



VCU

Virginia Commonwealth University
VCU Scholars Compass

Theses and Dissertations

Graduate School

2024

Exploring End-User Environments for the Control and Programming of Collaborative Robots

Luiz Felipe Fronchetti Dias
Virginia Commonwealth University

Follow this and additional works at: <https://scholarscompass.vcu.edu/etd>



Part of the [Computer and Systems Architecture Commons](#), and the [Robotics Commons](#)

© The Author

Downloaded from

<https://scholarscompass.vcu.edu/etd/7785>

This Thesis is brought to you for free and open access by the Graduate School at VCU Scholars Compass. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of VCU Scholars Compass. For more information, please contact libcompass@vcu.edu.

**Exploring End-User Environments for the
Control and Programming of
Collaborative Robots**

Luiz Felipe Fronchetti Dias

THESIS SUBMITTED TO
THE DEPARTMENT OF COMPUTER SCIENCE
OF VIRGINIA COMMONWEALTH UNIVERSITY
IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Program: Computer Science

Advisor: Dr. Rodrigo Spinola

During the development of this work, the author received
financial support from NSF – Grant #2024561

Richmond, Virginia

July, 2024

**Exploring End-User Environments for the
Control and Programming of
Collaborative Robots**

Luiz Felipe Fronchetti Dias

This is the original version of the
final thesis prepared by candidate
Luiz Felipe Fronchetti Dias, as
submitted to the Examining Committee.

Acknowledgements

This work is dedicated to my mother, Márcia, who inspired me to be my best, gave me more than enough support, and showed me that borders are made to be crossed. You will always be my biggest inspiration.

Nothing is permanent in this wicked world – not even our troubles.

– Charlie Chaplin



(Image courtesy of Modern Times – Roy Export S.A.S.)

Abstract

Luiz Felipe Franchetti Dias. **Exploring End-User Environments for the Control and Programming of Collaborative Robots**. Thesis (Doctorate). Virginia Commonwealth University, Richmond, Virginia, 2024.

The Fourth Industrial Revolution (4IR) is marked by the alignment between physical and digital technologies. Robots, only seen on industrial floors in the past, are now one of the promising technologies in this revolution. Advancements in hardware and software are bringing robots to a collaborative level, allowing users to interact directly with them to solve problems in multiple domains. Studies in the field have shown that robots are qualified to perform tasks beyond the industrial floor, making the lives of different professionals easier. Although robotics has been growing in the interest of a broader audience, most programming technologies available for robots are still based on the same solutions from the last century. To collaborate with the ongoing development of better programming solutions for robots, I propose three research contributions for collaborative robot programming. The first study evaluates the block-based paradigm as a potential alternative for end-users programming two-armed robots. In this study, a commercial beginner-friendly solution is put in contrast with a block-based programming language implemented as part of the contributions. Both programming solutions are evaluated by 52 participants in an experiment involving a pick-and-place task. From this first study, important insights into human-robot collaboration emerged, including challenges with robot positioning and interaction. The second study gave a sequence to the first study and focused on the challenges end-users face while manually positioning robots in a workspace. This study discusses the use of mixed-reality devices as a potential workaround to the manual positioning of industrial and collaborative robots. Five different control interfaces implemented in mixed reality were used in two experimental tasks involving 49 end-users and 11 experts from human-robot interaction. Finally, the third study brings a discussion about learning challenges and the use of learning resources by novices in robotics. In this last evaluation, 35 individuals without prior robot programming experience are invited to solve an experimental task using a collaborative robot and a block-based environment. Participants are instructed to report learning barriers faced throughout the experiment and to use a list of learning materials available on a computer placed next to the robot. Results from this study suggest that even participants with prior experience in programming tend to face challenges when trying a robot programming environment for the first time. This study also highlights novices' interest in videos and chatbots, rather than more traditional learning resources such as textbooks and technical manuals. With this work, I expect to bring insights into end-user robot programming and interaction, expanding the boundaries of applied research in Software Engineering.

Keywords: End-user robot programming. Human-robot interaction. Collaborative robots.

Abbreviations

| | |
|-------|--------------------------------|
| AR | Augmented Reality |
| HMD | Head-mounted display |
| MR | Mixed Reality |
| ROY | RobotStudio Online YuMi |
| VR | Virtual Reality |
| IR | Industrial Revolution |
| PLC | Programmable Logic Controller |
| COBOT | Collaborative Robot |
| VPL | Visual Programming Language |
| PLC | Programmable Logic Controllers |

List of Figures

| | | |
|-----|--|----|
| 1.1 | Introduction: Study workflow. | 4 |
| 2.1 | Background: Example of one and two-armed collaborative robots. | 9 |
| 2.2 | Background: Example of robot controller and teaching pendant. | 10 |
| 2.3 | Background: Example of online programming. | 11 |
| 2.4 | Background: Example of an offline programming environment. | 12 |
| 2.5 | Background: Example of a pick-and-place solution in RAPID. | 13 |
| 3.1 | Related work: Wizard Easy Programming Tool | 16 |
| 3.2 | Related work: RobotStudio AR Viewer. | 18 |
| 3.3 | Related work: Up for grabs. | 20 |
| 4.1 | Duplo: Programming interface. | 21 |
| 4.2 | Duplo: Programming interface of RobotStudio Online YuMi. | 22 |
| 4.3 | Duplo: Experimental design and procedure. | 25 |
| 4.4 | Duplo: Experimental setup. | 26 |
| 4.5 | Duplo: Experiment solution. | 28 |
| 4.6 | Duplo: Percentage of participants who completed each sub-task in a programming environment. | 31 |
| 4.7 | Duplo: Participants' completion times in ascending order. | 32 |
| 4.8 | Duplo: Box-plots of occurrence numbers of the top 3 programming obstacles. | 33 |
| 5.1 | Mixed reality: Methodology. | 42 |
| 5.2 | Mixed reality: Illustrations of user interaction with the evaluated interfaces. | 43 |
| 5.3 | Mixed reality: Task performed by participants in the first experiment. | 46 |
| 5.4 | Mixed reality: Completion rates for groups trying each interface, and average completion times for participants who finished the experiment. | 49 |
| 5.5 | Mixed reality: Average usability scores. | 50 |
| 5.6 | Mixed reality: Trial performed by experts in the second phase. | 53 |

| | | |
|-----|--|----|
| 5.7 | Mixed reality: Haptic gloves replacing hand tracking in the mixed reality prototype. | 58 |
| 5.8 | Mixed reality: Distant manipulation applied to the interfaces in mixed reality. | 59 |
| 6.1 | Learning: Task solution. | 68 |
| 6.2 | Learning: Experiment workspace. | 71 |
| 6.3 | Learning: Demographic composition of participants. | 74 |
| 6.4 | Learning: Distribution of participants across completed requirements and the distribution of requirements completed per participant. | 74 |
| 6.5 | Learning: Universal Robots Academy. | 79 |
| 6.6 | Learning: Wizard Easy Programming (Advertisement). | 80 |
| 7.1 | Conclusion: Programming in RoboArt. | 84 |
| 7.2 | Conclusion: Function-centric design for block-based environments. | 85 |

List of Tables

| | | |
|-----|---|----|
| 4.1 | Duplo: Variables annotated by the proctor during the experiment. | 29 |
| 4.2 | Duplo: Average completion time for each sub-task. | 32 |
| 4.3 | Duplo: Programming challenges mentioned by participants in the post-experiment questionnaire. | 34 |
| 5.1 | Mixed reality: Comments from the post-experiment questionnaire. | 52 |
| 5.2 | Mixed reality: Comments from experts during their interview. | 57 |
| 6.1 | Learning: List of software requirements given for the experimental task | 66 |
| 6.2 | Learning: List of learning resources available on the help desk | 70 |
| 6.3 | Learning: Appearance of learning barriers per request | 75 |
| 6.4 | Learning: Participants most used learning resources according to their format | 77 |
| 6.5 | Learning: Participants' most frequent challenges faced while using learning resources | 78 |

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | End-User Robot Programming | 2 |
| 1.2 | Research Contributions | 4 |
| 1.2.1 | Block-Based Programming For Two-Armed Robots | 5 |
| 1.2.2 | Evaluation of Robot Controls in Mixed Reality | 5 |
| 1.2.3 | Learning in End-User Robot Programming Environments | 7 |
| 1.3 | Structure | 8 |
| 2 | Background | 9 |
| 2.1 | Collaborative Robots | 9 |
| 2.1.1 | Programming Methods | 10 |
| 2.1.2 | Programming Languages | 11 |
| 2.1.3 | Known Challenges | 12 |
| 2.2 | End-User Robot Programming | 14 |
| 3 | Related Work | 15 |
| 3.1 | Visual Programming in Robotics | 15 |
| 3.2 | Robot Programming Assisted by Virtual, Mixed and Augmented Realities | 16 |
| 3.3 | Learning Barriers in Software Development | 19 |
| 4 | Block-Based Programming For Two-Armed Robots | 21 |
| 4.1 | Overview | 21 |
| 4.2 | Tools | 22 |
| 4.2.1 | ROY: RobotStudio Online YuMi | 23 |
| 4.2.2 | Duplo: Block-based Cooperative Programming | 24 |
| 4.3 | Method | 25 |
| 4.3.1 | Recruitment | 25 |
| 4.3.2 | Experimental Setup | 25 |
| 4.3.3 | Experimental Procedure | 26 |

| | | |
|----------|---|-----------|
| 4.3.4 | Data Collection and Analysis | 29 |
| 4.4 | Results | 29 |
| 4.4.1 | Demographics | 30 |
| 4.4.2 | Participant Performance | 30 |
| 4.4.3 | Completion Times | 31 |
| 4.4.4 | Programming Obstacles | 32 |
| 4.4.5 | Program Analysis | 33 |
| 4.4.6 | Feedback from Participants | 33 |
| 4.5 | Discussion | 34 |
| 4.5.1 | How Programming Environments Affect End-user Performance? | 35 |
| 4.5.2 | What Learning Barriers Do End-users Face? | 37 |
| 4.6 | Limitations | 38 |
| 4.7 | Conclusion | 39 |
| 5 | Evaluation of Robot Controls in Mixed Reality | 41 |
| 5.1 | Overview | 41 |
| 5.2 | Study Stage 1: Evaluation of MR Control Widgets for Robot Positioning . | 42 |
| 5.2.1 | Control Interfaces | 42 |
| 5.2.2 | Recruitment | 45 |
| 5.2.3 | Training and Instruction | 45 |
| 5.2.4 | Experimental Task | 46 |
| 5.2.5 | Measures | 47 |
| 5.3 | Study Stage 1: Results | 47 |
| 5.3.1 | Demographics | 48 |
| 5.3.2 | How Do Interfaces Affect Performance? | 48 |
| 5.3.3 | How Do Users Rate Interface Usability? | 48 |
| 5.3.4 | Which Aspects are Difficult and Easy to Understand? | 49 |
| 5.4 | Study Stage 2: Expert Feedback on Manipulating a Physical Robotic Arm | 51 |
| 5.4.1 | Prototype Implementation | 51 |
| 5.4.2 | Interviews | 53 |
| 5.4.3 | Interview Analysis | 54 |
| 5.5 | Study Stage 2: Results | 54 |
| 5.6 | Discussion | 57 |
| 5.6.1 | Precision and Hand tracking | 57 |
| 5.6.2 | Field of View | 58 |
| 5.6.3 | Practical Applications | 59 |
| 5.7 | Threats to Validity | 59 |
| 5.8 | Conclusion | 61 |

| | | |
|----------|---|-----------|
| 6 | Learning in End-User Robot Programming Environments | 63 |
| 6.1 | Experimental Method | 64 |
| 6.1.1 | Recruitment | 64 |
| 6.1.2 | Experimental Task | 65 |
| 6.1.3 | Wizard Easy Programming Tool | 65 |
| 6.1.4 | Workspace | 68 |
| 6.1.5 | Help Desk | 69 |
| 6.1.6 | Post-Experiment Questionnaire | 72 |
| 6.1.7 | Data Collection and Analysis | 72 |
| 6.2 | Results | 72 |
| 6.2.1 | Demographics | 73 |
| 6.2.2 | Participants Performance | 73 |
| 6.2.3 | Frequency of Learning Barriers | 74 |
| 6.2.4 | Usage of Learning Resources | 76 |
| 6.2.5 | Outcomes from Questionnaire | 76 |
| 6.3 | Discussion | 78 |
| 6.3.1 | Chatbots and the Next Generation of Software Developers | 78 |
| 6.3.2 | End-user robot programming: <i>Are We There Yet?</i> | 80 |
| 6.4 | Limitations | 81 |
| 6.5 | Conclusion | 82 |
| 7 | Conclusion | 83 |
| 7.1 | Complementary Studies | 83 |
| 7.1.1 | Artistic Robot Programming in Mixed Reality | 83 |
| 7.1.2 | Guided Decomposition in Larger Block-Based Programs | 84 |
| 7.1.3 | Language Impact on Programmable Logic Controllers | 85 |
| 7.2 | Papers Published | 86 |
| 7.3 | Future Work | 87 |
| | References | 89 |

Chapter 1

Introduction

The First Industrial Revolution (1760 - 1840) was marked by the transition from manual labor to mechanized processes (LOWE and LAWLESS, 2021). The adoption of steam-powered engines and the growth in manufacturing guided rural civilizations to factories, revolutionizing production standards and ideals of economy (VERMEULEN, 2020). In the following years, the Second Industrial Revolution (1870 - 1914) reshaped the purposes of mass production with assembly lines, bringing light to discoveries such as electricity, internal combustion engines, and the telephone (BERNAL and SRIDHAR, 2022). Later in the same century, the Third Industrial Revolution (1960 - 1990) opened space for a complete digitalization of our society. Computers and the internet renovated the ideals of human collaboration, taking humankind to space for the first time (DAL and DEBACQ, 2020).

The Fourth Industrial Revolution is now on track, bringing attention to the fields of robotics, artificial intelligence, cloud computing, and more (SCHWAB and DAVIS, 2018). If robots were one day considered science fiction, they are now an integral part of our reality. The International Federation of Robotics estimates that more than 2 million industrial robots were installed in the last five years (2018 - 2022), with a positive prospect for the future (I. F. o. ROBOTICS, 2023). Most robots still operate at an industrial level, offering automated labor in manufacturing processes such as welding, material handling, and palletizing. But the domain is changing. Over the past decade, collaborative robots (also known as *cobots*) have become one of the promising technologies in robotics (GRAU *et al.*, 2020). Different from industrial robots, cobots are envisioned as units that should be safe to work around humans and to be easy to operate (TARANTINO, 2022).

The ability of cobots to operate safely around humans opened space for a new range of robot applications, expanding the market of manufacturers to domains beyond the industrial floors (SHERWANI *et al.*, 2020). Adjustments were made to traditional standards, turning robots into "*a tool that anyone can use*" (VERLAG, 2022a). However, while areas like machine learning have consistently evolved during the Fourth Industrial Revolution, the study of robots is considered a work in progress. Many of the software solutions incorporated in cobots are seen as adaptations from industrial robots invented in the last century, making them incompatible with the perspectives of a new generation of customers (AALTONEN and SALMI, 2019). While computer developers now have access to cutting-edge programming solutions (e.g., large language models), new clients adapting

cobots to their domain still depend on last-century programming languages and tools to tell their robots what to do.

The challenges of promoting industrial and collaborative robots to new programming standards are many. For researchers working in Software Engineering, the opportunities are focused on building easy-to-use technologies to allow customers to program and interact with their robots (KNUDSEN and KAIVO-OJA, 2020). Because robots can now directly collaborate with humans, the number of companies interested in using such units is increasing, creating a need for programming tools that are easier to program and understand. Cobots in applications such as healthcare, security inspection, and personal assistance, for example, are growing in numbers over the years (TAESI *et al.*, 2023). The question that applies to those researching new programming alternatives is: How do we adapt robots for this new generation of consumers?

To respond to this question, an emerging topic named *end-user robot programming* is attracting the attention of researchers in robotics (AJAYKUMAR, STEELE, *et al.*, 2021). While end-user programming is concerned with users attempting to program their software solutions, end-user robot programming investigates strategies to allow users to program their robots (BARRICELLI *et al.*, 2019). Different solutions have been tested so far: Block-based languages are now replacing text-based solutions (RITSCHHEL, KOVALENKO, *et al.*, 2020). Lead-through programming is giving users the ability to program robots by demonstration (RAGAGLIA *et al.*, 2016). Virtual reality applications are allowing customers to operate robots from distance (BURGHARDT *et al.*, 2020), as many other technologies are also expanding the boundaries of robot programming, making robots easier to use, program, and comprehend.

1.1 End-User Robot Programming

The rise of collaborative robots started in 2008, with the introduction of the first units from Universal Robots in Denmark (TARANTINO, 2022; NIKU, 2020). From that point, other global robot manufacturers entered the collaborative industry, including FANUC from Japan, ABB from Sweden-Swiss, and KUKA from Germany (BOGUE, 2016). According to the International Organization for Standardization (ISO), cobots are defined as "*robots that allow direct interaction with humans*" (COHRSSSEN, 2021). In practice, while industrial robots tend to be isolated from human collaboration and require specialized professionals to operate them, cobots bring a new perspective to the use and application of robots, being safe to work around humans and targeting ease of operation.

Many advancements have been made regarding the safety of collaborative robots, but the ease of operating such units is a challenge that remains (SHERWANI *et al.*, 2020). In a global survey with 1,650 companies operating in multiple areas of expertise, ABB discovered that 53% of their clients "*identified a lack of skills and training in using and programming robots as a justification for not having made a switch to robotic automation*" (VERLAG, 2022b). To fill out this gap, a growing interest in solutions that make robot programming understandable is converging into this new research topic known as end-user robot programming. If end-user programming was at some point concerned with users implementing computer applications in an office (Capers JONES, 1995), end-

user robot programming now aims to give this same audience the ability to program robots (AJAYKUMAR and HUANG, 2020).


As end-user robot programming can be considered an emerging field, standard solutions prioritized by researchers in this domain may not yet be clear. However, AJAYKUMAR, STEELE, *et al.*, 2021 highlights some promising topics, including:

- **Visual programming:** The act of translating ideas into code can be a challenging task for users without major experience in programming (KO, B. A. MYERS, *et al.*, 2004). One common approach to make the coding experience more intuitive for end-users is to translate text-based instructions into visual components. A visual programming language (VPL) implements this approach using visual elements such as blocks, diagrams, icons, and forms to represent code (KUHAIL *et al.*, 2021). Throughout the years, VPLs have gained popularity in the robotics domain, leading to the creation of new programming solutions (CORONADO *et al.*, 2020). Open Roberta, for example, is a block-based language for educational robots (JOST *et al.*, 2014), and the ABB Wizard Easy Programming tool is one of the first commercially available block-based languages for one-armed cobots (ABB LTD, 2020).
- **Augmented and mixed reality:** Different from computer programming, developing solutions for robots goes beyond the computer screen. Robots are, in practice, mechanical units that must work in consonance with the environment where they are installed (HENTOUT *et al.*, 2018). Implementing a solution that does not make the robot collide, for example, can be challenging to achieve without visual feedback, and even harder to identify in a programming language. To create a more intuitive experience for end-users, multiple studies investigate the use of augmented and mixed reality technologies in support of robot programming (WALKER *et al.*, 2023). HÖRST and ORSOLITS, 2022, for example, proposes a solution in mixed reality for developers to control a collaborative robot via holographic gadgets (please, watch the promotional video¹). Other authors go beyond the point of using such devices as assistive technologies, proposing complete programming environments in the virtual domain (Mikhail OSTANIN *et al.*, 2020; QUINTERO *et al.*, 2018).
- **Natural language:** If groundbreaking technologies can immerse end-users in robot programming, nothing should beat the language they speak. Recently boosted by the popularization of large language models, different studies investigate the use of natural language as a trivial alternative to traditional robot programming methods (TANIGUCHI *et al.*, 2019). BEHRENS *et al.*, 2019, for example, proposes a multi-modal system where users combine speaking in natural language with demonstration to dictate instructions for dual-arm robots. In social robotics, large language models are being used to enhance human-robot interaction, reducing human discomfort over new technologies, and increasing the adaptability of robots into new workspaces (ZHANG *et al.*, 2023).

Although advancements have been made in the field, no ground truth was established for end-user robot programming. Aiming to move things forward, this thesis is concentrated on three new end-user studies, described in the following section.

¹ <https://www.youtube.com/watch?v=3Qv-cur4qxA>


1.2 Research Contributions



Study 1

Block-Based Programming For Two-Armed Robots
Enabling arm synchronization with jigsaw blocks


In this thesis, you will be introduced to Duplo, the first block-based language designed for robots with two arms. You will learn how Duplo makes arm synchronization easier for users without former experience in programming, and how Duplo performs in contrast with another solution.



Study 2

Evaluation of Robot Controls in Mixed Reality
Manipulating a robot using Microsoft HoloLens 2

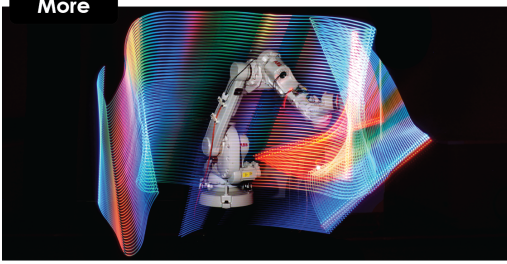
You will learn throughout this thesis how traditional forms of robot motion control can be replaced by mixed reality. You will be introduced to a detailed investigation of robot control interfaces in HoloLens 2, and two experiments involving end-users and experts in the domain.



Study 3

Learning in Robot Programming Environments
Understanding learning barriers in robotics

In my third experiment, you will learn about learning barriers that end-users face in robot programming environments, and be introduced to a discussion over their preferences among different learning resources. We will also discuss the applicability of large language models in robotics.



More

Artistic Programming, Logic Controllers...
The previous and future steps of my research

In the closing chapter of my thesis, you will be introduced to other studies executed throughout my doctorate research, including works on artistic robot programming, guided decomposition in block-based programming and ladder logic for programmable logic controllers.

Figure 1.1: Study workflow: The first study is about block-based programming in two-armed robots. The second is about mixed reality in robot motion control, and the third is about natural language in the support of end-users programming robots. Other studies are also presented in the final chapter.

In my thesis, I move forward the knowledge about end-user robot programming by exploring uncovered challenges in the aforementioned topics (See 1.1). My work is divided into three studies (See Figure 1.1): In the first study, I investigate the adoption of a block-based environment in the programming of two-armed robots. Although block-based environments have already been explored in related studies, this is the first work investigating their applicability for robots with two arms. In this work, I also provide a one-to-one comparison of block-based programming with a commercial solution designed for the same robot category, bringing insights for future end-user-related solutions. In the second study, I investigate the usability and application of mixed reality components as an alternative to the traditional controls found in articulated robots (e.g., joysticks and buttons). This study can be interpreted as a supplement to the work of HÖRST and ORSOLITS, 2022, bringing an experimental investigation with end-users and experts in robotics. Finally, in the third study, I evaluate the use of learning resources to assist end-users facing difficulties in robot programming tasks. This work involves an experimental task where individuals are asked to program a collaborative robot while they have assistance from a help desk. This work also lists common difficulties end-users faced while programming the robot,

following the same strategies of prior studies in end-user and novice programming (Ko, B. A. MYERS, *et al.*, 2004; STEINMACHER, SILVA, *et al.*, 2014). A more detailed overview of each study is given in the following subsections.

1.2.1 Block-Based Programming For Two-Armed Robots

Traditionally speaking, industrial and collaborative robots are programmed using text-based languages (KANDRAY, 2010b). Different from computer programming, the languages available in a robot usually vary based on the manufacturer. Robots from FANUC, for example, are restricted to Karel, a proprietary language similar to PASCAL. Due to their complex nature, most languages are designed with roboticists in mind, making them an unviable option for end-users (ROSSANO *et al.*, 2013). To overcome this problem, researchers have been investigating the applicability of block-based languages in the field of robotics (MAYR-DORN *et al.*, 2021; D. WEINTROP *et al.*, 2017). The block-based paradigm is historically known for being a viable alternative for novices without former programming experience (NOONE and MOONEY, 2018). Although solutions have been proposed for different robot categories, little is known about its use and application in robots with two arms.

In this study, we introduce Duplo, a block-based programming environment designed for end-users programming two-armed robots. Different from other block-based languages, Duplo is focused on the challenges of making two arms operate together. Duplo positions the program for each arm side-by-side, using the spatial relationship between blocks from each program to represent parallelism in a way that end-users can easily understand. This design was proposed by previous work (RITSCHEL, KOVALENKO, *et al.*, 2020), but never implemented or evaluated in a realistic programming setting. This work dives into an in-depth investigation of block-based programming using the end-user's perspective. A randomized experiment with 52 participants is applied to evaluate Duplo, including three experimental tasks and a post-experiment questionnaire.

To provide readers with a reference for comparison, this work also contrasts Duplo with RobotStudio Online YuMi (ROY), a commercial end-user-friendly programming solution developed by ABB. From this study, we can notice that the known benefits of block-based solutions are also observed in two-armed robot programming: Blocks are easier to add, edit, and understand. If debugging can be a concerning task for those without former experience in programming, Duplo makes it easier by allowing users to distribute chunks of code over a canvas. Generally speaking, participants who tested Duplo performed the same tasks in less time than those trying the commercial alternative. They also faced fewer problems (e.g., collisions), making Duplo a suitable reference for the next generations of two-armed robots. For a visual reference, a promotional video of Duplo is available².

1.2.2 Evaluation of Robot Controls in Mixed Reality

Robot programming could be an undemanding task if its unique obstacle was writing code (or using *blocks*). The truth is that programming a robot demands more than just code: When a physical robot is used, developers must know how to move it around, deal with

² <https://www.youtube.com/watch?v=MDmuNLtOmC4>

the limitations of its joints (e.g., singularity points), overcome workspace obstacles, and more (KANDRAY, 2010a). Among these tasks, efficiently moving the robot around can make a difference in the life of someone learning robot programming. This is because robots are precise machines, and minimal changes in translation and rotation can lead robots to collisions or even incorrect solutions (ROSSANO *et al.*, 2013). Industrial and collaborative robots are usually embedded with a solution for this problem: joysticks and button controllers. Although both may be considered reasonable tools, they contain imperfections: Button interactions can be tiresome when complex movements are performed, and joysticks depend on the manual ability of users to efficiently position a robot. As an alternative, we investigate in this study the use of mixed reality interfaces in the manual control of collaborative robots.

Recent advances in mixed reality (MR) have made it a viable solution for robotics (HOENIG *et al.*, 2015). The resolution and field of view of most devices are now bigger, and groundbreaking features such as hand and speech recognition are incorporated (PARK *et al.*, 2021). However, the most commonly used MR widgets have been designed to manipulate virtual objects, not robots. They are based on traditional web controls like buttons and sliders, making it unclear how effective they are at controlling physical objects, such as robotic arms and grippers. To investigate which MR widgets are the best for interacting with collaborative robots, we implemented five interfaces for the Microsoft HoloLens 2, an industry-leading headset. Each interface was based on a widget from the headset's development kit, including sliders, buttons, bounding boxes, object manipulators, and a custom joystick that we designed to mimic best practices from traditional robotics.

To evaluate the applicability of mixed reality in this context, we performed a randomized experiment with 49 participants. Participants were organized into five groups, each testing a different mixed reality interface. In individual sessions, we asked the participants to reposition a virtual robotic gripper over a target location and measured their performance and accuracy throughout the process. We also collected their overall feedback to calculate System Usability Scale (SUS) scores for each interface. This experiment showed that participants performed well when using the *Bounding Box* interface and gave it the most positive ratings, with a SUS score average of 77.8. Different from the traditional joysticks and button controllers, the *Bounding Box* interface provided a more suitable solution for users to grab and move the robot through specific axes.

Based on these results, we also implemented a robot control application in MR that uses the *Bounding Box* widget in the manipulation of a collaborative robot in real-time. We asked 11 professionals from different backgrounds to use the prototype and conducted follow-up interviews to get their opinions. In these interviews, the experts suggested additional use cases for the prototype and identified limitations restricting the widespread adoption of comparable systems. Our findings provide insights for designers of MR-based tools and can inform future research that aims to bridge the gap between robotics and virtually assisted technologies.

1.2.3 Learning in End-User Robot Programming Environments

How much knowledge is necessary for an end-user to program a collaborative robot? As professors play an important role in students' success in a classroom, the robot expertise end-users don't have may play an essential role in their programming success (AJAYKUMAR, STEELE, *et al.*, 2021). Although an expert could be of great benefit to end-users in robotics, having someone on their back does not sound like a trivial solution, nor does it align with the independent principles of end-user robot programming (BENOTSMANE *et al.*, 2018; SHERWANI *et al.*, 2020). To prepare end-users for the independence of writing their own coding solutions, companies like Universal Robots have invested in e-learning platforms designed to teach novices the essentials of the field³. Most of these e-learning platforms have limited video tutorials and provide access to private communities. However, to the best of our knowledge, no study in the literature investigates the use of different learning resources in training end-users in robotics. Are video tutorials the appropriate way of preparing end-users for robot programming tasks? We believe that the learning process should be investigated in detail if the goal is to give end-users independence over their implementation. Towards a similar end, we now see a rise in the popularity of large language models (MEYER *et al.*, 2023). If one-day independent learners were only supported by their hired experts, conversational chatbots like ChatGPT and Google Bard now offer them preprocessed knowledge for free (GURURANGAN *et al.*, 2020).

Following the conversational model's trend and the challenge of understanding end-users needs for robot apprenticeship, we worked on a third study investigating how different learning resources (e.g., videos, textbooks, and chatbots) can help novices complete robot programming tasks. In this study, we also investigated what common learning challenges users face when programming a collaborative robot for the first time using a list of six common barriers found in end-user programming environments KO, B. A. MYERS, *et al.*, 2004. Although other studies have already discussed the use of natural language in the field (VILLANI *et al.*, 2018; YOUNIS *et al.*, 2023), we also contrast in our work the success of traditional learning materials like textbooks and technical manuals with custom large language models in the assistance of end-users in robotics. To achieve this goal, we executed an experimental study with 35 end-users programming a one-armed robot. The study asked participants to solve a pick-and-place task similar to the one used in Duplo, with the difference that participants were able to access a help desk for assistance in the process. The programming language of choice was the Wizard Easy Programming tool from ABB, a block-based language designed for novices in robot programming.

Our results suggest that developers prefer video tutorials and conversations with chatbots rather than more traditional learning resources when programming a robot for the first time. Participants solving the experimental task opened 33 conversations with our custom language model. From a post-experiment questionnaire, 30 out of 35 participants (86%) suggested videos as the most useful resource when learning about programming, while 27 suggested the use of chatbots (77%). Technical manuals, the most common type of learning material in industrial robotics, were suggested as useful by only 15 participants (43%). From this experiment, we also learned that although end-user

³ <https://academy.universal-robots.com/>

programming environments may help novices program robotic tasks, these users still face many challenges translating their solutions into code. Among the most frequent challenges faced by participants with the Wizard Easy Programming Tool were barriers related to understanding the features available in the language, barriers associated with the difficulty of translating programming logic into instructions, and difficulties in dealing with unexpected results.

1.3 Structure

The thesis is organized as follows: Chapter 2 provides background information on collaborative robotics, including the most common programming methods, languages, and challenges. This chapter also provides readers with an introduction to end-user robot programming. Chapter 3 presents a collection of related studies, focusing on visual programming languages and robot programming assisted by virtual, mixed, and augmented realities. Chapter 4 presents the first contribution of this thesis, introducing a block-based programming language for two-armed robots. Chapter 5 brings up the second study, focused on the manual control of robots through mixed reality. Chapter 6 brings our study about learning resources and barriers in end-user robot programming environments. Finally, Chapter 7 presents a conclusion for this thesis, highlighting complementary studies, papers published, and future work.

Chapter 2

Background

In this chapter, a brief introduction to collaborative robots is given, including their definition, design, operation, and programming languages explained in detail. A quick overview of end-user robot programming is also presented.

2.1 Collaborative Robots

By definition, industrial robots are re-programmable, multi-functional manipulators designed to accomplish tasks using programmed motions (HÄGELE *et al.*, 2016; KUTZ, 2015). They automate repetitive procedures, performing tasks such as material handling, welding, and assembly (SAENZ *et al.*, 2018). Industrial robots can present a variety of structures, forms, and sizes. My work is focused on a trending category of industrial robots known as collaborative robots (or *cobots*). While some industrial robots are designed to work in isolated environments and require high programming experience, collaborative robots are made to work safely around humans and to be easy to use and operate (TARANTINO, 2022; MIHELJ *et al.*, 2019).



(a) Two-armed robot (ABB YuMi)



(b) One-armed robot (ABB GoFa)

Figure 2.1: Example of one and two-armed collaborative robots.

The mechanical structures of a collaborative robot are usually articulated, with three or more interconnected joints rotating about a given axis and a shape similar to a human

arm (MISRA *et al.*, 2020). Each robot usually contains no more than two arms and has an end effector (i.e., a peripheral device) connected to its extremity, designed to interact with the environment (See Figure 2.1). To operate a collaborative robot, users must first access its operating system through the robot controller, a custom computer used to operate robots. Users can access the system's interface by connecting the controller to a conventional computer using a computer network or manipulating it using a teaching pendant, a multi-functional device similar to a tablet connected to the robot controller (See Figure 2.2).

