to investigate some drawbacks revealed by the study. The 3D gizmo widget for axis selection was sometimes unclear for the participants as they were unsure which axis was selected or what distance / angle magnitude was currently selected for transformation. To check if the set spatial anchor is reachable by the robot, the participants had to execute an action attached to the anchor. It would be beneficial to visualize the reachability more clearly. We would also like to investigate more the possibilities for the robot motor's unlock button, as the dead-man-trigger concept causes trouble to the participants, forces them to hold the device in a non-ergonomic way, and causes troubles with robot manipulation. Moreover, we will evaluate the feasibility of the proposed concept in different contexts than robot programming in SMEs such as home automation, where there is also high demand for end-user programming techniques and, at the same time, a need to set spatial parameters as, e.g., the definition of various kinds of zones.

Code availability The source codes are available for download through GitHub.

Data availability The datasets generated during the current study are not publicly available due to the participants' privacy but are available from the corresponding author on reasonable request.

Funding This work has been performed with financial support from the state budget through the Ministry of Industry and Trade in the TRIO program (FV40052).

Declaration

Consent of participate Available, signed by each participant.

Conflict of interest There are no conflicts of interest/competing interests.

Ethical statement The manuscript complies with the Ethical Rules applicable for this journal as stated in the Instructions for Authors of the journal Virtual Reality.

References

[1] Ajaykumar, G., Steele, M., Huang, C.-M.: A survey on end-user robot programming. ACM Comput. Surv. **54**(8) (2021). https://doi.org/10.1145/3466819

[2] Weiss, A., Huber, A., Minichberger, J., Ikeda, M.: First application of robot teaching in an existing industry 4.0 environment: Does it really

work? Societies **6**(3), 20 (2016)

[3] Contero, M., Gomis, J.M., Naya, F., Albert, F., Martin-Gutierrez, J.: Development of an augmented reality based remedial course to improve the spatial ability of engineering students. In: 2012 Frontiers in Education Conference Proceedings, pp. 1–5 (2012). https://doi.org/10.1109/FIE.2012.6462312

[4] Scargill, T., Chen, J., Gorlatova, M.: Here to stay: Measuring hologram stability in markerless smartphone augmented reality. arXiv preprint arXiv:2109.14757 (2021)

[5] Morar, A., Băluţoiu, M.A., Moldoveanu, A., Moldoveanu, F., Butean, A., Asavei, V.: Evaluation of the arcore indoor localization technology. In: 2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet), pp. 1–5 (2020). IEEE

[6] Battegazzorre, E., Calandra, D., Strada, F., Bottino, A., Lamberti, F.: Evaluating the suitability of several ar devices and tools for industrial applications. In: International Conference on Augmented Reality, Virtual Reality and Computer Graphics, pp. 248–267 (2020). Springer

[7] ABB, R.: Technical reference manual: Rapid instructions, functions and data types. ABB Robotics (2014)

[8] Robot, U.: The urscript programming language for e-series. Universal Robot (2022)

[9] Ajaykumar, G., Huang, C.-M.: User needs and design opportunities in end-user robot programming. In: Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction, pp. 93–95 (2020)

[10] Schmidbauer, C., Komenda, T., Schlund, S.: Teaching cobots in learning factories–user and usability-driven implications. Procedia Manufacturing **45**, 398–404 (2020)

[11] Connolly, C.: Technology and applications of abb robotstudio. Industrial Robot: An International Journal (2009)

[12] Huang, J., Cakmak, M.: Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts. In: 2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI, pp. 453–462 (2017). IEEE

[13] Gao, Y., Huang, C.-M.: Pati: a projection-based augmented table-top interface for robot programming. In: Proceedings of the 24th International Conference on Intelligent User Interfaces, pp. 345–355 (2019)

[14] Paxton, C., Hundt, A., Jonathan, F., Guerin, K., Hager, G.D.: Costar: Instructing collaborative robots with behavior trees and vision. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 564–571 (2017). IEEE

[15] Mayr-Dorn, C., Winterer, M., Salomon, C., Hohensinger, D., Ramler, R.: Considerations for using block-based languages for industrial robot programming-a case study. In: 2021 IEEE/ACM 3rd International Workshop on Robotics Software Engineering (RoSE), pp. 5–12 (2021). IEEE

[16] Alexandrova, S., Cakmak, M., Hsiao, K., Takayama, L.: Robot programming by demonstration with interactive action visualizations. In: Robotics: Science and Systems, pp. 48–56 (2014). Citeseer

[17] Sefidgar, Y.S., Agarwal, P., Cakmak, M.: Situated tangible robot programming. In: 2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI, pp. 473–482 (2017). IEEE

[18] Sefidgar, Y.S., Weng, T., Harvey, H., Elliott, S., Cakmak, M.: Robotist: Interactive situated tangible robot programming. In: Proceedings of the Symposium on Spatial User Interaction, pp. 141–149 (2018)

[19] Fogli, D., Gargioni, L., Guida, G., Tampalini, F.: A hybrid approach to user-oriented programming of collaborative robots. Robotics and Computer-Integrated Manufacturing **73**, 102234 (2022)

[20] Blankemeyer, S., Wiemann, R., Posniak, L., Pregizer, C., Raatz, A.: Intuitive robot programming using augmented reality. Procedia CIRP **76**, 155–160 (2018)

[21] Materna, Z., Kapinus, M., Beran, V., Smrž, P., Zemčík, P.: Interactive spatial augmented reality in collaborative robot programming: User experience evaluation. In: 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 80–87 (2018). IEEE

[22] Quintero, C.P., Li, S., Pan, M.K., Chan, W.P., Van der Loos, H.M., Croft, E.: Robot programming through augmented trajectories in augmented reality. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1838–1844 (2018). IEEE

[23] Gadre, S.Y., Rosen, E., Chien, G., Phillips, E., Tellex, S., Konidaris, G.: End-user robot programming using mixed reality. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 2707–2713 (2019). IEEE

[24] Ostanin, M., Klimchik, A.: Interactive robot programing using mixed reality. IFAC-PapersOnLine **51**(22), 50–55 (2018)

[25] Ong, S.-K., Yew, A., Thanigaivel, N.K., Nee, A.Y.: Augmented reality-assisted robot programming system for industrial applications. Robotics and Computer-Integrated Manufacturing **61**, 101820 (2020)

[26] Yigitbas, E., Jovanovikj, I., Engels, G.: Simplifying robot programming using augmented reality and end-user development. In: IFIP Conference on Human-Computer Interaction, pp. 631–651 (2021). Springer

[27] Rosen, E., Whitney, D., Phillips, E., Chien, G., Tompkin, J., Konidaris, G., Tellex, S.: Communicating robot arm motion intent through mixed reality head-mounted displays. In: Amato, N.M., Hager, G., Thomas, S., Torres-Torriti, M. (eds.) Robotics Research, pp. 301–316. Springer, Cham (2020)

[28] Eschen, H., Kötter, T., Rodeck, R., Harnisch, M., Schüppstuhl, T.: Augmented and virtual reality for inspection and maintenance processes in the aviation industry. Procedia manufacturing **19**, 156–163 (2018)

[29] Barsom, E.Z., Graafland, M., Schijven, M.P.: Systematic review on the effectiveness of augmented reality applications in medical training. Surgical endoscopy **30**(10), 4174–4183 (2016)

[30] Werrlich, S., Nitsche, K., Notni, G.: Demand analysis for an augmented reality based assembly training. In: Proceedings of the 10th International Conference on PErvasive Technologies Related to Assistive Environments, pp. 416–422 (2017)

[31] Suzuki, R., Karim, A., Xia, T., Hedayati, H., Marquardt, N.: Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces. In: CHI Conference on Human Factors in Computing Systems, pp. 1–33 (2022)

[32] Liu, H., Chen, M., Zhang, G., Bao, H., Bao, Y.: Ice-ba: Incremental, consistent and efficient bundle adjustment for visual-inertial slam. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1974–1982 (2018)

[33] Taketomi, T., Uchiyama, H., Ikeda, S.: Visual slam algorithms: A survey from 2010 to 2016. IPSJ Transactions on Computer Vision and Applications **9**(1), 1–11 (2017)

[34] Terashima, T., Hasegawa, O.: A visual-slam for first person vision and mobile robots. In: 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA), pp. 73–76 (2017). IEEE

[35] Nowacki, P., Woda, M.: Capabilities of arcore and arkit platforms for ar/vr applications. In: International Conference on Dependability and Complex Systems, pp. 358–370 (2019). Springer

[36] Feigl, T., Porada, A., Steiner, S., Löffler, C., Mutschler, C., Philippsen, M.: Localization limitations of arcore, arkit, and hololens in dynamic large-scale industry environments. In: VISIGRAPP (1: GRAPP), pp. 307–318 (2020)

[37] Hart, S.G., Staveland, L.E.: Development of nasa-tlx (task load index): Results of empirical and theoretical research. Advances in psychology **52**, 139–183 (1988)

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

[39] Santos, M.E., Polvi, J., Taketomi, T., Yamamoto, G., Sandor, C., Kato, H.: A usability scale for handheld augmented reality. (2014). https://doi.org/10.1145/2671015.2671019

[40] Polvi, J., Taketomi, T., Yamamoto, G., Dey, A., Sandor, C., Kato, H.: Slidar: A 3d positioning method for slam-based handheld augmented reality. Computers & Graphics **55**, 33–43 (2016). https://doi.org/10.1016/j.cag.2015.10.013

ARCOR2: Framework for Collaborative End-User Management of Industrial Robotic Workplaces using Augmented Reality

Michal Kapinus, Zdeněk Materna, Daniel Bambušek, Vítězslav Beran, Pavel Smrž
Faculty of Information Technology, Brno University of Technology, Czech Republic
{ikapinus,imaterna,bambusekd,beranv,smrz}@fit.vut.cz

***Abstract*—This paper presents a novel framework enabling end-users to perform management of complex robotic workplaces using a tablet and augmented reality. The framework allows users to commission the workplace comprising different types of robots, machines, or services irrespective of the vendor, set task-important points in space, specify program steps, generate a code and control its execution. More users can collaborate at the same time for instance at a large-scale workplace. Spatially registered visualization and programming enable a fast and easy understanding of the workplace processes, while a high precision is achieved by a combination of kinesthetic teaching with a specific graphical tools for relative manipulation of poses. A visually defined program is for execution translated into Python representation, allowing efficient involvement of experts. The system was designed and developed in cooperation with a system integrator, based on an offline PCB testing use case and its user interface was evaluated multiple times during the development. The latest evaluation was performed by three experts and indicates the high potential of the solution.**

***Index Terms*—visual programming, augmented reality, collaborative robot, end-user programming**

I. Introduction

Most often, robots are used for highly repetitive tasks. For instance, in the automotive industry, the production line is programmed once and then works for several years without major changes. On the other hand, in Small and Medium Enterprises (SMEs) the production changes more often, each batch could be customized, the robot could be used for multiple purposes. If there is a need to reprogram the robot, the company needs its own highly skilled employee or must use the external supplier services which might be expensive or not flexible enough. Moreover, end-users might not be experts in programming or robotics but certainly might have invaluable task-domain knowledge. Therefore, there is a trend towards allowing end-users to program the robots: robot manufacturers are introducing simplified teach pendants and collaborative robots able to be hand-taught, third parties are developing visual programming tools, etc. Still, there are several pitfalls:

- Robot programming interface is vendor-specific.
- Missing visualization of spatial information.
- Robot-centric (does not allow to program the whole line).
- Requires textual coding or is not expressive enough.
- Doesn't allow to work within the task space.

This project was implemented with financial support from the state budget through the Ministry of Industry and Trade in the TRIO program (FV40052).

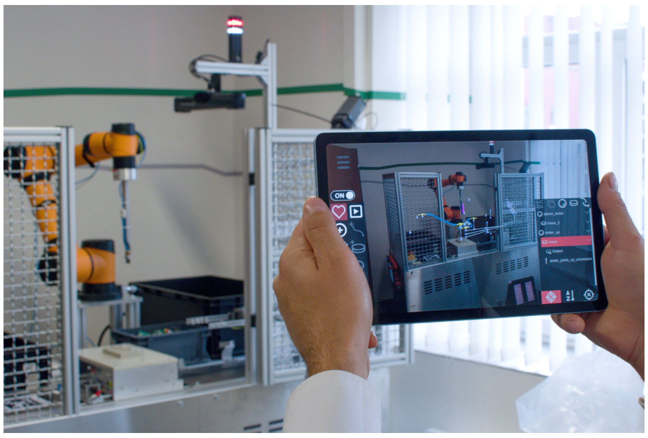


Fig. 1: A user observes the program for the PCB testing task through the tablet.

To overcome existing limitations, we present the ARCOR2 framework[1] which enables end-users to perform complete management of a robotic workplace or a production line: initial setup, programming, adaptation, releasing to production, controlling execution, etc. Its user interface can be seen as a universal teach pendant for all robot types, machines, or APIs where a new device or service can be integrated by writing a custom plugin in Python. This integrative approach eliminates the need to undergo training for the interface of each device involved. The user interface is designed for commodity tablets and utilizes augmented reality for visualization of program data, including spatial points, program instructions, and even a logic flow. One tablet can be used to manage multiple workplaces.

The framework was developed in cooperation with a system integrator[2], who provided the PCB testing use case and corresponding testing site, participated in gathering the requirements and continuous testing of both the backend part and

[1]Source code and documentation (including integration for ABB YuMi, Dobot M1 and Dobot Magician robots and for Kinect Azure sensor) are available at github.com/robofit/arcor2.

[2]www.kinali.cz/en

user interface of ARCOR2. The company has also developed an integration for its services and robots.

This paper is both culmination of the three-year project as well as our long-term research in the field of human-robot interaction within the industrial environments, which research questions could be summarized as follows:

- **Q1** – How interaction can be brought from a 2D screen back to the 3D world?
- **Q2** – Is (affordable) handheld AR a viable modality?
- **Q3** – How to allow precise work with imprecisely registered AR devices?
- **Q4** – What is the proper level of abstraction for end-users?

The goals of the paper are to summarize our work within the field and on the framework, to present the framework's general applicability and utility, perform overall evaluation (as previously published evaluations were focused on specific aspects), and share the gained experience with the community.

II. Related work

In industrial settings, an ordinary worker operates the robot most often at Level 0 (bystander) as defined in [1], while the higher levels are handled by a specially trained person, or by an external expert. With the increasing spread of collaborative robots, rising needs for flexible or customizable production, and deployment of robots into smaller industries, there is a trend towards allowing end-user to program robots.

A. Established Solutions

Although some off-the-shelf teach pendants offer a certain form of simplified programming, the usability seems to be rather low [2] due to missing visualization, inability to use common syntax structures as conditions and loops, high mental and physical demands or lack of tools for debugging [1,3–5].

Moreover, pendants are vendor-specific and limited to programming robots. On the other hand, offline programming tools as RoboDK[3] or ABB RobotStudio[4] offer comprehensive functionality but require extensive training. Additionally, PC-based and pendant-like user interfaces are in general likely not optimal for end-user programming, as they imply continuous switching of a user's attention [4] and therefore induce high cognitive and attention-related workload. A kinesthetic teaching is often employed in order to allow users to set target poses or waypoints intuitively and therefore simplify programming. However, depending on stiffness and size of a particular robot, it could be physically demanding and not desirable by users [3,5].

B. Proposed Approaches

Many alternative methods for simplified programming and even third-party complete solutions were proposed. Some of them are not intended as comprehensive tools and are rather focused on a specific task, aspect of the process or are limited to a certain robot. The simplification is usually achieved through some form of visual programming [6–10], spatial visualization enabling the user to work within the task context [8,11,12], commonly combined with a kinesthetic teaching [6,13,14] and/or perception [10]. Unfortunately, there still exist many limitations. Only a little portion of evaluations are carried out on non-trivial use-cases as in [9], or contain comparison with an existing method as in [7,8,15]. Often, there is only a simplified method available, which precludes the possibility of (remote) expert intervention, where it can be assumed that the expert prefers to work with source code. The issue could be for instance solved by generating the source code from visual representation [11,16] and probably optimally by bidirectional synchronization between those two.

C. Visualization Methods

Augmented reality seems to be a promising visualization method for simplified programming on a high level of abstraction. It may enable users to work within the task space, avoid superfluous attention switches, mental transformations, and related workload. However, only a few approaches really allow programming in augmented reality [8,11,17] or provides ability to setup a workspace [12] and therefore does not require an additional intermediate user interface.

Often, the augmented reality is used only as an extension, e.g. to visualize robot trajectories and in general, there is a lack of tools for precise manipulation of robot or virtual elements, which is necessary for industrial use cases. The AR may be for instance realized by spatial projection [8,18], which is however limited to visualization on surfaces. Head-up stereoscopic displays are able to convey depth, but on the other hand, are expensive, offer a limited field of view, and require learning of unconventional control (e.g. gestures). Usage of hand-held devices leads to problems with missing depth perception [12], however, those are affordable, portable, and easy to use.

D. Summary

As it can be seen from the previous related work overview, there exist many approaches that aim on lowering the barrier to entry robot programming by various means. However, there is a lack of comprehensive yet simple environments, allowing end-users to perform all tasks and steps that are necessary for industrial-like applications. Also, it has to be taken into account that modern workplaces may certainly contain not only a robot but multiple (programmable) machines or special-purpose devices. With Industry 4.0, there will be also a rising need to perform communication with various services through their APIs.

III. Use Case

Although the framework was designed to be general, the initial motivation came up from the use case of offline product testing in an SME company, where relatively small batches are tested and items' storage is highly variable. Therefore a program has to be adapted approximately once a week.

[3]url
[4]url

TABLE I: Comparison of the features of existing solutions for simplified programming of industrial applications.

property / feature	ARCOR2	ArtiMinds	CoStar	RoboDK	???
license	LGPL-3.0				
price ($)	0				
text programming	yes				
visual programming	yes				
integration of a custom hardware					
multi-user	yes				
motion planning	no				
workplace setup					
situated interaction	yes				
???					

The process consists of picking up either unorganized items from crates or organized ones from blisters, applying printed barcodes, putting items into a tester, starting the test, and placing items in boxes according to a functional test result. So far, the process was done by human operators but their time was not used efficiently, as they are idle for up to several minutes while the test runs. Moreover, the work is highly stereotypical. The goal was to optimize the use of a qualified workforce as most of the operators after robotizing the process could be reassigned to more creative work. The rest could be trained to be able to adapt programs of testing workplaces when needed and to supervise multiple workplaces during execution.

In the case of the traditional approach, the 6 DoF robot would be programmed using a teach pendant, the custom one and all the other devices would be operated by a PLC. To adapt such a heterogeneous workplace to a new product, a highly trained operator would be needed. In the proposed approach, the system integrator will develop an integration for all devices into the ARCOR2 system, providing functionality on the optimal level of abstraction for the task. The initial setup of each workplace (for its visualization see Fig. 2) will be also done by the system integrator. Then, changes can be either done by a trained operator or remotely by an expert programmer. The main advantage for end-user is that there is just one configuration, programming, visualization, and control interface.

Based on comprehensive discussion with the project partner, a set of requirements on the system were defined:

1) Convenient integration of new robots, machines, and services with variable level of abstraction.
2) Support collaboration between end-users and experts.
3) Ability to manage (perform CRUD operations on):
    a) Setups of the workplaces (available objects, their locations, and parameters).
    b) Important points in space and associated data.
    c) Program steps and their parameters.
    d) Self-contained executable snapshots of programs.
4) Robot as a source of precision.
5) Control and visualization of an execution state.
6) Debugging capabilities.

We have also defined a set of different user's roles, that are divided into two categories (see Table II) and for each role, there are different responsibilities and needs.

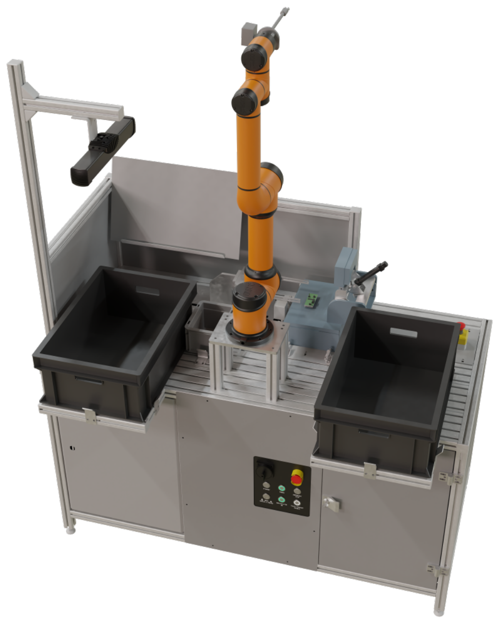


Fig. 2: Render of a PCB testing workplace with the Ensenso 3D camera for bin-picking, 6 DoF Aubo i5 robot, 2 DoF custom-build robot, functional tester, barcode reader, and printer, source, and target boxes.

IV. System Design

The design of the framework is mainly given by the defined requirements. However, it was also influenced by the knowledge gained during the development and evaluations of its previous generation called ARCOR [18], which utilized spatial augmented reality and a touch-enabled table for user interaction and was focused mainly on table-top scenarios. Later, in order to overcome inability to visualize information in free 3D space, HoloLens were integrated [19]. Although ARCOR was successfully evaluated in an industrial use case [20], its limitations (mainly complicated integration of new devices and program instructions) lead us to the development of the next generation.

A. Terminology

Object Type – plugin into the system that represents and provides integration with a particular type of real-world object, e.g. a certain type of robot, or a virtual object such as cloud API. It is written in Python and can benefit from (multiple) inheritance, in order to extend or share functionality. A set of

TABLE II: Expected types of users, divided into two main categories.

Category	Role	Responsibilities	Principal needs
End User	Operator	Manages program execution, solves simple problems.	Visualization of execution state, controls to start/stop program, notifications on errors.
	Standard User	Able to create a simple program or adapt an existing one.	Program management (edit, copy, etc.), tools to edit spatial points and program steps.
	Advanced User	Able to create complex program visually, can write simple code.	Visual definition of advanced concepts, simple and well-documented programming API.
Expert	Technician	Performs initial setup, called when serious problem occurs.	Debugging tools, entering exact numbers.
	Programmer	Integrates new devices, creates new functionality.	

built-in base classes is available, representing e.g. a generic robot or a camera and its required API. It could be associated to a model (representing both collision and visual geometry), which might be a primitive, or a mesh.

Action Object – an instance of an Object Type within the workplace, defined by its unique ID (UID), human-readable name, pose (optionally), and parameters (e.g. API URI, serial port, etc.).

Scene – set of Action Objects, represents a workplace, its objects and spatial relations.

Action Point – a spatially anchored container for orientations, robot joints configurations and actions. The container's position together with an orientation comprises a pose usable e.g. as a parameter for robot action.

Action – method of an Object Type exposed to the AR environment. A named and parameterized action is called an action instance. Actions may be implemented on different levels of complexity, according to the application needs and the target end-users competencies. However, in order to lower program complexity and to reduce training time, the actions should be preferably high-level and provide configurable skill-like functionality. Such actions can be seen as equivalent to reusable skills used e.g. in [7,13].

Project – set of Action Points, may contain logic definition. The project is associated with a scene.

Execution Package – a self-contained executable snapshot of a Project, which is created when there is a need to test the whole task or release a project into a production environment. The fact that the package is self-contained allows users to make further changes in the scene or project without any influence on already existing packages.

Main Script – contains a logic of the project, which may be defined visually or could be written manually with help of a set of generated classes providing access to project data as e.g. defined Action Points.

B. Integrating New Devices

A new device is integrated into the system by implementing an Object Type (Python wrapper) for it, that is based on some of the provided abstract base classes and is dynamically loaded into the system.

For instance, there is an abstract `Robot` class, and all Object Types representing particular robots are derived from it. It has a set of basic abstract methods representing mandatory, or robot's minimal functionality (e.g. method to get end effector pose), that have to be implemented. Then, there is a set of methods that may or may not be implemented, based on the available functionality of the particular robot (e.g. method for forward and inverse kinematics, or for toggling the hand teaching mode). After the wrapper is loaded, the system performs static analysis in order to determine in advance which optional methods are available and based on that, certain functionality is or is not made available to the user.

There are two main possibilities of how an Object Type could be interfaced to a real-world object, e.g. a robot:

1) Directly – if the robot provides API with the necessary functionality, the Object Type may communicate with it directly.
2) Through an intermediate service – for instance, if the robot lacks motion planning capability, there might be a ROS-based container between the robot and the Object Type.

In both cases, the Object Type is the main provider of all Actions, available to the user, regardless the interface between the Object Type and the real-world object.

C. Architecture

The framework is divided into a set of services (backend) and a user interface (frontend). The main service of the system is **ARServer**, which acts as a central point for user interfaces and mediates communication with other services (see Fig. 3).

So far, only one implementation of a user interface has been developed, a tablet-based app providing full functionality, however, the intention is to allow involvement of several simpler, complementary interfaces providing only some aspect of functionality, e.g. RGB LED strip indicating system status or hand tracking-based interface for controlling a robot. Therefore, the server must be able to deal with multiple connected interfaces even in single-user scenarios.

Interfaces are connected to ARServer through Websockets, which allows bidirectional communication. The ARServer holds the system state, while interfaces can manipulate it using a set of RPCs and receives notifications on changes. It is assumed, that each workplace runs its own instance of ARServer and therefore, the server maintains only one session for all users: if one user opens a project, the same project is shown to other connected users as well. In order to support efficient and safe collaboration between users, there is a locking mechanism that prevents multiple users to manipulate the same element (e.g. control the robot).

The ARServer also serves as a proxy between Python code and AR environment, which is code-agnostic. It analyzes code of Object Types in order to extract available Actions and their parameters and creates JSON metadata, that is available to user interfaces. The code analysis takes advantage of PEP 484 type hints[5] in order to extract e.g. parameter types and matching nodes of Abstract Syntax Tree (AST)[6] in order to e.g. inspect value ranges, that are defined using assertions or check if a method (feature) is implemented.

The scene or project opened within the server could be either offline or online. In the online state, instances for all objects are created meaning that e.g. connection to a robot is made. In an online state, robots could be manipulated with and any action instance added to a project may be executed, which simplifies a programming and debugging process. However, it is also possible to work offline, where just functionalities as controlling a robot are not available. Moreover, in the offline mode, the robot and other relevant machines does not have to be connected, therefore, the operator may prepare the base program in advance, without the need of the actual robot.

Project Service provides persistent storage for workplace-relevant data: scenes, projects, Object Types, models, etc.

Scene Service, used e.g. in cases where underlying implementation is based on ROS, is responsible for the management of collision objects.

The **Build Service** creates for a given project a self-contained executable package. The logic could be defined within the AR environment or provided in a standalone file. When generating logic from its JSON representation, it is first assembled in a form of AST and then compiled into Python code. Moreover, a set of supplementary classes e.g. simplifying work with Action Points are generated.

Execution Service manages execution packages created by the Build Service. The most important functionality is running the package when the service streams events regarding execution state (e.g. which action with what parameters is being executed) to ARServer. The execution can also be paused or resumed when needed.

Calibration Service provides a method to perform camera pose estimation based on ArUco marker detection [21]. The service is configured with IDs, poses, and sizes of available markers. When estimation is requested, markers are detected in the provided image, respective camera poses are computed and then all poses are averaged using a camera-marker distance and camera-marker orientation as weights in order to produce the final 6D pose. For averaging quaternions, a method from [22] is used. This estimation can be either used by user interfaces where e.g. an AR visualization needs to be globally anchored or it could be used by ARServer to update the pose of the camera in a scene. Another method of the service may be used to adjust the pose of the robot using an RGBD camera. The robot model in a configuration corresponding to the actual robot state is rendered from point of view of a camera in a robot's current position within the scene, which therefore serves as an initial guess. The virtual camera is used to generate a point cloud, which is then registered using a robust ICP (TukeyLoss kernel) with the point cloud from the real RGBD camera observing the scene (1024 frames averaged), that has been filtered to contain only close surroundings around the robot in its initial pose. If the precision of such calibration is not enough for the task, more precise methods must be used and the pose of the robot or camera can be entered manually.

[5]www.python.org/dev/peps/pep-0484/

[6]https://docs.python.org/3.8/library/ast.html

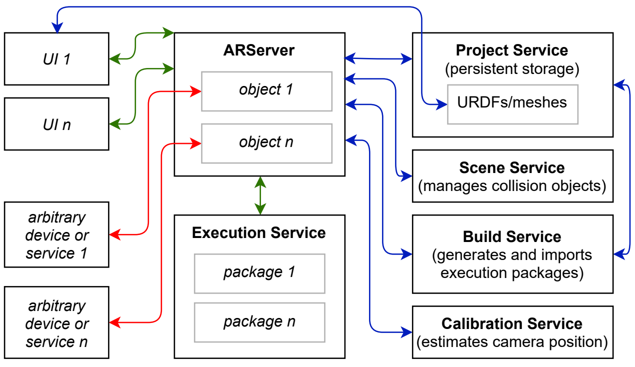


Fig. 3: Block diagram of the system in a state, when object instances are created in the ARServer (scene/project opened and online). Green lines depict WebSockets connections (two-way communication necessary), blue are REST APIs and for red, an implementer is free to choose appropriate technology.

D. Program Representation

One of the goals of the framework is to support collaboration between non-programmers who prefer creating programs visually and programmers who prefer to work with code. Because of this, there are two language representations. For visual programming, there is an intermediate program representation based on JSON format, which is language agnostic, easy to serialize, supports common programming techniques (cycles, conditions, variables), allows flexible parameter specification, and is easily manipulable from user interfaces. For execution, the intermediate format is translated into Python, which is currently the most popular scripting language[7]. The same language is also used for the implementation of Object Types, through which a new device can be integrated into the system. This also allows a use case when a non-expert user creates the program visually and the result is adjusted by an expert programmer. The form of Python code was designed with the possibility of transferring the code back into the intermediate format. However, this was not implemented yet.

The structure of the JSON format is as follows. Within a project, there might be $[0, n]$ action points, where each might contain $[0, n]$ actions. Each action is assigned a UID, unique human-readable name, type (scene object UID and corresponding underlying method/action), and $[0, n]$ parameters (corresponding to parameters of the method). Action parameter can be given as literal or referenced to either a project variable (constant shared by multiple actions) or a previous result (return value of precedent action). On the project level, there is an array of objects defining logical structure (visualized as blue lines, see Fig. 5), where each contains UID of source and target action and optional condition. Actions together with those linkages then form a directed acyclic graph, where the loops are forbidden at an application level. Without a

[7]According to PYPL Index for August 2021.

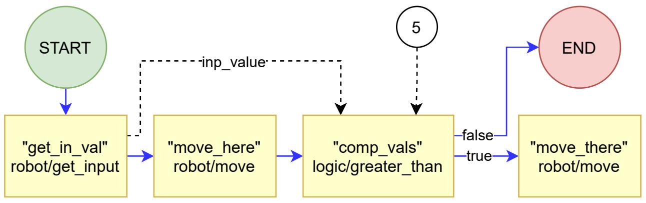


Fig. 4: The logical structure of an example program. Yellow boxes are actions (text in quotes is the user-entered name for the instance of action, below is an object and the corresponding method), blue lines denote logical connections (flow of the program), while black dashed lines denote data connections. The example shows how previous results can be used as parameters of a subsequent action and how logical flow can be branched based on a numerable value (boolean in this case).

condition, two actions could be connected only with one logic linkage. Conditions are meant to achieve simple branching for numerable types as boolean, enums, and integers. E.g. for branching according to a boolean value, two logical linkages are added, one for *true* and the other for *false* (see Fig. 4 and Listing 1). Any other type of condition has to be implemented in a form of action, for instance, *greater_than(float1, float2)* returning a boolean value. Also, loops are not part of the format definition and have to be implemented in form of custom actions. This restriction keeps the intermediate format simple and at the same time allows integrators to provide a customized set of actions to their end-users.

```
inp_value = robot.get_input(an="get_in_val")
robot.move(an="move_here")
comp_res = logic.greater_than(inp_value, 5, an="comp")

if comp_res == True:
  robot.move(an="move_there")
```

Listing 1: An example of generated Python code. Parameter `an` denotes action name, which is human-readable counterpart to action UID.

E. User Interaction

A working prototype, based on the concept presented by Kapinus et. al. [23], was implemented and iteratively tested, and improved in cooperation with the project partner. The design of the application was modified in order to support the two-handed operation of the tablet and control of interface elements using the user's thumbs, to lower the fatigue of arms and hands.

The primary concept of our tablet user interface deals with the fact, that most of the robotic programs interact with a real environment in some manner. An operator, using our user interface, is able to annotate the environment in a simplified way and subsequently design programs' logic. Thanks to the utilization of augmented reality, this can happen within the task space and therefore mental demands are lowered [5,17].