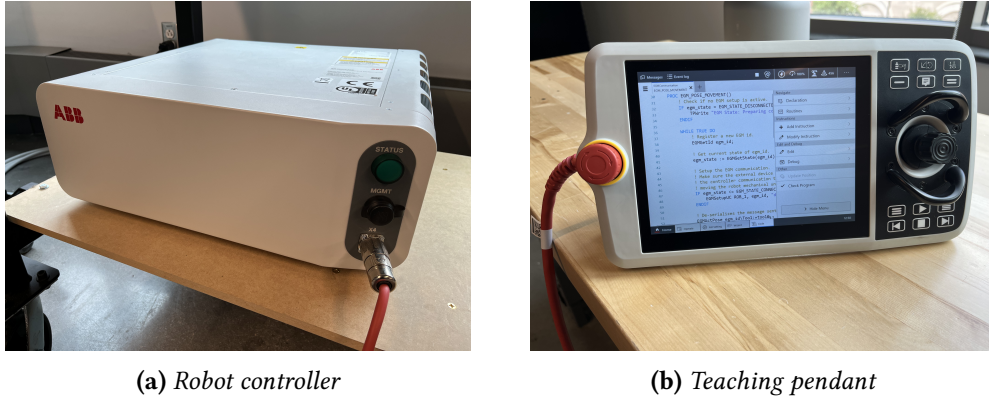


Figure 2.2: Example of robot controller and teaching pendant.

Operators can manipulate the robotic arms from the operating system and access other functionalities such as the program editor, file explorer, and system settings. In generic terms, collaborative robots are mechanical units humans program through customized computers to perform tasks. The difference between collaborative and other industrial robots is that cobots are supposed to be safe, easy to use, and operate.

2.1.1 Programming Methods

Programming a collaborative robot usually involves teaching the robot a desired task. To perform the task, the robot operator must define a path with positions that the robot will follow, implement the logic of each position using programming languages, and test if the robot performs the task correctly. More complex tasks may also involve other procedures, such as the control of end effectors (e.g., welding torches, grippers), which can also be implemented with programming languages. Different programming methods are defined based on the interaction between the operator and the collaborative robot and can be categorized in online or offline programming strategies (HEIMANN and GUHL, 2020; H. LIU, 2020).

Online Programming

In online programming methods, operators have direct contact with the robot while developing their solutions (SALVENDY, 2001). This usually means the robot is out of production, and is used to test new code implementations. Teaching pendant and lead-through programming are two common online programming strategies for collaborative robots.

Both follow similar approaches, and their major difference is based on how the operator manipulates the robot.

In lead-through programming, operators use their hands to lead the robot through the positions of the desired solution (See Figure 2.3a). The operator then saves the positions in the robot controller's memory and programs the final implementation using a teaching pendant or external computer. When the program is executed, the robot can repeat the positions defined by the operator. In teaching pendant programming, instead of using their hands, the operator controls the robot with the assistance of a pendant device (See 2.2b).

Depending on the layout of the teaching pendant, users can use a joystick controller, a touch-screen interface, or a set of directional buttons attached to the pendant case to move the robot to new positions. The final solution is then implemented using the pendant or with the assistance of an external computer. Other online methods may also be available on certain collaborative robots (e.g., walk-through programming), and the terminologies may vary from manufacturer to manufacturer.



(a) *Lead-through programming*



(b) *Teaching pendant programming*

Figure 2.3: *Example of online programming.*

Offline Programming

In offline programming, operators use programming tools on external computers to implement their solutions. These tools usually include simulation environments in which operators can interact with virtual representations of their robots. In these methods, there is no need for interaction between the operator and the physical robot as solutions are only deployed once they are completed in simulation. The number of features in an offline programming environment depends on the manufacturer. In ABB's RobotStudio, for example, the developer can simulate a real environment, including a virtual representation of the controller, the teaching pendant, and the robot (See Figure 2.4).

2.1.2 Programming Languages

Unlike traditional computers, the programming languages available in industrial and collaborative robots usually depend on the robot manufacturer. Most languages follow a procedural approach, with the developer defining the robot's steps through a sequence of instructions. Most languages contain the exact mechanisms as traditional programming

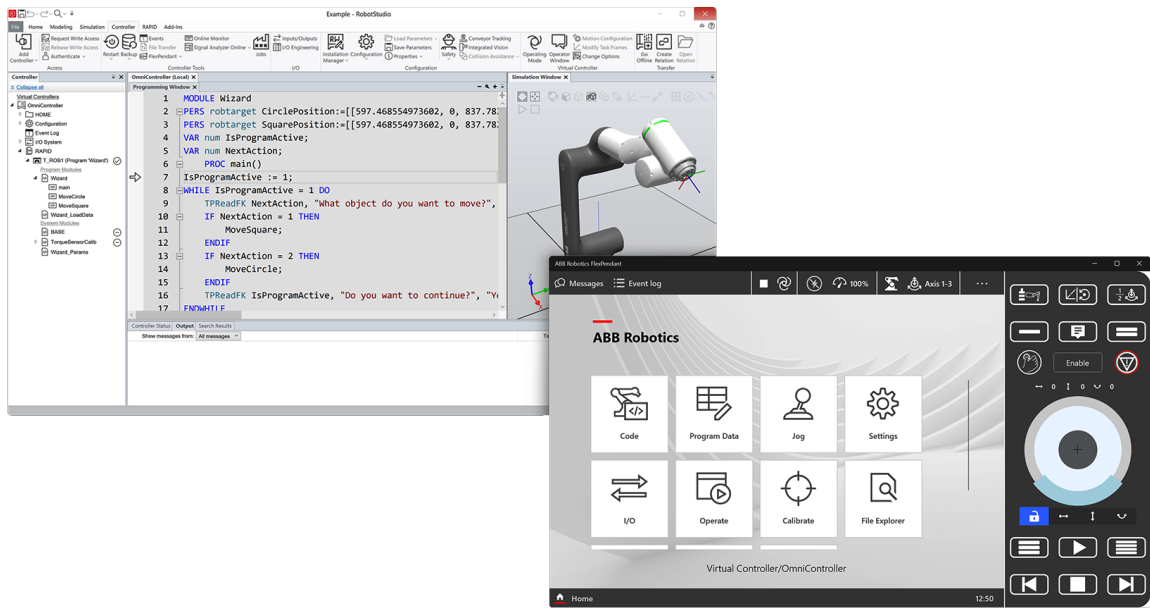


Figure 2.4: Example of an offline programming environment.

languages, including different data types (e.g., numbers, strings) and control structures (e.g., loops, conditionals). The major differences between these and traditional programming languages are the additional features a robot programming language must support, such as robot movement instructions (e.g., MoveJ) and robot-related data types (e.g., joint values).

RAPID is the main programming language available for robots produced by ABB (See Figure 2.5). It is based on a set of high-level instructions and follows a syntax similar to PASCAL. Other manufacturers provide different languages, such as URScript (an extension of Python) implemented by Universal Robots, and the KUKA Robot Language (also similar to PASCAL) made by KUKA. Targeting users without experience in programming, some manufacturers also provide beginner-friendly tools to assist novices in developing solutions for robots. ABB Wizard Programming, for example, is a block-based programming environment available in one-armed collaborative robots from ABB. Puzzle-shaped blocks represent the robot instructions on a canvas, and every code produced in the environment is translated to RAPID by the controller.

2.1.3 Known Challenges

Collaborative robots are becoming one of the key technologies in automation (SHERWANI *et al.*, 2020). Although such robots are changing how humans interact with industrial machinery, many challenges still compromise their adoption (VICENTINI, 2021; KNUDSEN and KAIVO-OJA, 2020). On the manufacturing side, engineers have to worry about the robots' compliance with safety standards and deal with variables such as speed and quality of work made by these machines (NG *et al.*, 2022; MALM *et al.*, 2019). In the business department, the challenge is to create a worldwide interest in cobots, spread their adoption, and establish a new worldwide market (BOGUE, 2016).

```

PROC main()
  VAR num Block;
  VAR num Continue;
  Continue := 1;

  WHILE Continue = 1 DO
    MoveJ InitialPosition, v200, fine, Gripper;
    TReadFK Block, "What block do you want to move?", "Circle", "Rectangle", "Triangle", "Square", "Hexagon";

    IF Block = 1 THEN
      MoveCircle;
    ELSEIF Block = 2 THEN
      MoveRectangle;
    ELSEIF Block = 3 THEN
      MoveTriangle;
    ELSEIF Block = 4 THEN
      MoveSquare;
    ELSEIF Block = 5 THEN
      MoveHexagon;
    ENDIF

    TReadFK Continue, "Continue?", "Yes", "No", stEmpty, stEmpty, stEmpty;
  ENDWHILE
ENDPROC

```

```

PROC MoveCircle()
  OR_RGX_MOVE 120,25,0;
  MoveJ CircleInitialPosition, v500, fine, Gripper;
  OR_RGX_MOVE 45,25,1;
  MoveJ CircleFinalPosition, v500, fine, Gripper;
  OR_RGX_MOVE 120,25,0;
ENDPROC

```

Figure 2.5: Example of a pick-and-place solution in RAPID.

However, such challenges may be of inconsiderable importance compared to the employability of such robots. How do manufacturers create interest in a product that clients can not employ? In a survey with 1,650 companies, ABB found that 53% of their clientele justified not having a robot due to *"a lack of skills and training in using and programming robots"* (VERLAG, 2022b). The same problem is reported in related studies with cobots, emphasizing programming, interaction, and learning difficulties as major problems found in collaborative robotics (SHERWANI *et al.*, 2020; VICENTINI, 2021; BISEN and PAYAL, 2022).

Programming barriers faced by users with different levels of experience should be expected. Especially considering that most robots are restricted to a single language, usually based on aged technologies such as PASCAL, and dependent on a heavy set of robot-related commands. The same holds for learning if we compare, for example, the wide range of materials available in more conventional languages (e.g., Python¹) with the limited resources from robot-related languages (e.g., RAPID²). Although technologies such as lead-through programming may promote better human-robot interaction, the interaction is also limited to basic physical and virtual commands, creating a need for programming alternatives.

Considering such challenges, my study explores how programming-related challenges can be mitigated with the adoption of research-proven strategies and up-to-date technologies. Using the knowledge acquired from studies with end-users in distinct programming environments, I propose a set of solutions to support them in collaborative robot programming such that the downsides of using and programming cobots can be reduced. Related to this, a brief introduction to end-user robot programming is given in the following section.

¹ <https://www.python.org/doc/>

² <https://library.abb.com/d/3HAC050947-001/>

2.2 End-User Robot Programming

End-user programming relates to the idea of allowing users with different levels of programming experience to develop software. The growing interest in the topic started with the popularization of computers in the early nineties (C. JONES, 1995). While more and more individuals started to deal with software solutions in their working routines, the number of experts in software development did not grow at the same rate. At that point, studies started to emerge on how to allow end-users to implement their own software solutions. As a consequence, end-user software engineering gained popularity, delivering multiple works and techniques, such as visual programming languages, programming by demonstration, and more (NARDI, 1993; SMITH *et al.*, 1994; M. BURNETT *et al.*, 2004).

Nowadays, end-user programming can be considered a consolidated topic in Software Engineering (BARRICELLI *et al.*, 2019). Tools such as AppInventor and Scratch are now disseminating programming concepts among users with different levels of expertise, including, for example, high and middle school students (RESNICK *et al.*, 2009; POKRESS and VEIGA, 2013). While researchers in Software Engineering have widely adopted the topic, the challenges of the last century persist for developers in the robotics domain. With robot manufacturers interested in promoting cobots as a viable solution for a broad range of clients, users with limited robot programming knowledge are still dealing with the difficulties of telling their robots what to do.

KO, ABRAHAM, *et al.*, 2011 highlights that end-users are not necessarily novices in programming. They state that users with different backgrounds should benefit from end-user studies as an alternative to professional approaches. The authors also state that a computer program in this context "*is not primarily intended for use by a large number of users with varying needs.*" Their perspective aligns with the current goals of robot manufacturers and collaborative robotics. If cobots reach a wide range of users, programmers with different backgrounds and programming experiences are expected. Professional programming methods found in industrial robotics may not fit the diversity of their goals, interrupting the applicability of such robots in a broader range of solutions.

Amid this scenario, end-user robot programming has become a prominent topic in the robotics domain (AJAYKUMAR, STEELE, *et al.*, 2021). If end-user programming research is focused on computers, end-user robot programming is concerned with robots. Many strategies proposed for cobots are still based on knowledge from traditional end-user programming studies. The use of visual programming as a replacement for more complex text-based programming languages, for example, is one of the approaches in common. However, because robot programming is also about understanding and interacting with the physical environment, other concepts not so well-explored in end-user programming are also being investigated, such as virtual, augmented, and mixed reality technologies.

The central point in end-user robot programming is to "*democratize robot programming by empowering end-users who may not have experience in it to customize robots to meet their individual contextual needs*" (AJAYKUMAR, STEELE, *et al.*, 2021). In my work, I contribute to the democratization of robot programming by exploring alternatives to make robots easier to interact and programming easier to understand. In the following chapter, I highlight related studies connected to my work.

Chapter 3

Related Work

As stated by HÄGELE *et al.*, 2016, "*the range of feasible applications could significantly increase if robots were easier to install, to integrate with other manufacturing processes, and to program.*" This chapter highlights studies on programming-related tools that can contribute to integrating collaborative robots in practice. Sections are divided into two end-user robot programming methods aligned with my research contributions: visual languages and virtual-assisted devices. A discussion section on learning barriers in software development is also included.

3.1 Visual Programming in Robotics

The use of visual affordances in programming is not a recent idea. Tools such as Boxer (DISESSA and ABELSON, 1986) and Fabrik (INGALLS *et al.*, 1988) already explored the advantages of programming using visual elements almost forty years ago. However, visual programming languages go beyond the use of visual cues, as their syntax is intrinsically connected to visual expressions (M. M. BURNETT and MCINTYRE, 1995). While text-based languages are purely based on writing, visual programming languages explore visual means as an alternative way of programming (CHANG, 2012). In practice, some studies consider visual languages as introductory learning tools, with many of these languages supporting a later transition to text-based programming (NOONE and MOONEY, 2018; LIN and David WEINTROP, 2021).

Visual languages provide significant support to end-users, and their strategies to allow such users to program are diverse (KUHAIL *et al.*, 2021). Among the most common methods, there is the block-based programming paradigm (BAU *et al.*, 2017). In a block-based programming language, statements are translated into puzzle-shaped blocks. One block usually serves as the starting checkpoint, and all blocks connected in the sequence compose the programming solution. The syntax of a block-based language tends to follow the same structures of imperative text-based languages such as C, Java, and Python. One of the most referenced block-based languages available is Scratch¹, a language primarily designed for kids learning the basics of computer programming (MALONEY *et al.*, 2010).

¹ <https://scratch.mit.edu/>

If visual languages can serve as a transition to more complex languages in traditional circumstances, the same transition should hold for robot-related languages. Based on a similar hypothesis, programming environments such as Coblox (David WEINTROP, AFZAL, SALAC, FRANCIS, B. LI, David C SHEPHERD, *et al.*, 2018a) started to emerge in robotics. Serving as a beginner-friendly interface for the RAPID language in ABB robots (See Figure 2.5), Coblox translates RAPID text-based instructions into visual blocks. The blocks are organized in categories such as *Move*, for robot movement commands, and *Logic*, for conditional expressions (See Figure 3.1). Inexperienced developers learning the basics of ABB’s collaborative robots are encouraged to start with Coblox as an introductory tool, later transitioning their workflow to RAPID programming.

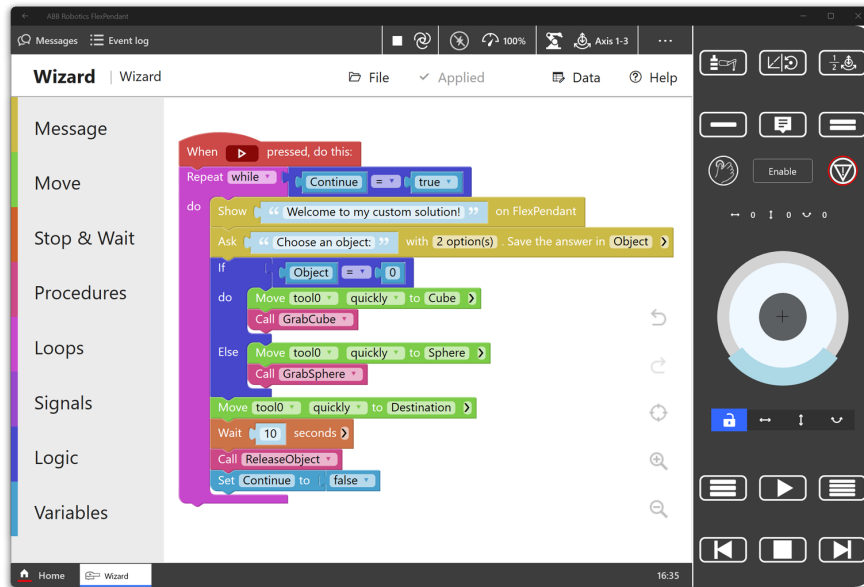


Figure 3.1: Wizard Easy Programming on a virtual teaching pendant.

However, this programming tool is not the only block-based language available in the robotics domain (CORONADO *et al.*, 2020). Tools such as OpenRoberta (JOST *et al.*, 2014), designed for educational robots, and Procrob (ZIAFATI *et al.*, 2017), supported by social robots, are within the same scope. Other visual programming paradigms are also explored by roboticists, including dataflow-based environments like RoboFlow (ALEXANDROVA *et al.*, 2015) and the Microsoft Visual Programming Language (JOHNS and TAYLOR, 2009). The significant difference between these environments and the block-based tools lies in the representation of visual components. While blocks in block-based environments visually represent instructions, dataflow programming represents instructions through directed graphs, with nodes and edges dictating the program workflow (SOUSA, 2012).

3.2 Robot Programming Assisted by Virtual, Mixed and Augmented Realities

Programming a robot is different from programming a computer. While the correctness of a computer program usually depends on the produced solution, programming a

robot also involves its consonance with the physical environment (AJOUDANI *et al.*, 2018). Positioning the robot in multiple locations and making it move without significant complications (e.g., collisions) are problems that most traditional programming environments do not consider. To mitigate this issue, many studies in robotics have been proposing the use of virtual-assisted devices for robot programming tasks. Devices supporting Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR) are among the most common solutions (AJAYKUMAR, STEELE, *et al.*, 2021; SHUMAKER and LACKEY, 2015).

Although working on the same spectrum, these three concepts diverge in practice, and their applicability depends on the devices available (SKARBEZ *et al.*, 2021). VR devices, for example, fully immerse individuals in virtual workspaces, providing limited interaction with the real world (CASINI, 2022). They are usually embedded in head-mounted displays, and the interaction with the virtual environment is given through joystick controllers. In robotics, VR has been used mostly as an alternative for offline programming environments (SHUMAKER and LACKEY, 2015). Developers recreate the robot and its working objects in the VR domain, program them in the virtual workspace, and export the solution to the physical robot in a later stage. Other applications, such as remote operation and training, are also typical in this type of technology (STOTKO *et al.*, 2019; MATSAS and VOSNIAKOS, 2017).

As a clear example of a VR-assisted solution, WANG *et al.*, 2019 implemented a human-robot collaborative welding system in VR. Their system allows welding operators to perform their tasks remotely with the assistance of a collaborative robot. The application displays the actual welding workspace in VR, and the user controls the welding tool in real time via a joystick controller. In a more complex solution, RobotStudio provides a collaborative programming environment in VR (ABB, 2023b). In this solution, users can recreate robotics procedures, manipulate industrial and collaborative robots using joystick controllers, and program new solutions using code instructions available through button-based interfaces (Please, watch the video²). Multiple users can connect and interact in the same environment, allowing a better design of robot solutions.

Many other applications are also available in the VR domain (ZHOU *et al.*, 2020; MOGLIA *et al.*, 2016). Different from simulation environments displayed on a computer screen, VR-supported applications allow developers to have a better understanding of the interactions between the robot and its workspace (O. LIU *et al.*, 2017). A downside of this technology is related to the fact that the quality of the implementation depends on the accuracy between the environment recreated in VR and the actual workspace. If the VR environment does not follow the same properties and sizes as its reference, the final program may result in inconsistent solutions. Problems associated with the VR domain, such as tracking issues and motion sickness, may also impact the applicability of this technology (CHATTHA *et al.*, 2020; CHANDRA *et al.*, 2019).

Another common type of virtual-assisted technology for robotics is found in augmented reality devices. With such technologies, users are not fully immersed in a virtual workspace. Instead, virtual elements are over-positioned in the real environment, creating the impression that they are part of the physical workspace. One of the most common

² <https://www.youtube.com/watch?v=WVUeJFIYBW8>

access points for AR applications are smartphones and tablets. Using the device's camera, users can visualize AR objects in real-time and interact with them using the touchscreen. Most mobile devices support AR features, and many popular applications use it in practice (e.g., Pokemon Go, Polycam).

In robotics, many studies also explore this concept in programming-related tasks (MAKHATAEVA and VAROL, 2020). A great example is presented in the work of GRADMANN *et al.*, 2018, who used a tablet to control collaborative robots from a distance. In their solution, a virtual representation of the robot is aligned next to the physical robot, and a back-end application establishes a network communication between the tablet and the robot controller. Among the features provided in their solution, users can move the robot, operate the gripper, change the control mode, and code basic instructions using the tablet interface. One of their outstanding features is the software's ability to detect objects in the physical workspace using the device's camera. The objects detected are registered into the application running on the mobile device, allowing users to incorporate them into code.

On a commercial level, the offline programming platform RobotStudio is also available on tablets and smartphones as an AR solution (ABB, 2023a). In this mobile application implemented by ABB, developers can easily position their computer-generated robot environments in physical workspaces (See Figure 3.2). Users can not only visualize if their solutions would work in the real workspace but also interact with multiple robots, display safety zones, and verify possible collisions in the working area. Other companies also support robot programming in AR and VR, including but not limited to the KUKA and Visual Components citepkukasim, visualcomp.

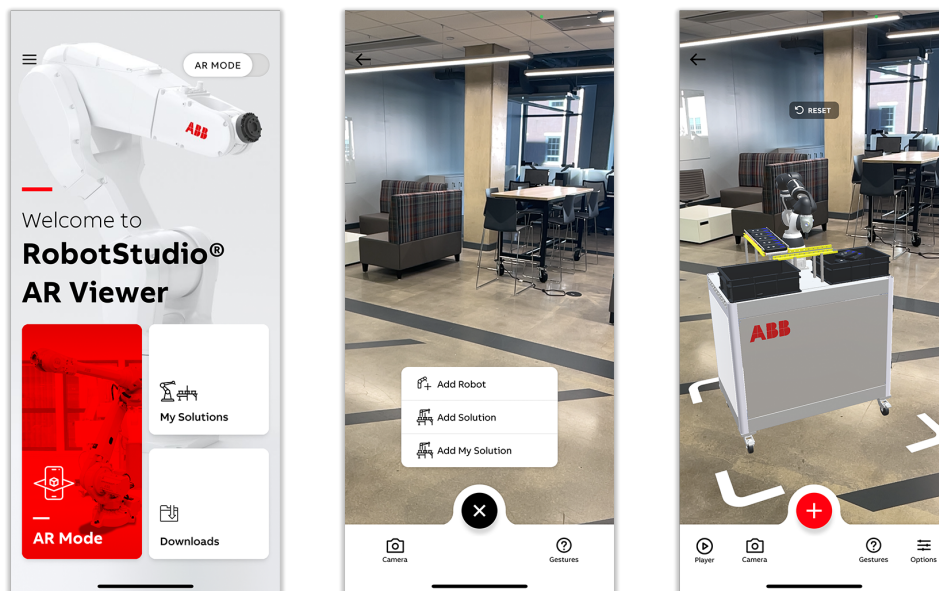


Figure 3.2: RobotStudio AR Viewer, an application to interact with computer-generated robot solutions.

Some augmented reality studies in robotics are also associated with the term mixed reality. This term can be considered a synonym for augmented reality or a more robust version of the concept (SPEICHER *et al.*, 2019). Most solutions associated with MR use

head-mounted displays instead of mobile devices (e.g., Microsoft HoloLens 2, Varjo XR-3). These displays usually come with see-through capabilities, allowing the user to interact with the real world while playing with virtual objects (SALVENDY and KARWOWSKI, 2021). Many displays also support features not available on mobile devices, such as hand-tracking for virtual object manipulation and spatial mapping for environment understanding.

There are many studies in the literature discussing the use of MR displays for robot programming (QUINTERO *et al.*, 2018; EVLAMPEV and OSPANIN, 2019; PICCINELLI *et al.*, 2021). NEVES *et al.*, 2018 discusses how such devices can be used in the programming of industrial robots by exploring two basic concepts: path visualization and path manipulation. In robotics, a path is defined as a sequence of positions the robot will follow in a program. By displaying the robot positions as holograms using a Microsoft HoloLens 2, the authors investigated the applicability of MR devices in path programming procedures. The idea behind the study was that by allowing developers to visualize and interact with the path the robot will take in their program, programmers would get a better understanding of their solution in real time.

Using a different perspective, HÖRBST and ORSOLITS, 2022 explored the use of Microsoft HoloLens 2 for manual control and visualization of collaborative robots. In their solution, holographic interfaces allow users to control a holographic copy of the robot (Watch ABB ROBOTICS, 2021). The movements made in the digital copy are translated in real-time to the physical robot, making it move according to the MR commands instructed by the user. The authors investigated two ways to control the robot manually: one by defining joint values and another by moving the virtual copy to a specific position. In their solution, the authors also discussed how MR can contribute to the visualization of important aspects of a collaborative robot, such as the robot pose and torque.

In general, studies on VR/AR/MR propose a valuable experience for developers working on robot programming tasks (AJAYKUMAR, STEELE, *et al.*, 2021). Although the constant visualization of the physical environment in real-time can be considered a benefit of AR and MR technologies, other limitations influence their applicability for robot programming tasks, making VR also a considerable option (BILLINGHURST, 2021; ROKHSARITALEMI *et al.*, 2020). Most limitations on these three technologies should vanish with advancements in hardware and software, making them essential tools to combat programming challenges beyond code in collaborative robotics. In my thesis, I explore different MR controls that could be adopted by users implementing solutions for cobots.

3.3 Learning Barriers in Software Development

It is not uncommon for novices to face challenges with software implementation (BUTLER, MORGAN, *et al.*, 2007; JIMENEZ *et al.*, 2018; CORNO *et al.*, 2019). Difficulties understanding a specific programming feature or gluing multiple features together are just examples of how complex programming tasks can be. In the open-source domain, STEINMACHER, SILVA, *et al.*, 2014 identified a series of development-related challenges new contributors face when attempting to join open-source communities. They categorized these challenges as a taxonomy of onboarding barriers. From difficulties in learning how to contact project maintainers to issues in understanding the community's documentation, their work ex-

plores the onboarding process in open-source communities through the perspective of inexperienced developers. Their work, extended by related studies (e.g., PADALA *et al.*, 2020; HANNEBAUER and GRUHN, 2017), is helping open-source communities to understand better the difficulties faced by newcomers and open space for solutions in the field. Among the solutions created so far, Up For Grabs is a web portal created to help newcomers find tasks to start their journey in the open-source domain (STEINMACHER, WIESE, *et al.*, 2015). This work inspired our third study on learning barriers in robot programming tasks. We believe that by exploring novices' difficulties, we can better understand what solutions they need.

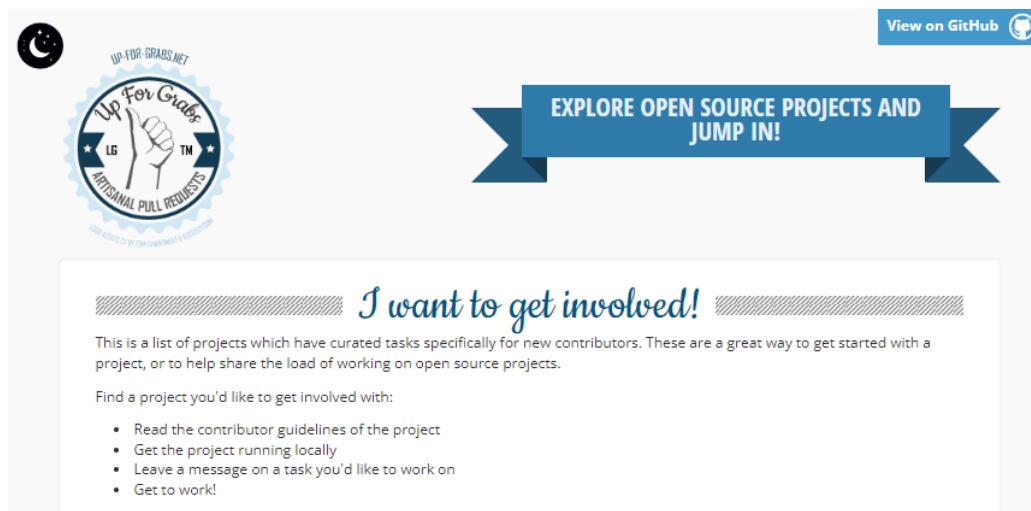


Figure 3.3: *Up for grabs*^a: a website to help newcomers find tasks in open-source communities.

^a <https://up-for-grabs.net/>

In the end-user context, KO, B. A. MYERS, *et al.*, 2004 realized an experiment with 40 programming novices to explore learning barriers in end-user programming systems. They put novices to work on a programming task using Visual Basic³ and asked them to report any difficulties throughout the process. In a posterior analysis, the authors summarized the challenges in six categories: design (*I don't know what I want the computer to do...*), selection (*I think I know what I want the computer to do, but I don't know what to use...*), coordination (*I think I know what things to use, but I don't know how to make them work together...*), use (*I think I know what to use, but I don't know how to use it...*), understanding (*I thought I knew how to use this, but it didn't do what I expected...*), and information (*I think I know why it didn't do what I expected, but I don't know how to check...*). At the end of their study, the authors suggest design solutions for end-user programming systems. These six categories of learning barriers are used in our third study, which is about the same topic but in robot programming environments. We aim to extend the contributions once found in computer programming environments to the robotics domain. In the following chapters, I discuss my three contributions. The first is Duplo, a block-based programming language for two-armed robots (Chapter 4). The second implements robot motion control interfaces in mixed reality (Chapter 5). The third is a deep investigation of learning barriers and resources in end-user robot programming tasks (Chapter 6).

³ <https://learn.microsoft.com/en-us/dotnet/visual-basic/>

Chapter 4

Block-Based Programming For Two-Armed Robots

4.1 Overview

This chapter presents Duplo, an easy-to-use block-based programming system for cooperative programs that controls a collaborative robot with two arms. As illustrated by Figure 4.1, Duplo presents users with two block-based programming canvases side-by-side and features cross-canvas blocks that are synchronized between the canvases to represent parallel commands. This design was explored by previous work through mock-up programs and a front-end prototype but never implemented or tested with actual robots (RITSCHEL, KOVALENKO, *et al.*, 2020). This work implements the design in a refined form and evaluates the use of the block-based paradigm using a two-armed robot.



Figure 4.1: Interface of the Duplo programming language. Users can drag blocks from a toolbox on the left onto the programming canvases and attach them to existing blocks to create a program. Blocks that affect both arms are duplicated across canvases and vertically aligned.

In our evaluation of Duplo, we compare it to RobotStudio Online YuMi (ABB LTD, 2015). ROY is the only existing robotics tool we are aware of that targets end-users and supports the programming of two-armed robots. In addition, ROY provides a similar set of features as Duplo and targets the same two-armed robot model, with equivalent robot commands being available in both languages. This makes ROY an ideal candidate for evaluating the impact that the block-based editing paradigm and synchronized blocks have on end-users programming two-armed robots.



Figure 4.2: RobotStudio Online YuMi programming interface for two-armed robots: Two canvases show RAPID code for each arm side-by-side, with buttons allowing the quick insertion of commands into the code.

To compare the two programming environments, we performed a randomized controlled experiment with 52 end-user participants. Our participants, primarily university students with little to no robot programming experience, were randomly assigned to one of the two programming systems and given a brief introduction on how to use it. They were then presented with a programming task that consisted of multiple stages in which participants had to coordinate two robot arms to jointly carry and assemble a series of items. We recorded whether participants completed each stage of the programming task, how long they took to do so, the number and kind of programming mistakes they made, and how many attempts they needed to test their programs. We found that participants who used the Duplo block-based environment made fewer programming mistakes, were able to solve the given tasks with greater success, and required less time on average to complete the task.

4.2 Tools

This section provides a detailed overview of the two programming environments compared in this experiment: the existing graphical-based, commercially available system, RobotStudio Online YuMi (ROY), and the block-based programming tool Duplo, introduced as the major contribution in this chapter.

4.2.1 ROY: RobotStudio Online YuMi

RobotStudio Online YuMi (ROY) is a programming interface created by the robot manufacturer ABB to control their two-armed collaborative robot YuMi. It is designed to demonstrate the range of tasks that two robot arms can solve when they cooperate. ROY is developed to be beginner-friendly and accessible to end-users who would be overwhelmed by traditional tools like ABB's RobotStudio (ABB LTD., 2023a). To the best of our knowledge, it is one of the only programming tools available in the market that is both beginner-friendly and supports the programming of a two-armed collaborative robot.