Our user interface uses several graphical elements for precise annotation of specific places in the environment (action points). These places may later be used as spatial anchors for elements representing specific actions. Visual elements representing actions (also known as *pucks* in our GUI) are therefore located at the place, where the action will be executed, which improved users' comprehension of spatial relations within the program.

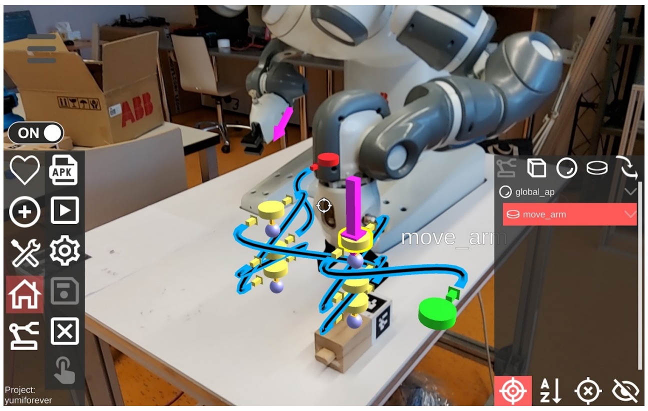


Fig. 5: The application screen with a tool menu (left), selector menu (right) and non-interactive 3D scene (center), with action points (violet), actions (yellow, green for program loop start, red for its end), logical connections (black/blue) and robot end effectors (magenta). The visualized program realizes a simple pick and place task consisting of low-level actions.

The whole interface consists of three main parts: *sight* in the middle of the screen, *main menu* on the left side of the screen, and *tool context menu* on the right side of the screen. The *sight* is used to select virtual objects by the physical pointing of the tablet. *The tool selection menu* shows actions available for a currently selected object (e.g. duplicate object, transform object, etc.) and the *tool context menu* serves as a sub-menu for currently performed operations (e.g. move / rotate tools when *action object* is being manipulated). See Fig. 5.

V. Evaluation

During development, the system and its user interface were evaluated multiple times using different methods and continuously refined to provide a plausible user experience and fulfill use case needs. Some of the presented evaluations focused on certain aspects of the problem were already published, and here, we only briefly report their results to provide a complete overview summarizing the framework's evolution.

A. Interface Concept

A high fidelity initial prototype of a user interface was implemented and evaluated in a qualitative experiment ($n = 7$) by Kapinus et. al. [20], focused on usability, mental workload, and user experience. Participants were asked to create a simple program using several highly abstracted actions. The task consisted of 21 steps that has to be done.

Regarding subjective metrics, the user interface was ranked A using SUS [24] (90-95th percentile) and it was overall rated Excellent in all UEQ [25] categories, i.e. Attractiveness (mean score 1.93, SD=0.58), Pragmatic attributes (mean score 2.26, SD=0.28), and Hedonic attributes (mean score 1.86, SD=0.72). Overall NASA-TLX [26] score was 27.38 (SD=9.41), making it lower than 80% of studies analyzed in [27].

From the objective point of view, the main finding was that all participants were able to successfully complete the task in a reasonable time (average completion time was 527 s, SD=130 s). No fundamental problem was discovered, although some minor issues were observed or reported by participants.

Most of the participants complained about the design of *pucks* (visual elements representing actions), mainly about their size, appearance, and placing strategy. Based on that, the design was altered and due to the fact that they are now placed above action points, the position of the puck could now be changed by the user, if the default position is not suitable.

The other important issue was related to the selection of virtual objects. In the prototype, the object was selected by touching it on the screen, causing the user has to hold the device with only one hand, so they can use the other hand to touch the object. This caused hand and arms fatigue. Moreover, the selection was sometimes difficult, especially in a cluttered environment. In order to cope with it, design of the interface was changed to allow controlling all elements by a user's thumbs, when holding the device with both hands in landscape mode, which also demanded utilization of some indirect selection method.

Only two of the participants found out, that they can benefit from the active movement of the mobile device inside the scene, one participant explicitly stated that they would like to stay on one spot and zoom inside the scene. To solve this issue, a non-immersive VR mode was introduced, which allow users to freely change position of a virtual camera and therefore to see and interact with the workplace from any angle and distance.

B. Virtual Object Selection

As the initial experiment with the prototype of the interface has shown, direct selection of virtual objects on the screen was problematic in some cases (high density of objects, occlusions, similar objects) and necessity to hold the device with one hand and touch the screen with the other lead to ergonomic issues. Therefore, the design of the application was changed in a way, that all interactable elements are within reach of the user's thumbs and we implemented and compared two indirect selection methods, which were described and preliminarily evaluated in [28]. Both of them work with spatially clustered objects. One method is based on a spatially anchored hierarchy menu, and the other utilizes a crosshair and a side menu that shows candidate objects according to a custom-developed metric. When compared with the direct touch method, the results indicate that indirect methods might lead to better precision and improved confidence for selecting the intended object, however, at the expense of worse task performance. The methods will be further developed and evaluated, however, based on the experiment's results, we chose the method with a side menu (the *selector menu*, see Fig. 5) as it seems to provide better performance (time to select an object), lower task load (TLX) and similar success rate as the hierarchical menu.

C. Non-immersive VR mode

When working with AR, there is often a need to move closer in order to distinguish or inspect virtual objects, or in contrary to move further in order to see the whole scene, which is amplified by the limited field of view of handheld devices and HMDs. Regarding handheld devices, it is also often necessary to see the scene from different angles, to correctly judge the placement of the objects, which is caused by a lack of depth perception due to monoscopic display. Moreover, within industrial environments there is typically limited floor space, there are safety curtains, fences around robot cells, etc. These constraints might make viewing the workplace from certain poses physically challenging, or even impossible. Therefore we proposed and evaluated an approach allowing temporal switching from AR to a non-immersive VR [29]. In VR, the application shows a 3D model of the workplace, and the viewpoint is controlled by device motion in conjunction with on-screen joysticks, with non-linear sensitivity. The conducted experiment ($n = 20$), based on the object alignment task, revealed that self-reported physical demands are significantly lower when users are allowed to arbitrarily switch between AR and VR. The usefulness of the VR mode was rated as high and during the task and users spent 70% of the time within it. Observations of users' behavior have revealed that the VR mode was often used to get an overview of the workspace, to find an occluded object, or to avoid an uncomfortable position.

D. Multi-user Collaboration

In order to evaluate the collaborative aspects of the framework, a small-scale user experiment ($n = 3$) was carried out in a lab-like environment. The experiment was focused on functionality, however also served as the very first usability evaluation. The workplace consisted of two robots (Dobot M1 and Magician), a conveyor belt, and several collision objects.

The task was to collaboratively set up the workplace and to create a simple program for moving cubes from one robot to the other and back using the conveyor belt. In the setup phase, each participant added one Action Object (a robot or the belt) to the scene and positioned it. Subsequently, a project was created and each participant created several *Action Points* (using a hand-teaching and visual positioning tools) and related *Actions*. Finally, the logical connections defining the program flow were added and the project was executed. The participants worked on separate sub-tasks most of the time but shared the same workspace. Moreover, they had to collaborate to successfully connect all sub-tasks into a working program.

From the technical perspective, the user experience during the collaboration was smooth and user interfaces were kept synchronized properly. Regarding usability, although users communicated verbally during the experiment, they also appreciated visual indication of which object is being used (locked) by someone else. Based on the course of the experiment, collaborative programming seems to be a viable approach and we will investigate it more deeply in the forthcoming research. In this case, the task was done by three users in approximately 15 minutes. It would be highly interesting to determine the relation between task time and a number of collaborating users on a significantly more complex task.

E. Iterative Testing and Refinement

The whole system was created in close cooperation with our industrial partner, who has over 15 years of experience in automation. During the development, the design of the user interface and key parts of the system were discussed in detail with the system integrator and potential end-users.

Besides individual testing, there were several integration meetings, where the whole task programming process was tested with the real production-like robotic cell, to maximize simplicity and comprehensibility for the end-users. This helped us with a better understanding of real-world scenario difficulties, which are not obvious during in-lab testing, e.g. object selection in heavily cluttered environments (e.g. caged robotic cell), cooperation with safety precautions, etc.

As a result, several improvements were added to the user interface, for example, better ray-casting strategy for virtual object selection enabling user to disable several objects (e.g. virtual safety walls), which has to be part of the scene, but once they are created, are barely used anymore.

Additionally, an unstructured interview with the system integrator representative was organized. They were asked to state their opinion on the framework being developed from the commercial perspective. Following key advantages were claimed:

- Clear visualization of position and distribution of individual actions in space.
- Ease and speed of programming for smaller-scale applications.
- Possibility of visual composition of the scene - collision objects, positional relationships of individual elements.
- Simple and useful controls for the robot - stepping, end effector alignment, integration of hand teaching.

And the following limitations were pointed out:

- Visual clutter for large-scale applications - too much graphical information.
- Ergonomically demanding method requiring the creation of the entire application in a standing position, while holding a tablet.

Regarding the large-scale applications, it was suggested to define categories of actions and to distinguish them e.g. by color-coding or by icons. Moreover, the visual clutter may be reduced by different techniques, as implementing e.g. level of detail [30], or by implementing more complex actions on a higher level of abstraction, which will reduce the number of individual actions needed to realize a given task. The ergonomy of use may be improved by the proper holding of the device (needs to be covered during training) [31] and also by already described VR mode, which allows users to temporarily switch from AR to VR in order to reach physically unreachable poses, zoom in, or to work while sitting.

F. Expert Review

At the point, where the system and interface design and features were mostly stable, an expert review was conducted in order to eliminate the most significant user experience problems and to validate the overall concept of the system. Three reviews were obtained.

The first reviewer (R_A) is an experienced software tester. The review was performed at the testing site of the project partner with an Aubo i5 and one custom-built two-axis robot. The second reviewer (R_B) is a 3D data visualization and processing specialist, with experience in the field of human-robot interaction. The third one (R_C) is an expert in cyber-physical systems, computer graphics, user interface design, and evaluation.

Reviews by R_B and R_C were performed at the university robotics laboratory with an ABB YuMi robot. Reviewers R_A and R_B used the same version of the interface, while R_C used a slightly updated one that was available at the moment. Each session lasted approximately one and a half hours. The Samsung Galaxy Tab S6 with a protective cover was used. A reviewer was given a technical document describing the solution in advance and then briefly introduced to the usage of the interface. Then, they went through the core functionality while commenting on their findings. The comments were recorded and after the experiment processed into a review protocol. The reviewer was then asked to verify the protocol, provide a brief comment on each issue, and assign severity on a scale of $[1, 5]$.

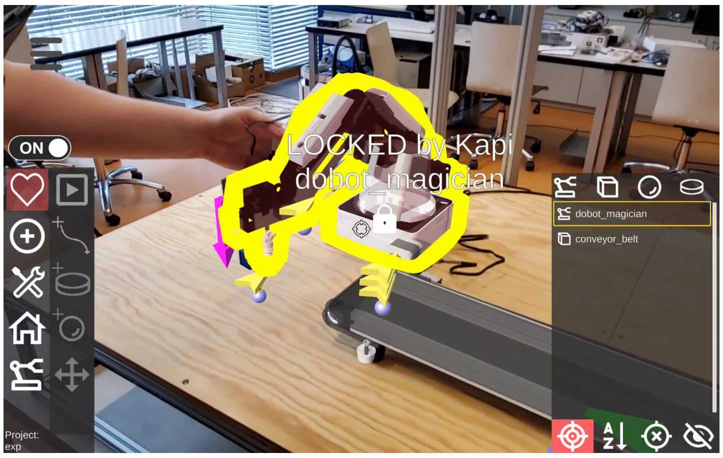
A 39 unique issues (48 in total) of different severity (see Fig. 7) were identified by all reviewers. Most of them were ranked with low severity, dealing mostly with some minor user interface usability issues, such as difficult number input (R_A) caused by the default Android keyboard, the unclear icon for a favorite group of actions (R_B), or issue with main menu actions grouping, forcing the reviewer to navigate through the menu to locate required action (R_C). The most severe issues are shown in Table III, together with self-reported severity and brief suggestion provided by the reviewer.

The collected feedback was categorized into the following groups:

- **Control** (12 unique issues, 14 total) – issues related to user interface control.
- **HUD design** (13 unique issues, 17 in total) – problems with the user interface itself - icons, menu design, etc.
- **System Status** (3 unique issues, 4 in total) – related to notifications and system state visualization.
- **Visualization** (11 unique issues, 13 in total) – visualization of 3D scene content (Action Points, Actions, etc.).

The R_A suggested, that there are too many icon buttons, and their purpose is not always very clear at the first sight. The only way how to understand individual icons is to hold a finger over them until a tooltip is shown. This issue will be solved by providing proper documentation of the application, together with onboarding mode, which will guide the novice user through the application.

The reviewer also pointed out, that it was quite physically demanding to hold the tablet for a longer period of time and it was necessary to take rest periods regularly. It was partially caused by improper holding of the device, where the R_A hold it by one hand on the left side of the screen for some time at the beginning of the review, which caused fatigue to the arm.



(a) During hand teaching, the robot is locked by the user and therefore unavailable for others, which is indicated within the 3d scene.

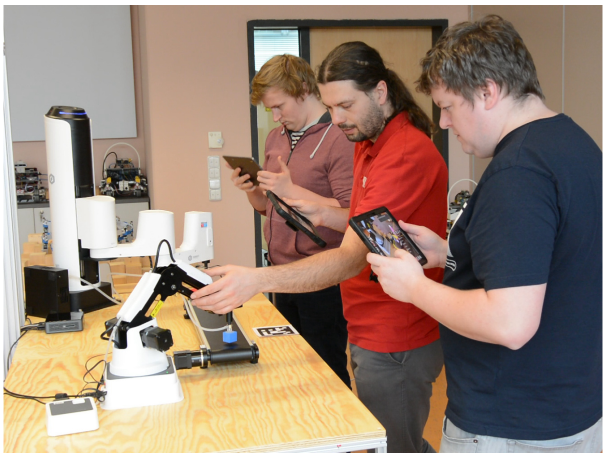

(b) Multiple users collaborating on the task of moving boxes between robots using a conveyor belt.

Fig. 6: Technically-oriented evaluation of the collaborative-related functionality.

TABLE III: The most severe issues (rated 4 or 5) and suggestions on how to mitigate them as reported by the reviewers.

	Issue	Severity	Suggestion	Category
R_A	Too much buttons and time-consuming determination of their meaning.	5	Implement a guide helping new users to get comfortable with application usage.	Documentation
	System status not visible in some cases (e.g. in case of long lasting processes as calibration).	4	Add persistent notifications.	System Status
	Inability to aim objects through another object.	4	Add possibility to temporarily disable an obstructing object.	Control
	Physical demands when working longer than 30 minutes.	4	Encourage both hand-holding, suggest rest period.	Control
R_B	A skill and a lot of physical movement are needed to judge the position of virtual objects.	5	Add shadows to the virtual objects, to improve depth perception.	Visualisation
	Complicated flow in order to add an action.	4	Allow to add an action without adding an action point first.	Control
R_C	Reachability of *action points* by selected robot is not visualised.	4	Add some indication of reachability.	Visualisation
	Difficult orientation in more complicated programs	4	Use different connections for different logic flow phases.	Visualisation

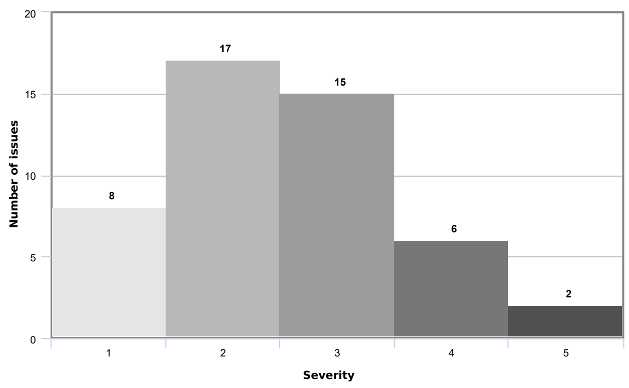


Fig. 7: Histogram of all reported issues clustered by their severity.

This issue could be addressed in the onboarding mode of the application, where proper holding should be demonstrated and rest sessions suggested.

The testing site of the project partner contained virtual walls around the robotic cell, assembled from dummy *action objects*. This complicated objects selections because these safety walls were the first objects to be hit by the sight, making virtual objects inside the cell virtually impossible to select by aiming. This issue was already solved by enabling users to put selected virtual objects on the blocklist, thus excluding them from the selection process.

On the other hand, the R_A really liked the visualization of the logical flow of the program, which helped them both to understand the meaning of the edited program and to change the behavior of the program. Moreover, the reviewer appreciated the simplicity of robot stepping available directly from the programming application, without the necessity of using the dedicated teach pendant. The reviewer explicitly mentioned that the ability to align the robot's end-effector with the underlying table was crucial for the fast navigation of the robot.

The main issue for the second reviewer, the R_B, was related to depth sensing. The reviewer had problems with understanding where the manipulated object (e.g. *action point*) is in the real world and they had to walk around heavily to see its position from multiple angles. They pointed out that it was probably caused by the lack of generated shadows, which usually helps people to sense the depth. As a (partial) solution, we have enabled shadows and light estimation in our application. To even more support the user's knowledge of object position, a kind of projection in the plane of the table, with information of the height above the table (or nearest object bellow).

The reviewer also stated that the flow of action definition is unnecessarily complex and difficult, meaning that the user has to insert an *action point* first and then assign an *action* to it. They suggested, that simplification of this process would be a significant improvement. Among others, this particular issue is currently being more deeply investigated in a separate study, which will be published later.

The last but not least reviewer, the R_C, stated that it makes sense for him to define actions inside the real environment. On the other hand, they were worried that it will be difficult to understand a more complex program, where a high amount of *actions* together with conditional execution will be incorporated. The reviewer suggested that selected parts of logical flow should be differentiated either by color or shape.

Another important issue for the R_C was that the robot's reachability of *action points* and executability of actions was not visualized in any way. For *actions*, there could be quite easily added a mark to indicate whether or not the action could be executed. In the case of the reachability of *action points*, the possibility of having more than one robot in the scene needs to be taken into account. Moreover, any *action point* could potentially have a number of orientations and each of them needs to be evaluated separately.

In summary, the reviewers approved the overall concept of the framework as suitable for end-users. As they tested the framework through the user interface, naturally, most findings were related to its usability, where several relevant suggestions were collected.

VI. Limitations

VII. Lessons Learnt

The problem of designing a framework that should act as a central point of integration for arbitrary devices and its user interface should provide great usability for end-users while serving a high amount of dynamically loaded functionality, is a complex one and there certainly exist different ways how to approach it. From our experience, it is crucial to have a realistic use case and then, to define what type of users are going to interact with the framework and its user interface in what ways. We see the iterative design process, for both the backend architecture and API as well as for user interface design, as a key factor determining success. We would also suggest making smaller and specifically focused experiments as we did, rather than one all-embracing because it allows faster iteration and provides better interpretable results.

The iterative design process led us to a state, where the user interface is minimalistic and optimized for best performance when used regularly, for several hours a day. The drawback is, that some of its aspects as the indirect selection of objects are slightly unintuitive for novel users. Moreover, there is a specific terminology, etc. Therefore, a need for some form of initial training is inevitable. The training, when executed efficiently (video, in-application guide), is in our opinion worth the improvement of performance during regular operation.

One of the main problems to solve was how to overcome the imprecise registration of AR visualization to the real environment. We chose to rely on work with relative coordinates, where the precise coordinate is obtained through a robot (usually the most precise device at the workplace) and then, further points are added as relative to it. Points are manipulated by visual tools providing virtually unlimited precision. In practice, it turned out that it is highly useful to let users navigate the robot to any point, then to step the robot and update the point.

From the practical point of view, it proved useful to organize the source codes of the backend (17 Python packages at the moment) in a monorepo, utilizing the Pants build system[8]. Especially in the early phases of development, monorepo makes big refactoring easier. Moreover, it significantly reduces necessary effort related to maintenance as there is only one configuration of continuous integration, etc. Although using tests may sound obvious, it is often skipped for academic software. The adoption of continuous integration and implementation of different levels of tests (unit, integration) definitely took some resources, but finally, it saved us a lot of valuable time and make us confident when performing changes to the code-base.

VIII. Conclusions

The paper presented the ARCOR2 framework, which allows end-users to program not only robots but whole complex workplaces or production lines consisting of multiple robots and other arbitrary machines. The framework was designed, developed, and iteratively tested in close cooperation with the industrial partner, who provided the realistic use case, testing site, and valuable feedback. One of the main advantages of the solution is that support for any device or service could be added in a form of a plugin. The visual programming in the AR interface allows specifying not only program steps and their parameters but also logical flow, including conditions. Kinesthetic teaching is utilized to obtain precise positions, however, its usage is minimized to limit users' fatigue only to get reference points, and then, other necessary points are manipulated by graphical tools relatively to the robot-originating points. This way, we also cope with the inaccuracy of AR registration to the real world. The framework also supports multiple users working at the same time, which can be useful for instance for large-scale workplaces or during training.

Many rounds of internal testing were performed, focused both on the user interface as well as on the API of the framework, making sure that it is usable from both end-user and expert-user point of view. The role of expert users is mainly to develop the integration of devices and services and prepare necessary functionality on a task-appropriate level of abstraction. The end-users role is to program the task but as the visual representation is compiled into a source code, expert users can get easily involved even in this stage, when needed. Despite internal testing, the initial concept of the user interface was evaluated in a user study. After that, the interface design

[8]https://www.pantsbuild.org/

was changed e.g. to allow controlling of all UI elements by users' thumbs and thus reduce fatigue coming from holding a tablet in one hand. After the functionality was stabilized, an expert review was carried out, which results are presented in this paper. Its main outcomes are the necessity to implement onboarding mode to support novice users, to improve depth perception by providing additional cues, and to simplify flow for adding actions.

In summary, the framework when adopted by a system integrator and its customer allows efficient collaboration between professional (robot) programmers and end-users, who are domain experts. By allowing end-users to set up a workplace, create or adapt the program, high flexibility needed for SMEs is achieved. Moreover, as SMEs are able to perform program modifications/adjustments on their own, expenses are reduced.

The gained experience also helped us to at least partially answer our long-term research questions. Within the context of visual programming, it turned out, that situated interaction could be realized using augmented reality and handheld devices, when supported by specifically designed interfaces (Q1-2). The solution we chose to overcome imprecise registration of AR visualization, was to utilize the robot's precision and then to move virtual objects relatively to precisely obtained positions. Then, visualization might be for instance slightly shifted, or not fully stable, but the spatial relations between objects are kept and users can manipulate objects with arbitrary precision (Q3). However, it has to be clearly communicated to users that such imperfection might occur and that it has no impact on the robot's precision. From our experience, a low level of abstraction makes visual representation incomprehensible, and therefore basic program steps have to be merged into more complex, parameterizable skill-like ones (Q4).

A. Future Work

There is ongoing work on comparing ARCOR2 with an existing, commercially available solution for simplified programming of robots. The paper under preparation will also contain an in-detail description of the user interface, which was here described only briefly. At the same time, we are planning a out of the lab long-term study that will start once the framework will be deployed at a PCB testing facility.

Moreover, collaborative programming by multiple end-users deserves to be investigated more deeply, including evaluation of effectiveness of such approach for larger workplaces. We will also make the effort to reveal how to optimally substitute the missing depth perception when using monoscopic displays. There is also a plan to implement decompilation of Python code into visual representation allowing bidirectional synchronization between AR environment and source code (for this to be possible, a source code would have to follow established conventions), which would in turn allow evaluation of collaboration between experts and end-users.

We also plan to port and adapt the user interface to other platforms as e.g. HoloLens 2 and to add support for advanced programming constructs like functions, or visually defined object actions allowing code reuse. It will be also highly interesting to investigate, how the framework could be applicable to other use cases than PCB testing, which predominantly consists of picking and placing parts.

Finally, during the development of the framework, many general questions related to handheld augmented reality arose. For instance, what are optimal cues to compensate for the limited depth perception, or how to motivate users (who tend to be rather still during AR usage) to take advantage of free movement.

REFERENCES

[1] T. B. Ionescu and S. Schlund, "A participatory programming model for democratizing cobot technology in public and industrial fablabs," *Procedia CIRP*, vol. 81, pp. 93–98, 2019.

[2] C. Schmidbauer, T. Komenda, and S. Schlund, "Teaching cobots in learning factories–user and usability-driven implications," *Procedia manufacturing*, vol. 45, pp. 398–404, 2020.

[3] G. Ajaykumar and C.-M. Huang, "User needs and design opportunities in end-user robot programming," in *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 93–95.

[4] A. Weiss, A. Huber, J. Minichberger, and M. Ikeda, "First application of robot teaching in an existing industry 4.0 environment: Does it really work?" *Societies*, vol. 6, no. 3, p. 20, 2016.

[5] C.-M. Huang, "Contextual programming of collaborative robots," in *International Conference on Human-Computer Interaction*. Springer, 2020, pp. 321–338.

[6] C. Paxton, A. Hundt, F. Jonathan, K. Guerin, and G. D. Hager, "Costar: Instructing collaborative robots with behavior trees and vision," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 564–571.

[7] M. Stenmark, M. Haage, and E. A. Topp, "Simplified programming of re-usable skills on a safe industrial robot: Prototype and evaluation," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, 2017, pp. 463–472.

[8] Y. Gao and C.-M. Huang, "Pati: a projection-based augmented tabletop interface for robot programming," in *Proceedings of the 24th international conference on intelligent user interfaces*, 2019, pp. 345–355.

[9] C. Mayr-Dorn, M. Winterer, C. Salomon, D. Hohensinger, and R. Ramler, "Considerations for using block-based languages for industrial robot programming-a case study," in *2021 IEEE/ACM 3rd International Workshop on Robotics Software Engineering (RoSE)*. IEEE, 2021, pp. 5–12.

[10] J. Huang and M. Cakmak, "Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts," in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*. IEEE, 2017, pp. 453–462.

[11] E. Yigitbas, I. Jovanovikj, and G. Engels, "Simplifying robot programming using augmented reality and end-user development," *arXiv preprint arXiv:2106.07944*, 2021.

[12] Y. J. Thoo, J. Maceiras, P. Abbet, M. Racca, H. Girgin, and S. Calinon, "Online and offline robot programming via augmented reality workspaces," *arXiv preprint arXiv:2107.01884*, 2021.

[13] Y. S. Liang, D. Pellier, H. Fiorino, and S. Pesty, "iropro: An interactive robot programming framework," *International Journal of Social Robotics*, pp. 1–15, 2021.

[14] G. B. Rodamilans, E. Villani, L. G. Trabasso, W. R. de Oliveira, and R. Suterio, "A comparison of industrial robots interface: force guidance system and teach pendant operation," *Industrial Robot: An International Journal*, 2016.

[15] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert, and A. Knoll, "Intuitive instruction of industrial robots: Semantic process descriptions for small lot production," in *2016 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE, 2016, pp. 2293–2300.

[16] M. Ladeira, Y. Ouhammou, and E. Grolleau, "Robmex: Ros-based modelling framework for end-users and experts," *Journal of Systems Architecture*, vol. 117, p. 102089, 2021.

[17] S. Y. Gadre, E. Rosen, G. Chien, E. Phillips, S. Tellex, and G. Konidaris, "End-user robot programming using mixed reality," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 2707–2713.

[18] Z. Materna, M. Kapinus, V. Beran, P. Smrž, and P. Zemčík, “Interactive spatial augmented reality in collaborative robot programming: User experience evaluation,” in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2018, pp. 80–87.
[19] D. Bambuŝek, Z. Materna, M. Kapinus, V. Beran, and P. Smrž, “Combining interactive spatial augmented reality with head-mounted display for end-user collaborative robot programming,” in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2019, pp. 1–8.
[20] M. Kapinus, Z. Materna, D. Bambušek, and V. Beran, “End-user robot programming case study: Augmented reality vs. teach pendant,” in *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 281–283.
[21] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, “Speeded up detection of squared fiducial markers,” *Image and vision Computing*, vol. 76, pp. 38–47, 2018.
[22] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, “Averaging quaternions,” *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007.
[23] M. Kapinus, V. Beran, Z. Materna, and D. Bambušek, “Spatially situated end-user robot programming in augmented reality,” in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2019, pp. 1–8.
[24] J. Brooke *et al.*, “Sus-a quick and dirty usability scale,” *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.
[25] M. Schrepp, “User experience questionnaire handbook,” 09 2015.
[26] S. G. Hart and L. E. Staveland, “Development of nasa-tlx (task load index): Results of empirical and theoretical research,” *Advances in psychology*, vol. 52, pp. 139–183, 1988.
[27] R. A. Grier, “How high is high? a meta-analysis of nasa-tlx global workload scores,” *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 59, no. 1, pp. 1727–1731, 2015.
[28] M. Kapinus, D. Bambušek, Z. Materna, and V. Beran, “Improved indirect virtual objects selection methods for cluttered augmented reality environments on mobile devices,” in *Companion of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*, 2022, accepted.
[29] D. Bambušek, Z. Materna, M. Kapinus, and V. Beran, “Handheld augmented reality: Overcoming reachability limitations by enabling temporal switching to virtual reality,” in *Companion of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*, 2022, accepted.
[30] M. Tatzgern, V. Orso, D. Kalkofen, G. Jacucci, L. Gamberini, and D. Schmalstieg, “Adaptive information density for augmented reality displays,” in *2016 IEEE Virtual Reality (VR)*, 2016, pp. 83–92.
[31] E. Veas and E. Kruijff, “Vesp’r: design and evaluation of a handheld ar device,” in *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2008, pp. 43–52.

Michal Kapinus is a Ph.D. student at the Faculty of Information Technology, Brno University of Technology, Czech Republic, where he successfully defended his master theses in the field of service robotics. The topic of his emerging doctoral thesis is “Bringing the Human-Computer Interaction Back to the Real World”. His research interests are mainly human-machine interaction in mixed/augmented reality and the end-user robot programming.

Zdeněk Materna, Ph.D. received a bachelor’s degree in computer systems in 2009 at College of Polytechnics Jihlava. Then at the Brno University of Technology, he received a master’s degree in cybernetics, control, and measurements in 2011 and later in 2019 defended a dissertation on the topic of advanced task-oriented user interfaces for non-expert users. His research interests are human-robot interaction, augmented reality, semi-autonomous systems, and intelligent home automation. He is currently a post-doc at BUT.

Daniel Bambušek is a Ph.D. student at the Faculty of Information Technology, Brno University of Technology, Czech Republic, where he focuses on the use of augmented reality in the field of human-robot interaction, including communication of spatial information and end-user robot programming. He is also interested in augmented reality and augmented virtuality for efficient user interfaces of drone operators.

Vítězslav Beran, Ph.D. is an assistant professor at the Faculty of Information Technology, Brno University of Technology, Czech Republic, where he leads the Human-Robot Interaction group. His research interests include human-machine interaction, computer vision, video processing and augmented reality. He has participated in several European and contractual research projects.

Pavel Smrž, Ph.D. is an associate professor at the Faculty of Information Technology, Brno University of Technology, Czech Republic, where he leads the Knowledge Technology Research Group. His research interests include human-machine interaction, embedded intelligence, machine learning, and big data processing. He has participated in many European and national research and development projects.