As Figure 4.2 illustrates, ROY's main interface contains two canvases on the left where the RAPID code is displayed and a panel on the right with graphical buttons used to implement new robot instructions. The two canvases on the left contain the program code for each robotic arm that is targeted by the environment and are used to visualize instructions created by the user. When the user presses a button on the right panel, the RAPID instruction assigned to that button is displayed on the canvas respective to that robot arm. The instructions displayed on the canvases are similar to the ones from professional tools of the same manufacturer, although boilerplate codes such as function headers and variable definitions are hidden. As users compile and run their code (using the button on the bottom right), the two programs are combined with their respective boilerplate code and deployed onto their respective robot arms. The entire project can also be saved into a single file and restored later to continue editing the code in ROY.

The programming process¹ in ROY is purely based on button interactions on the interface. The first button on the main interface, "Current position from both arms", generates code that simultaneously moves both arms to their current position. The second row has two buttons, "Current Position this arm" for each arm, that generates code to move only the corresponding arm. For these buttons, the target location is automatically set to a variable that contains the current location of the connected physical robot at the moment the button is clicked. The third row has four buttons, "Open Gripper" and "Close Gripper" for each arm, that generate code to open or close the grippers in its respective canvas. Lastly, the button on the left side of the bottom row adds a command to both canvases that make the program being executed in each arm wait for each other, for example, when one arm is supposed to remain idle while the other one conducts work.

All of the described buttons can only be used to insert new code at the currently selected line number. However, users can also edit the program's source code of each arm using the "Edit this arm" button on top of each canvas. This button switches into a similar interface that only shows a specific arm's instructions. To edit existing commands, users can manually delete code and replace it with new instructions using buttons on the "Edit this arm" interface. Existing variables can also be overridden with a new location, allowing users to fine-tune their existing programs without having to edit the code manually. By providing a programming environment where beginners can implement programs for two-armed robots using simple button interactions, ABB defines ROY as a "*fast introduction to robot programming*" (ABB LTD, 2015).

¹ <https://www.youtube.com/watch?v=jEbaaqNPh9c>

4.2.2 Duplo: Block-based Cooperative Programming

Duplo is a block-based programming tool that supports cooperative two-armed robot programs. Duplo uses similar commands as existing beginner-friendly block-based programming languages for robots (David WEINTROP and WILENSKY, 2015; FENG *et al.*, 2015; David WEINTROP, AFZAL, SALAC, FRANCIS, B. LI, David C. SHEPHERD, *et al.*, 2018b), which feature high-level commands such as “*Move arm <speed> to <position>*” and “*Open gripper*”. However, unlike previous block-based languages, which focused on a single robot arm executing a single program, Duplo targets two robot arms at once and allows users to write programs that are executed simultaneously. It further integrates lead-through programming to define locations, similar to how positions are declared in ROY.

Figure 4.1 shows Duplo’s user interface. Similar to ROY, it features two programming canvases side-by-side, each containing the program for one robot arm. On the left side of the environment, a sidebar provides several drawers with available programming blocks. The blocks are grouped thematically into movement commands, gripper commands, and synchronization commands. Blocks can be dragged from the drawers onto one of the two canvases and attached to existing blocks, as illustrated by their jigsaw shape. The design of Duplo is based on the findings of previous work (RITSCHEL, KOVALENKO, *et al.*, 2020), which evaluated different design alternatives for coordinating two-armed robots. The Duplo environment follows the design approach that was deemed best by that work, using explicit synchronization blocks and vertical alignment to represent concurrent robot behavior. However, unlike previous work, which used mock-ups and front-end prototypes to compare design alternatives, Duplo features a fully functional implementation that targets a real two-armed robot².

A unique feature of Duplo compared to other block-based languages is the availability of blocks that target both robot arms at once, and that therefore exist in both of the programming canvases. The “*Wait for each other*” block synchronizes the state of both arms before proceeding to the next instruction, and the “*Move arm <speed> to <position> and follow on the other side*” block is used to perform a simultaneous movement of both arms at once. When a programmer drags one of these blocks onto one of the programming canvases, it is automatically inserted into the other canvas as well. The blocks can be edited and moved to a different point in the program by dragging either of the two representations, with the other representation following along accordingly. The environment ensures that complementary blocks are always vertically aligned, making it easy to identify how they correspond to each other. This allows users to visually track the timing of the two programs as they edit them and signals them to potentially add more synchronization blocks to ensure the correct sequence of commands.

This implementation allows us to compare Duplo to ROY, which features a similar complexity of programming features and targets the same robot model. Even though both languages adopt distinct programming styles (*block-based* vs. *graphical-based*), the similarities between Duplo and ROY make them suitable for comparison. Both are among the few tools available for two-armed robot programming. They are specifically designed to be user-friendly for individuals without experience in robot programming and offer

² <https://www.youtube.com/watch?v=MDmuNLtOmC4>

comparable movement and synchronization commands. The collaborative robot and the *lead-through capability* used in both languages are also the same. In other words, equivalent solutions can be implemented in both languages.

4.3 Method

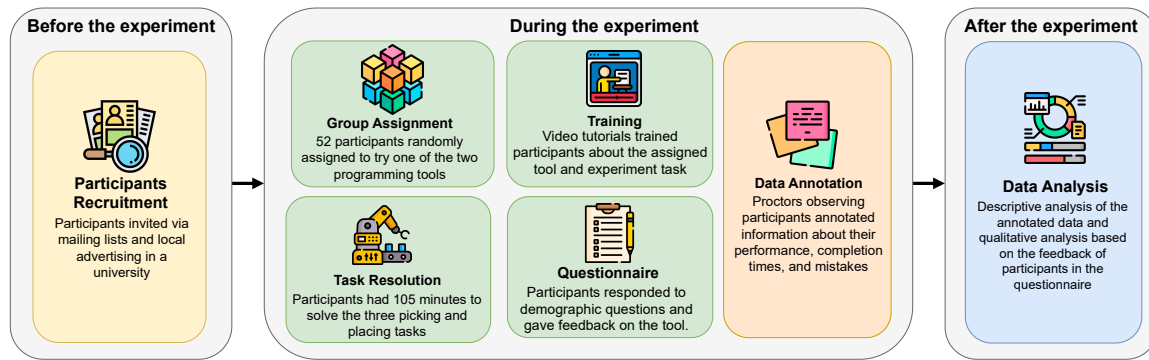


Figure 4.3: Experimental design and procedure divided into three phases: before, during, and after the experiment.

This section describes the controlled experiment conducted to compare Duplo and ROY using a two-armed collaborative robot. Figure 4.3 presents an overview of the experimental procedure and the individual steps that were split across three experimental phases. The entire methodology was refined through a pilot study using feedback from 31 individuals and approved in advance by an institutional review board.

4.3.1 Recruitment

The experiment was advertised to undergraduate students enrolled at a single university in the United States. To ensure a diverse range of participant backgrounds, advertisement emails were sent to different departments, and flyers were distributed to students around the campus. The materials clearly described that the experiment involved a collaborative robot programming task but emphasized that it was focused on end-users without any previous programming experience. A 50 USD gift card for a local bookstore was offered as an incentive to all participants.

4.3.2 Experimental Setup

Each participant was provided with one of the two programming environments and a robot to execute their code and record positions via lead-through programming. The robot used in the experiment was an ABB YuMi (ABB LTD., 2023b), a collaborative robot with lead-through capabilities. Figure 5.3 shows the physical layout of the experiment as it was presented to participants. A laptop running the assigned programming environment was placed on a table adjacent to the workstation where the two-armed robot was mounted. Participants were able to program using either the laptop's touch-screen monitor or the built-in keyboard and touchpad. In addition to the robot, the workstation contained 3D-printed objects relevant to the task participants were asked to solve.



Figure 4.4: *Experimental setup: Participants used a touch-screen-enabled laptop (left) to program a two-armed collaborative robot (right).*

4.3.3 Experimental Procedure

Participants were randomly assigned to one of two groups after they consented to join the experiment. One group was assigned the Duplo environment, and the other group used the ROY environment. Although participants were divided into two groups, each participant was scheduled for an individual session to try the experimental procedure. The only individual rather than the participant in a session was a proctor, who was instructed not to provide extra information to the participants. Other than the assigned programming environment and respective training, both groups were provided with the same setup and task descriptions. The experiment was limited to a maximum of 120 minutes. During the first 15 minutes of the experiment, participants were introduced to their assigned programming environment and trained on how to use it. Next, participants received the task they were to solve along with a clear understanding of what would constitute successfully accomplishing the task. During the following 105 minutes, participants were allowed to solve the given task at their own pace.

Training Procedure

The training for both environments was designed to be as similar as possible, consisting of two brief videos (Please, watch the playlist ³). The first video explained how to use the assigned programming environment. Although the two environments required different instructions and consequently different videos, we made them as similar as possible in both length and content. Each video was approximately 7 minutes long and covered all the basic programming features of the respective environment. Neither video referenced the concrete task that participants were expected to solve.

The second video was two minutes long and introduced the task to the participants. This video introduced them to the 3D-printed objects they were to assemble. It also showed

³ <https://www.youtube.com/playlist?list=PLQHWcSK2-Zw4bAUNAWVuZ1BAZ8oYiUt6l>

them the desired, fully assembled state of the components after all steps of the task were completed. The video did not show the assembling process, as we expected participants to implement their own solutions. It did, however, suggest the order in which participants could tackle the assembly steps, effectively breaking the task down into three smaller sub-tasks that allowed participants to keep track of their progress.

After watching both training videos, the proctor supervising the experiment gave participants a brief in-person introduction to the robot workspace. The proctor demonstrated how to execute programs and how to move the robot arm in lead-through mode to capture its current position. Participants were also given a “cheat sheet” with some reminders and tips, including how to open and close the robot’s grippers while recording positions.

Task Procedure

The experiment task involved writing a program that could perform a series of steps that involved picking and placing 3D-printed objects and executing this program on the physical robot to assemble the final object. Pick-and-place tasks are commonly used in robot experiments because they often occur in practice and typically lead to challenges that participants encounter and have to overcome (BAUMGARTL *et al.*, 2013). Participants were asked to assemble three components using the two-armed robot. First, participants had to pick up a “spacer”, a small cuboid-shaped plastic component, and place it onto a narrow metal shaft in the center of the robot workstation. This part of the task only involved one robot arm and was, therefore, suitable as a warm-up step. However, only one of the two robot arms could reach the item, so this step also served as a check of whether participants could identify and program the correct robot arm for the sub-task.

Second, participants were instructed to pick up a “gear”, a larger and differently shaped component, and move it on top of the previously placed spacer. To solve this programming step, participants had to use the robot’s second arm as only that arm was able to reach the item. While this sub-task seems similar to the previous one (requiring only one robot arm), it did involve coordination because the two tasks had to be executed in the right order. This and the previous sub-task could be parallelized by picking up the two items simultaneously and then placing them one after another, but the materials did not instruct participants to solve them this way. However, a naive implementation is likely to cause a concurrent execution of both tasks in both programming environments. If participants decided to retain the parallelism, they had to ensure the two arms do not collide, which can occur if they try to place objects simultaneously or do not move out of each other’s way.

For the last sub-task, participants were asked to pick and place a “propeller”, a wide plastic object that could not be carried and assembled accurately using a single robot arm. Instead, participants had to carry the item with both robot arms and use synchronized movements to move it into the correct position on top of the previously placed items. Unlike in prior sub-tasks, this required the two arms to work in tandem. This also required synchronization to ensure it was only executed after both previous sub-tasks were completed. Figure 4.5 shows a potential solution for the assembly task, although the specific sequence in which objects had to be placed was not specified or enforced in the experiment. For example, while Figure 4.5 shows the order envisioned above (spacer

first, gear second, and propeller third), participants were allowed to start by placing the propeller and then placing the gear or spacer. This means that participants could move on to a different sub-task if they felt like they were stuck. The task description video did, however, show the assembly in the order as shown in Figure 4.5, as this order was expected to be the most appropriate difficulty curve for participants.

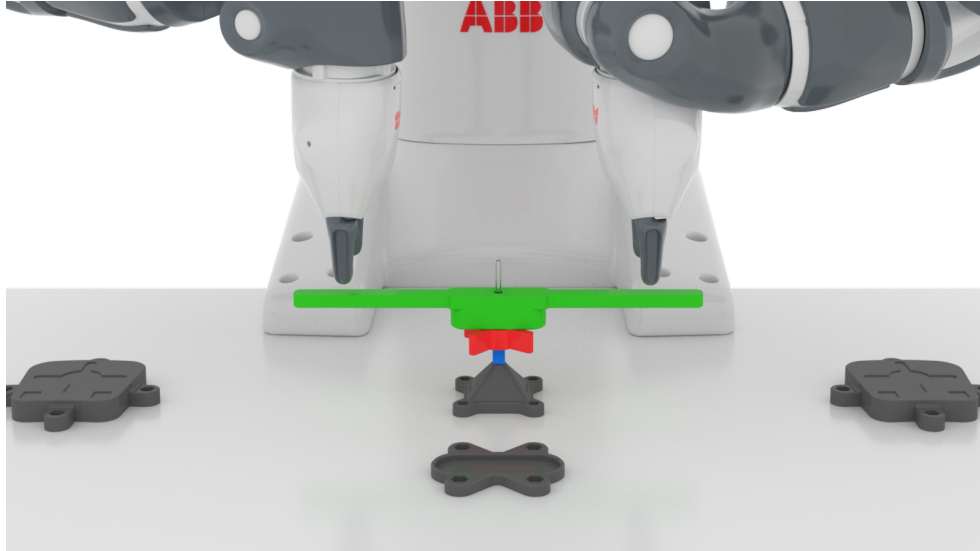


Figure 4.5: *Potential solution: The spacer at the bottom (in blue), the gear in the middle (in red), and the propeller at the top (in green) are all placed into the small metal shaft. There is no specific order of how the objects had to be placed.*

Participants were free to spend the 105 minutes provided for the experiment as they wished. In particular, we did not direct them to move on to a new sub-task if they spent longer than anticipated on a single step. The proctor would only intervene if participants explicitly requested them to repeat previously given instructions or the task description, or if there were technical issues. After participants indicated that had finished the task, the proctor would ask them to run their program one final time to verify the solution. If the program did not solve the task and there was time left, participants could resume programming and try to fix their mistakes.

Post-Experiment Questionnaire

After completing the experiment, participants were asked to answer a post-experiment questionnaire. The questionnaire was composed of eight questions: The first five questions asked participants about their demographic information; this included participant age, area of study, overall programming experience, experience with robot programming, and whether they had ever used a block-based programming language before. The last three questions were open-ended and asked what participants found easy, what they found difficult, and if they had any other feedback about the experiment or the environment they used.

4.3.4 Data Collection and Analysis

A proctor supervised the experiment at all times and collected data in a spreadsheet for later analysis. Using a digital clock, the proctor recorded the duration the participant took to complete each sub-task in the experiment, from picking up their first object to placing the last one. They also counted occurrences of particular events in the experiment, including the number of times a participant executed their program solution, the number of objects accidentally dropped during executions, and the number of times the robot collided with itself or the surrounding workspace. Once a participant had successfully finished the task, the proctor inspected the code and collected information about the participant's solution, including the number of used lines or blocks of code used and the number of robot positions the participant defined in their solution. All the collected variables are defined in Table 4.1.

The data collected by the proctors and provided in the post-experiment questionnaire were analyzed after the experiment. While most experimental data was quantitative, the written responses to the questionnaire required qualitative analysis. The analysis was performed by three researchers using open card sorting to organize and categorize responses (SPENCER, 2009). The researchers were instructed to create codes for participant comments that described features or attributes they found easy and difficult to use. At first, each researcher performed the analysis individually. Once they were done with their analysis, the researchers met to compare their results, aiming to arrive at a final, common set of codes. Constant comparison was employed to guarantee consistency in the codes (CORBIN, 2015). Finally, two additional researchers inspected the final set of codes to ensure they were easy to understand.

| Type | Variables |
|----------|---|
| Integer | # Program Executions, # Robot Positions Created, # < Blocks, Lines > Implemented, # Objects Dropped, # Robot Collisions with < Environment, Robot > |
| Datetime | When participant started the experiment, When participant < picked / placed > the < spacer / gear / propeller > |

Table 4.1: Variables annotated by the proctor during the experiment. Each variable corresponds to a different column in the spreadsheet. Similar or equivalent variables are grouped using < symbols > in the table.

4.4 Results

This section presents the major outcomes of the experiment. Starting with an analysis of the demographic data, followed by the performance of participants throughout the experiment. The feedback given by participants in the post-experiment questionnaire is presented at the end of the section.

4.4.1 Demographics

A total of 52 participants joined and completed the experiment. Participants were randomly assigned into one of two groups of 26 participants, with one group using Duplo and the other using ROY for the experimental task. The participants indicated that they pursued 31 distinct majors. Some were from computing-related domains such as Electrical Engineering (4 participants) and Computer Science (4 participants), but the vast majority were from other areas of study, such as Biology (6 participants), Cinema (3 participants), and Nursing (2 participants). The average participant age was 22 years (min: 17, max: 50, sd.: 6.61). When asked about their programming experience, 10 participants using Duplo (38%) and 11 participants using ROY (42%) declared no prior programming experience. The remaining participants declared some level of programming experience, with 9 participants of each group (34%) indicating one or more years of experience in programming. For Duplo, 12 participants (46%) indicated having at least some experience with block-based languages, while only 5 participants testing ROY (19%) indicated the same. Only 2 participants testing Duplo (8%) and 1 participant testing ROY (4%) indicated at least some experience programming robots.

4.4.2 Participant Performance

During the experiment, the proctor recorded the times and outcomes at which participants completed the sub-tasks and the overall task. Figure 4.6 shows the success rates for each sub-task in the experiment, split by their assigned programming environment. Note that, as explained in Section 4.3.3, sub-tasks were not strictly sequential, so participants might have completed later sub-tasks despite missing previous ones.

For Duplo, 21 out of 26 participants (80%) completed all sub-tasks successfully. All participants in the Duplo group successfully completed the first sub-task, placing the spacer on the rod. One participant failed to place the gear for the second sub-task. Four participants could not pick up or place the propeller in the third sub-task. For the ROY alternative, only 12 out of 26 participants completed all six sub-tasks (46%). 3 participants failed to pick up the spacer during the first sub-task, and 2 more were unable to place it correctly. For the second sub-task, 4 participants did not succeed at picking up the gear and 3 more failed to place it. For the third sub-task, 11 participants failed to pick up the propeller and 3 more were unable to place it.

The Chi-squared test of independence (ZIBRAN, 2007) was employed to examine if the sub-tasks completion rates differ among groups trying each programming environment. The hypothesis that these variables were independent was rejected for p-values < 0.05 . The analysis yielded a significant result for the "Place Propeller" sub-task, with a p-value equal to 0.008. The "Place Spacer" (p-value: 0.059), "Place Gear" (p-value: 0.054), and "Pick up Propeller" (p-value: 0.066) tasks also presented p-values close to the threshold. Such results suggest that there may be a difference in completion rates for certain sub-tasks depending on the programming environment.

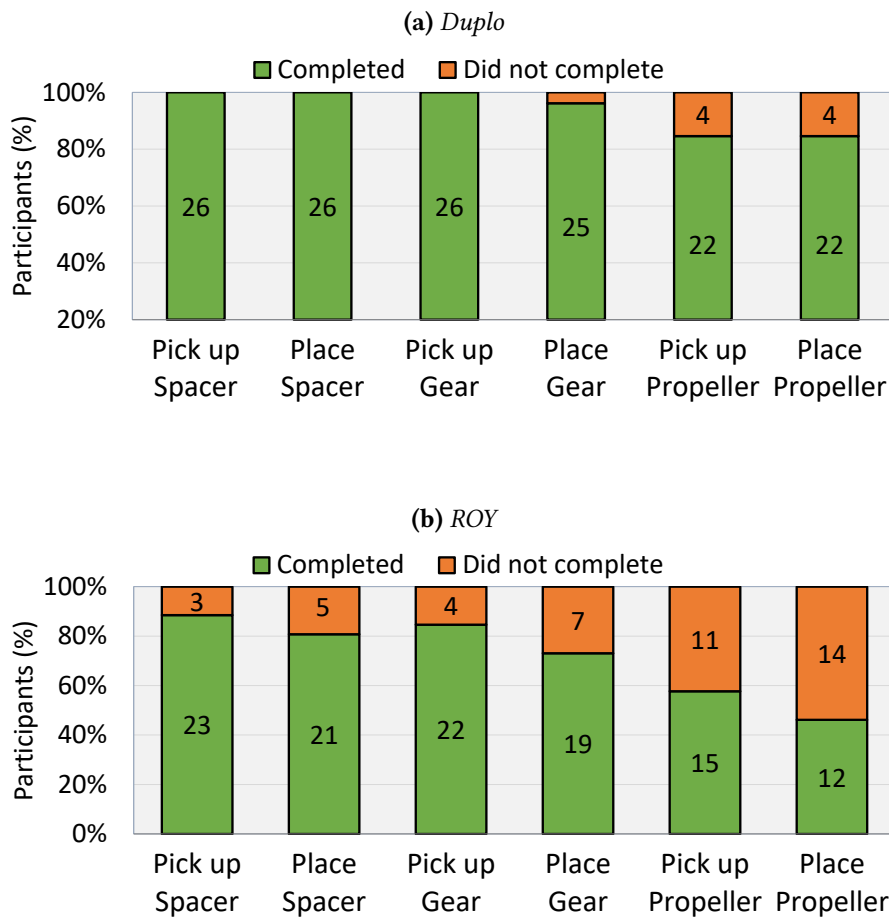


Figure 4.6: Percentage of participants who completed each sub-task in a programming environment. The bar labels provide the exact number of participants per sub-task.

4.4.3 Completion Times

During the experiment, the proctors also recorded participants' completion times for each sub-task and the overall task. Figure 4.7 shows the time participants took to complete the experiment, with bar colors indicating each participant's assigned environment. Participants who ran out of time and did not complete all sub-tasks are marked with a black circle. As the Figure illustrates, Duplo participants were more successful and faster overall. This observation also holds when only considering participants who successfully completed all sub-tasks. Those who completed all the sub-tasks using Duplo took 52.7 minutes on average to finish the experiment (min: 28, max: 97, sd: 22.06), and those who used ROY took 74.6 minutes (min: 35, max: 105, sd: 18.33). Table 4.2 shows the participants' average completion times for each sub-task, divided by their assigned group.

A survival analysis was also performed on the results (GOEL *et al.*, 2010). This statistical method allows the comparison of the completion times of the two groups while factoring in the unsuccessful participants who were cut off after 105 minutes. Note that "survival" in this context means that participants have not yet completed the task at a given point in

time. A log-rank test on the survival curves (WELLEK, 1993) found that there is a statistically significant difference between the two groups ($df = 1$, $\chi^2 = 9.59$, $p < 0.005$).

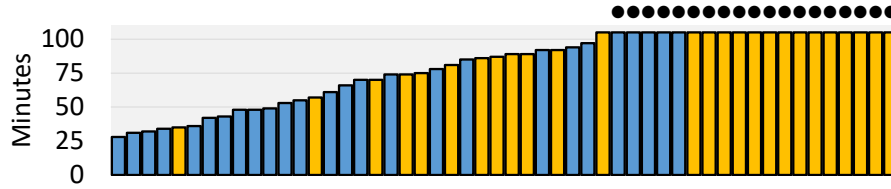


Figure 4.7: Participants' completion times (in minutes) in ascending order. Each bar represents one participant, and its color indicates the assigned environment (blue for Duplo, yellow for ROY). A black circle marks participants who didn't complete all sub-tasks.

| | Pick up Spacer | Place Spacer | Pick up Gear | Place Gear | Pick up Prop. | Place Prop. | Total |
|-------|-------------------|-----------------|-----------------|---------------|------------------|----------------|-------|
| ROY | 18.70 | 14.05 | 22.68 | 15.26 | 30.67 | 21.25 | 74.58 |
| Duplo | 8.15 | 5.45 | 9.27 | 11.24 | 14.64 | 14.50 | 52.67 |

Table 4.2: Average completion time for each sub-task (in minutes). Only participants who completed the task were considered. The total average only considers participants who completed all sub-tasks.

4.4.4 Programming Obstacles

In addition to participant performance, the obstacles participants faced were also investigated. Three errors ended up being the most common: dropping the object held by the robot in the wrong position, collisions of the robot with its surrounding workspace, and collisions of the robot with itself. Note that, unlike the other two errors, the robot's controller detected and automatically prevented self-collisions. This provided quicker feedback to participants and prevented damage to the robot hardware.

Figure 4.8 shows how often participants encountered the top three programming obstacles using each method. On average, participants using Duplo dropped blocks 4.7 times during the experiment (min: 0, max: 15, sd.: 4.26), while participants using ROY dropped blocks 6.9 times (min: 1, max: 21, sd: 5.22). For workspace collisions, the average number of occurrences was also higher for ROY, with an average of 15.1 workspace collisions per participant (min: 7, max: 33, sd: 6.32) compared to an average of 10.7 collisions (min: 1, max: 27, sd: 8.84) from participants using Duplo. The number of collisions prevented by the robot controller is closer for both languages: ROY users encountered an average of 2.0 prevented collisions (min: 0, max: 9, sd: 2.54), and Duplo users encountered 2.3 of them (min: 0, max: 8, sd: 3.09).

To determine if there were significant differences between the obstacle occurrences in both groups, the Mann-Whitney-Wilcoxon Test was conducted (NAHM, 2016). The hypothesis that the occurrences were identical for both groups was rejected for p -values < 0.05 . A significant difference in the number of workspace collisions was found in both groups (p -value: 0.008). However, the number of objects dropped (p -value: 0.067), and

predicted collisions (p-value: 0.684) did not reach the threshold for statistical significance. These results suggest that although two out of three obstacles were less frequent for Duplo participants, it is not possible to confirm a statistically significant difference in occurrence values.

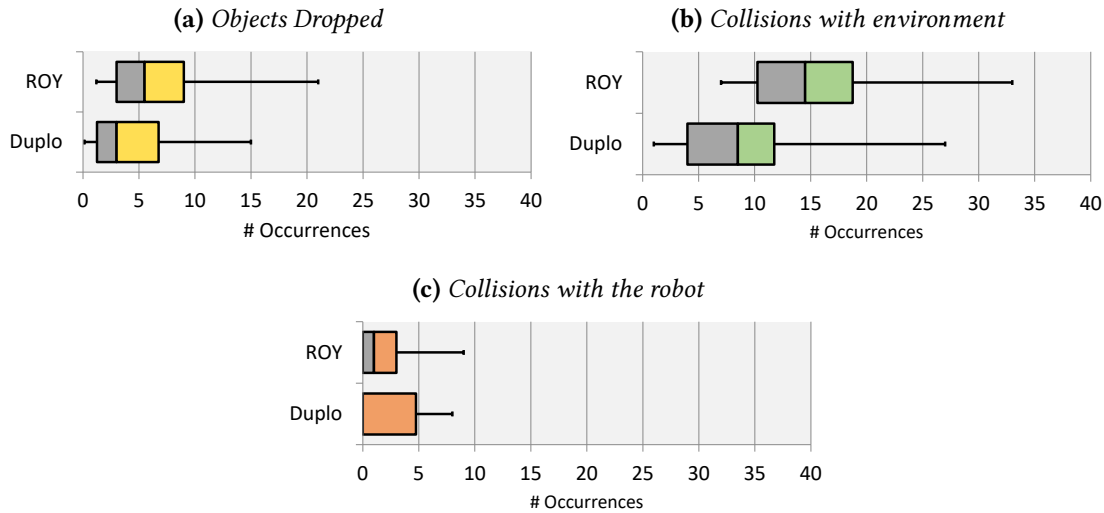


Figure 4.8: Box-plots of occurrence numbers of the top 3 programming obstacles.

4.4.5 Program Analysis

To gain further insight into the participants' programming experience, their final code, and the number of times they executed code while programming were also analyzed. Both participant groups used approximately the same number of test executions during the study: the Duplo group ran their code 37.4 times on average (min: 9, max: 85, sd: 18.59) compared to the ROY group which ran theirs 37.1 times (min: 19, max: 64, sd: 12.27). Duplo users required an average of 33 blocks to write their final solution for the entire task (min: 22, max: 44, sd: 6.42), while participants using ROY wrote an average of 45 lines of text-generated code (min: 11, max: 68, sd: 36.61). Note that these numbers include incomplete programs from unsuccessful participants but do not include empty lines in ROY. Participants using ROY defined 24 robot positions (min: 8, max: 39, sd: 18.5) on average using lead-through programming, compared to 16 on average for Duplo (min: 8, max: 29, sd: 4.8). These results suggest that the necessary workforce to solve the tasks was similar for both groups.

4.4.6 Feedback from Participants

Participants also had the opportunity to comment on the experiment and their assigned programming environment after they completed the task. The only comments considered were those in some way related to the programming interfaces. Comments that provided feedback on the robot hardware, the training, or the experimental task were ignored in this stage but saved for future studies. Table 4.3, presents an overview of the codes generated during the qualitative analysis. These codes described programming challenges and their prevalence in the provided feedback.

For ROY, the biggest challenge found was editing code, with 12 participants mentioning this problem. Among their comments, participants highlighted that it was not possible to reorganize lines of code without deleting them, that they were unable to rename robot positions, that it was not possible to edit both arms at the same time, and there was no option to undo changes. A further 5 participants also mentioned difficulties debugging code, primarily because ROY does not always provide clear feedback when errors occur, which makes it difficult to locate issues. Another 3 participants complained about not being able to execute subsections of code (which could be useful for debugging), and two mentioned how the *MoveSync* command, used to move both arms at the same time, was confusing to them. For Duplo, participants mostly commented that using arm synchronization confused them, including how to program both arms to execute different commands simultaneously. They also mentioned issues understanding the “wait for each other” synchronization block. Another 3 participants also mentioned issues with the position reteaching feature as it is hidden in a drop-down menu and was not mentioned in the training tutorial. Lastly, three more participants mentioned problems with debugging their code, similar to those encountered in ROY.

Participants also commented on what aspects of the two systems they found particularly useful or easy to use. These comments were mostly similar for both systems. For both interfaces, participants mentioned that they were simple and more intuitive than writing code manually. Also, for both interfaces, participants pointed out that defining positions using lead-through programming was intuitive to them. Some participants also noted that they liked the commands for synchronized movement (the *MoveSync* command in ROY and the “follow other arm” block in Duplo, respectively). The only repeated comment we identified that was specific to one environment was that 3 participants found it helpful that they could review the definition of robot positions in Duplo.

| ROY | |
|--|----|
| Difficult to edit code. | 12 |
| Difficult to debug problems. | 5 |
| It is impossible to execute specific chunks of code. | 3 |
| <i>MoveSync</i> command is counter-intuitive. | 2 |
| Duplo | |
| Arm synchronization features are confusing. | 6 |
| Difficult to reteach positions. | 3 |
| Difficult to debug problems. | 3 |

Table 4.3: Programming challenges mentioned by participants in the post-experiment questionnaire.

4.5 Discussion

This section discusses the experimental findings and how different factors may have influenced participants throughout the study. Speech balloons indicate feedback given by participants in the post-experiment questionnaire.

4.5.1 How Programming Environments Affect End-user Performance?

The findings in Section 4.4.2 indicate that the programming environment assigned to participants substantially affected how well they solved the given task. As shown in Figure 4.6, while less than half of the participants (46%) using ROY completed the assigned task, nearly twice as many (80%) Duplo participants succeeded. Participants testing Duplo not only solved the task more effectively but did so faster. As Table 4.2 indicates, participants using Duplo spent less than half the time required by participants using ROY on almost all of the sub-tasks. The performance observations match our expectations, as Duplo was designed to improve ROY's usability. However, performance numbers alone cannot account for Duplo as a better solution. The feedback given by participants should also be considered.

As indicated in Table 4.3, 12 of the 26 participants using ROY (46%) complained about the difficulty of editing code while using the interface. This feedback matches our observations, as ROY's graphical elements are only integrated with text-based editing on a superficial level. Although inserting new code in ROY is easy, to edit an existing line of code, users must either delete and re-write their code or access a secondary interface to redefine positions. It was also not straightforward for users to move lines of code (for example, swap their order). In ROY, users have to manually copy and paste code similar to text-based editing, which can easily interfere with the alignment of instructions and the synchronization between the two robot arms. This introduces a potential source of errors or confusion. Conversely, in Duplo users can drag and drop blocks within the canvas as desired, automatically updating the surrounding code's alignment.