How Do I Get There? Overcoming Reachability Limitations of Constrained Robotic Workspaces in Augmented Reality Applications

Daniel Bambušek, Zdeněk Materna, Michal Kapinus, Vítězslav Beran, Pavel Smrž
Faculty of Information Technology, Brno University of Technology
Brno, Czech Republic
{bambusekd,imaterna,ikapinus,beranv,smrz}@fit.vut.cz

***Abstract*—The paper presents an approach for handheld augmented reality in constrained industrial environments, where it might be hard or even impossible to reach certain poses within a workspace. Therefore, a user might be unable to see or interact with some digital content in applications like visual robot programming, robotic program visualizations, or workspace annotation. To overcome this limitation, we propose a temporal switching to a non-immersive virtual reality that allows the user to see the virtual counterpart of the workspace from any angle and distance using a unique combination of on-screen control complemented by the physical motion of the handheld device. Using such a combination, the user can position the virtual camera roughly to the desired pose using the on-screen control and then continue working just as in augmented reality. To explore how people would use it and what the benefits would be over pure augmented reality, we chose a representative task of object alignment and conducted a study. The results revealed that mainly physical demands, which is often a limiting factor for handheld augmented reality, could be reduced and that the usability and utility of the approach are rated as high. In addition, suggestions for improving the user interface were proposed and discussed.**

***Index Terms*—augmented reality; virtual reality; mixed reality; transitional interface; constrained industrial environments**

I. Introduction

With the broad availability of augmented reality (AR) capable devices, such as tablets or head-mounted displays (HMD), more industry applications are emerging. AR has been found useful in various areas – visualizations of robot programs and motions [1], inspection and maintenance [2], training [3,4] or robot programming [5,6]. Although the AR seems promising and useful, it may not be usable universally, especially in the context of industry environments. These environments tend to be very limited in terms of space because of both safety and high utilization of floor space **[[nějakej zdroj, co by to zkoumal/potvrzoval?]]**. In scenarios, such as spatially situated visual programming [7], where the user needs to interact with some digital content bound to the physical places of the workplace (fitting bounding boxes around the real objects, workplace annotation, positioning of spatial actions, etc.), it may be physically challenging or even impossible to reach those places, due to the inconvenient workplace layout,

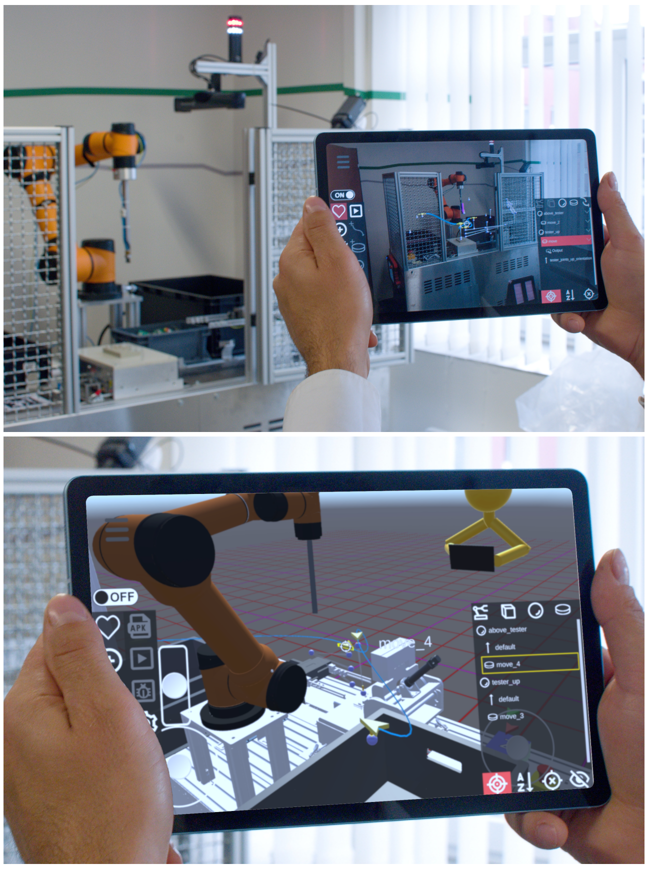


Fig. 1. A use case of the proposed solution. **Top:** An example of workspace, where the robot is caged and accessible from the front and left sides only. A user programming the robot in AR by setting spatial actions bound to physical places has a limited working area and cannot easily access the places behind the robot or the cage. **Bottom:** The user switched to VR, where the virtual counterpart of the workspace is displayed, and positioned the virtual camera to physically inaccessible pose to get a view from behind the workspace.

large machines, or safety fences (see Fig. 1). When comes to handheld AR applications, users also suffer from a limited depth perception, mainly because of monoscopic displays. Although this problem could be mitigated by applying the depth cues, such as shadows, occlusion, or shading [8–10], it still requires looking at the content from at least two

This project was implemented with financial support from the state budget through the Ministry of Industry and Trade in the TRIO program (FV40052).

different points of view to achieve the most precise depth estimation results [11]. This further supports the idea that within the constrained industrial environments, it may be dangerous or even impossible to get to certain viewpoints to estimate the depth of AR content precisely. Moreover, the design of handheld AR forces users to have their arms in mid-air position, pointing the device's camera towards the observed scene, which may result in discomfort, fatigue, or pain when using the device for longer periods of time [12,13].

To resolve the problems of inaccessible workplace positions, depth perception, and high physical load associated with the use of handheld AR, we propose a solution that enables a temporal switching from the AR to a non-immersive VR, where the user can see the virtual counterpart of the workspace from any angle and distance and can work with the virtual content without any physical limitations. Additionally, in case of spatial visual programming, it enables the user to program the robot even during program executions, where it would be dangerous to move around the workspace. It also gives users the option to sit down, place the device in a comfortable position and do the work in VR instead of AR, when it is convenient.

We focus on mobile-AR on a tablet device mainly, because they are more accessible for small and medium enterprises, thanks to low price (compared to AR head-up displays), and are simple to control using the touch screen, which most people are already familiar with because of the widespread use of smartphones.

II. Related Work

Throughout the past years, many works on using AR for HRI emerged. Some explored robot programming through setting trajectory waypoints using HMD [14–16], or HMD in combination with handheld pointer [17], some combined visual programming with visualization of spatial waypoints in the workplace [18], and some explored visualization of robot motions [1]. However, most of the literature relies on the ideal workplace layout and completely omits possible limitations of industry environments (hard-to-reach places, safety fences) **[[pohledat zdroje, který by se zabývaly rozložením pracoviště, či programováním robotů v klecích, na nepřístupných místech, atp.]]**.

Recently, frameworks as Google ARCore[1] or Apple ARKit[2] enabled fast and easy development of AR applications for phones and tablets, which are in general significantly more affordable than HMD devices as i.e. Microsoft HoloLens, and thanks to their widespread, their usage is well known for users. Compared with a phone, a tablet provides better ordinal depth perception and seems to be preferred by users [19,20]. However, consumer-grade handheld devices can not so far deliver stereoscopic visualization, and therefore the need for viewing a scene from different angles increases, especially in tasks of, e.g., aligning virtual and real objects [21,22]. At the same time, for certain tasks, it might be beneficial to observe the scene from a physically demanding or even impossible pose, e.g., to zoom in or to switch to a bird's eye perspective. The issue has been addressed multiple times by enabling users to switch from AR (egocentric) visualization to a VR (exocentric) one for various applications such as 3D model exploration [23,24], points of interest navigation [25] or furniture arrangement [26]. Despite such a feature inherently increases the complexity of the user interface and therefore mental effort, existing results show that it might be faster and more preferred by users [21] or lead to better scene understanding [20,26]. Additionally, tasks that were not possible with pure AR can be performed [27].

Transitioning into immersive VR requires either HMD capable of both AR and VR [26] or a user must put on a VR-capable HMD when needed [28]. When using a tablet-based AR without any additional device, there is the only possibility to transition into non-immersive VR. In this case, there is a question of how to control the virtual camera position, where for instance [23] proposes the usage of real physical objects and natural motions. Moreover, according to our knowledge, the utility and usability of a transitional approach have not been so far evaluated within an industrial context.

[1]developers.google.com/ar
[2]developer.apple.com/augmented-reality/arkit/

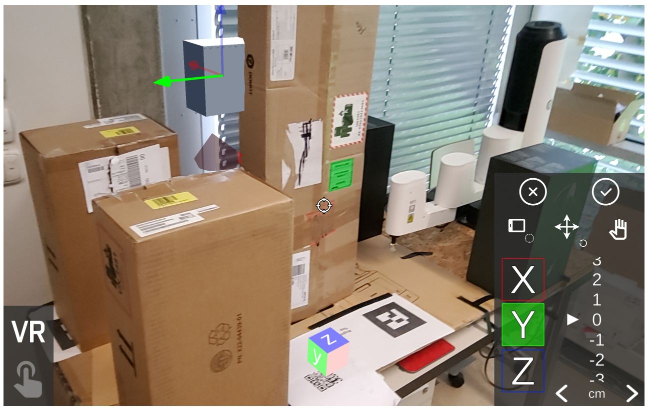


Fig. 2. An example of a fitting task, where a user is trying to fit the grey object onto the red one, which is occluded by the real-world environment.

III. Proposed Approach

Handheld AR applications in constrained industrial environments suffer from reachability limitations, where it may be hard or even impossible to reach certain locations within the workspace in order to interact with the AR content. Users are also limited in terms of moving around the workplace freely, because of inconvenient workplace layout or safety, due to presence of large robots or robot fences. Handheld devices also imply a high physical load when used for a longer periods of time [12,13,29]. Therefore we propose an approach of temporal switching to a VR environment, where the user controls a virtual camera thanks to which is able to see the workplace and the virtual content from any position.

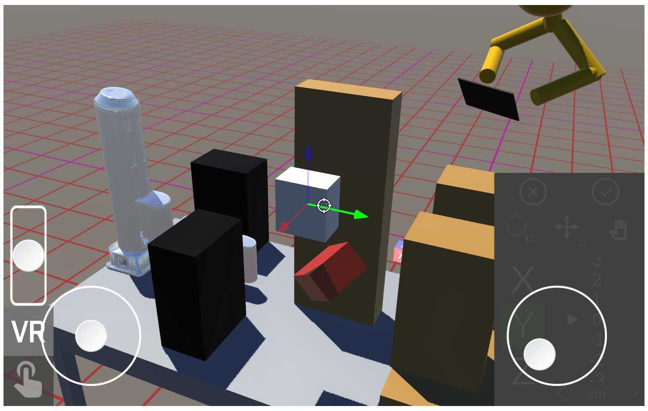


Fig. 3. A user switched to the VR and moved the camera to the physically inaccessible position in order to get a better view of the scene.

A. *Seamless Switching Between AR and VR*

Two cameras are needed to enable switching between AR and VR – one mediating the AR (*AR camera*), controlled by AR-capable framework (e.g. Google's ARCore of Apple's ARKit), and the second one for displaying the VR environment (*VR camera*). When the user switches the interface to VR, a virtual counterpart of the workspace is displayed and the VR camera is transformed to precisely match the AR camera transformation. Using this approach, where the virtual environment is displayed on the exact same place as the real one, should limit potential disorientation issues. While in VR, having the AR framework still active to continuously track the device's position in the real world enables immediate switch back to AR, without the need of tracking re-establishment, which would normally consume some time. Additionally, available tracking data can be used to render a virtual model of the person in the scene, which could help the user to better orient in VR environment. Switching back to AR resets the VR camera back to the position of the AR camera, thus to the user's position.

B. *Movement in VR Environment*

To control the VR camera, we propose a unique interaction mode using a combination of on-screen controls and physical motion of the handheld device. As a representative of the on-screen controls, we chose on-screen joysticks. The joystick approach was chosen mainly because of the effort to reduce the physical load associated with the use of handheld devices. The controls can be situated on the sides of the device, allowing the user to hold it with both hands, thus spreading its weight. There are three joysticks in total – one for moving the camera in the axis perpendicular to the ground, second for moving in the plane parallel to the ground, and third for rotating the camera. Apart from joysticks approach, conventional gestures such as drag and pinch can be used.

The on-screen controls are complemented by the physical motion of the handheld device. That means the VR camera is mapped to the motion of the AR camera, as tracked by the AR framework, which enables the same level of interaction as in AR. This functionality is achieved by keeping the AR framework for device's tracking enabled the whole time to get the device's transformation matrix for each frame and use it to transform the VR camera.

Using this combination of controls, a user can move the VR camera using the joysticks to the position where they want to operate and then continue the work by moving the tablet physically, just like when they are in AR.

C. *Acquisition of the VR Environment*

To complete the AR to VR switching concept, a virtual counterpart of the real environment must exist, either modelled in advance or dynamically generated. We assume that for most industrial workplaces, there is an available model. This model can be used as a base offline layer, which is sufficient in scenarios, where the workplace is static and no live-action is needed in the VR.

The base offline layer can be complemented by online data acquired from various sensors, either present on the handheld device or integrated to the workplace, in the form of point clouds [30], detected surfaces [22,31] or reconstructed 3D scene [32]. Since the current configuration of the robot is known to the system, its 3D model in the same configuration could be visualized as well, to further support the immersity of the visualization and help the user with the orientation.

IV. Experiment Application Design

For the sake of experimental evaluation of the proposed approach, the prototype application was prepared and implemented using the Unity3D and the AR Foundation framework[3], which encapsulates the Google's ARCore. The application was specifically developed for the Samsung Galaxy Tab S6 device.

The virtual counterpart of the workspace was modelled in advance, to match the physical workspace where the experimental evaluation should happen. It comprises of a table, a robot and several cardboard boxes simulating an industrial-like workspace (see Fig. 3). Besides the static virtual model of the workspace, a simplified model of a person holding a tablet is rendered in the scene with the position and orientation of the user (using the camera pose estimation provided by the ARCore framework).

The VR camera is controlled by the proposed interaction mode that uses a combination of three on-screen joysticks with a physical motion of the tablet. We have designed the user interface to support long-term usage by allowing the user to hold the tablet with both hands, which spreads the weight of it. According to that, all controllable elements of the user interface are situated in bottom corners of the screen to be reachable by thumbs. Therefore, two joysticks are placed in the left bottom corner (one for moving in the axis perpendicular to the ground, second for moving in the plane parallel to the ground) and one is placed in the right bottom corner and serves

[3]docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual

for rotating the camera. Besides the proposed interaction mode – the *hybrid joystick* control, we also included a *pure joystick* control, which disables the mapping of the tablet motion to the VR camera, so the camera can be controlled by the on-screen joysticks solely. We added the possibility to switch between these two control modes at any time to observe what type of control would users prefer the most.

As a representative task for our experiment, a simple fitting problem was selected, where users had to align a virtual object with another one (details in the Section V-A). According to that, the user interface offers a set of elements for manipulating with the virtual object, consisting of a transform menu with several 2D elements and a 3D gizmo (see Fig. 2). To select an axis, three buttons with corresponding colors (matching the colors of the different axis on the 3D gizmo) can be used. The selected axis is represented by the solid background of corresponding button and a high-contrast color on the gizmo. A simple two-state button is used for switching between translation and rotation of the object. To translate or rotate the virtual object, the so-called transform wheel can be used. It can be scrolled either up or down, depending if the object should go in the direction of the gizmo arrow (scrolling down to positive values) or against it (scrolling up to negative values). The numbers indicate the number of units by which the object will be moved or rotated. Unit precision (centimeters, millimeters, etc.) can be changed by the units selector placed beneath the wheel.

The transform wheel was designed to be used for a precise manipulation with the virtual object. For fast manipulation, we included a mode of free-form movement, where the movement of the object copies the movement of the physical device. It can be activated by holding the button with an icon of palm and is especially useful for the initial, rough positioning of the objects. A last set of buttons contains a button with an x-cross inside, which can be used to reset the currently manipulated object to its initial pose and a button with a check-mark that is used to confirm the final pose.

V. Evaluation

A study was conducted to evaluate the proposed approach and observe users' behavior. It was designed as a within-subject and took place in an office-like environment. As a representative task, a simple fitting was selected (details in the Section V-A). Prior to the experiment, a preliminary test took place.

A. Task

The task consisted of multiple alignments of two virtual objects, inspired by [33]. It was also inspired by the industry, where is a common need for annotating real-world objects, encapsulating them by collision bounding boxes, etc. However, to be able to evaluate the precision of alignment, we chose to align a virtual on a virtual object instead of a virtual on a real one. A grey object was first displayed above the workplace for each sub-task to be easily visible. A slightly larger red object with a different orientation was displayed within the workplace. The user had to align the grey one over the red one. The positions of red objects were chosen to be in hard-to-reach and hard-to-see areas (occluded by the environment) and were the same for all participants and both conditions. The alignment was done using a menu with controls for axis selection and manipulation (see Fig. 2). After a user was satisfied with the precision of alignment, they pressed the confirmation button, and another set of objects was shown.

B. Conditions

Two conditions were tested: C_{AR} as AR-only baseline and $C_{AR/VR}$, which allowed switching between AR and VR. The order of conditions was randomized.

C. Hypotheses

Prior to the experiment, the following hypotheses were formulated:

- **H1** – C_{AR} will have a higher perceived physical load.
  In C_{AR}, users will have to perform all motions physically while with $C_{AR/VR}$, part of the physical motion will be compensated by moving a virtual camera.
- **H2** – C_{AR} will have a lower perceived mental load.
  $C_{AR/VR}$ will introduce new interface controls and will be more difficult to learn; therefore, the perceived mental load will be lower for C_{AR}.
- **H3** – There will be no significant difference in achieved precision.
  Achieving the same precision will be possible, just more demanding with C_{AR}, due to the need to move physically.
- **H4** – With C_{AR}, manipulation trajectory will be significantly longer.
  In $C_{AR/VR}$, users will have the option to position the virtual camera to a better viewpoint from which they will be more effective in positioning the virtual object.