- *Participant I (ROY/Difficult): "I couldn't move lines of code after placing them, so I had to delete them and remake them in the correct line."*
- *Participant II (ROY/Difficult): "The fact that you could not move lines up and down the sequence in the code was frustrating because errors could not easily be fixed..."*
- *Participant III (Duplo/Easy): "It was very easy to use the blocks to ask the robot to make actions, link the blocks together, and break them apart. It was also easy to move the robot's arms."*
- *Participant IV (Duplo/Easy): "Organizing the actual commands was pretty simple and intuitive."*

Another indicator that features affected our participants' performance is present in the second and third most frequent comment highlighted by ROY participants in Table 4.3: the difficulty of debugging problems and being unable to execute specific chunks of code. This feature is particularly important for a use case such as robot programming, where commands can take several seconds to execute, making it tedious to repeatedly re-execute long code sequences. Although we did receive a similar comment from a Duplo participant, the block-based environment made it easier for users to test partial programs because it allowed them to detach code from the main program. This feature, while conceptually similar to commenting out code in a text-based system like ROY, ensures that

even temporarily unused code remains valid and is not accidentally forgotten. ROY, being a fundamentally text-based environment, only had regular comments available, which were also not graphically represented in the user interface. This might be responsible for ROY users spending more time solving the given tasks.

Another debugging feature both environments provided is highlighting the currently executed line of code when running a program. This feature is found in many end-user languages, but for the specific use case of Duplo and ROY, it can become confusing for users since there are two separate executions (for the left arm and the right arm) that take place simultaneously. In Duplo, the vertical alignment of blocks guarantees that synchronized blocks that get executed, such as synchronized movements, are always in a single line. However, this is not necessarily the case in ROY, which creates additional mental effort for users. We speculate that this additional mental strain might have been a factor in why some ROY users found the command for synchronized movements unintuitive.

● *Participant IV (ROY/Difficult): “It took me longer than it would have otherwise taken me because I could not start in the middle of my program...”*

● *Participant V (ROY/Difficult): “Setting the robot back to a designated position required running the program from the start and pausing before it began a new cycle.”*

● *Participant VI (Duplo/Difficult): “Debugging, it was extremely difficult to control the robot outside of a program. If I wanted to open the arms so that I could replace a piece before another cycle, I had to start the program and stop it before it got too far.”*

Our previously discussed findings align with studies where block-based languages and other end-user robot programming tools are evaluated (D. WEINTROP *et al.*, 2017; David WEINTROP, AFZAL, SALAC, FRANCIS, B. LI, David C SHEPHERD, *et al.*, 2018a; MAYR-DORN *et al.*, 2021). In a study where participants had to program a pick-and-place task using Polyscope, researchers pointed out design recommendations for new end-user programming interfaces (AJAYKUMAR and HUANG, 2020). According to the authors, “*interfaces should minimize the use of tabs and keep similar actions and commands coherently grouped together...*”. They also emphasized that “*...end-user robot programming interfaces should have easy-to-use replay capabilities to visualize contextualized portions of the robot program...*”. The ROY interface does not implement either of these features. One participant also highlighted the lack of options to undo commands in ROY, another recommendation proposed by the same prior study.

Summary 6.1: By representing robot commands as puzzle pieces, block-based programming contributed to end-users ability to insert, edit, and debug code. The freedom to reorganize blocks using the puzzle metaphor and the alignment of instructions on vertical columns allowed Duplo participants to complete more tasks in less time. For future work, other common features should be included, such as the ability for users to undo commands.

4.5.2 What Learning Barriers Do End-users Face?

Participants only had a short time to learn how to use either environment they were assigned. Similar time constraints are not unusual when industrial workers have to learn jobs on-task, and previous studies found that end-users can overcome learning challenges quickly as they get hands-on experience with a reasonably end-user-friendly system (RITSCHER, SAWANT, *et al.*, n.d.). Identifying and addressing potential learning barriers can substantially improve how quickly end-users become familiar with a new system.

In a study about learning barriers in end-user programming systems (KO, B. A. MYERS, *et al.*, 2004), researchers identified six categories of challenges end-users face while solving tasks in a programming environment: design barriers (*what to do?*), selection barriers (*what to use?*), use barriers (*how to use?*), coordination barriers (*how to combine different things?*), understanding barriers (*what is wrong?*), and information barriers (*how to check what is wrong?*). The difficulties highlighted by participants in our post-experiment questionnaire show that neither of the evaluated systems is free of those barriers, although end-users encounter them in different situations.

Understanding and information barriers. Programming a collaborative robot involves the understanding of both virtual (software) and physical (mechanical) concepts. For example, to teach a robot a new position, users have to use lead-through to manually move the robot to a new physical location and use the programming environment to record the position. If a defect occurs in their code, they must determine whether the problem is in the programming logic or the physical workspace. In some cases, the logic behind the code may be correct, but the physical locations taught to the robot may still produce errors (e.g., collisions, robot singularities). In the proposed experiment, some users struggled to identify what was wrong with their implementation and reported it as a difficulty in the questionnaire.

☞ *Participant VII (ROY/Difficult): “...I also did not like that it did not always show me which movement code line had a problem if it was after a movement error...”*

☞ *Participant VIII (ROY/Difficult): “...I didn’t know why there were errors sometimes when it looked like it worked...”*

☞ *Participant IX (Duplo/Difficult): “Figuring out what I did wrong.”*

We believe lead-through programming can make defining positions substantially easier, and participants have echoed that sentiment in their feedback. However, there might be room to provide more guidance or training for lead-through, for example in the form of visual aids or immediate feedback during the programming process. This is discussed in detail in Chapter 5.

Selection and use barriers. Some participants also reported difficulties understanding certain features of both programming systems. In particular, both systems involved features to program two arms simultaneously, and some users commented that the synchronization commands provided were unintuitive (i.e., MoveSync in ROY, “wait for each other” in Duplo). While this feedback was primarily received from ROY users, it was observed that

some participants in Duplo were not aware that the block-based language allows them to access and review an already-defined location. This feature, which other participants explicitly named as a useful tool to understand their programs, could have been represented more prominently in the system to make users aware of its existence and explained better to convey its usefulness.

● *Participant (ROY/Difficult): "...I wasted a lot of time deleting sync steps before I understood how to combine one-sided steps with sync steps... It would also be nice to tell the robot, "move to the locations I told you in Step 10 and then stop". It's unclear to me whether that's possible to do. "*

● *Participant (Duplo/Difficult): "... The first (difficulty) was thinking the "Wait for each other" blocks could be placed anywhere instead of only in line with each other. My solution was instead of placing those blocks, I slowed down the movement speed of one of the arms (The block did come in handy later)..."*

● *Participant (Duplo/Difficult): "At first, I didn't know which arm was the right or left. Also, I didn't know how exactly to reattach a position, but after a while, I got it."*

Summary 6.2: Because Duplo eliminated many of the programming challenges for end-users, it enabled the identification of second-order problems. Primary among these was physical positioning and mapping. It became clear that end-users had trouble mapping between position names (e.g., "AboveGround") and physical positions in 3D space.

4.6 Limitations

This section discusses some of the limitations found in this study.

Number and background of participants. This study was conducted with 52 participants, which required a substantial effort to recruit, considering the time and effort participants had to invest in the in-person experiment. The recruitment process aimed at students from diverse backgrounds to represent the wide range of possible end-users interacting with collaborative robots. However, it is possible that the limited number of participants and the fact that they were all students from the same university limit how well this group of participants represents the overall population of end-users.

The fact that 12 out of 26 participants testing Duplo had prior experience with block-based programming may also limit the conclusions made. To evaluate this issue, the Pearson's Chi-squared test of independence was executed to compare participants' success with their experience in block-based programming. It was found a significant association between the variables (p-value < 0.05). The same association wasn't found for other demographic values, such as general and robot programming experience. We hope that our findings inspire additional work that can be evaluated more thoroughly with other end-users of robotic systems.

Training and time constraints. Time to train participants and allow them to work on the given task was limited. It is likely that with more available time or resources,

participants of both groups might have performed better overall when solving the task. However, as outlined in Section 4.5.2, we believe that restrictive time constraints are not uncommon in practice and can be particularly useful in investigating a system's learnability and usability qualities.

Task choice. One unique task was evaluated in this study, which could also be considered a limitation. However, the task was split into several sub-tasks that required different forms of coordination. We believe the combination of sub-tasks covered various challenges and requirements end-users face in practice. Future work is needed to investigate other tasks, particularly those requiring coordination in ways neither of the tested environments supports.

Selected tools. Comparing Duplo and ROY introduces limitations to our work as these programming environments make specific implementation decisions that may not generalize to all programming tools. The difference in programming method (ROY is graphical-based while Duplo is block-based) provides insight into these different methods but also introduces some confounds. Although differences exist between the two programming systems, both are designed to be beginner-friendly and represent two of the most capable end-user tools available for two-armed industrial robots, making them a nice pick for comparison.

4.7 Conclusion

The work proposed in this chapter presents Duplo, a new block-based programming environment for real-world two-armed robots. Duplo is the first practical evaluation of a design concept that uses blocks to visualize the flow of time and coordination between two robot arms. The system was evaluated in comparison to ROY, a commercial tool that uses graphical elements as an alternative to text-based programming and targets end-users. We found that Duplo allowed participants to solve a complex two-armed pick-and-place task faster and more successfully. Based on our observations and our participants' responses, we have identified which differences between the two environments might have caused this effect. We have further identified barriers that remain when participants try to learn either system. We believe that this work can inform future work on how to design robot programming environments that are more user-friendly and make use of the strengths of existing visual frameworks.

Chapter 5

Evaluation of Robot Controls in Mixed Reality

5.1 Overview

Mixed Reality (MR) devices allow workers to access and manipulate virtual elements in real-world environments. Thanks to the advancements in MR technology, entirely new workflows have been introduced in several applications, allowing, for example, car designers to simulate prototype consoles inside real cars (GHIURĂU *et al.*, 2020; GOEDICKE *et al.*, 2022), and enabling stage designers to visualize their work before construction (SZEDMÁK, 2021). While seminal work on MR has taken place decades ago (MILGRAM *et al.*, 1995), its adoption and the interest in studying its benefits and limitations has recently accelerated (FLAVIÁN *et al.*, 2019; ROKHSARITALEMI *et al.*, 2020). The growing popularity of MR devices such as Microsoft HoloLens and Oculus Quest are opening new borders for human-computer collaboration, making it a key technology for collaborative robotics (MAKHATAEVA and VAROL, 2020).

To contribute to the advancement of MR control mechanisms in robotics, this chapter presents an experimental study investigating the usability of MR widgets for the manual control of collaborative robots. Figure 5.1 provides an overview of this study, which took place in two stages:

1. In the first stage, five robot control interfaces based on common MR widgets were implemented using the Microsoft Mixed Reality Toolkit for HoloLens 2 (MRTK2). A user study with 49 undergraduate participants who were trained to use the Microsoft HoloLens 2 headset was conducted in sequence. Participants were randomly divided into five groups, each assigned to a control interface. Each participant was asked to reposition a virtual robotic gripper that was projected into the real world via MR. The time and accuracy of participants were measured, and a survey was conducted to evaluate their satisfaction with the widget assigned. The first stage showed that one control widget, the *Bounding Box*, offered the best combination of user performance and satisfaction for manually controlling grippers in MR.
2. In the second stage, a fully functional prototype using the *Bounding Box* widget

was implemented for users to control a collaborative robot. A series of qualitative interviews was conducted with 11 experts, including robotics, computer science, and user experience professors. In these interviews, interrogators focused on the experts' perception of the interface, especially on the challenges and limitations that need to be addressed before using MR in real-world applications. The study and interviews contribute to the understanding of how users can interact with physical collaborative robots in mixed reality and outline directions for future research to improve such interactions.

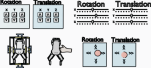







| | Stage 1: Evaluation of MR Control Widgets in a Gripper Positioning Task | Stage 2: Expert Evaluation on the Use of MR Widgets to Control an Industrial Robot |
|-----------------------|---|--|
| Widgets | Buttons, Sliders, Bounding Box Object Manipulator and Joystick  | Bounding Box (Highest in usability score in Stage 1)  |
| Participants | 49 Undergraduates in CS  | 11 Professionals in CS-related fields  |
| Collection Methods | Gripper positioning task, Questionnaire  | Semi-structured Interviews  |
| Data Collected | Performance in positioning task, SUS scores, impressions on MR widgets  | Impressions, suggestions and applications for MR in robotics  |

Figure 5.1: Method description, divided into two stages: first, a study with undergraduate students to compare MR controls in a gripper positioning task, and second, an implementation of the best-evaluated control revised by experts in semi-structured interviews.

5.2 Study Stage 1: Evaluation of MR Control Widgets for Robot Positioning

In the first stage, the goal was to compare five control interfaces for manipulating robot arms in mixed reality. To evaluate the interfaces, an experiment where participants positioned a robot gripper using a Microsoft HoloLens 2 headset was designed. This task, although simple, mimics the control challenges posed by more complex robotics tasks in MR. For the experiment, 49 undergraduate students were recruited. The participants were divided into five groups, each testing one of five interfaces in MR. For each participant, their success in completing the task was recorded, and, in case of success, how long they took to complete the task. Feedback about the prototype was also collected from each participant. In the remainder of this section, the elements of the study design are detailed.

5.2.1 Control Interfaces

The proposed investigation focused on the widgets available in the Microsoft Mixed Reality Toolkit 2 (MRTK2) (ONG and SIDDARAJU, 2021), a toolkit that can be used with

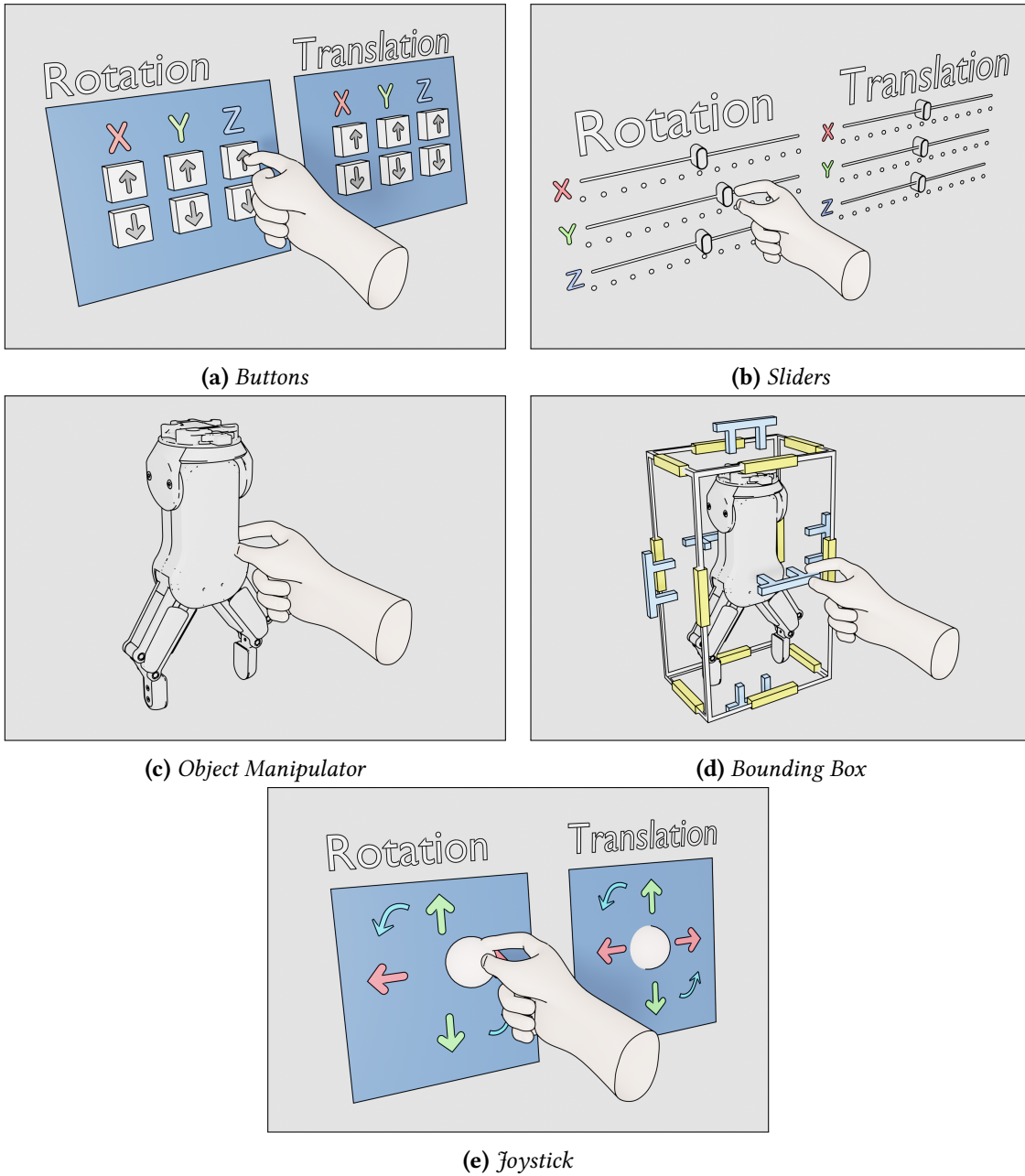


Figure 5.2: Illustrations of user interaction with the evaluated interfaces in mixed reality.

the Unity gaming engine to implement applications for Microsoft HoloLens 2 (PARK *et al.*, 2021; SPEICHER *et al.*, 2019). The interfaces created using each specific widget are described below:

- **Buttons (Figure 5.2a):** From arcade machines to web pages, buttons are well-understood by users in both physical and virtual settings. MRTK2 provides buttons for use in mixed reality, where the user can press the mixed reality button with their hand, mimicking pressing a physical button. An interface to control a holographic gripper using these buttons was implemented. For both translation (i.e., positioning the gripper without rotating it) and rotation, buttons were divided in a per-axis base, with pairs of buttons used to increase and decrease the gripper's relevant coordinate or rotation value. Six buttons were used to translate the gripper, one for each dimension in three-dimensional space and six for rotation. To interact with the buttons, users trying the mixed reality headset could touch them using their own hands as if they were real. All buttons react to being pushed with feedback animations so individuals can determine when they have successfully pressed a button.

- **Sliders (Figure 5.2b):** Sliders are a popular widget in many applications, from mixing music to selecting a departing time for an upcoming flight, as they allow users to change a quantity relatively without caring about the exact value. Six different sliders, three for rotation and three for translation, were used to create a slider-based gripper control. The slider handle begins in the middle, and users pinch the handle with their fingers to move it left or right, increasing or decreasing the respective axis they are controlling.

- **Object manipulator (Figure 5.2c):** While previous controls were familiar in more general domains, some controls are specific to MR. The object manipulator is a simple yet powerful new control enabled by MRTK2. Instead of introducing indirect control, the object manipulator allows users to directly pinch a given virtual object, translating and rotating that object by simply moving their pinched fingers. Using artificial intelligence to interpret the user's intent, the user pinches near or on the virtual gripper and then moves and rotates their closed fingers to control the object. This approach resembles the intuitive lead-through alternatives found in industrial robots but might be a less efficient approach to perform precise positioning tasks (GUPTA A.K., 2017).

- **Bounding Box (Figure 5.2d):** The bounding box control is also specific to MRTK2. This interface consists of handles that are set around the holographic gripper. Six blue handles, when pinched, are used to translate the gripper in the three-dimensional space, while twelve yellow handles are used to rotate it. This interface offers much of the flexibility of the object manipulator but with the ability to isolate movements to a certain axis, which can help with exact positioning.

- **Joystick (Figure 5.2e):** As the last option, an interface that resembles a traditional joystick control system found on commercial robots was also implemented (ARYANIA *et al.*, 2012). Since the joystick is not a standard control provided by MRTK2, a custom version of a joystick in MR was implemented by combining objects from the Unity gaming engine (i.e., spheres and planes) and interaction assets of MRTK2. This interface consists of two joysticks, each represented by a sphere on a plane, one used to control the translation of the holographic gripper and the other to control the rotation. Note that these joysticks work as a traditional joystick found on commercial robots—moving the joystick moves/rotates the

gripper along the X and Y axes– but it can also be rotated clockwise or counterclockwise to control the gripper in the Z axis.

5.2.2 Recruitment

Undergraduate students in Computer Science were recruited from two user-experience classes taught at a large North American research university. Email invitations were distributed online, inviting the students to participate in the user study and get a chance to experience programming in MR. Sixty participants confirmed their participation and were divided into five groups of 12 individuals, each testing one of the prototype widgets. In the end, 49 participants showed up to the experiment: 12 participants tested the buttons interface, 7 participants the sliders, 11 participants the object manipulator, 10 participants the bounding box, and 9 participants the joystick interface. The experimental processes described in the study were approved in advance by an institutional review board.

5.2.3 Training and Instruction

Using MR for the first time can be a disconcerting experience, as controlling holographic objects by pinching and moving your hands mid-air is odd for most users. The correct positioning of the headset (i.e., keeping your hands in view of the headset) also dramatically affects accuracy, and users must be trained on how to position the headset. To mitigate the learning effects of MR, two training sessions were conducted with the participants. As revealed in prior studies, training new MR participants may contribute to a more consistent experiment (BENEDICT *et al.*, 2019). The first training session focused solely on using the Microsoft HoloLens 2 headset and experiencing the MR environment for the first time. Participants were asked to familiarize themselves with the system and were given simple MR controls to interact with, including an interface to resize geometric shapes. During this session, participants were free to ask questions and encouraged to become familiar and comfortable with the MR environment. Each participant's session lasted 30 minutes.

The second and final training session was held immediately before participants started the experimental task and focused on explaining their assigned interface and the task to them. The training started with the participants watching two videos. The first video introduced participants to the interface they would use for the experiment, and the second video introduced the task (Please, watch the playlist¹). The first video was filmed in a first-person perspective with detailed instructions on how to use the interface and visual examples of moving the robotic gripper along each axis. Because different interfaces had different complexities, the first video's length differed for each group, but none exceeded 4 minutes. The second video introduced participants to the concrete task that they should perform. This video also introduced a window behind the robotic gripper that would guide them on where to position the gripper and when they had successfully completed the task. In total, this training session took participants approximately 10 minutes to complete.

¹ <https://www.youtube.com/playlist?list=PLQHWcSK2-Zw5qIpRAZgEsMKL8Bzx1rmho>

5.2.4 Experimental Task

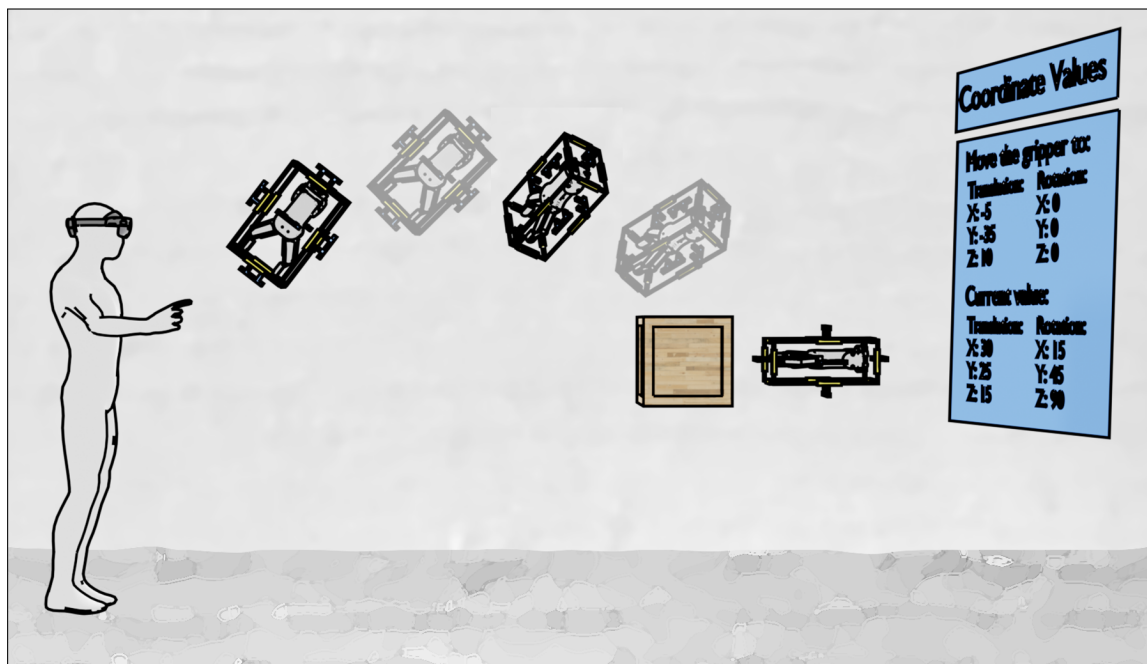


Figure 5.3: Illustration representing the task performed by participants in our first experiment. Participants were required to move a holographic gripper to a specific position and orientation in the mixed-reality space. A console provided the axes values that participants should reach to complete the task, and a box was provided as a reference for the final position. In this example, the bounding box interface is being used.

Immediately after each participant finished watching the two instructional videos, they were asked to put on the MR headset and begin solving their task. Participants were given 15 minutes from this point to complete the task. For participants who exceeded the time limit, proctors counted the task as failed and asked them to stop. This cut-off time was intended to be generous, as successful informal test runs rarely took longer than 5-8 minutes. As illustrated in Figure 5.3, participants were asked to move a virtual robotic gripper to a specific, pre-defined position that required movement and rotation on all three axes. The virtual display panel, seen in the upper right (blue in mockup view), specified the goal position and the current position.

To give the user feedback on their progress towards the goal, if the axis location or rotation values moved into an acceptable range, the text of that axis would turn green. The task was considered complete once all positional and rotational values were within an acceptable range. As additional guidance, participants were also provided with a virtual small shipping box as a reference point (shown in Figure 5.3). Once positioned correctly, the robotic gripper would fit perfectly around the box as if it were supposed to pick it up. This box made the goal of the task more intuitive for participants, providing them with a visual goal, not just a numerical goal.

Due to the limited number of Hololens 2 headsets available, the experiment took place over a period of five days, where participants of the same group were assigned to different

hours on the same day. The training was also staggered to avoid bias due to different delays between the first training session and the experiment. Therefore, all participants had an almost identical delay between their first training session and the experiment. While proctors monitored participants, they were not allowed to help participants use the interfaces or answer questions related to them. They were, however, allowed to clarify other elements of the task, such as how they could tell when the task was complete. Proctors had to step in and reset the experiment for a small number of participants when they encountered unexpected technical issues (e.g., the application closed, the headset turned off). In these cases, the duration time required to reset the experiment was not counted towards the participants' task completion time.

5.2.5 Measures

There were two main ways of measuring the effects of the MR interfaces on participants: measuring performance and surveying satisfaction. To measure their performance on the tasks, proctors annotated whether participants were successful at completing the positioning task and, if so, the time in minutes and seconds that it took for them to complete the task. After participants had completed the experiment, either successfully or due to running out of time, they were asked to complete a brief post-experiment survey. The survey consisted of three parts, and there was no fixed time limit to complete it.

The first part of the survey consisted of a set of demographic questions, where participants were asked about their major, current year in college, and experience with mixed reality and robotics. The second part of the post-experiment survey consisted of a standardized questionnaire to determine the System Usability Score (SUS) of the used system. The SUS was developed as a straightforward way for users to quantify the perceived usability of a system (BANGOR, P. T. KORTUM, *et al.*, 2008). This standard questionnaire contains ten Likert-style items on a 5-point scale that are general enough for different application types. Once completed, participants' responses were converted into a single score ranging from 0 to 100 points. Since the SUS has been determined for a wide range of systems, comparing their usability and determining the percentile of a system's score is possible.

Following the SUS questionnaire, there was the third part of our survey, where participants were asked two open-ended questions about their assigned interface, one where they could report what they found easy when using the interface and another what they found difficult. These questions were analyzed qualitatively using open card sorting and constant comparison by three researchers contributing to the experiment (SPENCER, 2009; DYE *et al.*, 2000). While these questions were optional, most participants responded to them, allowing researchers to understand the difficulties they might have faced. At the end of the questionnaire, an open space was also provided for participants to report their overall experience while participating in the experiment. Although this final feedback was not analyzed in the qualitative analysis, it can be used as a reference for future studies.

5.3 Study Stage 1: Results

This section discusses the results of the first phase of this work, starting with demographics, followed by participants' performance, and then their usability scores and

feedback.

5.3.1 Demographics

A total of 49 participants attended the study. Overall, 12 participants used the button-based interface, 11 participants used the object manipulator, 10 used the bounding box, 9 participants used the joystick, and 7 participants used the slider-based interface. As the study was advertised through Computer Science classes, all participants except for two had Computer Science as their primary field of study, with the remaining participants being in Biomedical Engineering and Music. Participants were also asked about their level of studies. A total of 33 participants (67%) indicated being seniors, 14 being juniors (29%), 1 being a sophomore student, and 1 part of a graduate program (2% each). When asked about their experience with mixed reality, 37 of our participants (76%) reported no prior experience. Forty-one participants (83%) reported that they had never worked with robots before in any context or through any mode of interaction. Participants were randomly assigned to an interface, and a post-hoc sanity check showed an approximately even distribution across interfaces for both MR and robot experience.

5.3.2 How Do Interfaces Affect Performance?

The primary focus of the first phase was to evaluate how well participants performed on the task using the different interfaces they were assigned. Figure 5.4 shows the completion rates and average time participants spent using each interface. The x-axis shows the different interfaces, while the bar charts illustrate the completion rates per group using each interface. The black dots show the participants' average time to complete the task using each interface.

Overall, 27 participants (55%) were able to complete the task. However, substantial differences can be observed among the different interfaces. All the 7 participants who used the *Sliders* interface could complete the task, followed by 10 out of 11 participants who used the *Buttons* (92%) and 8 out of 9 participants trying the *Bounding Box* interface (89%). In contrast, only 1 out of 8 participants who used the *Joystick* interface completed the task (11%), followed by the *Object Manipulator* interface with 1 out of 11 participants completing the task (9%). Fisher's exact test was employed to verify if there is an association between the success of participants and the interfaces they tested (KIM, 2017). The test results rejected the null hypothesis ($p\text{-value} < 0.05$), suggesting an association between the success in completing the assigned task and the interface participants used.

5.3.3 How Do Users Rate Interface Usability?

Following the performance analysis, the average SUS score was calculated for each interface based on the feedback given by participants in the post-experiment questionnaire. As suggested by prior studies (HARRISON *et al.*, 2013), SUS scores can not be compared on a linear scale, so the average percentile scores for each interface are reported. Figure 5.5 shows the results of the SUS analysis. The dashed black line is based on literature and depicts the percentile curve that was derived from scores reported across a wide range of systems. The vertical lines represent the average usability scores of each interface analyzed

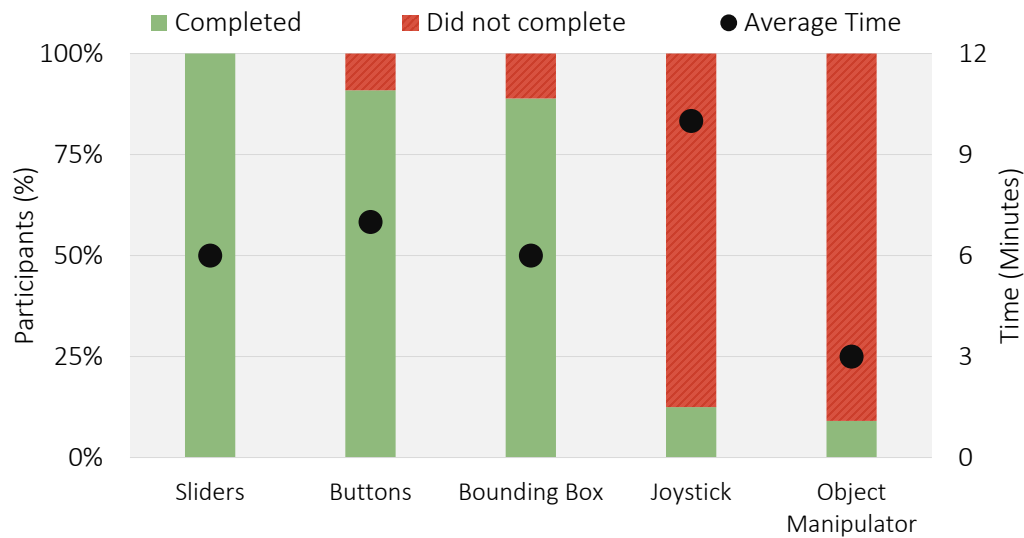


Figure 5.4: On the left axis, completion rates (in percentage) for groups of participants trying each interface. On the right axis average completion times (in minutes) for participants who finished the experiment.

in this experiment. By determining the intersection of the percentile curve and each interface’s vertical line, one can determine their percentile rank across all systems.

The *Bounding Box* interface achieved the highest average score of 77.8 (min: 70, max: 92.5, sd: 7.3), placing it in the 80th percentile of all interfaces. This was the only interface to fall into the “Good” or “Acceptable” category, according to the commonly used SUS interpretation (BANGOR, P. KORTUM, *et al.*, 2009). All other interfaces were “Marginal” or worse. The *Buttons* interface was rated with an average score of 69.4 (min: 50, max: 87.5, sd: 11.7), placing it around the 58th percentile. The *Sliders* interface, reaching an average score of 57.5, ranked a little lower than the 30th percentile (min: 35, max: 75, sd: 12.2). The *Joystick* and the *Object Manipulation* interfaces were rated the worst, scoring respectively 41.4 (min: 22.5, max: 67.5, sd: 15.0) and 42.5 (min: 20, max: 67.5, sd: 15.0), placing them both in the 10th percentile.