D. Participants

For the evaluation, 20 participants were recruited, mainly from university staff and students with the mean age 28.50 ($SD = 6.225$) years. 16 participants identified them as *male*, 4 as *female*, and 0 as *other / prefer not to specify*. Participants were prior to the experiment asked to assess on a Likert scale from 1 (no experience) to 5 (very experienced) their experience with 3D authoring software (mean value 2.25, CI $[1.59, 2.91]$), AR (2.15, CI $[1.59, 2.91]$), playing computer games (3.25, CI $[2.65, 3.85]$) and usage of either real or virtual controls as joysticks or gamepads (2.60, CI $[2.04, 3.16]$). After completing the evaluation, each participant was given a small material reward.

E. Setup

The setup consisted of a table equipped with a non-functioning robot (Dobot M1) in order to create an industrial-like impression and several cardboard boxes simulating obstacles or machines. The table was fully accessible only from the front, non-accessible from the rear, while left and right sides were moderately difficult to access. The purpose was to simulate realistic settings, where unobstructed access from any side can be hardly achieved. There was a chair available next to the table. The user interface for evaluation was run on the Samsung Galaxy S6 tablet. The application was also used to collect objective metrics such as task time, achieved accuracy, or traveled distances. Subjective metrics (NASA TLX [34] and a set of custom questions) were collected using a questionnaire. The whole experiment was video recorded for later analysis.

F. Protocol

Each participant was first briefly told about the experiment's purpose and then asked to give a signed consent with it and its recording. Then they filled in the first part of the questionnaire containing demographic data and custom questions regarding experience with several factors that may influence results. After that, a participant was assigned the first condition and completed a training consisting of alignment of one easy-to-reach object. They were asked to work fast and precisely at the same time, no exact duration of the test was mentioned. Practically, there was a soft 20 minutes deadline meaning they could finish the last object after the given time passed out. After completing the first part, they filled in the TLX questionnaire and continued with the other condition. When both conditions were finished, participants answered a set of questions related to VR mode usability and preference, and then debriefing with a moderator took its place.

G. Preliminary test

In advance of the main evaluation, a preliminary test with two participants was conducted to validate the task, the experimental protocol, and the developed application.

In the initial draft of the experimental protocol, each participant was supposed to align 20 targets in both conditions. The preliminary test showed us that this would be too demanding for most of the participants and the session would be too long, potentially causing arms fatigue for the participants. This could negatively affect the performance of the participant's second condition, as the arms would be already tired from the first condition. For that reason, a soft limit of 20 minutes was incorporated into the experimental protocol.

Also, issues with the virtual joysticks were revealed and therefore, non-linear control was implemented and sensitivity and dead zone of all joysticks were tuned, so both fast and precise movements are comfortably available.

VI. Results

As the experiment was designed as a within-subject, we treated all obtained subjective and objective measures as paired data. In order to examine differences between conditions, for each measure, a paired t-test was done if the distribution was normal or Wilcoxon's test otherwise. The False Discovery Rate (FDR) method was applied to cope with multiple comparisons problem, namely Benjamini/Hochberg approach.

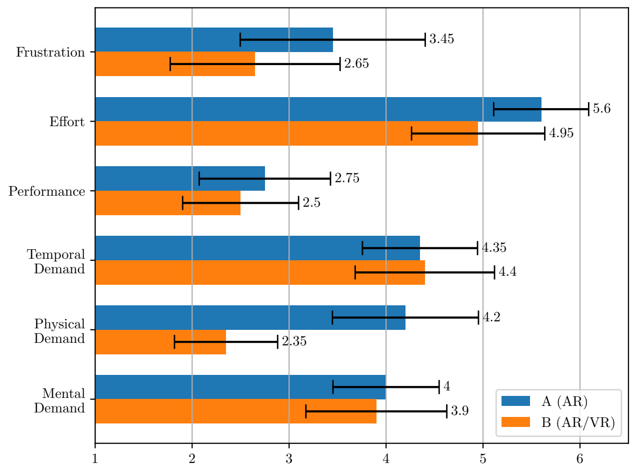


Fig. 4. Mean values for subscales of (raw) NASA TLX (in the range of $[1, 7]$) with their respective 95 % confidence intervals for both conditions.

A. Statistical Data

The obtained data can be seen in Table I. A paired t-test was used to detect significant differences between conditions for the first seven normally distributed metrics and Wilcoxon's test was used for the rest. The FDR method (Benjamini/Hochberg) was used to compensate for multiple comparisons. We found a significant difference for the *Physical Demand* dimension of TLX ($p = 0.002$), which was lower for $C_{AR/VR}$, and for *Time to Align*, which is the time needed to align one object and which was lower for C_{AR}. There were also noticeable differences in *Raw TLX* (overall TLX) and *Effort*, however not significant. The overall (raw) mean TLX for condition C_{AR} was 50.97 and for $C_{AR/VR}$, it was 40.97, which is lower than at least 50 % of studies analyzed [35]. The individual subscales of TLX (scaled to range $[1, 7]$) are shown in Fig. 4.

The **H1** hypothesis is supported by subjective metric *Physical Demand*, which is significantly higher for condition C_{AR}. For the metric *Mental Demand*, there is no significant difference, and therefore we have to reject the **H2** hypothesis, which is actually positive. Although $C_{AR/VR}$ does not seem to be more mentally demanding than C_{AR}, as we expected, there is still room for improvements, especially regarding control using virtual joysticks, which was rated as rather hard to use (question "VR easy to control by joysticks", see Fig. 5). Regarding **H3**, a significant difference was not found for *Alignment Accuracy*. Moreover, a two-one-sided t-tests procedure (TOST) was used to test for equivalence of both modalities, with bounds $\Delta_L = -5$ and $\Delta_U = 5$. With a p-value 0.00002, we have to reject the TOST hypothesis that the difference between the two samples is larger than the selected bounds, and therefore we can not reject the hypothesis **H3**. For the *Object Distance* metric, there is some difference, however, not a significant one, so the hypothesis **H4** has to be rejected.

Surprisingly, there is a significant difference in *Time to Align* objective metric (time to align one object), meaning that the performance was higher under condition C_{AR}. In another word,

TABLE I
SUBJECTIVE (TLX) AND OBJECTIVE METRICS, MEANS WITH 95 % CI, STATISTICS, FDR-CORRECTED P VALUES, AND THE RAW ONES. A PAIRED T-TEST WAS USED TO DETECT SIGNIFICANT DIFFERENCES BETWEEN CONDITIONS FOR THE FIRST SEVEN METRICS WHICH WERE NORMALLY DISTRIBUTED AND WILCOXON'S TEST WAS USED FOR THE REST. THE FDR METHOD (BENJAMINI/HOCHBERG) WAS USED TO COMPENSATE FOR MULTIPLE COMPARISONS. METRICS MARKED WITH * ARE NORMALIZED ACCORDING TO A NUMBER OF OBJECTS THAT WERE ALIGNED.

metric	**C_{AR} (mean/CI)**	**$C_{AR/VR}$ (mean/CI)**	**statistic**	**corrected p-val**	**original p-val**
Raw TLX (0-100)	50.97 [43.90, 58.05]	40.97 [32.24, 49.71]	2.05	0.1683	0.055
Mental Demand (1-7)	4.00 [3.45, 4.55]	3.90 [3.17, 4.63]	0.29	0.943	0.772
Physical Demand (1-7)	4.20 [3.45, 4.95]	2.35 [1.82, 2.88]	4.63	**0.002**	0.0002
Temporal Demand (1-7)	4.35 [3.76, 4.94]	4.40 [3.68, 5.12]	-0.13	0.968	0.899
Performance (1-7)	2.75 [2.07, 3.43]	2.50 [1.90, 3.10]	0.75	0.717	0.460
Effort (1-7)	5.60 [5.11, 6.09]	4.95 [4.26, 5.64]	1.99	0.1683	0.061
Frustration (1-7)	3.45 [2.50, 4.40]	2.65 [1.77, 3.53]	1.54	0.31	0.141
Alignment Accuracy* [%]	94.18 [92.51, 95.85]	93.54 [91.33, 95.76]	94.00	0.968	0.968
Time to Align* [s]	162.43 [127.76, 197.10]	230.99 [175.94, 286.03]	23.00	**0.007**	0.001
Physical Distance* [m]	14.67 [12.11, 17.23]	15.28 [10.29, 20.27]	3.33	0.7173	0.521
Object Distance* [m]	25.05 [16.42, 33.69]	21.10 [13.42, 28.79]	2.17	0.712	0.388

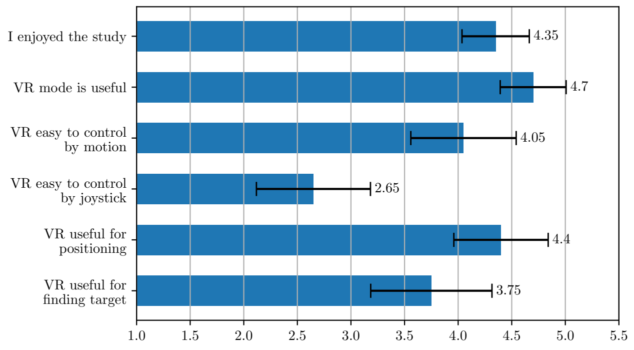


Fig. 5. Mean values for participants' agreement with a set of custom questions on a Likert scale (where 1 means "I totally disagree" and 5 means "I totally agree"), with their respective 95 % confidence intervals.

users were able to align more objects within a given time. We speculate, that users were slower with $C_{AR/VR}$ condition because of usability issues with controlling the viewpoint. In contrast, the self-reported *Temporal Demand* and *Performance* differ very little for both conditions.

In summary, participants rated the usefulness of the VR mode as rather high, with exception of controlling viewpoint by virtual joysticks (see Fig. 5), and spent 70 % of the time within it.

B. Observation Findings

When using the $C_{AR/VR}$ condition, 11 participants took advantage of sitting down on a nearby chair to get to a more comfortable and relaxed body pose while in VR (they were not previously instructed to do so). The 5 of them sat most of the time and were in VR for the whole experiment duration. The most typical poses observed during the experiment are depicted in Fig. 6. Regarding the movement, 17 participants used the combination of the on-screen joysticks and the motion of the tablet, of which 5 preferred joystick movement slightly more. The remaining 3 participants chose to use joysticks only, complaining about the troubles of simultaneous coordination of two different sources of movement. 7 participants used the option of fixing the view by disabling the motion-based control in order to finish the task by using on-screen joysticks solely.

In favor of pure AR, 3 users claimed that it is easier to use, faster, and enables a faster and more natural look-around about the workplace. On the other hand, 5 claimed that while in VR, they felt more confident and productive. Using their words, the more time to position the virtual camera was compensated by lower physical demands. Some claimed that VR is good for improved spatial imagination about the workplace, good for fast search of a virtual object. VR felt more comfortable for the experiment task, mostly because of the possibility to sit down and relax comfortably, instead of moving around the workplace physically. 3 participants used the AR only to quickly find the virtual object in the real world, position it roughly using the physical motion of the tablet, and then switched to VR, sat down, and finished the fitting task from the chair. This supports the idea, that AR and VR can coexist together in industrial applications when AR can be more effective for natural interaction with the virtual content within the workspace, while VR can be used to do the work more comfortably when AR is no longer needed.

5 participants had large troubles using joysticks. They complained about the difficulty of controlling them, non-intuitiveness, and low sensitivity. 3 of them would replace the joysticks with conventional gestures, like drag and pinch. 2 would prefer moving in the camera coordinate system, instead of the global one (parallel to the ground plane). 1 even suggested connecting the physical joysticks to the tablet to control the VR movement.

During the experiment, we also observed how the participants were searching for the virtual objects within the scene. 1 used the VR to get the camera to the bird's eye view in order

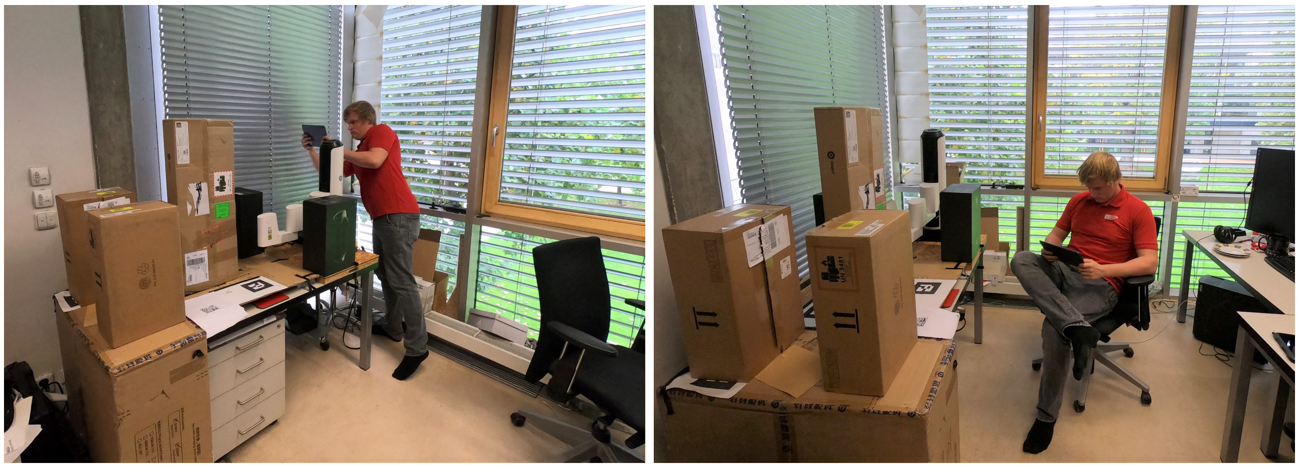


Fig. 6. Demonstration of typical poses during the experiment. **Left:** While in C_{AR} condition, users often had to stretch themselves into uncomfortable postures in order to get the needed view about the virtual content. **Right:** When having the option to switch to VR, approximately half of the participants used a chair to sit down and performed the task or part of it in a more comfortable and relaxed pose.

to get a fast outlook on the scene and to find the task-related objects faster. Some used the AR camera solely for finding virtual objects in the workplace. Exactly 3 started always in AR, searched for the virtual object, moved it approximately to the target position, switched to VR, moved the virtual camera to better view, and finished the task in VR.

Switching from VR back to AR seemed confusing for one participant. After moving the virtual camera, working on the task from a different angle than the participant's physical position, and switching back to AR, they expected the virtual content to appear at the same place as it was in VR. This problem could be mitigated by smooth automatic camera movement back to the user's physical position.

VII. Discussion and Proposed Improvements

[[

- **Konkrétní návrhy, jak teda zlepšit ovládání VR scény:**
  - **Přednastavené viewpointy, kterými by klikali (viz paper "Exploring real world points of interest: Design and evaluation of object-centric exploration techniques for augmented reality", kde jim ale právě ty viewpointy vyšly jako nejhůř používané - dalo by se zhodnotit a toto zavrhnout).**
  - **Manipulace přímo se scénou - otáčím a zoomuju scénu, nikoliv free-look kameru.**

]]

Based on the gained experience from the conducted experiment, we propose several improvements of our approach for handheld-based AR in the context of industrial environments. As movement in the VR environment using joysticks appeared to be non-intuitive and hard to control to some participants, especially those without any experience with game-pads, we focused mainly on improving this particular problem. One possible solution would be to replace the joysticks with predefined viewpoints, as Tatzgern et al. [24] proposed in their work for exploring distant landmarks in AR. They compared the viewpoint approach for exploring the virtual model of the landmark with free-form manipulation of the model. Based on the results, the free-form manipulation outperformed the viewpoint switching, mainly due to disorientation problems when changing the views. Therefore, we transformed our free-form movement using joysticks into a free-form manipulation with the scene.

A. *Improved Movement in VR Environment*

Instead of free-form VR camera movement in the VR environment using three joysticks, which was rated as rather hard to use during the conducted experiment, we propose a control method that should reduce the mental demand induced by too many control inputs (three joysticks with different functionality) by reducing the number of inputs and employing a more natural interaction technique – gestures. The method uses two gestures only – pinch and drag and fully compensates the joystick-based approach. As depicted in Fig. 7, the VR camera can be rotated around the z axis

VIII. Conclusions

The presented approach aims to improve the usability of handheld AR, especially in constrained industrial environments, where reaching certain poses might be difficult or impossible due to limited space or safety measures. It allows users to temporarily switch from AR to non-immersive VR, where they can freely adjust viewpoint in the virtual counterpart of the real environment using on-screen controls in conjunction with moving a tablet physically. In order to evaluate the approach and gain knowledge about users' behavior, the within-subject user study ($n = 20$) was conducted, where an object alignment was selected as a general representative task.

The results have shown that the ability to switch temporarily between AR and VR has potential as physical demands were significantly lower when compared to pure AR, whereas

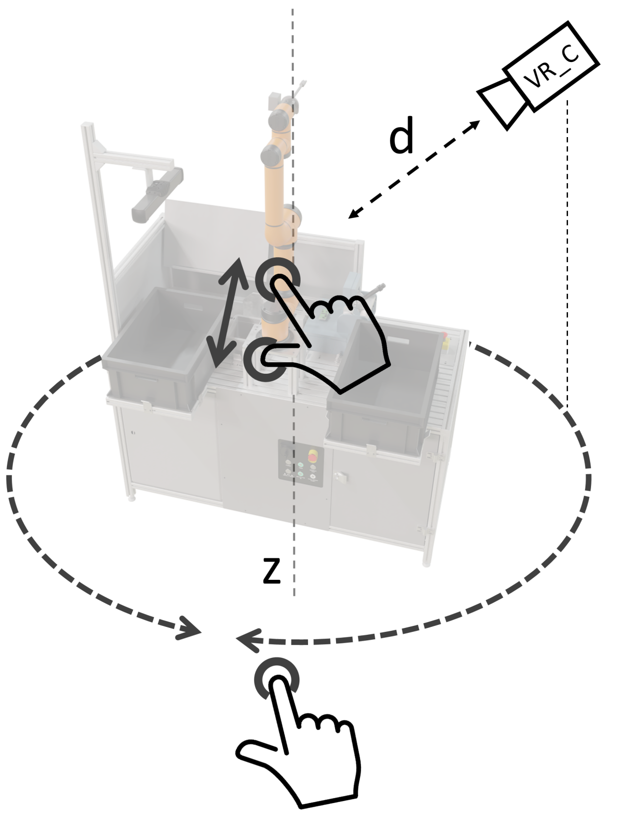


Fig. 7. Proposed approach for improved movement in the virtual scene where users, instead of moving the virtual camera using joysticks, are manipulating the scene directly using pinch gesture for zooming and drag gesture for rotating the scene in vertical axis.

physical demands could be considered as a limiting factor for AR usage. User observations have revealed that the VR mode was often used to get an overview of the workspace, to find an occluded object, or to avoid an uncomfortable position.

In summary, the approach provided a clear advantage regarding the physical load of users and was well accepted. The possible applications could be, i.e., workspace annotation, spatially-oriented visual programming, monitoring of processes, etc.

In future work, we will investigate other possibilities of controlling the free-viewpoint in order to improve usability, as joysticks were evaluated as cumbersome and complicated. Moreover, adding smooth transitions between AR and VR displays may help to enhance users' experience. It could also be interesting to compare the proposed approach with a pure VR. Last but not least, the proposed solution will be integrated into a framework for visual robot programming in AR.

REFERENCES

[1] E. Rosen, D. Whitney, E. Phillips, G. Chien, J. Tompkin, G. Konidaris, and S. Tellex, "Communicating robot arm motion intent through mixed reality head-mounted displays," in *Robotics research*. Springer, 2020, pp. 301–316.

[2] H. Eschen, T. Kötter, R. Rodeck, M. Harnisch, and T. Schüppstuhl, "Augmented and virtual reality for inspection and maintenance processes in the aviation industry," *Procedia manufacturing*, vol. 19, pp. 156–163, 2018.

[3] E. Z. Barsom, M. Graafland, and M. P. Schijven, "Systematic review on the effectiveness of augmented reality applications in medical training," *Surgical endoscopy*, vol. 30, no. 10, pp. 4174–4183, 2016.

[4] S. Werrlich, K. Nitsche, and G. Notni, "Demand analysis for an augmented reality based assembly training," in *Proceedings of the 10th International Conference on PErvasive Technologies Related to Assistive Environments*, 2017, pp. 416–422.

[5] Z. Materna, M. Kapinus, V. Beran, P. Smrž, and P. Zemčík, "Interactive spatial augmented reality in collaborative robot programming: User experience evaluation," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2018, pp. 80–87.

[6] S. Blankemeyer, R. Wiemann, L. Posniak, C. Pregizer, and A. Raatz, "Intuitive robot programming using augmented reality," *Procedia CIRP*, vol. 76, pp. 155–160, 2018.

[7] M. Kapinus, V. Beran, Z. Materna, and D. Bambušek, "Spatially situated end-user robot programming in augmented reality," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2019, pp. 1–8.

[8] D. Schmalstieg and T. Hollerer, *Augmented reality: principles and practice*. Addison-Wesley Professional, 2016.

[9] M. Berning, D. Kleinert, T. Riedel, and M. Beigl, "A study of depth perception in hand-held augmented reality using autostereoscopic displays," in *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2014, pp. 93–98.

[10] T. D. Do, J. J. LaViola, and R. P. McMahan, "The effects of object shape, fidelity, color, and luminance on depth perception in handheld mobile augmented reality," in *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2020, pp. 64–72.

[11] J. E. Swan, L. Kuparinen, S. Rapson, and C. Sandor, "Visually perceived distance judgments: Tablet-based augmented reality versus the real world," *International Journal of Human–Computer Interaction*, vol. 33, no. 7, pp. 576–591, 2017.

[12] S. Boring, M. Jurmu, and A. Butz, "Scroll, tilt or move it: using mobile phones to continuously control pointers on large public displays," in *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7*, 2009, pp. 161–168.

[13] J. D. Hincapié-Ramos, X. Guo, P. Moghadasian, and P. Irani, "Consumed endurance: a metric to quantify arm fatigue of mid-air interactions," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2014, pp. 1063–1072.

[14] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. M. Van der Loos, and E. Croft, "Robot programming through augmented trajectories in augmented reality," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1838–1844.

[15] S. Y. Gadre, E. Rosen, G. Chien, E. Phillips, S. Tellex, and G. Konidaris, "End-user robot programming using mixed reality," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 2707–2713.

[16] M. Ostanin and A. Klimchik, "Interactive robot programing using mixed reality," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 50–55, 2018.

[17] S. Ong, A. Yew, N. Thanigaivel, and A. Nee, "Augmented reality-assisted robot programming system for industrial applications," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101820, 2020.

[18] E. Yigitbas, I. Jovanovikj, and G. Engels, "Simplifying robot programming using augmented reality and end-user development," *arXiv preprint arXiv:2106.07944*, 2021.

[19] A. Dey, G. Jarvis, C. Sandor, and G. Reitmayr, "Tablet versus phone: Depth perception in handheld augmented reality," in *2012 IEEE international symposium on mixed and augmented reality (ISMAR)*. IEEE, 2012, pp. 187–196.

[20] U. Riedlinger, L. Oppermann, and W. Prinz, "Tango vs. hololens: A comparison of collaborative indoor ar visualisations using hand-held and hands-free devices," *Multimodal Technologies and Interaction*, vol. 3, no. 2, p. 23, 2019.

[21] M. Sukan, S. Feiner, B. Tversky, and S. Energin, "Quick viewpoint switching for manipulating virtual objects in hand-held augmented reality using stored snapshots," in *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2012, pp. 217–226.

[22] B. Nuernberger, E. Ofek, H. Benko, and A. D. Wilson, "Snaptoreality: Aligning augmented reality to the real world," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 1233–1244.

[23] M. Billinghurst, H. Kato, and I. Poupyrev, "The magicbook: a transitional ar interface," *Computers & Graphics*, vol. 25, no. 5, pp. 745–753, 2001.

[24] M. Tatzgern, R. Grasset, E. Veas, D. Kalkofen, H. Seichter, and D. Schmalstieg, "Exploring real world points of interest: Design and evaluation of object-centric exploration techniques for augmented reality," *Pervasive and mobile computing*, vol. 18, pp. 55–70, 2015.

[25] A. Mulloni, A. Dünser, and D. Schmalstieg, "Zooming interfaces for augmented reality browsers," in *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, 2010, pp. 161–170.

[26] M. Mori, J. Orlosky, K. Kiyokawa, and H. Takemura, "A transitional ar furniture arrangement system with automatic view recommendation," in *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*. IEEE, 2016, pp. 158–159.

[27] R. Grasset, A. Dünser, and M. Billinghurst, "Moving between contexts-a user evaluation of a transitional interface," 2008.

[28] J. S. Roo and M. Hachet, "Towards a hybrid space combining spatial augmented reality and virtual reality," in *2017 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 2017, pp. 195–198.

[29] E. E. Veas and E. Kruijff, "Handheld devices for mobile augmented reality," in *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia*, 2010, pp. 1–10.

[30] S. N. B. Gunkel, H. M. Stokking, M. J. Prins, N. van der Stap, F. B. t. Haar, and O. A. Niamut, "Virtual reality conferencing: Multi-user immersive vr experiences on the web," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 498–501.

[31] Y.-C. Kung, Y.-L. Huang, and S.-Y. Chien, "Efficient surface detection for augmented reality on 3d point clouds," in *Proceedings of the 33rd Computer Graphics International*, ser. CGI '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 89–92.

[32] T. Zhou, Q. Zhu, and J. Du, "Intuitive robot teleoperation for civil engineering operations with virtual reality and deep learning scene reconstruction," *Advanced Engineering Informatics*, vol. 46, p. 101170, 2020.

[33] M. Krichenbauer, G. Yamamoto, T. Taketom, C. Sandor, and H. Kato, "Augmented reality versus virtual reality for 3d object manipulation," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 2, pp. 1038–1048, 2017.

[34] S. G. Hart, "Nasa-task load index (nasa-tlx); 20 years later," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 50, no. 9. Sage publications Sage CA: Los Angeles, CA, 2006, pp. 904–908.

[35] R. A. Grier, "How high is high? a meta-analysis of nasa-tlx global workload scores," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 59, no. 1, pp. 1727–1731, 2015.

Effective Remote Drone Control Using Augmented Virtuality

Kamil Sedlmajer[1], Daniel Bambušek[1] and Vítězslav Beran[1]
[1]*Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Bozetechova 1/2, Brno, 612 66, Czech Republic.*
kamilsedlmajer@gmail.com, {bambusekd, beranv}@fit.vutbr.cz

Keywords: Augmented Virtuality, UAV, Drone Piloting, Virtual Scene, Navigation Elements, First Person View, Third Person View

Abstract: Since the remote drones control is mentally very demanding, supporting the pilot with both, first person view (FPV) and third person view (TPV) of the drone may help the pilot with orientation capability during the mission. Therefore, we present a system that is based on augmented virtuality technology, where real data from the drone are integrated into the virtual 3D environment model (video-stream, 3D structures, location information). In our system, the pilot is mostly piloting the drone using FPV, but can whenever switch to TPV in order to freely look around the situation of poor orientation. The proposed system also enables efficient mission planning, where the pilot can define 3D areas with different potential security risks or set navigation waypoints, which will be used during the mission to navigate in defined zones and visualize the overall situation in the virtual scene augmented by online real data.

1 INTRODUCTION

Nowadays, the use of unmanned aerial vehicles (UAVs) extends to a wide range of areas, from rescue services and police forces to the commercial sector. Drones are used to monitor the quality of high voltage structures, the development of infrastructure outages, or to support complex interventions by rescue or police units. In all cases, the use of a drone requires high skill and mental demand for the drone operator.

Recent research let arise to various autonomous modes, where the drones are able to perform a precisely predefined mission independently and without the need for operator intervention. Unfortunately, today there is no problem in solving a wide range of tasks with the autonomous capability of the drone, but with the operator's legal constraints. For this reason, it is necessary to look for a different solution. This article deals with solving this situation by linking autonomous drone functions to operator control. The article seeks to reduce the operator's mental load in controlling drone in action using semi-autonomous drone functions.

Legal constraints today do not allow the full use of autonomous drone functions. Similarly, existing drone control solutions are now extremely burdensome for the operator. Orientation in space, keeping in safe zones, tracking key mission points, all of this makes the operation of the drone operator quite challenging.

Based on the study of existing tools and published results in the field, supported by own drone control experience, this work aims to define the key attributes of attention to the drone's operator and proposes a range of visualization and interactive features to reduce the drone operator's mental load, using augmented virtuality.

The key elements, that this study builds on, are the use of available topography maps, elevation maps, 3D data and virtual objects to enhance mission navigation clarity. The proposed solution combines existing data sources into a 3D virtual scene, augmented by online drone camera video-stream and other drone sensor data, as well as user-defined virtual objects such as safe zone boundaries or key points of a planned mission.

2 RELATED WORK

With an increasing popularity of drones, new effective methods for piloting them are emerging. Some are concentrating solely on autonomous flights with autonomous obstacle avoidance algorithms (Gageik et al., 2015) and (Devos et al., 2018), others are focusing on various user interfaces for manual drone controlling, such as gesture-based, voice-based, or remote controllers along with head-mounted displays (HMD). Recently, there were some experiments for controlling drones using brain-computer interfaces (Nourmohammadi et al., 2018) and (Mamani and Yanyachi, 2017). More common are attempts of direct drone control using gestures, which are detected with use of vision based methods from either drone attached cameras (Fernández et al., 2016), (Natarajan et al., 2018) or from additional hand tracking devices, such as the Leap Motion (Gubcsi and Zsedrovits, 2018). Unfortunately, such approaches are usable only for piloting the drone within a close distance rather than piloting it remotely. When piloting the drone remotely, but still from a visible distance, using some remote controller, it is often difficult to distinguish the drone's front face, which causes high workload on the operator. In order to lower this workload, Cho et al. introduced an egocentric drone control approach, which keeps the drone's back face automatically rotating towards the operator, who is piloting the drone from his/hers perspective (Cho et al., 2017).

To enable the user to control the drone completely remotely, without directly observing it, it is crucial to provide the user some sort of vision from the drone perspective. Currently, there are many commercially available products that are able to transmit the image from the drone attached camera into either the HMD, handheld display or regular screen in order to provide the first person view (FPV) for the operator. However, most of them only sends monocular video feed, which is insufficient in terms of perceiving distances, depths and proportions inside the FPV. The solution to overcome this problem has proved to be the attachment of two cameras to the drone to enable the stereoscopic vision inside the HMD (Smolyanskiy and Gonzalez-Franco, 2017).

Using remote controllers require attention and developed skills during piloting. Replacing them with wearable interfaces, such as exoskeleton suit with smart glove, could enable a more natural and intuitive drone control, where the operator feels more immersed (Rognon et al., 2018). Unfortunately, the operator could still struggle with awareness of drone surroundings that are out of drone camera's field of view (FoV). There comes in place the idea of placing the camera image into a virtual environment model, in order to extend the limited FoV (Calhoun et al., 2005). The image, along with the virtual model, can be augmented with waypoints, danger zones or points of interest. We believe that enabling the operator to whenever switch from the FPV into a third person view (TPV), where he/she can see whole drone situated in a virtual environment, should improve operator's situation awareness and reduce the mental load.

3 PROPOSED INTERFACE

The main goal of this work is to design a visualization system and navigation elements that will reduce the operator's mental strain when piloting the drone over longer distances. We define the following key attributes and how to resolve them:

1. Off-line and on-line spatial and sensory data fusion.
2. Virtual cameras.
3. Navigation or security structures.

Operator orientation might be improved by the TPV. The presented solution is based on virtual 3D scene augmented by real on-line data registered and visualized in the virtual scene (*Augmented Virtuality*). The 3D scene is created from the available free map data sources (topological, elevation). The 3D model of the drone inserted to the scene is controlled (position and orientation) by the data transmitted from the real drone. A video-stream from the camera on the drone is also rendered to the virtual scene. All combined spatial data must be properly registered into a local coordination frame. The pilot is then able to control the drone as usual (FPV), but can also *unlock* the camera and move to the TPV and see the drone and its surroundings. The operator's FoV is significantly expanded. This concept is depicted in the Figure 1.

The **rendering of the video-stream to the virtual scene** can be realized in several different ways. The first is to project it into a huge *virtual screen* (see Figure 1). The screen size depends on the FoV of the physical camera with either a plane or a suitably curved surface. The *screen* is moving at a fixed distance from the drone and follows the drones' position and orientation. The other option is to project the image directly onto objects in the scene. The usability of the video projection onto virtual scene requires very precise registration between the 3D virtual scene and the drone.