5.3.4 Which Aspects are Difficult and Easy to Understand?

In addition to the SUS questionnaire, participants were asked about the aspects of each system that they found easy or hard to use. Table 5.1 summarizes participants’ feedback broken down by interface. Several users pointed out issues that were related to **hardware limitations**. These issues are highlighted in orange or blue. Issues in orange are fully caused by hardware limitations, and issues in blue are at least partially caused by hardware limitations.

For instance, *Joystick* users mentioned that “[positioning was imprecise] due to the HoloLens not always picking up hand movements”, “if the [HoloLens] camera moved out of the arms view you lose the ability to move/control the arm”, and “The FOV of the object also was very narrow, which provided some difficulty”. Similar hardware issues were mentioned for other interfaces and have been highlighted in orange or blue. Note that these hardware

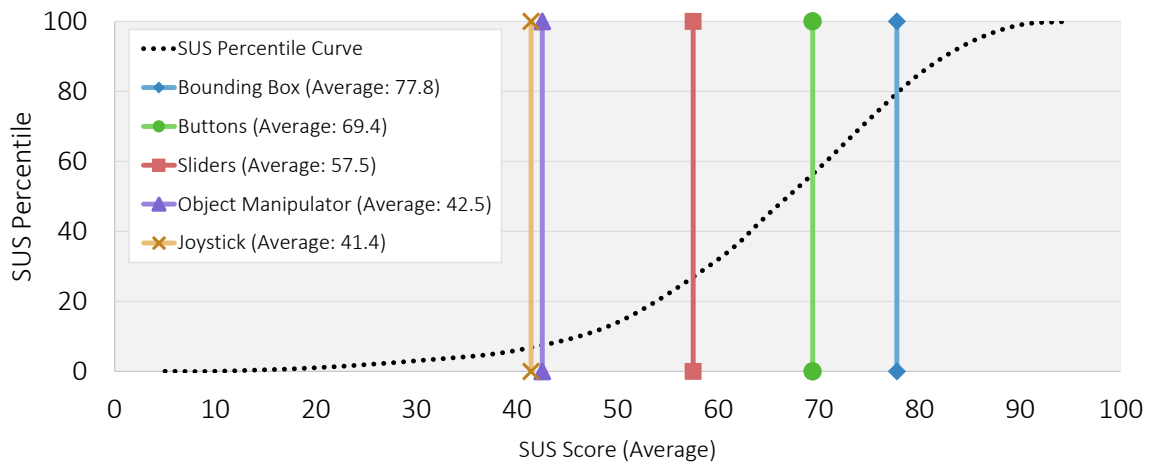


Figure 5.5: Average usability score for each mixed reality interface. The dashed black line defines a SUS score threshold line from a wide range of evaluated systems.

issues are often due to the relative immaturity of MR technology and will likely decrease or even disappear as improved hardware is released.

The top of Table 5.1 contains comments from those that used the *Bounding Box* interface. Most participants (7 of 10) commented that this interface was **easy-to-use** for them, saying “*I thought moving it around was **easy***” and “*..it was **easy** to physically grip the objects used to move the arm and move them*”. Several mentioned specifically that this interface was **intuitive**, saying “*..being able to determine which actions would be performed by the different motions you could do was pretty **self-explanatory***” and “[*it was*] **easy-to-tell** where I was supposed to grab/use”.

Next, the responses of participants who used the *Object Manipulator* are shown. Most participants (8 of 11) mentioned that, with this interface, achieving **precision was difficult**, saying “*precision movements were **very difficult** to use*”, “*doing fine motions is **near impossible**. I am not myself a robot; I cannot hold my hand perfectly still, as this task required*”, and “*it was basically **impossible** to position the object with a fine degree of control*”. Despite these complaints, several participants praised its direct and easy-to-understand manipulation. They thought it was effective for **gross positioning**, saying “*..you could roughly **approximate** an orientation and position with relative ease*”, “*..**general** movement was easier*”, and “*easy to move the object around **..generally***”.

Participants gave mixed feedback for the *Sliders* interface, although all seven participants successfully used it. Several users found the interface easy to comprehend and operate, but opinions on its fit for precise movements were split. In theory, sliders allow very fine-grained adjustments, but it appears that some users found it hard to control the interface with the necessary level of precision, saying “[*I had a*] **difficult** time to get the exact number” and “*..minute changes are difficult*”. As with other interfaces, some users mentioned problems related to **hardware limitations**, such as difficulty grabbing the slider handles in MR (e.g., “[*I had difficulty*] **having my gesture register in the interface**”).

Participants gave mixed feedback on the *Buttons* interface. Several participants (6 of 10) had difficulties getting the interface to recognize their button presses, again a symptom of

MR hardware immaturity, saying “*the biggest issue I found was the button only registered every 3-5 clicks*” and “*it wouldn’t register some taps so I would have to tap more than once*”. Despite these difficulties, participants found this interface easy to understand. The proctors confirmed that participants understood and were able to use this interface, but they also noted that, because it often took many button presses to position the robot correctly, this interface took time to use, an insight that is corroborated by its average completion time which is the highest of the top three interfaces.

The participants’ feedback for the *Joystick* interface was overwhelmingly negative. Many participants stated that they struggled with the joystick moving the robot arm in multiple axes simultaneously, saying “*trying to manipulate one axis after adjusting another (i.e., doing the z-axis after the x) moved the different coordinates as well, which confused me a lot*” and “*..it was really difficult to move just X or Y by itself, often the sensitivity was so high that if I moved it in the X direction, it would simultaneously move in the Y direction as well, and the Z direction would activate almost every time I grabbed the ball*”. Although joysticks may be a great physical controller for cobots, their implementation in MRTK2 still suffers from hardware limitations.

5.4 Study Stage 2: Expert Feedback on Manipulating a Physical Robotic Arm

In the second stage of the study on MR interfaces, the best-rated interface from Stage 1 was used: the *Bounding Box* interface. The hypothesis in the second stage is that such an interface could facilitate the manipulation of a physical robotic arm and gripper. A full interface prototype was implemented based on an almost identical user-facing presentation from Stage 1. The primary difference for this study stage was that the prototype targeted a physical robot arm and gripper instead of a virtual robotic gripper. A total of 11 experts in the field were recruited to conduct interviews and gather their opinions on this final prototype. They were asked about its potential use and limitations. The following section explains in detail how the second stage of this study was conducted.

5.4.1 Prototype Implementation

A prototype that targets a collaborative ABB GoFa CRB 15000 robot arm was implemented. As shown in Figure 5.6, the *Bounding Box* control interface was placed in the same position as the physical gripper installed on the robot. Users were able to use the *Bounding Box* holders to control the gripper and, consequently, the robot. A panel with two sliders was also placed beside the robot so users could control the robot’s movement speed and open and close the robot’s gripper hand. To implement this prototype, a similar approach from the first phase was used. The Unity engine and the Mixed Reality Toolkit 2 (MRTK2) were used to implement the application in mixed reality, and the ABB Externally Guided Motion (EGM) interface was used to establish communication with the robot hardware. To make the robot move with little delay as users adjusted the *Bounding Box*, any updates in MR were continuously converted to physical joint parameters, making the robot move into the updated state. Using this approach, the response time between the *Bounding Box* and robot hardware was less than one second. The robot speed available in the slider interface

| Bounding Box | |
|--|--------------------|
| Comment | Occurrences |
| It is easy to identify and use bounding box handles to move objects. | 7 |
| It is difficult to release objects with precision (**). | 4 |
| It is difficult to position objects with precision (**). | 3 |
| The layout is easy and intuitive. | 2 |
| The field of view was small (*). | 2 |

| Object Manipulator | |
|---|--------------------|
| Comment | Occurrences |
| The interface was too sensitive and made it difficult to be precise (**). | 8 |
| It is easy to pick up objects. | 6 |
| It is easy to translate objects. | 6 |
| It is difficult to release objects with precision (**). | 5 |
| It is easy to understand how the overall manipulation works. | 3 |
| It is not possible to manipulate a singular axis at a time (**). | 2 |

| Sliders | |
|--|--------------------|
| Comment | Occurrences |
| It is difficult to grip the slider (**). | 4 |
| It is easy to manipulate the slider handle. | 3 |
| It is difficult to perform precision movements (**). | 3 |
| It is easy to grip the slider handle. | 2 |
| It is easy to get precise values with the slider handle. | 2 |
| The field of view made it difficult to handle the slider while positioning the object (*). | 2 |

| Buttons | |
|---|--------------------|
| Comment | Occurrences |
| It is difficult to press the buttons (**). | 6 |
| It is difficult to recognize a button being pressed (**). | 5 |
| Pressing the buttons is easy. | 3 |
| The buttons are intuitive to use. | 2 |
| The interface is well organized and easy to understand. | 2 |

| Joystick | |
|--|--------------------|
| Comment | Occurrences |
| It is difficult to manipulate one axis at a time. | 5 |
| It is easy to translate the object using the joystick control. | 3 |
| It is difficult to perform precision movements (**). | 3 |
| It is difficult to grab the joystick control (**). | 3 |
| It is easy to pinch the joystick control. | 2 |

Table 5.1: Most frequent comments from participants in the post-experiment questionnaire. Comments highlighted in blue (*) represent issues fully caused by hardware limitations, and issues in green (**) at least partially caused by hardware limitations.

was also controlled with the EGM interface, and the opening of the gripper was controlled using the REST API provided by the gripper manufacturer.

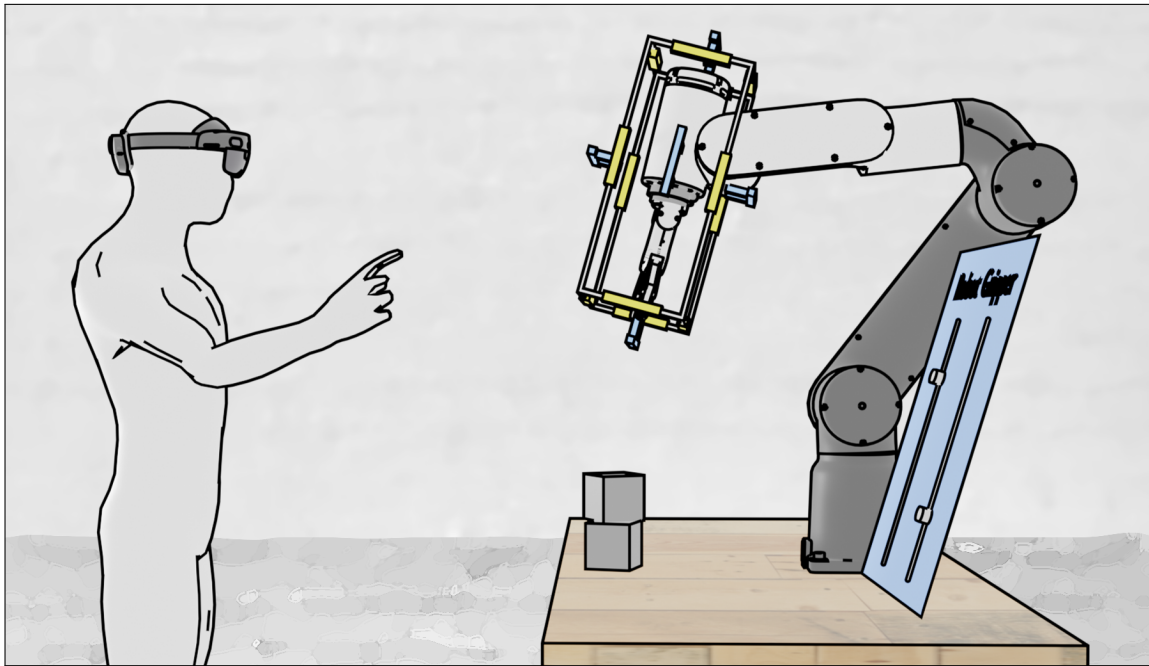


Figure 5.6: Illustration representing the trial performed by experts in the second phase. The bounding box was adjusted to fit the robot manipulated by the users. A panel with sliders allowed users to adjust the gripper opening and robot speed.

5.4.2 Interviews

A total of 11 expert participants were invited to use the prototype and participate in follow-up interviews. The recruitment process was focused on professionals with expertise in related fields, as they were expected to have the highest likelihood of using similar systems and, therefore, the ability to compare the prototype to other technologies. Nine of the participants invited were employees of the same university in the United States, and the other two were invited researchers from nearby institutions. Among their backgrounds, there were three professors in Computer Science, one in Design, and two in Electrical and Computer Engineering. Two Ph.D. students in Computer Science and three in Mechanical Engineering also participated.

A proctor scheduled a private session for each participant to try the experiment and perform the interview. The same proctor instructed the participant on how to use the prototype by following predefined guidelines². The experimental workspace consisted on the HoloLens 2 running the prototype, the ABB robot connected to the prototype application, and three cubes on the table where the robot was mounted to. The participants were not instructed to solve any concrete task but were suggested to interact with the cubes while playing with the prototype. We considered that picking the cubes up at the

² https://github.com/fronchetti/TOCHI-2024/blob/main/resources/proctors_guide_experts.pdf

right angle would be a valuable way for participants to explore the robot's controls and the precision of its positioning in MR.

Each interview lasted approximately 45 minutes, and three questions guided the discussion: First, the proctor asked each expert about their overall impression of the prototype. Then, participants were questioned about the practical applications for which they could see the prototype being used. Finally, the proctor asked the participants how the prototype could be improved, for example, to make it more widely applicable or to overcome existing limitations.

5.4.3 Interview Analysis

A qualitative analysis of the interviews was performed to get insights from the feedback provided by the experts, following the same open card sorting and constant comparison strategies used in Stage 1. First, one reviewer transcribed each interview in its entirety, and the three different discussion topics were separated. Then, the same reviewer divided each topic into multiple paragraphs, each containing one statement made by the participants or a question or response by the proctor. This raw material allowed a more in-depth analysis of the interview responses.

The remaining qualitative analysis was performed by three researchers collaborating in this study, over three separate rounds. In the first round, researchers randomly selected three out of the eleven interviews to establish a common coding standard. Each researcher analyzed the selected spreadsheets independently. For each statement made by a participant, they assigned several codes that they found appropriate. The codes assigned by the authors were discussed in a follow-up meeting with all of them present. During the meetings, the researchers organized the codes in a table for use in the subsequent rounds. They also calculated their agreement by dividing the number of rows where all the researchers agreed and by the total number of rows. During this stage, the agreement rate was 58%.

In the second round, researchers selected two more of the remaining spreadsheets to validate the codes established in the first round. The researchers applied the codes from the first round and potentially added additional ones. Repeating the same process of discussing and refining the codes, the researchers reached an agreement of 85% during this second round. Considering the high agreement reached in the second stage, researchers used a different approach for the third coding stage. They divided the six spreadsheets that had not been analyzed in the first rounds among the three researchers, with two individually assigned to each researcher. In a final meeting, they discussed the codes generated and created a final set of codes to make them self-explanatory and unambiguous for later presentation. All three researchers organized the final codes in a table and mapped previous codes to the final set.

5.5 Study Stage 2: Results

In this section, the codes generated in the qualitative analysis of the second stage are presented. Table 5.2 presents the final set of codes derived from interview responses, as well

as the number of participants in whose responses the code appears. For conciseness, only comments that at least two interviewed participants have made were listed. The table is divided into three parts: general impressions (related to the interview question: “*What were your impressions on using this interface to move the robot?*”), practical applications (related to the interview question: “*How could this interface be utilized in real-world settings?*”), and improvement suggestions (related to the interview question: “*How this interface could be improved?*”).

When asked about their overall impression of the prototype, participants once again highlighted hardware and software limitations found in Microsoft HoloLens 2. However, this time, their responses were more fine-grained, and they commented on their frustration about the physical object (i.e., the gripper) obstructing their interactions with the *Bounding Box*. The most frequent comment highlighted by 8 participants (72%) was that pinching an object in mixed reality was challenging to them. Because the *Bounding Box* was placed around a robot gripper, some of the interface’s handles were harder to reach, and the physical presence of the robot gripper seemed to make these handles more challenging to pinch. One participant said: “*The challenge I’m having is grabbing anything. Because of the way these are all positioned, [let’s] say, in the center of the Bounding Box, it makes it very difficult to grab them*”.

Often, the hand detection system did not work as expected, replicating issues with pinching that were already present in phase one of the study: “*I’m going correctly to the handle; but, when I close my hand sometimes it does not respond correctly to my command*.” While some issues reported by experts were indeed caused by the presence of the physical object—if the gripper was between the headset and the hand, the headset could not scan the hand movements—we do not believe that this was the most common cause for their frustration. Instead, it was observed that the success rate of the participants as they tried to pinch increased substantially throughout their session with the tool. We speculate that part of their issues might be caused by a lack of experience in interacting with physical objects through mixed reality.

Another frequent comment participants made was about the limited field of view of Microsoft HoloLens 2. Similar comments were reported in the first stage. This issue is related to the inherent limitations of the used hardware and could not be fixed. A total of 7 participants (63%) mentioned the field of view as being too small for this specific interface and task. One said: “*Honestly, (it would be better) if the lenses were bigger. Um, because currently, like, I can’t see the whole box at the same time. Like I can (only) see the middle section, or the top section, or the bottom section*.” Note that the *Bounding Box* used in this prototype was pretty large as it covered the gripper and the extreme part of the robot arm. The limited field of view would be less of an issue for smaller physical objects.

Another five users (45%) complained about the behavior of the *Bounding Box* interface being unclear or not intuitive to them. Among their comments, participants mentioned difficulty in understanding how to use the interface: “*I thought maybe the robot was not tied to how the Bounding Box is (moving), but, as I’m watching its motion, it is*.” This issue may have occurred mostly because participants were unfamiliar with this specific type of manipulator and because they did not receive explicit training materials or instructions. As most participants quickly became better at using the interface throughout their session,

this wasn't considered to be a substantial issue. Systems in production typically come with clear instructions or training tools, which was not the case for this experiment.

Participants also made other, less frequent comments, including the difficulty of using mixed reality without proper training, issues while manipulating the gripper interface, issues with the pinpointing feature used to manipulate objects from a distance, and the lack of interaction between the mixed reality workspace and the physical environment. In general, most of their issues are related to hardware limitations of the headset or would only require minor changes to the prototype to be addressed. Incremental improvements in mixed reality devices and more extensive user testing of interfaces could eliminate most of these issues.

When asked about improvement suggestions, 6 participants (54%) mentioned the idea of moving specific handles to more reachable positions, which could potentially solve the challenges of pinching them, as pointed out by one participant: *"The ones [handles] on the back are useless. So if you could give me this one right here [from the back] in the front..."* While this is a reasonable suggestion, it is valid to note that even "back" handles can be easily reached by moving to the opposite side of the table, a solution unique to MR that might not have been obvious to participants. Another 3 participants (27%) also suggested the idea of including visual markers to support users during the manipulation of the interface. One said: *"I think the color of when grabbed it [the gripper] and when you've let go [should change]. I think that would help you know that you have actually let it go."* This suggestion seems quite reasonable, and we expect it to be incorporated into future versions of the underlying widget library. Additionally, 3 participants (27%) suggested explicit training tools to support first-time users of mixed reality. A few participants also made other suggestions, such as the use of other types of control to manipulate the robot, design suggestions for the gripper interface, and an increase in the size of the *Bounding Box* handles.

When asked about how the *Bounding Box* interface could be used in practice, 5 participants (45%) suggested it as valuable to manipulate objects remotely. One said: *"I guess if you got to be dealing with toxic chemicals or something. You're on one side of, like, an airlock or something and got to do something from the other side."* Another participant combined the idea of precision with distant manipulation: *"I think the best use case for me would be: I have to stand off from something and manipulate it, but the automated algorithms can't do what I want it to do. I have to do it in a more precise way. That's where I think it would be useful."* Other similar ideas were given by the participants, including manipulating heavy objects, which would require a certain distance from the robot, manipulating sensitive objects, and using this interface in assistive health care. While these suggestions are certainly valid, they focus on a different use case than those we targeted, remote manipulation. Our study intended to investigate the physical-virtual interaction through in-person manipulation. Manipulating heavy or dangerous objects without direct touch may be good in-person use cases to consider.

| General Impressions | |
|---|-------------|
| Comment | Occurrences |
| Pinching in mixed reality is challenging. | 8 |
| The field of view of the headset view is small. | 7 |
| The interface behavior is not clear. | 5 |
| The gripper interface may not work as expected. | 2 |
| Pinpointing in mixed reality is challenging. | 2 |
| The mixed reality application has no sense of its physical workspace. | 2 |

| Improvement Suggestions | |
|--|-------------|
| Comment | Occurrences |
| Some bounding box controls should be moved to reachable positions. | 6 |
| Markers could be included to improve visual feedback. | 3 |
| Training on mixed reality should be given to first-time users. | 3 |
| Other types of controls could be included within the interface (e.g., joystick). | 2 |
| The grippers' interface should be redesigned. | 2 |
| The size of the bounding box controls should be increased. | 2 |

| Practical Applications | |
|--|-------------|
| Comment | Occurrences |
| The interface could be used to manipulate distant objects. | 5 |
| The interface could be used to manipulate heavy objects. | 3 |
| The interface could be used in assistive health care. | 2 |
| The interface could be used to manipulate sensitive objects. | 2 |

Table 5.2: Most frequent comments from experts during their interview (i.e., commented by > one participant).

5.6 Discussion

This section summarizes discussions over the two phases of this study. The focus of this section is to bring light to future studies, illustrating how MR can be applied in the manual control of cobots.

5.6.1 Precision and Hand tracking

Many of the issues that were pointed out by both the novices and experts in the proposed study can be traced to a lack of precise hand tracking, including problems with object positioning and pinching detection. Because many MR headsets use some form of hand-tracking technology to interact with the user interface, it is expected that this problem will persist, independent of the used hardware. However, the Microsoft HoloLens 2 hardware used in this study amplifies the problem further, as hands must be visible in the limited field of view of the headset to be detected. Although it is convenient to manipulate an object using only one's hands, industry-level applications may require additional tools to achieve higher precision. Other studies have already explored the restricted capabilities of hand-tracking features in MR devices, and proposed solutions

for this problem (PEDDIE, 2023; SOARES *et al.*, 2021; MILLER *et al.*, 2020). For instance, the performance of object positioning could be improved to millimeter accuracy by combining MR with other technologies, such as haptic gloves and external hand trackers. This trade-off between the ease of use and the current limitations in MR technology should be considered by developers who target applications where precision is essential. We hope that further studies and prototypes can provide them with additional insights into the best compromise that can be achieved with currently available technologies.

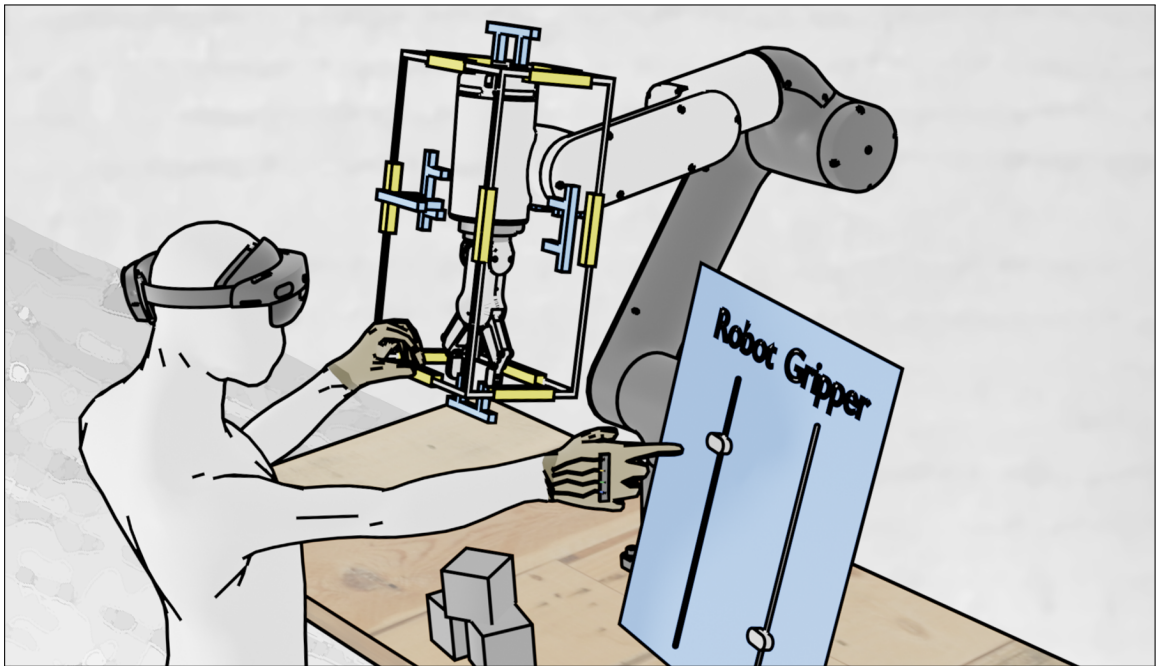


Figure 5.7: Illustration representing how haptic gloves (in brown) could replace hand tracking in our prototype. A robot operator visualizes and controls a Slider interface on the right side of the robot, while he also manipulates a Bounding Box interface positioned over the robot gripper without looking at it.

5.6.2 Field of View

The limited field of view in mixed reality was another common issue among participants. However, the related problems with hand tracking might have caused them to pay more attention to this issue than they would have otherwise. If, instead of hand tracking, an alternative input method not tied to the field of view of the headset was used, the field of view would likely become less of an issue for most users. They would be able to rely on the feedback from the real robot when it is moved by the controller, as well as the information in the field of view of the headset.

Based on our experience using the Microsoft HoloLens 2, we believe that its field of view might be sufficient if the confounding hand-tracking issue is removed. Figure 5.8 illustrates how the replacement of hand tracking could minimize issues associated with the field of view. Although additional devices may enhance the experience of users controlling robots in MR, it is essential to highlight that each device may impose new limitations on users' experience, and further studies are necessary to understand better which solutions

work best for collaborative robots.

5.6.3 Practical Applications

Using holographic projections to allow a user to manipulate a robot from further distances was the most frequently suggested application for the final prototype. Prior studies have also suggested this use case in the context of applications for alternative MR interfaces MAKHATAEVA and VAROL, 2020. Remote operation is not only a standard application in industrial and collaborative robotics but is also especially important when safety concerns may impose restrictions on human collaboration SHERIDAN, 2016. Many industrial robots can work in hazardous environments but depend on human interaction to be programmed. For a long time, the manipulation of robots in hazardous environments depended on the use of cameras, physical controls, and computer screens TREVELYAN *et al.*, 2016. Mixed reality might be able to introduce new ways for users to interact with holographic copies of robots and controls in real time without depending on the limited capabilities of solutions such as VR-based environments.

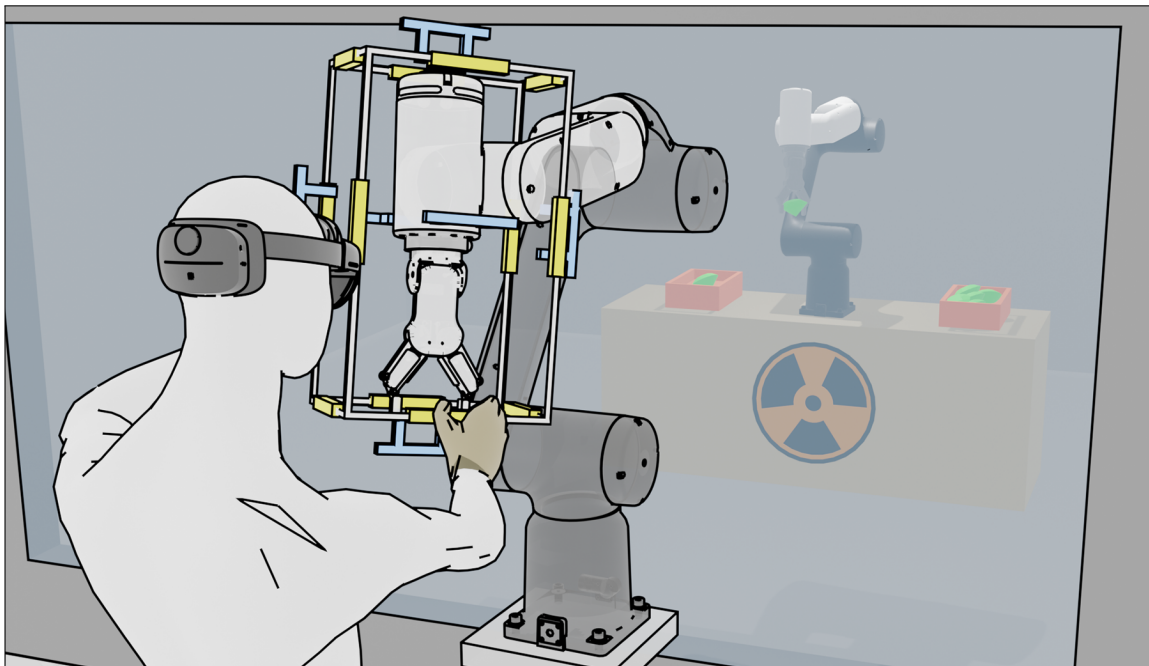


Figure 5.8: Illustration representing how distant manipulation could be applied to mixed reality. An operator uses HoloLens to manipulate a holographic copy of a robot to manipulate it from another room. The robot is placed in a room where radioactive objects are placed.

5.7 Threats to Validity

We have identified the following threats to validity regarding the presented work:

Construct Validity: The five interfaces compared in the first phase were prototype implementations based on frameworks provided by MRTK2. These prototypes might not be comparable to commercial-grade, more thoroughly polished, and tested implementations.

The results provided are relative to these prototypes, and the design decisions made in our implementation may have impacted the outcomes found for this study. In addition, while we tried to cover a wide range of interface types, other interfaces or frameworks might use MR differently or more effectively. Also, the Microsoft HoloLens 2 MR device in this study, although well-established in the market and used in literature, could be replaced by other device options, resulting in different user feedback and performance.

It is important to stress that the joystick interface created was not based on a native feature of MRTK2 but mirrored the design of an analog joystick. Although we used the best of our knowledge to implement it, we believe that it could function better if it were adapted more thoroughly to the capabilities of MR. The task provided for participants trying the interfaces in the first phase may have also impacted the results of this study.

Internal Validity: When we trained our participants, we attempted to re-create a realistic experience where users were first exposed to MR and later given a realistic task in a separate session. However, one session might not have been sufficient to introduce our participants to the environment, and they might have performed better overall if more thorough or differently staged training had been made available to them. While we believe that this bias would have affected all of the environments we compared similarly, we cannot verify whether this is the case.

External Validity: Our study was performed on undergraduate students, who might not be representative of real-world end-users using MR interfaces as part of their work. In particular, robot workers or programmers might have more experience with manipulating robot arms through other non-MR means, and that experience might affect their performance. In addition, the task we asked users to solve was only one example of a practical MR task. While we designed this task to capture a wide range of positioning steps, real tasks might bring different challenges or focus on different steps that our task was not able to emulate.

To better evaluate how mixed reality interfaces can be used for real-world tasks, the *Bounding Box* interface was implemented to control a real-world system. The *Bounding Box* interface was chosen for this development because it contained the highest usability score from the experiment completed in this paper. To connect the *Bounding Box* with a real-world system, a mixed-reality application was created that utilized WiFi to communicate with an ABB Gofa robot. The application consisted of a *Bounding Box* that would be used to control the positioning of the robot. In addition, a slider interface was used to vary the speed at which the robot repositioned itself. This was to enhance the precision of the system. Several experts in robotics were asked to evaluate the application and provide feedback on the system. This feedback was provided via an open discussion led by the following prompts: "Tell us your impressions about using Mista to move the robot."; "Do you believe that Mista could be utilized for real industrial applications? Why or why not?"; "For future developments, how would you improve Mista?". After analyzing the feedback provided by these users, several trends were discovered.

For the project's future development, the weaknesses of some interfaces could be enhanced or joined with other interfaces. For example, as one participant mentioned, rather than allowing the robot to move whenever the button is pressed, a feature could be implemented that would move the robot whenever the button is pressed and held down.

This would create a less repetitive method for moving the robot. In addition, multiple interfaces could be used together so that the strengths of each feature could complement each other.

One idea would be that a slider or button interface would be used to change the speed at which the robot will move, and the *Bounding Box* would be used to implement the position. This means that a higher value of speed could be used for more general positions, where precision is not as necessary, and then a slower value of speed would allow users to position the robot in the location they would like.

Another additional tool that could be implemented with the robot would be path programming. Currently, our experiment seeks to analyze the positioning of a robot at a single point in space. Path programming is another programming method used in robot applications today. Path programming utilizes a succession of many points in space to accomplish a task. Using MR, multiple points could be implemented in the virtual space, where some could be more precisely defined if so desired. We also leave this suggestion for future studies.

5.8 Conclusion

In this chapter, mixed reality (MR) was explored as an alternative solution to the manual control of collaborative robots. The study was divided into two phases: The first phase identified which MR interface was the most appropriate for the manual control of a virtual robotic gripper and what challenges users still face in MR, especially given the early stage of the hardware. The second phase identified how experts thought the best-evaluated interface should be deployed in a real context. Experts in the second phase had the opportunity to play with a real collaborative robot and control it using a prototype in MR. From this study, it was possible to identify important challenges that still limit the applicability of MR devices in robotics, including hardware constraints (e.g., limited field of view) and software-related issues (e.g., low accuracy of hand tracking systems). Future work is necessary to expand the knowledge about MR in industrial and collaborative robotics.

Chapter 6

Learning in End-User Robot Programming Environments

Collaborative robot programming is usually carried out in an industrial setting (SHERWANI *et al.*, 2020). As most cobots are sold to factories, the practices followed by robot manufacturers are still based on industrial standards, and ignore what works best for end-users (EL ZAATARI *et al.*, 2019). While in computer programming learners have access to many informative resources online, technical manuals are the primary source of information for developers in cobot programming. Furthermore, robots are so expensive that most developers do not have access to one, which stunts the growth of online user communities that would otherwise produce shared learning materials (HENTOUT *et al.*, 2018). If a novice decides to learn a computer programming language (e.g., Java), they have instant access to a huge array of learning resources. In contrast, a novice learning a robot programming language (e.g., RAPID) is largely on his own (ROSSANO *et al.*, 2013).

Robot manufacturers are beginning to understand that they must provide prospective customers with the learning resources they need to operate and program their robots. ABB¹ and Universal Robots², for example, have been investing in online learning academies, conducting in-person robot training, and creating publicly available training videos. But what kind of learning resources work best to support novices? What learning challenges do they tend to encounter? To investigate these questions, we invited 35 students with (some) previous experience in general purpose programming to try collaborative robot programming for the first time. We asked them to solve a pick-and-place task using a one-armed robot. To program the robot, they used the Wizard Easy Programming Tool³, a block-based programming approach released by ABB Robotics in 2020.

To assist participants with any learning barriers, we provided them with a help desk. On the help desk, participants had access to a web application containing learning resources divided into three categories: videos, textbooks, and a chatbot trained to answer questions about robot programming. Every time a participant accessed the help desk, we asked them

¹ [://new.abb.com/products/robotics/service/training/online-learnings](https://new.abb.com/products/robotics/service/training/online-learnings)

² <https://academy.universal-robots.com/>

³ <https://new.abb.com/products/robotics/software-and-digital/application-software/wizard>

to describe what challenge/barrier they were facing. To help them classify their problem, we listed six learning barriers from a study on end-user programming systems (Ko, B. A. MYERS, *et al.*, 2004). The list included learning barriers associated with programming, such as "*I think I know what I want the robot to do, but I don't know what to use...*" and "*I think I know what to use, but I don't know how to use it...*".

Our experimental study identified that participants prefer chatbots when getting assistance in a robot programming task. Of the 35 students, only 25 used the helpdesk during the experimental task. Of those 25, 16 preferred the chatbot, 6 the videos, and 3 the textbooks. Among the most frequent challenges that participants faced while trying to program our collaborative robot were on how to translate programming logic to the block-based environment (i.e., "*I think I know what I want the robot to do, but I don't know what to use...*"), how to use the programming features available (i.e., "*I think I know what to use, but I don't know how to use it...*"), and how to deal with mistakes (i.e., "*I thought I knew how to use this, but it didn't do what I expected...*"). In the following sections, we explain how we designed our robot programming task and how we captured participants' learning barriers and favorite learning resources. We also explain in detail our results and discuss outcomes to the robot programming domain.

6.1 Experimental Method

Our study aims to explore learning barriers and potential learning resources in end-user robot programming environments. To this end, we conducted a robot programming experiment involving a one-armed robot and a block-based programming environment. Thirty-five participants were invited to implement a solution equivalent to a data sorting algorithm in Computer Science. Throughout the process, we captured their feedback regarding the challenges they faced in implementing the experimental task. We also provided participants with different types of learning materials to evaluate how different learner-content interactions can help end-users program robot solutions. In the following subsections, we provide details about the development and execution of this experiment.

6.1.1 Recruitment

We recruited participants from two software engineering classes within a single Computer Science department to participate in our experiment. Our goal was to explore learning barriers in robot programming, so we targeted students with former programming experience to avoid learning barriers not associated with robotics (e.g., difficulties in comprehending relational or conditional expressions). Other robot programming studies have followed a similar approach, showing students can be an effective participant pool for studying robot programming (DJURIC *et al.*, 2017; KARLI *et al.*, 2024; FRONCHETTI, RITSCHEL, SCHORR, *et al.*, 2023).

Our recruitment process involved sending out personalized invitations via email, explaining the purpose and benefits of the study, and verbally advertising the experiment in class. Each participant was scheduled for an individual session through an online scheduling platform. Participation was voluntary and performed in person in our laboratory, with a maximum completion time of one hour and twenty minutes per participant. As

compensation, students who tried our experiment received extra credit points in class. All participants had their involvement approved by an institutional review board.

6.1.2 Experimental Task

Our goal in designing the task was to exercise most of the programming features available in the Wizard Easy Programming Tool, as we wanted the participants to learn the entire programming system. To achieve this goal, we designed a queue sorting exercise, similar to those found in a typical data structure course. We proposed a task where participants would rearrange an existing queue of values into two new queues. In the beginning, the queue would be randomly filled with two types of values (e.g., zeros and ones). By the end of the experiment, one queue should contain only zeros and the other queue should only contain ones.

To represent the two queues and the binary values in a physical environment, we used coffee cans containing two distinct flavors of coffee and two can dispensers (See Figure 6.2). The cans represented the binary values and were randomly placed in the dispensers representing the queues. We asked participants to implement an application that allows a robot to rearrange the cans among the two dispensers. To avoid using a computer vision system, which would complicate the programming task, we allowed users to interactively indicate to the robot which type of coffee it currently held using a prompt. By the end of the experiment, the cans containing one flavor should be in the first dispenser and the remaining ones in the second dispenser.

Participants were treated as the developers responsible for implementing this software application, and a list of nine software requirements was presented to them before the beginning of their session (See Table 6.1). To complete the experimental task, participants should follow all nine requirements. They were given a printed copy of the software requirements, along with guidelines on how to operate the robot and the block-based programming language proposed for this experiment. A proctor also read for them the guidelines and the requirements out loud before the beginning of the experiment as a form of training.

6.1.3 Wizard Easy Programming Tool

Participants were restricted to a single programming environment to implement a solution for the experimental task. The environment of choice was the Wizard Easy Programming Tool (See Figure 3.1). This tool consists of a block-based programming environment for collaborative robots made by the ABB robot manufacturer. It supports most of the manufacturer's collaborative robots, including the one used in this experiment, the ABB GoFa. The programming environment translates complex robot commands into colorful blocks that are easy to use and understand, and it is one of the few programming options available for end-users in robotics. In this experiment, participants programmed in the environment using the robot's teaching pendant. They were responsible for implementing every software requirement using only the blocks available.

The Wizard Easy Programming Tool provided eight categories of blocks: *message*, *move*, “*stop and wait*”, *procedures*, *loops*, *signals*, *logic*, and *variable*. Under the *message* category,

| Software Requirements |
|--|
| R1. Your software should only use the features available in the block-based language. |
| R2. Your software should only use the robot positions available in the block-based environment. You should not create new robot positions for your solution, but you can instantiate as many other variables as you want (e.g., numbers, strings). |
| R3. Customers should not interact directly with your code, including but not limited to moving your blocks or editing your variables. |
| R4. Your software must be written within a single file, provided for you beforehand by the experiment. You should not create or load other files in the programming environment or even rename the one opened for you. |
| R5. Your software must start printing a welcome message to clients on the teaching pendant screen. |
| R6. Your software must receive the customer's input to decide the next actions of the robot, and the customer's input must be received as touchscreen interactions on the teaching pendant screen. |
| R7. In terms of actions, the robot must be able to move the coffee cans from one organizer to the other and to move the first can of each organizer to the last position of the same organizer, based on the customer's input, while your application is running. |
| R8. Customers should be informed about which one of the actions the robot is performing. |
| R9. Your software must allow customers to decide when to stop the application. This decision must be received as a touchscreen interaction on the teaching pendant screen. |

Table 6.1: List of software requirements given for the experimental task.

four blocks were available to receive user commands through touchscreen interactions and to print messages on the teaching pendant. To capture user commands, developers could use a "ask a question" block and receive the user input in numerical format (a numeric answer is inserted by the user in a text field and saved as a variable) or categorical format (a list of categorical options is selected by the user through button interactions on the screen and saved as a numeric variable).

Two blocks under the *move* category were available for developers to move the robot around the workspace, one using linear and other joint-based movements. Three blocks under the "stop and wait" category were available to make the program execution stop or make the robot wait for a defined period. In the *procedures* category, developers could encapsulate a set of blocks by creating custom procedures (i.e., functions) in the language. This category displayed all the custom procedures created by the developer, plus a call block to run them during execution time. Each participant started the experiment with three custom procedures to manipulate the robotic gripper: open gripper, close gripper, and restart gripper. Restart was used in cases where the gripper stopped working after a problem in the execution (e.g., after a collision with the environment). The participants

were allowed to create as many procedures as they judged necessary.

The *loops* category contained two loop blocks, one representing a *for* and another a *while* loop. The *signals* category was available to manipulate input and output signals in the robot's controller. Participants in this experiment were instructed to ignore this category, as no input/output manipulation was required for the experimental task. The *logic* category contained four blocks for developers to control the execution flow of their programming, including conditional blocks and operators. Finally, the *variables* category provided a set of blocks for users to set, update, and use variables. Three types of variables were available for implementation: numbers, booleans, and strings.

Figure 6.1 presents an example of an expected solution for the experimental task, covering all the nine software requirements. The program starts displaying a welcome message on the teaching pendant screen (R5). The robot is moved to a starting position named *Home*. A boolean variable and a *while* block keeps the program running in a loop. A question is displayed on the screen at the beginning of every loop iteration to capture the user input (R6). Five input options are available for the user to choose from: (a) Move the can from the first dispenser to the second dispenser, (b) move the can from the second dispenser to the first dispenser, (c) move the can from the first dispenser to the last position of the same dispenser, (d) move the can from the second dispenser to last position of the same dispenser, (e) stop the loop and close the software application (R7).

The user decision is saved in a variable named *Command*, and if statements are used to control the execution flow. A message is printed on the teaching pendant screen for every possible decision (R8). For each movement decision, custom procedures are used to encapsulate move blocks and the opening and closing of the gripper. A final condition is used to close the application (R9). The program is executed every time a user presses a start button on the teaching pendant. This solution and the experimental task requirements cover most of the block categories available in the Wizard Easy Programming tool. The only categories not used to complete the task are the “*stop and wait*” and *signals*, although stop blocks could also be used to make the program flow smoother. This solution also indirectly covers the first four requirements not mentioned above.

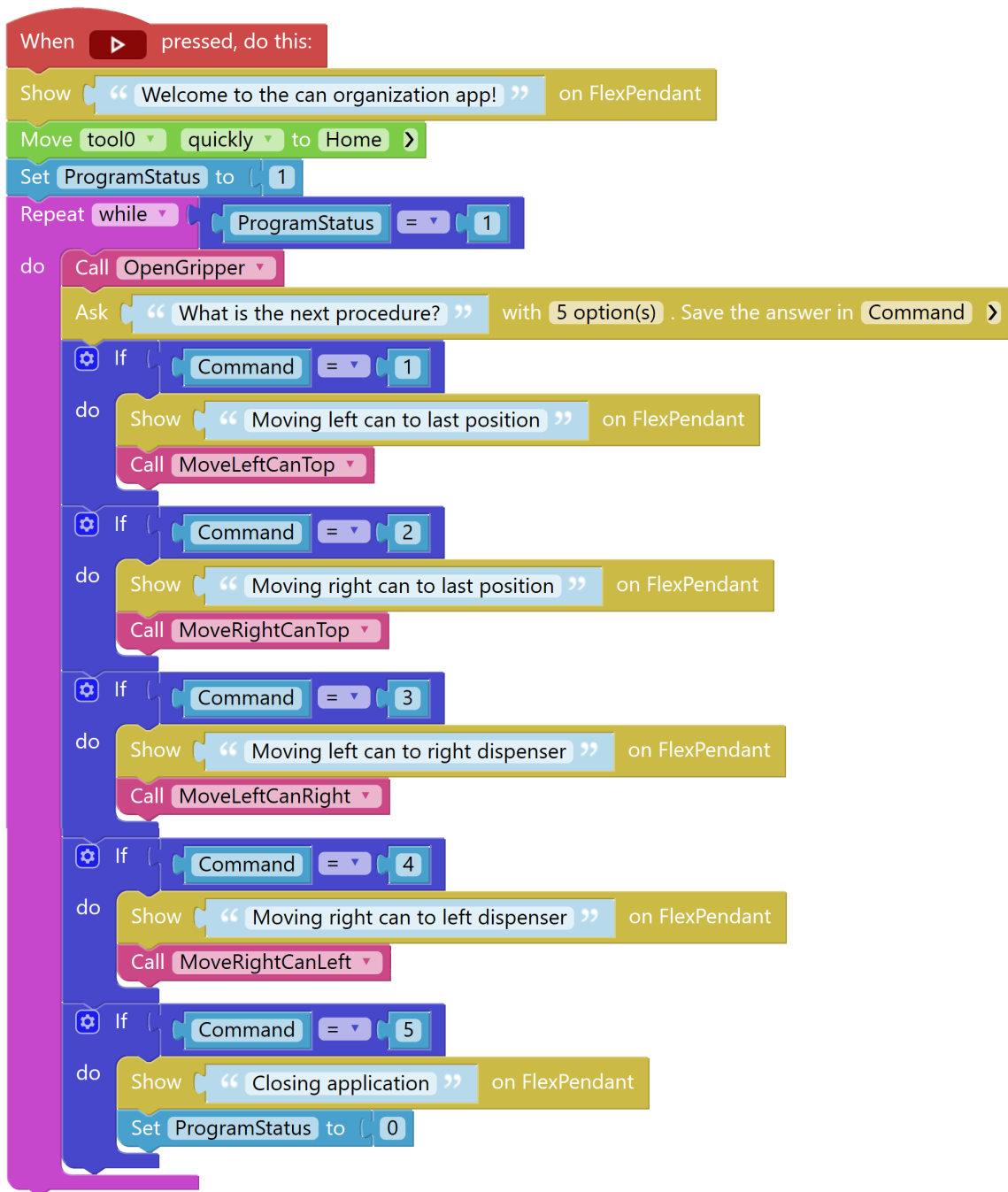


Figure 6.1: Potential task solution. The solution is composed of multiple block categories and complies with the nine software requirements proposed for the experimental task.

6.1.4 Workspace

Each participant visited our laboratory at a time to participate in the experiment. Figure 6.2 displays the workspace presented to participants during their session. On the right side was the robot operating area. The robot was fixed on a table behind two dispensers filled with cans of coffee. Predefined in the robot operating system, there were robot positions for the robot to reach the upper and bottom parts of the two dispensers. A proctor randomly

organized the cans and tested the robot before the beginning of every session. Participants were instructed not to interact with the robot workspace as they only needed to use the block-based environment to complete the task.

The teaching pendant and a desktop computer were placed on the left side of the experiment area. Participants accessed the block-based environment and programmed their solution through the pendant device, which also contained physical buttons to play and stop the program's execution. Before the beginning of every session, a proctor instructed the participants on using the teaching pendant. We also provided the participants with a desktop computer acting as a "help desk". Participants were advised that they could use the computer at any time to assist them in completing the task. Running on the computer, a website for this experiment was available for the participants. Participants were instructed not to use any other applications other than the website during the experiment.

6.1.5 Help Desk

The website presented participants with a workflow divided into three pages. On the first page, participants were asked to complete a form containing a multiple-choice question and a text field. The question asked participants what learning challenge they faced, and the text field asked them to describe their request in detail. The available choices for the question were based on six learning barriers of end-user programming systems discovered in a related study (Ko, B. A. MYERS, *et al.*, 2004), including:

1. "I don't know what I want the robot to do..."
2. "I think I know what I want the robot to do, but I don't know what to use..."
3. "I think I know what things to use, but I don't know how to make them work together..."
4. "I think I know what to use, but I don't know how to use it..."
5. "I thought I knew how to use this, but it didn't do what I expected..."
6. "I think I know why it didn't do what I expected, but I don't know how to check..."

An "Other" option was also available where participants could define a challenge not described by the six learning barriers available. This first page captured participants' learning challenges and their descriptions. Once they filled out the form, participants were redirected to the *resources* page. In this second stage, multiple learning materials were listed for use in three categories: video, text, and chat. Our goal was to provide participants with potential resources to help them overcome their learning barriers and evaluate how different resources impact their understanding of the programming task. Table 6.2 shows the complete list of learning resources available at the help desk.

For the video category, we provided participants with all the tutorials posted by ABB on YouTube about the Wizard Easy Programming Tool. Only videos from their official channel were listed. We also recorded and included a short training video about the block-based environment, explaining what blocks were available in the language. The script for this video was written using the tool's official application manual as a reference. For the text category, we provided participants with the video script we recorded and the application manual extracted from the ABB website. In the block-based environment, users also had

| Resource | Type | Source |
|--|-------|------------------------------------|
| <i>Quickstart Guide: Wizard Easy Programming</i> | Video | (FRONCHETTI, 2024a) |
| <i>Using Wizard to create a PCB assembly application in minutes</i> | Video | (A. ROBOTICS, 2020a) |
| <i>ABB Wizard easy programming for single arm YuMi</i> | Video | (A. ROBOTICS, 2020b) |
| <i>How to program collaborative robot GoFa with Wizard Easy Programming</i> | Video | (A. ROBOTICS, 2021a) |
| <i>Step-by-step guide on pick and place application with Wizard Easy Programming</i> | Video | (A. ROBOTICS, 2021b) |
| <i>Wizard Easy Programming: Advanced Application Overview</i> | Video | (A. ROBOTICS, 2023a) |
| <i>Wizard Easy Programming: For everyone and all new robots</i> | Video | (A. ROBOTICS, 2023b) |
| <i>Quickstart Guide: Wizard Easy Programming</i> | Text | (FRONCHETTI, 2024b) |
| <i>Wizard Easy Programming: Application manual</i> | Text | (A. ROBOTICS, 2024) |
| <i>Wizard Easy Programming: Wiki</i> | Text | Only available on teaching pendant |
| <i>Chatroom with an expert in Wizard Easy Programming</i> | Chat | Only available on the help desk |

Table 6.2: List of learning resources available on the help desk.

access to a Wiki page explaining each block’s functionality, which we also included as part of the text category.

Finally, we created a chat room for the chat category using a customized version of ChatGPT 4 as the respondent (WU *et al.*, 2023). Using prompt engineering (EKIN, 2023), we trained the chatbot to answer questions about the Wizard Easy Programming Tool. The chatbot was trained using the same input of the video script we recorded about the tool, creating an even comparison for all three categories, as all of them shared this same resource (i.e., read, watch, and chat). We also provided the chatbot with basic information about what type of robot was used and what programming language was available. To restrict the chatbot from giving answers unrelated to robot programming, we trained it with a list of rules, such as “never tell participants to change robot settings” and “never tell participants to teach robot positions manually”. We used the Python programming language

to build the ChatGPT⁴ integration and NiceGUI⁵ to build the UI of our website.



Figure 6.2: *Experiment workspace. On the left, the desktop computer and the teaching pendant device are placed on a tool cart. On the right, the robot is fixed on a table along with the two dispensers randomly filled with cans containing two distinct flavors. At the bottom of the table, it is also possible to see the robot controller connected to the teaching pendant by a cable and a preview of the robot placing the can on the dispenser.*

There were no hints about the solution of the experiment in any resources provided, just instructions for the block-based environment. Once a participant completed their use of the *help desk* for a specific issue, they could close the request created by clicking on a close button on the *resources* page. This button redirected them to the third page, where a feedback questionnaire was presented with three multiple-choice questions. The first question asked them to rate their satisfaction with the assistance given for the request. The second asked participants to choose, when applicable, the category of learning resource that was the most useful to solve the request. Finally, the third asked if the learning barrier they selected in the first stage was appropriate for that request. If not, participants could assign another learning barrier to the request or create a new one.

Every input interaction in the help desk application was recorded for posterior analysis. This includes, but is not limited to, the responses to the feedback questionnaire, the learning barriers and descriptions assigned to each request, and the interactions with the available resources. We also recorded the date and time they started and ended their participation in our experiment.

⁴ <https://platform.openai.com/docs/api-reference/>

⁵ <https://nicegui.io/>

6.1.6 Post-Experiment Questionnaire

As the final step of our experiment, we invited participants to answer a post-experiment questionnaire. We used this questionnaire to get their overall feedback regarding the experiment, the block-based tool, the learning barriers and resources, and the robot programming task. The questionnaire was divided into four pages. The first page asked participants for demographic information, including their experience with robotics, block-based programming, and robot programming languages. No sensitive information was collected through the questionnaire. On the second page, we asked participants about their learning experience in computer and robot programming, when applicable. For each programming domain, we asked them two questions: what learning resources do they use the most (e.g., videos, textbooks, audiobooks), and what challenges do they typically face when consuming these learning materials (e.g., outdated information, information not concise, or clear)? On the third page, we asked participants about their overall experience with the help desk, including what category of learning materials were most useful to solve the programming task and their feedback regarding the materials available. On the last page, we allowed them to give feedback about the Wizard Easy Programming Tool and their experience in our study.

6.1.7 Data Collection and Analysis

This study's main data sources are the post-experiment questionnaire and the forms and interactions from the help desk application. We also saved the solution implemented by our participants. As most data is considered quantitative information, descriptive statistical methods were used to analyze them. We applied a qualitative description approach for the open-ended questions and forms, with two researchers analyzing the data and describing the participants' feedback in detail. We opted for a more immediate qualitative approach as the information collected throughout the experiment was insufficient for a more detailed analysis (e.g., open card sorting). The same researchers also analyzed the program solutions made by participants to check if they completed the task. Each participant was identified in our dataset by a unique identifier based on the date and hour they participated in our experiment. No sensitive information was recorded.

6.2 Results

In this section, we present the outcomes of our study. Our analysis focused on the existence of learning barriers in robot programming tasks and the interaction of end-users with different learning materials. We start introducing the demographic information of our participants, followed by an analysis of their performance in the experimental task. We follow the section presenting the learning challenges faced by participants while programming our experimental task and their interactions with learning materials in our help desk. At the end, we also highlight answers from our post-experiment questionnaire.

6.2.1 Demographics

Thirty-five students participated in our experiment, with 33 students being undergraduate (94%) and 2 master students (6%). Among the undergraduate ones, we found 21 seniors (60%), 9 juniors (26%), and 3 sophomores (9%). Figure 6.3 shows their computer, block-based, and robot programming experiences. Regarding computer programming, 4 participants reported more than five years of experience (11%), 16 participants between four to five years (46%), 14 participants between one to three years (40%), and 1 participant less than one year of programming experience (3%). For experience with block-based languages, only one participant reported more than five years of experience (3%), 5 participants between 1 and 3 years of experience (14%), 10 participants had less than one year of experience (29%), and 19 participants had no experience with block-based environments (54%).

Regarding experience with robot programming, 4 participants reported less than one year of experience (11%), and 31 participants stated no experience in the area (89%). In another question we also asked participants about their overall experience with robotics, whether it is part of industrial robotics or not. Twelve participants reported limited experience in robotics (34%), while 21 suggested no domain experience (60%). The demographic composition of our participants suggests they fit in our definition of end-users in robotics and the target population of our experimental task, as most participants are experienced in computer programming but have limited to no experience in robotics. We also hold a reasonable sample size compared to other studies in the field.

6.2.2 Participants Performance

Two authors of this study manually inspected the solutions made by participants using the nine software requirements (See Table 6.1). A requirement was considered complete if the participant satisfied its definition. For example, requirement five (R5, "*Your software must start printing a welcome message to clients on the teaching pendant screen.*") was considered complete if the participant used a message block to print a welcome message on the pendant screen. Figure 6.3b presents the number of participants who completed each software requirement, and Figure 6.3c the number of requirements completed per participant. All 35 participants completed the first four requirements (R1, R2, R3, and R4). In contrast, 31 participants (89%) wrote the welcome message on the teaching pendant screen (R5), 23 participants (66%) used input blocks to receive customers' directions over the next actions of the robot (R6), and 19 participants (54%) used blocks to move the cans between the organizers (R7).

Only 10 participants (29%) informed users about which one of the actions the robot was performing (R8), and 14 participants (40%) implemented a logic that allowed the user to stop the application (R9). Regarding the number of requirements completed per participant, we have that 3 participants (9%) completed only 4 out of 10 requirements, 8 participants (23%) completed 5 requirements, and 6 participants (17%) completed 6 requirements. On higher completeness, we found that 3 participants (9%) completed 7 requirements, 7 participants (20%) completed 8 requirements, and 8 participants (23%) completed all 9 software requirements. Using a secondary application on the help desk, the proctors computed the time taken per participant to complete their participation in the experiment. On average, participants spent one hour and five minutes working on the

experimental task, fifteen minutes below the expected time limit.

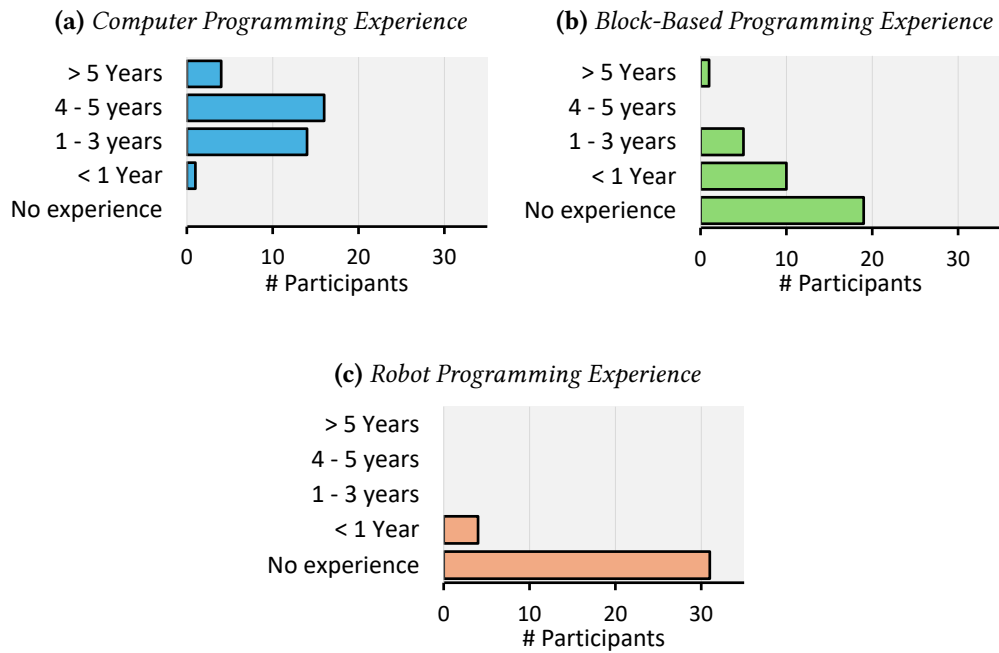


Figure 6.3: Demographic composition of participants: Computer programming experience, block-based programming experience, and robot programming experience.

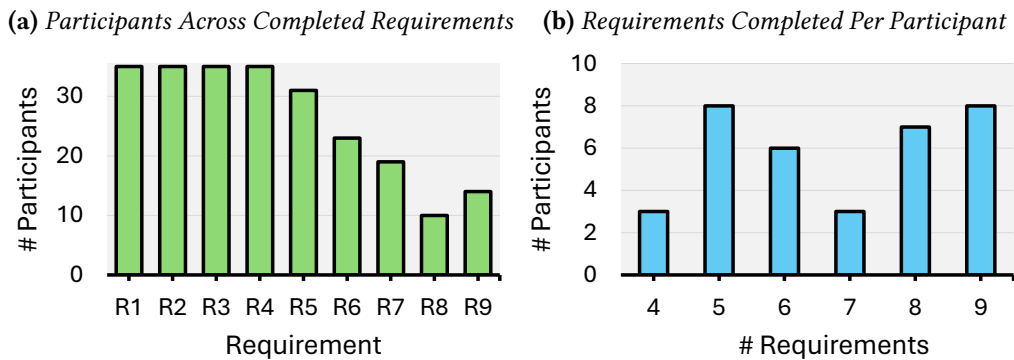


Figure 6.4: Distribution of participants across completed requirements and the requirements completed per participant. See Table 6.1 for requirements list.

6.2.3 Frequency of Learning Barriers

In total, thirty-six help requests were opened on the *help desk*, with an average of one request per participant (min: 0, max: 4, sd.: 1). Regarding the barrier assigned for each request, participants stated in three requests, "I don't know what I want the robot to do...". Seven requests were related to "I think I know what I want the robot to do, but I don't know what to use...". Six requests were associated with "I think I know what things to use, but I

don't know how to make them work together...". Seven requests were related to *"I think I know what to use, but I don't know how to use it..."*, and seven requests to *"I thought I knew how to use this, but it didn't do what I expected..."*. When accessing the help desk, none of our participants selected the learning barrier, *"I think I know why it didn't do what I expected, but I don't know how to check..."*. However, six other requests were assigned with the "Other" option, stating that the help request created was not associated with one of the six barriers available. None of these six requests provided substantial information to organize a new learning barrier. Table 6.3 summarizes the frequency of learning barriers in help requests.

| Learning Barrier | # Requests |
|---|------------|
| I thought I knew how to use this, but it didn't do what I expected... | 7 |
| I think I know what to use, but I don't know how to use it... | 7 |
| I think I know what I want the robot to do, but I don't know what to use... | 7 |
| I think I know what things to use, but I don't know how to make them work together... | 6 |
| I don't know what I want the robot to do... | 3 |
| I think I know why it didn't do what I expected, but I don't know how to check... | 0 |
| Other | 6 |

Table 6.3: *Appearance of learning barriers per request.*

Along with the barriers, participants also provided a description of the reason they requested help. Two authors of our study organized their descriptions into three categories according to the context of each request. The requests were related to programming, bugs, and context issues. The programming category covered requests related to difficulties with the programming environment. Twenty-two out of the thirty-six requests were found in this category. Examples of descriptions associated with this category included questions such as *"How do I save a response from a question to a variable so I can use it in an if statement?"*, *"How can I end the program using user input?"*, and *"How do I copy more than one block at the same time?"*. In another category, ten out of the thirty-six requests were related to participants complaining about bugs/problems in the workspace. Descriptions in this category include *"The robot is stuck in a position, and it is not moving no matter how many times I run..."* and *"I am running the code to go to the left dispenser and close the grip, but it keeps getting stuck..."*. The four descriptions left were grouped in a generic category, as they didn't belong to any of the categories above. The examples included questions related to the experiment, such as *"What is the role of the customer?"* and *"I'm confused about what you meant by what the client wants."*

6.2.4 Usage of Learning Resources

We also quantified the use of the learning resources at the help desk. Participants used the resources available sixty-eight times throughout the experiment, with the chatbot being used 33 times (49%), videos 24 times (35%), and textbooks 11 times (16%). Each participant used, on average, two learning materials per request. We counted a video or textbook used when the participant opened it in our web application. For the chatbot, we counted the number of requests where a new chatroom was opened. In the video category, the most watched videos were the "*Quickstart Guide: Wizard Easy Programming*" (FRONCHETTI, 2024a) with 9 views, followed by "*Using Wizard to create a PCB assembly application in minutes*" (A. ROBOTICS, 2020a) with 5 views. The videos "*How to program collaborative robot GoFa with Wizard Easy Programming - Tutorial for beginners*" (A. ROBOTICS, 2021a) and "*Wizard Easy Programming: Advanced Application Overview*" (A. ROBOTICS, 2023a) received both 3 views. The videos "*Wizard Easy Programming – For everyone and all new robots*" (A. ROBOTICS, 2023b) and "*Step-by-step guide on pick and place application with Wizard Easy Programming tool*" (A. ROBOTICS, 2021b) received 2 views.

The remaining video, "*ABB Wizard easy programming for single arm YuMi*" (A. ROBOTICS, 2020b) didn't receive any views. The lack of interest from participants in this video may be justified by the fact that this is the only video that uses another type of robot. In the text category, we only counted the views for the materials available on our web application. The most used textbook was the "*Application Manual of the Wizard Easy Programming Tool*" (A. ROBOTICS, 2024) with 7 views, followed by the "*Quickstart Guide for the Wizard Easy Programming Tool*" (FRONCHETTI, 2024b), with 4 views. In the chat category, participants exchanged 152 messages with ChatGPT, averaging 5 messages per chatroom/request. The majority of the messages sent by participants were simple questions about programming, such as "*Does the if statement work with strings?*", "*How do I receive user input?*", "*Is there an else if feature?*", and "*How to reset robot to original position?*". On average, participants spent 13 minutes talking to the chatbot (min: 1, max: 53, sd.: 15), 5 minutes watching videos (min: < 1, max: 41, sd.: 10), and 2 minutes reading textbooks (min: < 1, max: 8, sd.: 2). These results suggest the chatbot as the most engaging learning resource available on the help desk.