Virtual cameras in the 3D scene provides the user with ability to look into the scene from any viewpoint and to manipulate the camera freely. E.g. the operator can stop the drone in a safe position (safe distance from the obstacle) and move around the scene by *flying with the virtual camera*. When the virtual camera is fixed, the operator can switch back to con-

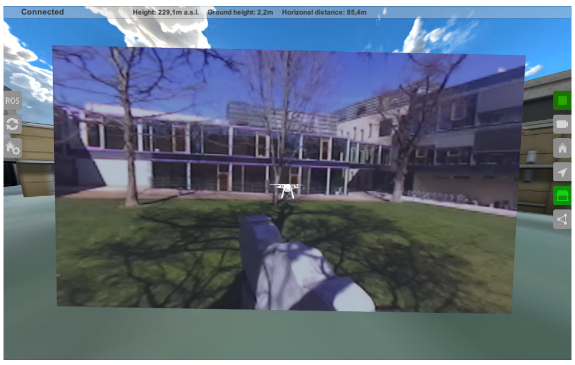


Figure 1: Concept of controlling the drone from the third person view. The virtual screen (with the video feed) moves at a certain distance in front of the drone and is synchronized with the physical gimbal configuration.

trol the drone and observe the scene from a new virtual position.

Another way to increase the safety of remote drone control is visualization of other sensor data, such as depth data, usually presented by 3D point cloud, that samples outer surfaces of objects around. This data significantly refine and complement the world model created from off-line data and allow the pilot to have a much better overview of the static obstacles around the drone, such as trees, buildings, cars, etc.

The use of augmented virtuality further provides the operator with the ability to add **navigation or security structures** to the scene. One such element may be **virtual walls**. These can define a space in the scene, such as unauthorized entry or a safe zone. Since such zones are often up to a certain height, displaying them in 3D scene is far more perspicuous than displaying them on a 2D map.

Another elements, e.g. for a flight in a more complex environment with a number of obstacles, are **navigation arrows** that point to waypoints. As a result, the pilot will always know which direction to fly and the task is just to avoid the obstacles. These waypoins can be placed not only on the ground, but also at a specific height in the air. The application should also be able to navigate back to the starting point and the pilot should be able to display both navigation arrows at once. One will point to the starting point, the other to the current waypoint. In addition to the arrows, the distance to a given point may also be displayed.

4 System Architecture

The testing application was developed using the Unity game engine. For creating the 3D virtual environment, we used the plugin MapBox[1]. The plugin automatically loads the environment maps and heightmaps in order to create the terrain in Unity. MapBox also includes several map layers with 3D building models, which are not perfect (rough building shapes, approximate heights, unrealistic textures or fake rooftop shapes), but they are still sufficient for our proof of concept. For a better quality virtual environment, virtual objects and textures from Google Maps, which are currently the best on the market, could be used. However, Google does not provide them to third parties yet.

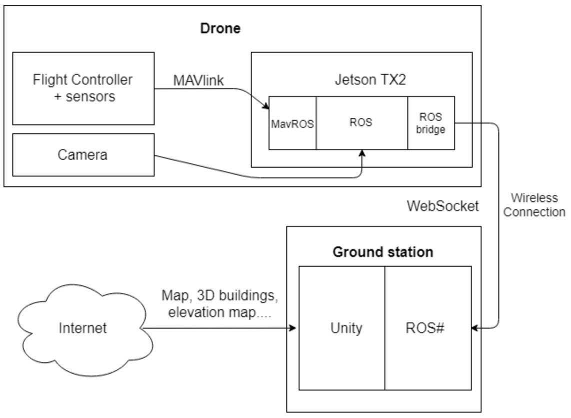


Figure 2: Communication between the drone and the ground station. On the application side, the ROS# library communicates with the drone via Rosbridge, where the data are transmitted via the WebSocket protocol. The application also automatically downloads maps, 3D buildings and heightmaps from the Internet.

For a successful combination of the virtual scene with the real drone, it is necessary to transmit following data from the drone into the ground station:

- Drone's position (pair of coordinates in WSG84 format).
- Drone's rotation (yaw – angle obtained from an electronic compass on the drone; pitch and roll).
- Compressed drone-attached camera stream.
- Drone-attached camera rotation (in respect to the drone).
- Other available sensor data (e.g. point cloud, battery status, flight speed, flight mode).

The method is based on 3D virtual scene model. The 3D virtual data are integrated from existing data sources and the coordination system is registered to actual real drone position. The GPS drone coordinates and orientation data is used for registration with virtual 3D frame. The augmentation of the virtual 3D scene is achieved by rendering the live video-stream

[1]https://docs.mapbox.com/unity/maps/overview/

from the drone front camera to projection plane in front of the virtual drone. The video latency is an important issue and highly influence the quality of the interaction with the proposed system. The video latency hardly depends on the quality of the WiFi signal, i.e. distance between the drone and the station. The additional navigation UI elements, like mission points, direction to next point or virtual walls, are rendered into virtual scene and presented to the user. The registration method might be improved by computer vision techniques, but this step will be considered later according to user tests, when the effect of rough GPS and compass based registration and caused video latency will be considered by professional pilots as an important issue.

The communication protocol is dependent on the drone manufacturer, drone's control unit and used software. In order to have customization possibilities, we built our custom experimental drone (see Figure 3), which comprises of the Pixhawk control unit with PX4 Autopilot software, Nvidia Jetson TX2 with the Ubuntu OS, stereoscopic camera, GPS and a compass. We chose the communication over WiFi between the drone and the base station. WiFi communication is limited in transmit distance, but has a high data throughput. The PX4 Autopilot uses the MAVlink communication protocol, which is transferred into the MavRos protocol of the ROS operating system (running on the Nvidia Jetson) over the Rosbridge tool. The communication scheme is depicted in the Figure 2. The system architecture is described in more detail in (Sedlmajer, 2019).

5 RESULTS

A test application was created with the implementation of several basic elements described in the previous chapter; and several real drone tests were performed. Since it was only an experimental platform with a gimbal-free camera and connection to a ground station via WiFi, which was limited in scope, it was only possible to perform the testing with limited restrictions. In spite of this, it was possible to try out a few basic use cases where the tests showed that the concept works and it is possible to control the drone with it. The application was built and tested on a laptop.

5.1 Test 1: Monitoring the area where the drone must not fly

When using a drone, for example, in the service of the police, it is often necessary to monitor an area (road,

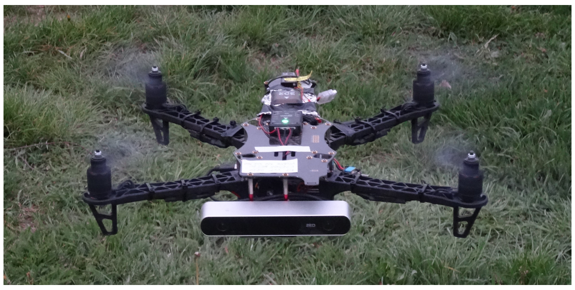


Figure 3: Test drone with Pixhawk control unit, ZED stereoscopic camera and Nvidia Jetson supercomputer, shot just above ground.

demonstration area, etc.). However, the police must also comply with the legal constraints, therefore their drones must not be flown, for example, near a highway or over a square full of people. If a pilot wants to use the drone to track the area, but to keep it out of its protective zone, he must constantly check where the drone is.

The first designed test flight was supposed to check whether adding virtual walls, representing borders of such areas, would help the pilot stay on their edge while observing what is happening around.

Virtual transparent walls were very useful in this test when flying inside the area, because the pilot sees clearly that he is approaching the border, or that he has just flown through it (see Figure 4). But the flight at the walls' border proved surprisingly difficult. When flying near these walls (about 5-8 m) it is really hard to estimate their distance. Therefore, it would be useful to hide them and display only the nearest part of the border when the drone really approaches it. The second option could be to gradually make the walls more transparent if the drone moves away from them. Most clearly, the part of the border that is closest to the drone would shine, the distant parts would be completely hidden. So if the pilot saw a glowing grille in front of him, he would know with absolute certainty that the border is very close. The distance, at which the boundary would appear, would be good to adjust to the speed at which the drone is approaching it, so that the pilot always has enough time to react, but at the same time the boundary does not unnecessarily interfere with the pilot's vision.

Another problem was the blending of the virtual wall with a virtual screen that was distracting and unpleasant. But even the proposed solution could at least partially solve this problem, because only the part of the boundary that is really needed at that moment would always be displayed.

5.2 Test 2: Exploration of a distant object and flight between obstacles

This test mimicked another fairly common task – exploring a more distant object (such as a house or a parked car) that is too far away from the drone, causing the pilot to fly closer to it. Here, the aim was to test whether the application really could help to improve the pilot's spatial orientation during the flight to this location, the object being explored, and the return to the starting point. In this test, it was assumed that the pilot knows the position of the object in advance and can create a waypoint on the object's position and navigate to it. The second part of the test was a low altitude flight between obstacles, when it is not possible to use an autonomous flight, but the pilot must manually get through a lot of obstacles (e.g. trees) and not to lose the spatial orientation and direction of the flight even with no landmarks around. It is not possible to use map data in this mode, because it is necessary to fly using a video only in order to watch obstacles carefully. Here, however, navigation arrows could help, because they keep the pilot informed about the direction to the waypoint and back to the starting point. This allows the pilot to accomplish the task faster.

When testing a remote object survey, an object was placed on the ground at a distance of approximately $60m$ from the starting point, and a waypoint was created at that point. Between the starting point and this point was an asphalt cycle path and several rows of freshly planted young trees that created natural obstacles. Since there were only meadows and low trees nearby, there were hardly any natural landmarks. Orientation only by the poor quality video stream was very demanding. However, during this test, the navigation arrows, which performed perfectly in their role, had greatly facilitated spatial orientation and significantly helped to reduce cognitive stress during driving. It was not necessary to search for natural landmarks, it was enough to observe nearby obstacles, a navigation arrow and a gradually decreasing distance to the destination. After a while, a target point indicator appeared on the ground.

But at this stage, there was a problem, because the target waypoint was constantly traveling the ground a few meters in all directions from a relatively small target. This was probably due to the inaccuracy of GPS navigation. In an effort to circumvent the examined object and explore it from all sides, the constant movement of this point and the confused rotation of the arrow was very annoying and confusing. Surprisingly, this subject was complicated.

It was much easier to do the fly around and survey with the navigation arrow turned off, by video only. At this point, it would probably be better if the navigation elements were automatically hidden after arriving close to the target and appeared again only if the drone moved away from the target again.

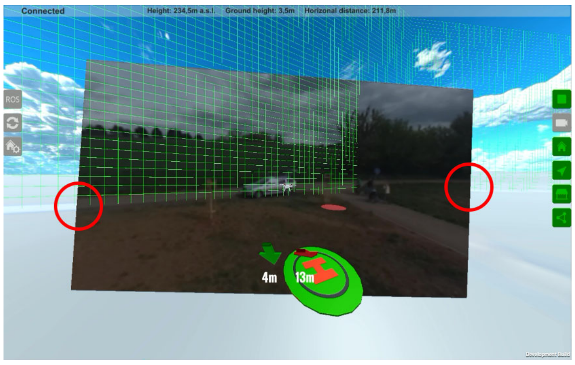


Figure 4: Screen of the implemented application based on augmented virtuality. The virtual environment model is augmented with the data from the real drone – position and orientation, which are used to render the virtual drone in the scene; camera image that is aligned with the virtual environment model (marked with the red circles); and other sensor data. This environment model can be enriched by virtual walls (the green grille) that can mark restricted areas or with a waypoints and direction arrows.

On the other hand, the second arrow pointing to the starting point could be used for orientation, as a some sort of compass that makes clear how the drone is being turned. In addition, for a pilot, a pointer to the starting point is somewhat more natural than an ordinary compass pointing to the north.

5.3 Discussion

Overall, the implemented application was relatively pleasant to use and the fact that it was able to partially compensate the absence of gimbal was positively appreciated. Indeed, by moving the video according to the current tilt of the drone, the objects on the video were still displayed at approximately the same location in the scene. Of course, the convenience of using the application was reduced by controlling the camera's rotation with the arrow keys of the laptop keyboard. This convenience could be increased by connection of VR glasses with head-tracker, which would allow the pilot to naturally look around the scene.

On the other hand, the application did not work very well at very low altitudes (up to $3m$), where the flight altitude and the distance from smaller objects were very poorly estimated. However, this problem also occurs in the first person view (FPV). Surprisingly, adding a drone model to the scene did not reduce the problem, but slightly increased it.

Quite surprisingly, the virtual screen with the camera image was relatively well-connected to the virtual scene most of the time, which even slightly exceeded expectations, especially given that the test drone certainly did not have the most accurate sensors available. Professional drone data is likely to be even more accurate.

The further development will be primarily focused on solving problems that were discovered during testing and on other designed ideas that were not implemented (e.g. the visualization of other sensor data, the point cloud, and the completion of area boundary visualization). Then, VR glasses with a head-tracker, which allows natural looking around the scene, will be connected to the application. Another such thing is to implement a free camera and test its capabilities.

6 CONCLUSIONS

The aim of this work was to improve pilot's orientation and to reduce his mental load during the drone remote control. Based on research and experience, a system has been designed that is based on augmented virtuality, where on-line data from drone sensors (video-stream, flight data, etc.) are integrated into the virtual environment model. The 3D virtual model consists of the data from external data sources like topography maps, elevation maps and 3D building models. The model also includes the user-specified planned mission information like waypoints, safe zone boundaries or flight directions.

The system architecture is designed to be scalable to communicate with multiple drones simultaneously. This could be useful in situations where more pilots are simultaneously carrying out a mission and have to work together.

The preliminary user tests proved that the proposed concept and technical implementation of the entire system improves the operator's orientation and navigation skills and so reducing the mental load. More user tests are planned in future work. The professional pilots will test the system to refine the concept, to improve or include more UI elements and for further development based on their needs.

ACKNOWLEDGEMENTS

The work was supported by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science – LQ1602" and by Ministry of the Interior of the Czech Republic project VRASSEO (VI20172020068, Tools and methods for video and image processing to improve effectivity of rescue and security services operations).

REFERENCES

Calhoun, G., H. Draper, M., F Abernathy, M., Delgado, F., and Patzek, M. (2005). Synthetic vision system for improving unmanned aerial vehicle operator situation awareness. *Proceedings of SPIE - The International Society for Optical Engineering*, 5802.

Cho, K., Cho, M., and Jeon, J. (2017). Fly a drone safely: Evaluation of an embodied egocentric drone controller interface. *Interacting with Computers*, 29(3):345–354.

Devos, A., Ebeid, E., and Manoonpong, P. (2018). Development of autonomous drones for adaptive obstacle avoidance in real world environments. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 707–710.

Fernández, R. A. S., Sanchez-Lopez, J. L., Sampedro, C., Bavle, H., Molina, M., and Campoy, P. (2016). Natural user interfaces for human-drone multi-modal interaction. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1013–1022.

Gageik, N., Benz, P., and Montenegro, S. (2015). Obstacle detection and collision avoidance for a uav with complementary low-cost sensors. *IEEE Access*, 3:599–609.

Gubcsi, G. and Zsedrovits, T. (2018). Ergonomic quadcopter control using the leap motion controller. In *2018 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, pages 1–5.

Mamani, M. A. and Yanyachi, P. R. (2017). Design of computer brain interface for flight control of unmanned air vehicle using cerebral signals through headset electroencephalograph. In *2017 IEEE International Conference on Aerospace and Signals (INCAS)*, pages 1–4.

Natarajan, K., Nguyen, T. D., and Mete, M. (2018). Hand gesture controlled drones: An open source library. In *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, pages 168–175.

Nourmohammadi, A., Jafari, M., and Zander, T. O. (2018). A survey on unmanned aerial vehicle remote control using brain–computer interface. *IEEE Transactions on Human-Machine Systems*, 48(4):337–348.

Rognon, C., Mintchev, S., Dell'Agnola, F., Cherpillod, A., Atienza, D., and Floreano, D. (2018). Flyjacket: An upper body soft exoskeleton for immersive drone control. *IEEE Robotics and Automation Letters*, 3(3):2362–2369.

Sedlmajer, K. (2019). User interface for drone control using augmented virtuality. Master's thesis, Brno University of Technology, Faculty of Information Technology.

Smolyanskiy, N. and Gonzalez-Franco, M. (2017). Stereoscopic first person view system for drone navigation. *Frontiers in Robotics and AI*, 4.