When asked about their satisfaction level with the resources available in a post-request form, 4 participants (11%) declared being very satisfied, 16 participants were satisfied (44%), 14 participants were neutral (39%), and 2 participants (6%) were dissatisfied with the learning materials at the help desk. Regarding the most useful category of information, participants selected the chatroom 20 times as their favorite category (56%), the textbooks 6 times (16%), and the videos 2 times (6%). For 8 times (22%), participants opted out from selecting one of the categories. In the same post-request form, we asked participants whether the learning barrier they assigned for the request was correct or should be replaced. Participants kept the same learning barrier for 29 requests (81%) and replaced 5 of them (14%). In 2 requests (5%), participants opted for not answering this question.

6.2.5 Outcomes from Questionnaire

On the post-experiment questionnaire, we repeated the question about what categories of learning materials were useful to solve the task. At this point, participants could choose

more than one category as useful. Their answer repeated similar trends from the post-request form, with 16 participants (46%) choosing the chatbot as the best learning resource, 6 participants choosing the videos (17%), and 3 participants choosing textbooks (9%). Fifteen participants (43%) stated that they did not use the help desk at any moment, and 3 participants stated that no categories were useful to them (9%). Throughout the questionnaire, we also asked participants what type of learning materials they use the most while studying computer programming and what challenges they face when using these resources. For those with experience in robotics, we repeated the same questions using robot programming as context. These questions were not a requirement to complete the questionnaire. For both questions, participants could select multiple options from a predefined list of options and add new ones if necessary. The predefined options were based on two related studies (AL-FRAIHAT *et al.*, 2020; ABDULRAHAMAN *et al.*, 2020).

Table 6.4 presents the most used resource types according to our participants. The table is divided into two columns, one for the first question asking all participants about computer programming and the second for those with experience in robotics. Regarding the most used resource types, while studying computer programming, videos were selected by 30 out of 35 participants (86%), followed by learning from online communities and chatbots, both selected by 27 participants (77%). Twenty-six participants also selected lecture materials (74%) as a common resource used while studying computer programming. In the last positions, we found textbooks selected by 14 participants (40%), technical documentation selected by fifteen participants (43%), and audiobooks selected by just one participant (3%).

| Resource Type | Computer Programming (N = 35) | Robot Programming (N = 11) |
|-------------------------|----------------------------------|-------------------------------|
| Videos | 30 (86%) | 10 (91%) |
| Textbooks | 14 (40%) | 2 (18%) |
| Audiobooks | 1 (3%) | 1 (9%) |
| Online Communities | 27 (77%) | 6 (55%) |
| Lecture Materials | 26 (74%) | 5 (45%) |
| Chatbots | 27 (77%) | 9 (82%) |
| Technical Documentation | 15 (43%) | 5 (45%) |

Table 6.4: Participants most used learning resources according to their format. Each column represents the number of participants who chose a format for a specific programming domain. N is the number of individuals who selected at least one format.

Eleven participants selected the resource types they used the most to study robot programming. Similar trends from computer programming were found with students in robotics, with 10 of them (91%) selecting video as their favorite resource type, followed by chatbots (82%) and online communities (55%). The least used types also presented similar numbers, starting with lecture materials and technical documentation (45%), followed by textbooks (18%) and audiobooks (9%). In Table 6.5, we present participants' most frequent challenges while consuming learning materials in the computer and robot programming domains. For the computer programming domain, 20 out of 35 participants (57%) complained about the materials not providing sufficient information, not being concise and

clear, and not being organized into logical and understandable components. Fourteen participants (40%) also complained about the information and resources not being readily useable.

Nine participants (26%) reported problems with resources not being up to date, and eight participants (23%) stated that resources are not always accessible. One participant added a challenge to the list, stating, "*the materials are so scattered that it takes time to gather them for your specific case.*" The eleven participants who selected their favorite resource types to study robot programming also reported the challenges they usually face with learning resources in this domain. Five out of six challenges listed for participants were selected by six participants (17%). The only challenge that did not receive the same selection was the one about information not being up to date, selected by five participants (14%). Participants did not include custom challenges in this question.

| Challenges | Computer Programming (N = 35) | Robot Programming (N = 11) |
|--|----------------------------------|-------------------------------|
| The materials do not provide the sufficient or required information | 20 (57%) | 6 (17%) |
| The information and resources are not always accessible | 8 (23%) | 6 (17%) |
| The information and resources are not in a form that is readily useable | 14 (40%) | 6 (17%) |
| The information and resources are not concise and clear | 20 (57%) | 6 (17%) |
| The information and resources are not organized into logical and understandable components | 20 (57%) | 6 (17%) |
| The information and resources provided are not up to date | 9 (26%) | 5 (14%) |

Table 6.5: Participants most frequent challenges while using learning resources. Each column represents the number of participants in a specific programming domain. N is the number of individuals who selected at least one challenge in a domain.

6.3 Discussion

In this section, we discuss some of the outcomes of this chapter.

6.3.1 Chatbots and the Next Generation of Software Developers

With the advancements of the Internet, a multitude of learning technologies is now available online for those interested in learning computer programming. Not only are the resources made by programming language maintainers and manufacturers available, but a universe of professionals is also engaged in teaching newcomers the path to software

development. In recent years, chatbots sustained by large language models have attracted the attention of those attempting to learn something new, and results have shown that they can also support learners in software development. If one day, all a programmer had was the printed manual of a programming language, now one-on-one support is also available from a language model.

This chapter shows that incoming professionals in software development are opting for new learning representations. Twenty-seven out of thirty-five undergraduate students in Computer Science who participated in our experiment are using chatbots to study computer programming. Videos and online communities like Stack Overflow are also widely used by students as more recent forms of learning. On the other hand, less than 50% of our participants mentioned the use of more traditional formats, such as textbooks and technical documentation. But does the same hold for novices in robot programming? What resources do they access when working on a programming task? Our experimental task reinforces the same trends in the robotics domain. Our chatbot was voted the most useful learning resource at the help desk, receiving 33 votes. The videos and textbooks available received 24 and 11 votes, respectively. If companies like Universal Robots are now investing in e-learning academies filled with video tutorials (See Figure 6.5), maybe training a chatbot to support their prospective programmers should also be considered.



Figure 6.5: *Universal Robots Academy^a: e-learning platform for novice robot programmers.*

^a <https://academy.universal-robots.com/>

However, it is important to emphasize that chatbots' work in traditional programming environments may not work in robotics, as the answers generated by the language model will depend on context information such as the robot type, brand, and infrastructure. Defining these rules is necessary to avoid common challenges in participants' understanding of robot programming. As shown in Table 6.5, although the digital revolution in education is bringing new ways of learning computer programming, problems already seen before, such as lack of information clarity, are still present in students' perceptions nowadays. In future studies, we expect to explore strategies to support robot companies in releasing valuable learning resources to newcomers in robotics.

6.3.2 End-user robot programming: *Are We There Yet?*

This chapter also explores participants' interaction with a commercial block-based programming environment designed for end-users in robotics. The tool of choice is one of a few programming options for end-users working on collaborative robots. Created by the ABB robot manufacturer, the Wizard Easy Programming Tool promises to empower first-time robot users to "*program collaborative and industrial robots easily, quickly, and efficiently in a wide range of applications*" (See Figure 6.6). In our evaluation, we asked participants with computer programming experience to try a collaborative robot of their brand for the first time. Only four participants had minimal robot programming experience in our sample. Our results suggest that, although most users could implement code using ABB's block-based tool, only eight out of thirty-five participants (23%) could complete the pick-and-place task assigned in our experiment. One may argue that these results tell more about the experimental task than the language itself, but the learning challenges identified in our help desk also tell the opposite. As shown in Table 6.5, seven times, participants who accessed the help desk had difficulties trying to comprehend what to use in the block-based environment. Participants also had problems translating instructions to blocks and had difficulties dealing with mistakes in the block-based environment. These results suggest that more in-depth studies are necessary on block-based programming environments for robotics.



Teach your robot within 10 minutes without prior experience

Wizard Easy Programming is a graphical programming tool that empowers users – be it first-time robot users or professionals – to program collaborative and industrial robots easily, quickly and efficiently in a wide range of applications. Add Wizard Easy Programming blocks by dragging and dropping them, or by pushing a button. Then press play to run your program. It's that simple.

Wizard Easy Programming comes with a set of standard pre-installed blocks, from robot movements, to messages, and signal instructions. For situations where highly specialized blocks are required, experts can easily create custom blocks with the Skill Creator.

Figure 6.6: Advertisement for the Wizard Easy Programming Tool at ABB website^a.

^a <https://new.abb.com/products/robotics/software-and-digital/application-software/wizard>

Although it might be challenging for end-users to program robot solutions, participants' feedback about the block-based language they tried was mostly positive. One participant said: "*I think block-based programming can be easy for people to learn because a lot of technical theory and other stuff is abstracted. It helps people learn programming at a very basic level, which is useful.*" We believe that combining end-user programming environments and suitable learning resources may be the right direction to make robot programming an easier task. In future work, we plan to expand our understanding of this combination

to other scenarios, including their applicability in offline programming environments, non-collaborative robots, and applications other than pick-and-place tasks.

6.4 Limitations

In this section, we highlight some of the limitations of this chapter:

Experimental task and choice of programming environment. In this work, we explore end-user learning barriers and the use of learning resources in robot programming tasks. We use a commercial block-based environment and a custom robot programming task for evaluation. To reduce the complexity of the task given to participants, we asked them to implement a simple pick-and-place procedure using the block-based environment. Pick-and-place tasks are among the most common types of robot programming procedures and are easy to understand due to their simplicity. We also provided them with experiment guidelines and a list of software requirements. We comprehend that choosing a block-based tool rather than other programming designs for end-users may impact our outcomes, including but not limited to the conclusions about learning barriers and participant feedback. We also understand that the design proposed for our experimental task impacts our results and that other tasks could lead to distinct conclusions.

Learning barriers and resources. As our goal was to evaluate learning barriers and learning resources in robot programming environments, the choice of barriers listed on the help desk and the resources provided may also influence participants' perceptions. We tried to mitigate this issue by using a predefined list of barriers from a related study working in a similar context and by providing only the learning resources made by the robot manufacturer. We also allowed participants to include new learning barriers at the help desk and opened space for them to give feedback about the resources in the post-experiment questionnaire.

Training and time constraints. One may argue that the amount of time and training given to participants is too small. We understand that these decisions may impact our results, but, at the same time, we must emphasize that they were made based on our prior experience with the Duplo experiment and a pilot study with four participants. Each participant received a printed copy of the experiment guidelines and the software requirements they should follow. A proctor read these copies to the participant before starting the experiment and was allowed to answer any questions regarding these documents. We did not provide any hints on solving the task, as we expected participants to access the help desk when necessary. The expected time limit was calculated based on the time participants took in the pilot study and the complexity of the task. The time definition was also made in agreement with an institutional review board.

Number and origin of participants. Due to budget and time constraints, participants in this experiment were students recruited from two software engineering classes. Thirty-five students participated in our experimental task, all with previous experience in computer programming and a few with very limited experience in robotics. To increase participation, we offered students with extra credits in class. We understand that the recruitment process and the sample size may also impact our outcomes. College students may not totally represent end-users in robotics, and the conclusions made throughout

our work may be limited by the feedback given by a small sample of participants. To validate our conclusions, we comprehend that future work will be necessary with more participants and a more diverse experiment design.

6.5 Conclusion

Learning resources are important in helping beginners understand a subject, especially when they face challenges that hinder the learning process. In this chapter, we explore how robotics novices learn to use a block-based tool to solve a simple robot programming task. We provided them with learning resources to master the tool and asked them to report what challenges they faced throughout the process. We learned from their interactions and feedback that although end-user environments are a promising technology in robotics, users without experience in the field still face difficulties in solving simple programming tasks. As robots are present in the physical domain, translating the programming logic to a movable arm can be more challenging than traditional programming exercises. Our results also highlight that the new generation of robot developers is interested in less traditional learning resources, opting for videos and conversations with chatbots rather than technical textbooks and manuals. For future work, we plan to explore the use of chatbots in collaborative robot programming tasks, and the use of other learning resources in the process.

Chapter 7

Conclusion

End-user robot programming is a domain with many open questions to explore. In this thesis, I expand the domain by investigating: the use of block-based languages in two-armed robots (Chapter 4), the application of mixed-reality interfaces to the manual control of articulated robots (Chapter 5), and how learning resources can contribute to collaborative robot programming tasks (Chapter 6). In the following sections, I present a collection of complementary studies executed throughout my doctorate to complement this work. I also introduce my published papers and a brief description of future work.

7.1 Complementary Studies

The studies presented in this thesis are part of my main investigation into end-user robot programming. However, I have also participated in other related studies in software engineering and human-robot interaction, and this section summarizes three of them. Other studies published throughout the Ph.D. can also be found in Section 7.2.

7.1.1 Artistic Robot Programming in Mixed Reality

Paper:

FRONCHETTI, POPIELA, *et al.*, 2024

Articulated robots are attracting the attention of different areas worldwide, including the artistic domain. Due to their precise, tireless, and efficient nature, robots are now being deployed in different forms of creative expression, such as sculpting (MA *et al.*, 2021), choreography (H. PENG *et al.*, 2018), and cinematography (GSCHWINDT *et al.*, 2019). While there is a growing interest among artists in robotics, programming such machines is a challenge for most professionals in the field, as robots require extensive coding experience and are primarily designed for industrial applications and environments.

To enable artists to incorporate robots in their projects, we created a robot programming application using an intuitive spatial computing environment designed for Microsoft HoloLens 2. This solution could be considered an extension of our work in robot manipulation using mixed reality and our first practical application for a non-engineering domain.

This application synchronizes the robot’s movements with a mixed-reality hologram via network communication. Using natural hand gestures, users can manipulate, animate, and record the hologram movements, similar to 3D animation software. The hologram animation is then translated to robot coordinates, making the robot move in the path taken by the hologram. Our solution allows artists to translate their creative ideas and movements into industrial and collaborative robots. It also makes human-robot interaction safer, as robots can accurately and effectively operate from a distance.

The solution has two modes of hologram interaction: animation and creation mode. In animation mode, the hologram’s movements are recorded over a period of time (See Figure 7.1a). The user can then save and replay these movements in the robot through a control interface in the mixed-reality application. In creation mode, the movements are dictated by 3D vertices inserted by the user in the mixed-reality workspace. A sequence of vertices represents the path that the robot will replicate. To evaluate our technological intervention, we invited a group of artists from Virginia Commonwealth University to try it in a preliminary study. The artists recorded a light trail through a timelapse using LED lights attached to an industrial robot (See Figure 7.1b). Although their feedback was mostly positive, we expect to evaluate our intervention in detail in a future study.

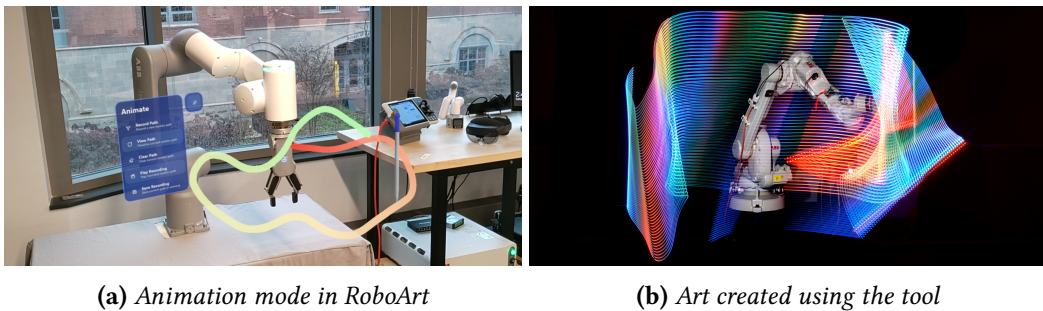


Figure 7.1: Example of path programming and artistic solution created in RobotArt.

7.1.2 Guided Decomposition in Larger Block-Based Programs

Papers:

RITSCHER, FRONCHETTI, *et al.*, 2022a, RITSCHER, FRONCHETTI, *et al.*, 2022b, RITSCHER, FRONCHETTI, *et al.*, 2024

Block-based environments may be considered an inviable solution for more complex problems involving collaborative robots. Organizing and debugging large chunks of blocks can be as challenging as difficulties found in non-beginner-friendly solutions. In this complementary study, we propose a function-centric block-based environment that enables end-users to write larger programs by decomposing their solutions into functions (See Figure 7.2). In our approach, the block-based environment is divided into two canvases aligned side by side. On the left canvas, users use function blocks to define tasks that the robot will accomplish. By clicking on a function block, users can program a solution for the respective task on the right canvas using a set of robot commands, also represented as blocks. In the current state of our implementation, a simple simulation environment on the right side of the block-based environment is available for study.

This block-based design aims to provide a scaffolded environment where end-users can implement functions without struggling with more advanced concepts (e.g., function overloading). To evaluate how this environment would hold in a practical scenario, an online user study was conducted with 92 participants from Amazon Turk. They were randomly assigned to two groups, one group testing this new approach and another testing a traditional block-based environment with support for functions. The findings suggest that the function-centric environment encourages end-users to organize their code into functions, enabling the implementation of larger block-based solutions. In a future study, we aim to expand our analysis to a real scenario, where we will ask users to solve tasks in different workstations using a mobile industrial robot.

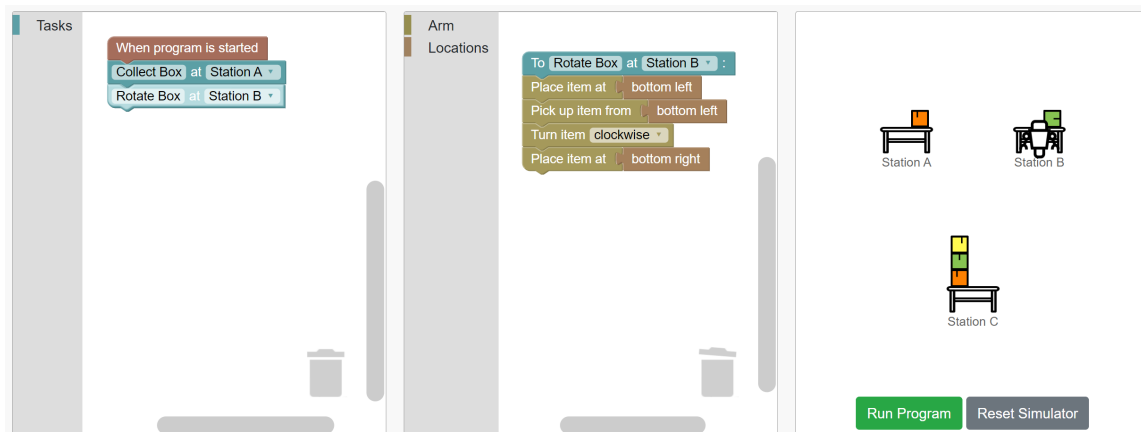


Figure 7.2: Example of a function-centric solution in the proposed environment. The current program state shows the inner blocks of the "Rotate Box" function executed in Station B in the simulation (right-most canvas).

7.1.3 Language Impact on Programmable Logic Controllers

Paper:

FRONCHETTI, RITSCHEL, HOLMES, *et al.*, 2022

Robots are not the only technology participating in the fourth industrial revolution. Programmable Logic Controllers (PLCs), a type of computer that controls industrial equipment, are also growing in popularity. As in robotics, most programming environments available for PLCs are restricted to outdated languages from the last century, restricting their adoption by a broader audience. To investigate the challenges of PLC programming, this study proposes an online survey with 175 technical employees from an industrial manufacturer. Participants were invited to respond to a set of questions related to Ladder Logic, the most popular programming language available in PLCs.

The online survey was divided into four stages. The first stage asked participants about their demographic information. Nearly half of the participants claimed to have over five years of programming experience (100 out of 175), with most of the respondents being engineers. Aiming to support participants with lower levels of experience, the second stage provided a video tutorial on PLCs and Ladder Logic. The third and most crucial stage asked participants to solve ten simple automation tasks using Ladder Logic (e.g., turn on a

light using a button). The questions were divided into interpretative and writing questions. In the fourth and final stage, participants were invited to respond to a questionnaire containing a system usability scale and an open-ended question regarding their opinion about the respective programming language.

The study pointed out significant problems in participants understanding of basic concepts in Ladder Logic. Approximately 70% of the participants, including those with more than five years of experience in programming, failed at least one of the tasks. In terms of usability, the 175 participants rated Ladder Logic as a marginal system (50th percentile) and stated that certain characteristics of the language make it harder to read and understand. This study expands the discussion on the difficulties most programmers face in industrial programming settings. In future work, we plan to bring up solutions that could also benefit developers working on PLCs.

7.2 Papers Published

The papers published during the execution of my thesis are listed in this section:

- **Software Fairness Debt**
(2024) ACM TOSEM 2030 Software Engineering
- **Blocks? Graphs? Why Not Both? Designing and Evaluating a Hybrid Programming Environment for End-users**
(2024) IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings
- **RoboART: Artistic Robot Programming in Mixed Reality**
(2024) IEEE Conference on Virtual Reality and 3D User Interfaces: Abstracts and Workshops (VRW)
- **Block-based Programming for Two-Armed Robots: A Comparative Study**
(2024) IEEE/ACM 46th International Conference on Software Engineering
- **Ready Worker One? High-Res VR for the Home Office**
(2023) ACM 29th Symposium on Virtual Reality Software and Technology
- **Enabling end-users to implement larger block-based programs**
(2022) ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings
- **Project-sized scaffolding for software engineering courses**
(2022) First International Workshop on Designing and Running Project-Based Courses in Software Engineering Education
- **Language impact on productivity for industrial end users: A case study from Programmable Logic Controllers**
(2022) Elsevier Journal of Computer Languages (Volume 69)
- **Can Guided Decomposition Help End-Users Write Larger Block-Based Programs? A Mobile Robot Experiment**
(2022) ACM SIGPLAN 13th Conference on Systems, Programming, Languages, and Applications: Software for Humanity

7.3 Future Work

Although contributions have been made in this thesis, we have many open questions for future studies. In future work, we plan to extend the applicability of robot control in mixed reality to two-armed robots. As in block-based programming, controlling two arms in synchrony may bring new challenges to our mixed reality solution, and an in-depth investigation is necessary. We also want to extend robot control in mixed reality to other domains, including but not limited to mobile robots and drones. Considering the increasing popularity of large language models, we also plan a more detailed investigation of their applicability in robot programming tasks, extending our current work to more complex features such as code generation and analysis by chatbots. Our plans also explore their ability to support end-users in robot programming tasks. Finally, we expect that our conglomerate of future studies will also contribute to generating a taxonomy of challenges end-users face in robot programming tasks, providing researchers and industry with a better understanding of their difficulties for future work.

References

- [AALTONEN and SALMI 2019] Iina AALTONEN and Timo SALMI. “Experiences and expectations of collaborative robots in industry and academia: barriers and development needs”. In: *Procedia Manufacturing* 38 (2019), pp. 1151–1158 (cit. on p. 1).
- [ABB 2023a] ABB. *ABB AR Viewer*. <https://new.abb.com/products/robotics/robotstudio/ar-viewer-app>. [Accessed: November-2023]. 2023 (cit. on p. 18).
- [ABB 2023b] ABB. *RobotStudio*. <https://new.abb.com/products/robotics/robotstudio/robotstudio-desktop>. [Accessed: November-2023]. 2023 (cit. on p. 17).
- [ABB LTD 2015] ABB LTD. “RobotStudio Online YuMi”. In: URL: <https://apps.microsoft.com/store/detail/9NBLGGH2SQFM> (2015) (cit. on pp. 22, 23).
- [ABB LTD 2020] ABB LTD. *ABB Industrial Robots Get Wizard Easy Programming Software*. Dec. 2020. URL: <https://new.abb.com/news/detail/72178/abb-industrial-%20robots-get-wizard-easy-programming-software> (cit. on p. 3).
- [ABB LTD. 2023a] ABB LTD. “Robotstudio suite”. In: URL: <https://new.abb.com/products/robotics/robotstudio> (2023) (cit. on p. 23).
- [ABB LTD. 2023b] ABB LTD. “Yumi - irb 14000 collaborative robot”. In: URL: <https://new.abb.com/products/robotics/robots/collaborative-robots/yumi/irb-14000-yumi> (2023) (cit. on p. 25).
- [ABB ROBOTICS 2021] Jakob Hörbst ABB ROBOTICS. *GoFa AR Control by HoloLens 2*. <https://www.youtube.com/watch?v=3Qv-cur4qxA>. [Accessed: November-2023]. 2021 (cit. on p. 19).
- [ABDULRAHAMAN *et al.* 2020] MD ABDULRAHAMAN *et al.* “Multimedia tools in the teaching and learning processes: a systematic review”. In: *Heliyon* 6.11 (2020) (cit. on p. 77).
- [AJAYKUMAR and HUANG 2020] Gopika AJAYKUMAR and Chien-Ming 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 (cit. on pp. 3, 36).

- [AJAYKUMAR, STEELE, *et al.* 2021] Gopika AJAYKUMAR, Maureen STEELE, and Chien-Ming HUANG. “A survey on end-user robot programming”. In: *ACM Computing Surveys (CSUR)* 54.8 (2021), pp. 1–36 (cit. on pp. 2, 3, 7, 14, 17, 19).
- [AJOUDANI *et al.* 2018] Arash AJOUDANI *et al.* “Progress and prospects of the human–robot collaboration”. In: *Autonomous Robots* 42 (2018), pp. 957–975 (cit. on p. 17).
- [ALEXANDROVA *et al.* 2015] Sonya ALEXANDROVA, Zachary TATLOCK, and Maya CAKMAK. “Roboflow: a flow-based visual programming language for mobile manipulation tasks”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 5537–5544 (cit. on p. 16).
- [ARYANIA *et al.* 2012] Azin ARYANIA, Balazs DANIEL, Trygve THOMESSEN, and Gabor SZIEBIG. “New trends in industrial robot controller user interfaces”. In: *2012 IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom)*. IEEE. 2012, pp. 365–369 (cit. on p. 44).
- [BANGOR, P. KORTUM, *et al.* 2009] Aaron BANGOR, Philip KORTUM, and James MILLER. “Determining what individual SUS scores mean: adding an adjective rating scale”. In: *Journal of usability studies* 4.3 (2009), pp. 114–123 (cit. on p. 49).
- [BANGOR, P. T. KORTUM, *et al.* 2008] Aaron BANGOR, Philip T. KORTUM, and James T. MILLER. “An empirical evaluation of the system usability scale”. In: *International Journal of Human–Computer Interaction* 24.6 (2008), pp. 574–594. DOI: 10.1080/10447310802205776. eprint: <https://doi.org/10.1080/10447310802205776>. URL: <https://doi.org/10.1080/10447310802205776> (cit. on p. 47).
- [BARRICELLI *et al.* 2019] Barbara Rita BARRICELLI, Fabio CASSANO, Daniela FOGLI, and Antonio PICCINNO. “End-user development, end-user programming and end-user software engineering: a systematic mapping study”. In: *Journal of Systems and Software* 149 (2019), pp. 101–137 (cit. on pp. 2, 14).
- [BAU *et al.* 2017] David BAU, Jeff GRAY, Caitlin KELLEHER, Josh SHELDON, and Franklyn TURBAK. “Learnable programming: blocks and beyond”. In: *Communications of the ACM* 60.6 (2017), pp. 72–80 (cit. on p. 15).
- [BAUMGARTL *et al.* 2013] Johannes BAUMGARTL, Thomas BUCHMANN, Dominik HENRICH, and Bernhard WESTFECHTEL. “Towards easy robot programming-using DSLs, code generators and software product lines.” In: *ICSOFT*. Citeseer. 2013, pp. 548–554 (cit. on p. 27).
- [BEHRENS *et al.* 2019] Jan Kristof BEHRENS, Karla STEPANOVA, Ralph LANGE, and Radoslav SKOVIERA. “Specifying dual-arm robot planning problems through natural language and demonstration”. In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 2622–2629 (cit. on p. 3).

REFERENCES

- [BENEDICT *et al.* 2019] Jacob D BENEDICT, Jacob D GULIUZO, and Barbara S CHAPARRO. “The intuitiveness of gesture control with a mixed reality device”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 63. 1. SAGE Publications Sage CA: Los Angeles, CA. 2019, pp. 1435–1439 (cit. on p. 45).
- [BENOTSMANE *et al.* 2018] Rabab BENOTSMANE, L DUDÁS, and Gy KOVÁCS. “Collaborating robots in industry 4.0 conception”. In: *IOP Conference Series: Materials Science and Engineering*. Vol. 448. 1. IOP Publishing. 2018, p. 012023 (cit. on p. 7).
- [BERNAL and SRIDHAR 2022] Joey BERNAL and Bharath SRIDHAR. *1.3.4 moving forward and the fourth industrial revolution*. 2022. URL: <https://app.knovel.com/hotlink/khtml/id:kt0134OFW2/industrial-iot-architects/moving-forward-fourth> (cit. on p. 1).
- [BILLINGHURST 2021] Mark BILLINGHURST. “Grand challenges for augmented reality”. In: *Frontiers in Virtual Reality 2* (2021), p. 12 (cit. on p. 19).
- [BISEN and PAYAL 2022] Asmita Singh BISEN and Himanshu PAYAL. “Collaborative robots for industrial tasks: a review”. In: *Materials Today: Proceedings 52* (2022), pp. 500–504 (cit. on p. 13).
- [BOGUE 2016] Robert BOGUE. “Europe continues to lead the way in the collaborative robot business”. In: *Industrial Robot: An International Journal 43.1* (2016), pp. 6–11 (cit. on pp. 2, 12).
- [BURGHARDT *et al.* 2020] Andrzej BURGHARDT *et al.* “Programming of industrial robots using virtual reality and digital twins”. In: *Applied Sciences 10.2* (2020), p. 486 (cit. on p. 2).
- [M. BURNETT *et al.* 2004] Margaret BURNETT, Curtis COOK, and Gregg ROTHERMEL. “End-user software engineering”. In: *Communications of the ACM 47.9* (2004), pp. 53–58 (cit. on p. 14).
- [M. M. BURNETT and McINTYRE 1995] Margaret M BURNETT and David W McINTYRE. “Visual programming”. In: *COMputer-Los Alamitos- 28* (1995), pp. 14–14 (cit. on p. 15).
- [BUTLER, MORGAN, *et al.* 2007] Matthew BUTLER, Michael MORGAN, *et al.* “Learning challenges faced by novice programming students studying high level and low feedback concepts”. In: *Proceedings ascilite Singapore 1.99-107* (2007) (cit. on p. 19).
- [CASINI 2022] Marco CASINI. *6.3.1.1 virtual reality technologies*. 2022. URL: <https://app.knovel.com/hotlink/khtml/id:kt012RPRT4/construction-4-0-advanced/virtual-reality-technologies> (cit. on p. 17).

- [CHANDRA *et al.* 2019] Ananth N Ramaseri CHANDRA, Fatima EL JAMIY, and Hassan REZA. “A review on usability and performance evaluation in virtual reality systems”. In: *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE. 2019, pp. 1107–1114 (cit. on p. 17).
- [CHANG 2012] Shi-Kuo CHANG. *Visual languages*. Springer Science & Business Media, 2012 (cit. on p. 15).
- [CHATTHA *et al.* 2020] Umer Asghar CHATTHA *et al.* “Motion sickness in virtual reality: an empirical evaluation”. In: *IEEE Access* 8 (2020), pp. 130486–130499 (cit. on p. 17).
- [COHRSSSEN 2021] Barbara COHRSSSEN. *12.4.1 industrial robots - risk management*. 2021. URL: <https://app.knovel.com/hotlink/khtml/id:kt012XLV17/pattys-industrial-hygiene/risk-manag-industrial> (cit. on p. 2).
- [CORBIN 2015] Juliet M. CORBIN. *Basics of qualitative research : techniques and procedures for developing grounded theory*. eng. Fourth edition. Thousand Oaks, California: SAGE, 2015. ISBN: 1412997461 (cit. on p. 29).
- [CORNO *et al.* 2019] Fulvio CORNO, Luigi DE RUSSIS, and Juan Pablo SÁENZ. “On the challenges novice programmers experience in developing iot systems: a survey”. In: *Journal of Systems and Software* 157 (2019), p. 110389 (cit. on p. 19).
- [CORONADO *et al.* 2020] Enrique CORONADO, Fulvio MASTROGIOVANNI, Bipin INDURKHYA, and Gentiane VENTURE. “Visual programming environments for end-user development of intelligent and social robots, a systematic review”. In: *Journal of Computer Languages* 58 (2020), p. 100970 (cit. on pp. 3, 16).
- [DAL and DEBACQ 2020] Jean-Pierre DAL and Marie DEBACQ. *5.1.3.2 the period 1960-1990*. 2020. URL: <https://app.knovel.com/hotlink/khtml/id:kt012IQIO1/process-industries-2/the-period-1960-1990> (cit. on p. 1).
- [DISSA and ABELSON 1986] Andrea A. DISSA and Harold ABELSON. “Boxer: a reconstructible computational medium”. In: *Communications of the ACM* 29.9 (1986), pp. 859–868 (cit. on p. 15).
- [DJURIC *et al.* 2017] Ana DJURIC, Jeremy L RICKLI, Vukica M JOVANOVIĆ, and Daniel FOSTER. “Hands-on learning environment and educational curriculum on collaborative robotics”. In: (2017) (cit. on p. 64).
- [DYE *et al.* 2000] Jane F DYE, Irene M SCHATZ, Brian A ROSENBERG, and Susanne T COLEMAN. “Constant comparison method: a kaleidoscope of data”. In: *The qualitative report* 4.1/2 (2000), pp. 1–9 (cit. on p. 47).
- [EKIN 2023] Sabit EKIN. “Prompt engineering for chatgpt: a quick guide to techniques, tips, and best practices”. In: *Authorea Preprints* (2023) (cit. on p. 70).

REFERENCES

- [EL ZAATARI *et al.* 2019] Shirine EL ZAATARI, Mohamed MAREI, Weidong LI, and Zahid USMAN. “Cobot programming for collaborative industrial tasks: an overview”. In: *Robotics and Autonomous Systems* 116 (2019), pp. 162–180 (cit. on p. 63).
- [EVLAMPEV and OSTANIN 2019] A EVLAMPEV and M OSTANIN. “Obstacle avoidance for robotic manipulator using mixed reality glasses”. In: *2019 3rd School on Dynamics of Complex Networks and their Application in Intellectual Robotics (DCNAIR)*. IEEE. 2019, pp. 46–48 (cit. on p. 19).
- [FENG *et al.* 2015] Annette FENG, Eli TILEVICH, and Wu-chun FENG. “Block-based programming abstractions for explicit parallel computing”. In: *Proceedings of the 2015 Blocks and Beyond Workshop*. IEEE. 2015, pp. 71–75 (cit. on p. 24).
- [FLAVIÁN *et al.* 2019] Carlos FLAVIÁN, Sergio IBÁÑEZ-SÁNCHEZ, and Carlos ORÚS. “The impact of virtual, augmented and mixed reality technologies on the customer experience”. In: *Journal of business research* 100 (2019), pp. 547–560 (cit. on p. 41).
- [AL-FRAIHAT *et al.* 2020] Dimah AL-FRAIHAT, Mike JOY, Jane SINCLAIR, *et al.* “Evaluating e-learning systems success: an empirical study”. In: *Computers in human behavior* 102 (2020), pp. 67–86 (cit. on p. 77).
- [FRONCHETTI 2024a] Felipe FRONCHETTI. *Quickstart Guide: Wizard Easy Programming*. Youtube. 2024. URL: <https://www.youtube.com/watch?v=2y1DmG-57JA> (cit. on pp. 70, 76).
- [FRONCHETTI 2024b] Felipe FRONCHETTI. *Replication Package, Thesis. Quick Start Guide: Wizard Easy Programming*. GitHub. 2024. URL: <https://github.com/fronchetti/thesis> (cit. on pp. 70, 76).
- [FRONCHETTI, POPIELA, *et al.* 2024] Felipe FRONCHETTI, Miles POPIELA, Rodrigo SPINOLA, and Shawn BRIXEY. “Roboart: artistic robot programming in mixed reality”. In: *2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE. 2024, pp. 1192–1193 (cit. on p. 83).
- [FRONCHETTI, RITSCHER, HOLMES, *et al.* 2022] Felipe FRONCHETTI, Nico RITSCHER, Reid HOLMES, *et al.* “Language impact on productivity for industrial end users: a case study from programmable logic controllers”. In: *Journal of Computer Languages* 69 (2022), p. 101087 (cit. on p. 85).
- [FRONCHETTI, RITSCHER, SCHORR, *et al.* 2023] Felipe FRONCHETTI, Nico RITSCHER, Logan SCHORR, *et al.* “Block-based programming for two-armed robots: a comparative study”. In: *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE Computer Society. 2023, pp. 494–505 (cit. on p. 64).
- [GHIURĂU *et al.* 2020] Florin-Timotei GHIURĂU, Mehmet Aydın BAYTAŞ, and Casper WICKMAN. “Arcar: on-road driving in mixed reality by volvo cars”. In: *Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 2020, pp. 62–64 (cit. on p. 41).

- [GOEDICKE *et al.* 2022] David GOEDICKE *et al.* “Xr-oom: mixed reality driving simulation with real cars for research and design”. In: *CHI Conference on Human Factors in Computing Systems*. 2022, pp. 1–13 (cit. on p. 41).
- [GOEL *et al.* 2010] Manish Kumar GOEL, Pardeep KHANNA, and Jugal KISHORE. “Understanding survival analysis: kaplan-meier estimate”. In: *International journal of Ayurveda research* 1.4 (2010), p. 274 (cit. on p. 31).
- [GRADMANN *et al.* 2018] Michael GRADMANN, Eric M ORENDT, Edgar SCHMIDT, Stephan SCHWEIZER, and Dominik HENRICH. “Augmented reality robot operation interface with google tango”. In: *ISR 2018; 50th international symposium on robotics*. VDE. 2018, pp. 1–8 (cit. on p. 18).
- [GRAU *et al.* 2020] Antoni GRAU, Marina INDRI, Lucia Lo BELLO, and Thilo SAUTER. “Robots in industry: the past, present, and future of a growing collaboration with humans”. In: *IEEE Industrial Electronics Magazine* 15.1 (2020), pp. 50–61 (cit. on p. 1).
- [GSCHWINDT *et al.* 2019] Mirko GSCHWINDT *et al.* “Can a robot become a movie director? learning artistic principles for aerial cinematography”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 1107–1114 (cit. on p. 83).
- [GUPTA A.K. 2017] Westcott Jean Riescher GUPTA A.K. Arora S.K. *16.5 lead-through programming*. 2017. URL: <https://app.knovel.com/hotlink/khtml/id:kt0119K69L/industrial-automation/lead-through-programming> (cit. on p. 44).
- [GURURANGAN *et al.* 2020] Suchin GURURANGAN *et al.* “Don’t stop pretraining: adapt language models to domains and tasks”. In: *arXiv preprint arXiv:2004.10964* (2020) (cit. on p. 7).
- [HÄGELE *et al.* 2016] Martin HÄGELE, Klas NILSSON, J Norberto PIRES, and Rainer BISCHOFF. “Industrial robotics”. In: *Springer handbook of robotics* (2016), pp. 1385–1422 (cit. on pp. 9, 15).
- [HANNEBAUER and GRUHN 2017] Christoph HANNEBAUER and Volker GRUHN. “On the relationship between newcomer motivations and contribution barriers in open source projects”. In: *Proceedings of the 13th International Symposium on Open Collaboration*. 2017, pp. 1–10 (cit. on p. 20).
- [HARRISON *et al.* 2013] Rachel HARRISON, Derek FLOOD, and David DUCE. “Usability of mobile applications: literature review and rationale for a new usability model”. In: *Journal of Interaction Science* 1.1 (2013), pp. 1–16 (cit. on p. 48).
- [HEIMANN and GUHL 2020] Oliver HEIMANN and Jan GUHL. “Industrial robot programming methods: a scoping review”. In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. IEEE. 2020, pp. 696–703 (cit. on p. 10).

REFERENCES

- [HENTOUT *et al.* 2018] Abdelfetah HENTOUT, Aouache MUSTAPHA, Abderraouf MAOUDJ, and Isma AKLI. “Key challenges and open issues of industrial collaborative robotics”. In: *2018 The 27th IEEE International Symposium on Workshop on Human-Robot Interaction: from Service to Industry (HRI-SI2018) at Robot and Human Interactive Communication. Proceedings. IEEE*. 2018 (cit. on pp. 3, 63).
- [HOENIG *et al.* 2015] Wolfgang HOENIG *et al.* “Mixed reality for robotics”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 5382–5387 (cit. on p. 6).
- [HÖRBST and ORSOLITS 2022] Jakob HÖRBST and Horst ORSOLITS. “Mixed reality hmi for collaborative robots”. In: *International Conference on Computer Aided Systems Theory*. Springer. 2022, pp. 539–546 (cit. on pp. 3, 4, 19).
- [INGALLS *et al.* 1988] Dan INGALLS, Scott WALLACE, Yu-Ying CHOW, Frank LUDOLPH, and Ken DOYLE. “Fabrik: a visual programming environment”. In: *ACM SIGPLAN Notices* 23.11 (1988), pp. 176–190 (cit. on p. 15).
- [JIMENEZ *et al.* 2018] Yerika JIMENEZ, Amanpreet KAPOOR, and Christina GARDNER-MCCUNE. “Usability challenges that novice programmers experience when using scratch for the first time”. In: *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE. 2018, pp. 327–328 (cit. on p. 19).
- [JOHNS and TAYLOR 2009] Kyle JOHNS and Trevor TAYLOR. *Professional microsoft robotics developer studio*. John Wiley & Sons, 2009 (cit. on p. 16).
- [C. JONES 1995] C. JONES. “End user programming”. In: *Computer* 28.9 (1995), pp. 68–70. DOI: 10.1109/2.410158 (cit. on p. 14).
- [Capers JONES 1995] Capers JONES. “End user programming”. In: *Computer* 28.9 (1995), pp. 68–70 (cit. on p. 2).
- [JOST *et al.* 2014] Beate JOST, Markus KETTERL, Reinhard BUDDE, and Thorsten LEIMBACH. “Graphical programming environments for educational robots: open roberta-yet another one?” In: *2014 IEEE International Symposium on Multimedia*. IEEE. 2014, pp. 381–386 (cit. on pp. 3, 16).
- [KANDRAY 2010a] Daniel E. KANDRAY. *7. robot programming*. 2010. URL: <https://app.knovel.com/hotlink/khtml/id:kt007WJHZ6/programmable-automation/robot-programming> (cit. on p. 6).
- [KANDRAY 2010b] Daniel E. KANDRAY. *7.3 robot programming languages*. 2010. URL: <https://app.knovel.com/hotlink/khtml/id:kt007WJ141/programmable-automation/robot-programming-languages> (cit. on p. 5).

- [KARLI *et al.* 2024] Ulas Berk KARLI, Juo-Tung CHEN, Victor Nikhil ANTONY, and Chien-Ming HUANG. “Alchemist: llm-aided end-user development of robot applications”. In: *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*. 2024, pp. 361–370 (cit. on p. 64).
- [KIM 2017] Hae-Young KIM. “Statistical notes for clinical researchers: chi-squared test and fisher’s exact test”. In: *Restorative dentistry & endodontics* 42.2 (2017), pp. 152–155 (cit. on p. 48).
- [KNUDSEN and KAIVO-OJA 2020] Mikkel KNUDSEN and Jari KAIVO-OJA. “Collaborative robots: frontiers of current literature”. In: *Journal of Intelligent Systems: Theory and Applications* 3.2 (2020), pp. 13–20 (cit. on pp. 2, 12).
- [KO, ABRAHAM, *et al.* 2011] Amy J Ko, Robin ABRAHAM, *et al.* “The state of the art in end-user software engineering”. In: *ACM Computing Surveys (CSUR)* 43.3 (2011), pp. 1–44 (cit. on p. 14).
- [KO, B. A. MYERS, *et al.* 2004] Amy J Ko, Brad A MYERS, and Htet Htet AUNG. “Six learning barriers in end-user programming systems”. In: *2004 IEEE Symposium on Visual Languages-Human Centric Computing*. IEEE. 2004, pp. 199–206 (cit. on pp. 3, 5, 7, 20, 37, 64, 69).
- [KUHAİL *et al.* 2021] Mohammad Amin KUHAİL, Shahbano FAROOQ, Rawad HAMMAD, and Mohammed BAHJA. “Characterizing visual programming approaches for end-user developers: a systematic review”. In: *IEEE Access* 9 (2021), pp. 14181–14202 (cit. on pp. 3, 15).
- [KUTZ 2015] Myer KUTZ. *11.4.3 robot control and programming*. 2015. URL: <https://app.knovel.com/hotlink/khtml/id:kt011JIPL1/mechanical-engineers/robot-control-programming> (cit. on p. 9).
- [LIN and David WEINTROP 2021] Yuhan LIN and David WEINTROP. “The landscape of block-based programming: characteristics of block-based environments and how they support the transition to text-based programming”. In: *Journal of Computer Languages* 67 (2021), p. 101075 (cit. on p. 15).
- [H. LIU 2020] Hui LIU. *3.1.4.4 programming method*. 2020. URL: <https://app.knovel.com/hotlink/khtml/id:kt012LA462/robot-systems-rail-transit/programming-method> (cit. on p. 10).
- [O. LIU *et al.* 2017] Oliver LIU, Daniel RAKITA, Bilge MUTLU, and Michael GLEICHER. “Understanding human-robot interaction in virtual reality”. In: *2017 26th IEEE international symposium on robot and human interactive communication (RO-MAN)*. IEEE. 2017, pp. 751–757 (cit. on p. 17).
- [LOWE and LAWLESS 2021] Andrew LOWE and Steve LAWLESS. *1.3.5 the industrial revolutions*. 2021. URL: <https://app.knovel.com/hotlink/khtml/id:kt0132UFIA/artificial-intelligence/industrial-revolutions> (cit. on p. 1).

REFERENCES

- [MA *et al.* 2021] Zhao MA *et al.* “Stylized robotic clay sculpting”. In: *Computers & Graphics* 98 (2021), pp. 150–164 (cit. on p. 83).
- [MAKHATAEVA and VAROL 2020] Zhanat MAKHATAEVA and Huseyin Atakan VAROL. “Augmented reality for robotics: a review”. In: *Robotics* 9.2 (2020), p. 21 (cit. on pp. 18, 41, 59).
- [MALM *et al.* 2019] Timo MALM, Timo SALMI, Ilari MARSTIO, and Iina AALTONEN. “Are collaborative robots safe?” In: *Automaatiopäivät23*. Suomen automaatioseura. 2019, pp. 110–117 (cit. on p. 12).
- [MALONEY *et al.* 2010] John MALONEY, Mitchel RESNICK, Natalie RUSK, Brian SILVERMAN, and Evelyn EASTMOND. “The scratch programming language and environment”. In: *ACM Transactions on Computing Education (TOCE)* 10.4 (2010), pp. 1–15 (cit. on p. 15).
- [MATSAS and VOSNIAKOS 2017] Elias MATSAS and George-Christopher VOSNIAKOS. “Design of a virtual reality training system for human–robot collaboration in manufacturing tasks”. In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 11 (2017), pp. 139–153 (cit. on p. 17).
- [MAYR-DORN *et al.* 2021] Christoph MAYR-DORN, Mario WINTERER, Christian SALOMON, Doris HOHENSINGER, and Rudolf 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 (cit. on pp. 5, 36).
- [MEYER *et al.* 2023] Jesse G MEYER *et al.* “Chatgpt and large language models in academia: opportunities and challenges”. In: *BioData Mining* 16.1 (2023), p. 20 (cit. on p. 7).
- [MIHELJ *et al.* 2019] Matjaž MIHELJ *et al.* “Collaborative robots”. In: *Robotics* (2019), pp. 173–187 (cit. on p. 9).
- [MILGRAM *et al.* 1995] Paul MILGRAM, Haruo TAKEMURA, Akira UTSUMI, and Fumio KISHINO. “Augmented reality: a class of displays on the reality–virtuality continuum”. In: *Telemanipulator and telepresence technologies*. Vol. 2351. Spie. 1995, pp. 282–292 (cit. on p. 41).
- [MILLER *et al.* 2020] Jack MILLER, Melynda HOOVER, and Eliot WINER. “Mitigation of the microsoft hololens’ hardware limitations for a controlled product assembly process”. In: *The International Journal of Advanced Manufacturing Technology* 109 (2020), pp. 1741–1754 (cit. on p. 58).

- [MISRA *et al.* 2020] Ashwin MISRA, Anuj AGRAWAL, and Vihaan MISRA. “Robotics in industry 4.0”. In: *Handbook of Smart Materials, Technologies, and Devices: Applications of Industry 4.0*. Ed. by Chaudhery Mustansar HUSSAIN and Paolo DI SIA. Cham: Springer International Publishing, 2020, pp. 1–35. ISBN: 978-3-030-58675-1. DOI: 10.1007/978-3-030-58675-1_68-1. URL: https://doi.org/10.1007/978-3-030-58675-1_68-1 (cit. on p. 10).
- [MOGLIA *et al.* 2016] Andrea MOGLIA *et al.* “A systematic review of virtual reality simulators for robot-assisted surgery”. In: *European urology* 69.6 (2016), pp. 1065–1080 (cit. on p. 17).
- [NAHM 2016] Francis Sahngun NAHM. “Nonparametric statistical tests for the continuous data: the basic concept and the practical use”. In: *Korean journal of anesthesiology* 69.1 (2016), pp. 8–14 (cit. on p. 32).
- [NARDI 1993] Bonnie A NARDI. *A small matter of programming: perspectives on end user computing*. MIT press, 1993 (cit. on p. 14).
- [NEVES *et al.* 2018] João NEVES, Diogo SERRARIO, and J Norberto PIRES. “Application of mixed reality in robot manipulator programming”. In: *Industrial Robot: An International Journal* 45.6 (2018), pp. 784–793 (cit. on p. 19).
- [NG *et al.* 2022] AHC NG *et al.* “Challenges for manufacturing smes in the introduction of collaborative robots”. In: *SPS2022: Proceedings of the 10th Swedish Production Symposium*. Vol. 21. IOS Press. 2022, p. 173 (cit. on p. 12).
- [NIKU 2020] Saeed B. NIKU. *1.19 social issues*. 2020. URL: <https://app.knovel.com/hotlink/khtml/id:kt0137S7C1/introduction-robotics/social-issues> (cit. on p. 2).
- [NOONE and MOONEY 2018] Mark NOONE and Aidan MOONEY. “Visual and textual programming languages: a systematic review of the literature”. In: *Journal of Computers in Education* 5 (2018), pp. 149–174 (cit. on pp. 5, 15).
- [ONG and SIDDARAJU 2021] Sean ONG and Varun Kumar SIDDARAJU. “Introduction to the mixed reality toolkit”. In: *Beginning Windows Mixed Reality Programming*. Springer, 2021, pp. 85–110 (cit. on p. 42).
- [Mikhail OSTANIN *et al.* 2020] Mikhail OSTANIN, Stanislav MIKHEL, Alexey EVLAMPIEV, Valeria SKVORTSOVA, and Alexandr KLIMCHIK. “Human-robot interaction for robotic manipulator programming in mixed reality”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2805–2811 (cit. on p. 3).
- [PADALA *et al.* 2020] Hema Susmita PADALA *et al.* “How gender-biased tools shape newcomer experiences in oss projects”. In: *IEEE Transactions on Software Engineering* 48.1 (2020), pp. 241–259 (cit. on p. 20).

REFERENCES

- [PARK *et al.* 2021] Sebeom PARK, Shokhrukh BOKIJONOV, and Yosoon CHOI. “Review of microsoft hololens applications over the past five years”. In: *Applied sciences* 11.16 (2021), p. 7259 (cit. on pp. 6, 44).
- [PEDDIE 2023] Jon PEDDIE. “Technology issues”. In: *Augmented Reality : Where We Will All Live*. Cham: Springer International Publishing, 2023, pp. 253–364. ISBN: 978-3-031-32581-6. DOI: 10.1007/978-3-031-32581-6_8. URL: https://doi.org/10.1007/978-3-031-32581-6_8 (cit. on p. 58).
- [H. PENG *et al.* 2018] Hua PENG, Jing LI, Huosheng HU, Changle ZHOU, and Yulong DING. “Robotic choreography inspired by the method of human dance creation”. In: *Information* 9.10 (2018), p. 250 (cit. on p. 83).
- [PICCINELLI *et al.* 2021] Marco PICCINELLI, Andrea GAGLIARDO, Umberto CASTELLANI, and Riccardo MURADORE. “Trajectory planning using mixed reality: an experimental validation”. In: *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE. 2021, pp. 982–987 (cit. on p. 19).
- [POKRESS and VEIGA 2013] Shaileen Crawford POKRESS and José Juan Dominguez VEIGA. “Mit app inventor: enabling personal mobile computing”. In: *arXiv preprint arXiv:1310.2830* (2013) (cit. on p. 14).
- [QUINTERO *et al.* 2018] C Perez QUINTERO *et al.* “Robot programming through augmented trajectories”. In: *VAM-HRI Workshop at The International Conference on Human Robot Interaction*. 2018 (cit. on pp. 3, 19).
- [RAGAGLIA *et al.* 2016] Matteo RAGAGLIA, Andrea Maria ZANCHETTIN, Luca BASCETTA, and Paolo ROCCO. “Accurate sensorless lead-through programming for lightweight robots in structured environments”. In: *Robotics and Computer-Integrated Manufacturing* 39 (2016), pp. 9–21 (cit. on p. 2).
- [RESNICK *et al.* 2009] Mitchel RESNICK *et al.* “Scratch: programming for all”. In: *Communications of the ACM* 52.11 (2009), pp. 60–67 (cit. on p. 14).
- [RITSCHEL, FRONCHETTI, *et al.* 2022a] Nico RITSCHEL, Felipe FRONCHETTI, Reid HOLMES, Ronald GARCIA, and David C SHEPHERD. “Can guided decomposition help end-users write larger block-based programs? a mobile robot experiment”. In: *Proceedings of the ACM on Programming Languages* 6.OOPSLA2 (2022), pp. 233–258 (cit. on p. 84).
- [RITSCHEL, FRONCHETTI, *et al.* 2022b] Nico RITSCHEL, Felipe FRONCHETTI, Reid HOLMES, Ronald GARCIA, and David C SHEPHERD. “Enabling end-users to implement larger block-based programs”. In: *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*. 2022, pp. 347–349 (cit. on p. 84).

- [RITSCHER, FRONCHETTI, *et al.* 2024] Nico RITSCHER, Felipe FRONCHETTI, Reid HOLMES, Ronald GARCIA, and David C SHEPHERD. “Blocks? graphs? why not both? designing and evaluating a hybrid programming environment for end-users”. In: *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*. 2024, pp. 326–327 (cit. on p. 84).
- [RITSCHER, KOVALENKO, *et al.* 2020] Nico RITSCHER, Vladimir KOVALENKO, Reid HOLMES, Ronald GARCIA, and David C SHEPHERD. “Comparing block-based programming models for two-armed robots”. In: *IEEE Transactions on Software Engineering* 48.5 (2020), pp. 1630–1643 (cit. on pp. 2, 5, 21, 24).
- [RITSCHER, SAWANT, *et al.* n.d.] Nico RITSCHER, Anand Ashok SAWANT, *et al.* “Training industrial end-user programmers with interactive tutorials”. In: *Software: Practice and Experience* () (cit. on p. 37).
- [A. ROBOTICS 2020a] ABB ROBOTICS. *Using Wizard to create a PCB assembly application in minutes*. Youtube. 2020. URL: <https://www.youtube.com/watch?v=nj0X8fLj1SE> (cit. on pp. 70, 76).
- [A. ROBOTICS 2020b] ABB ROBOTICS. *Webinar | ABB Wizard easy programming for single arm YuMi*. Youtube. 2020. URL: <https://www.youtube.com/watch?v=OKlcUcLMHQM> (cit. on pp. 70, 76).
- [A. ROBOTICS 2021a] ABB ROBOTICS. *How to program collaborative robot GoFa with Wizard Easy Programming - Tutorial for beginners*. Youtube. 2021. URL: <https://www.youtube.com/watch?v=zPnEOQX4jUA> (cit. on pp. 70, 76).
- [A. ROBOTICS 2021b] ABB ROBOTICS. *Step-by-step guide on pick and place application with Wizard Easy Programming tool*. Youtube. 2021. URL: <https://www.youtube.com/watch?v=eUgqXsWMmwl> (cit. on pp. 70, 76).
- [A. ROBOTICS 2023a] ABB ROBOTICS. *Wizard Easy Programming – Advanced Application Overview*. Youtube. 2023. URL: <https://www.youtube.com/watch?v=CVfQJoM8KsY> (cit. on pp. 70, 76).
- [A. ROBOTICS 2023b] ABB ROBOTICS. *Wizard Easy Programming: For everyone and all new robots*. Youtube. 2023. URL: <https://www.youtube.com/watch?v=Kmv5jUl3WF0> (cit. on pp. 70, 76).
- [A. ROBOTICS 2024] ABB ROBOTICS. *Wizard Easy Programming: Application Manual*. ABB. 2024. URL: <https://new.abb.com/products/robotics/software-and-digital/application-software/wizard> (cit. on pp. 70, 76).
- [I. F. o. ROBOTICS 2023] International Federation of ROBOTICS. *World Robotics*. https://ifr.org/img/worldrobotics/2023_WR_extended_version.pdf. [Accessed: January-2024]. 2023 (cit. on p. 1).

REFERENCES

- [ROKHSARITALEMI *et al.* 2020] Somaiieh ROKHSARITALEMI, Abolghasem SADEGHI-NIARAKI, and Soo-Mi CHOI. “A review on mixed reality: current trends, challenges and prospects”. In: *Applied Sciences* 10.2 (2020), p. 636 (cit. on pp. 19, 41).
- [ROSSANO *et al.* 2013] Gregory F ROSSANO, Carlos MARTINEZ, Mikael HEDELIND, Steve MURPHY, and Thomas A FUHLBRIGGE. “Easy robot programming concepts: an industrial perspective”. In: *2013 IEEE international conference on automation science and engineering (CASE)*. IEEE. 2013, pp. 1119–1126 (cit. on pp. 5, 6, 63).
- [SAENZ *et al.* 2018] José SAENZ, Norbert ELKMANN, Olivier GIBARU, and Pedro NETO. “Survey of methods for design of collaborative robotics applications-why safety is a barrier to more widespread robotics uptake”. In: *Proceedings of the 2018 4th International Conference on Mechatronics and Robotics Engineering*. 2018, pp. 95–101 (cit. on p. 9).
- [SALVENDY 2001] Gavriel SALVENDY. *12.8.4.2 robot simulation*. 2001. URL: <https://app.knovel.com/hotlink/khtml/id:kt0061XYV5/handbook-industrial-engineering/robot-simulation> (cit. on p. 10).
- [SALVENDY and KARWOWSKI 2021] Gavriel SALVENDY and Waldemar KARWOWSKI. *30.2.1.1 Augmented Reality and Mixed Reality Technology, Handbook of human factors and ergonomics*. John Wiley & Sons, 2021 (cit. on p. 19).
- [SCHWAB and DAVIS 2018] Klaus SCHWAB and Nicholas DAVIS. *Shaping the future of the fourth industrial revolution*. Currency, 2018 (cit. on p. 1).
- [SHERIDAN 2016] Thomas B. SHERIDAN. “Human–robot interaction: status and challenges”. In: *Human Factors* 58.4 (2016). PMID: 27098262, pp. 525–532. DOI: 10.1177/0018720816644364. eprint: <https://doi.org/10.1177/0018720816644364>. URL: <https://doi.org/10.1177/0018720816644364> (cit. on p. 59).
- [SHERWANI *et al.* 2020] Fahad SHERWANI, Muhammad Mujtaba ASAD, and Babul Salam Kader K IBRAHIM. “Collaborative robots and industrial revolution 4.0 (ir 4.0)”. In: *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*. IEEE. 2020, pp. 1–5 (cit. on pp. 1, 2, 7, 12, 13, 63).
- [SHUMAKER and LACKEY 2015] Randall SHUMAKER and Stephanie LACKEY. *Virtual, Augmented and Mixed Reality*. Springer, 2015 (cit. on p. 17).
- [SKARBEZ *et al.* 2021] Richard SKARBEZ, Missie SMITH, and Mary C WHITTON. “Revisiting milgram and kishino’s reality-virtuality continuum”. In: *Frontiers in Virtual Reality* 2 (2021), p. 647997 (cit. on p. 17).
- [SMITH *et al.* 1994] David Canfield SMITH, Allen CYPHER, and Jim SPOHRER. “Kidsim: programming agents without a programming language”. In: *Communications of the ACM* 37.7 (1994), pp. 54–67 (cit. on p. 14).

- [SOARES *et al.* 2021] Inês SOARES, Ricardo B. SOUSA, Marcelo PETRY, and António Paulo MOREIRA. “Accuracy and repeatability tests on hololens 2 and htc vive”. In: *Multi-modal Technologies and Interaction* 5.8 (2021), p. 47 (cit. on p. 58).
- [SOUSA 2012] Tiago Boldt SOUSA. “Dataflow programming concept, languages and applications”. In: *Doctoral Symposium on Informatics Engineering*. Vol. 130. 2012 (cit. on p. 16).
- [SPEICHER *et al.* 2019] Maximilian SPEICHER, Brian D HALL, and Michael NEBELING. “What is mixed reality?” In: *Proceedings of the 2019 CHI conference on human factors in computing systems*. 2019, pp. 1–15 (cit. on pp. 18, 44).
- [SPENCER 2009] Donna SPENCER. *Card sorting: Designing usable categories*. Rosenfeld Media, 2009 (cit. on pp. 29, 47).
- [STEINMACHER, SILVA, *et al.* 2014] Igor STEINMACHER, Marco Aurélio Graciotto SILVA, and Marco Aurélio GEROSA. “Barriers faced by newcomers to open source projects: a systematic review”. In: *Open Source Software: Mobile Open Source Technologies: 10th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2014, San José, Costa Rica, May 6-9, 2014. Proceedings 10*. Springer. 2014, pp. 153–163 (cit. on pp. 5, 19).
- [STEINMACHER, WIESE, *et al.* 2015] Igor STEINMACHER, Igor WIESE, Tayana Uchoa CONTE, and Marco Aurelio GEROSA. “Increasing the self-efficacy of newcomers to open source software projects”. In: *2015 29th Brazilian Symposium on Software Engineering*. IEEE. 2015, pp. 160–169 (cit. on p. 20).
- [STOTKO *et al.* 2019] Patrick STOTKO *et al.* “A vr system for immersive teleoperation and live exploration with a mobile robot”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 3630–3637 (cit. on p. 17).
- [SZEDMÁK 2021] Borbála SZEDMÁK. “Business model innovation and the first steps of digitalization in the case of symphony orchestras”. In: (2021) (cit. on p. 41).
- [TAESI *et al.* 2023] Claudio TAESI, Francesco AGGOGERI, and Nicola PELLEGRINI. “Cobot applications—recent advances and challenges”. In: *Robotics* 12.3 (2023), p. 79 (cit. on p. 2).
- [TANIGUCHI *et al.* 2019] Tadahiro TANIGUCHI *et al.* “Survey on frontiers of language and robotics”. In: *Advanced Robotics* 33.15-16 (2019), pp. 700–730 (cit. on p. 3).
- [TARANTINO 2022] Anthony TARANTINO. *13.9 collaborative robots*. 2022. URL: <https://app.knovel.com/hotlink/khtml/id:kt013ETGU1/smart-manufacturing-lean/collaborative-robots> (cit. on pp. 1, 2, 9).
- [TREVELYAN *et al.* 2016] James TREVELYAN, William R HAMEL, and Sung-Chul KANG. “Robotics in hazardous applications”. In: *Springer handbook of robotics* (2016), pp. 1521–1548 (cit. on p. 59).

REFERENCES

- [VERLAG 2022a] VDE VERLAG. *51.2 cobots and people - the best of both worlds*. 2022. URL: <https://app.knovel.com/hotlink/khtml/id:kt01380C91/54th-international-symposium/cobots-people-best-both> (cit. on p. 1).
- [VERLAG 2022b] VDE VERLAG. *51.7 the gateway to the workers of the future*. 2022. URL: <https://app.knovel.com/hotlink/khtml/id:kt01380CG1/54th-international-symposium/gateway-workers-future> (cit. on pp. 2, 13).
- [VERMEULEN 2020] Andreas François VERMEULEN. *13. fourth industrial revolution (4ir)*. 2020. URL: <https://app.knovel.com/hotlink/khtml/id:kt01340CU1/industrial-machine-learning/industrial-fourth-revolution> (cit. on p. 1).
- [VICENTINI 2021] Federico VICENTINI. “Collaborative robotics: a survey”. In: *Journal of Mechanical Design* 143.4 (2021), p. 040802 (cit. on pp. 12, 13).
- [VILLANI *et al.* 2018] Valeria VILLANI, Fabio PINI, Francesco LEALI, Cristian SECCHI, and Cesare FANTUZZI. “Survey on human-robot interaction for robot programming in industrial applications”. In: *Ifac-PapersOnline* 51.11 (2018), pp. 66–71 (cit. on p. 7).
- [WALKER *et al.* 2023] Michael WALKER, Thao PHUNG, Tathagata CHAKRABORTI, Tom WILLIAMS, and Daniel SZAFIR. “Virtual, augmented, and mixed reality for human-robot interaction: a survey and virtual design element taxonomy”. In: *ACM Transactions on Human-Robot Interaction* 12.4 (2023), pp. 1–39 (cit. on p. 3).
- [WANG *et al.* 2019] Qiyue WANG, Yongchao CHENG, Wenhua JIAO, Michael T. JOHNSON, and YuMing ZHANG. “Virtual reality human-robot collaborative welding: a case study of weaving gas tungsten arc welding”. In: *Journal of Manufacturing Processes* 48 (2019), pp. 210–217. ISSN: 1526-6125. DOI: <https://doi.org/10.1016/j.jmapro.2019.10.016>. URL: <https://www.sciencedirect.com/science/article/pii/S1526612519303330> (cit. on p. 17).
- [D. WEINTROP *et al.* 2017] D. WEINTROP, D.C. SHEPHERD, P. FRANCIS, and D. FRANKLIN. “Blockly goes to work: block-based programming for industrial robots”. In: *Proc. of Blocks and Beyond Workshop (B&B)*. 2017, pp. 29–36 (cit. on pp. 5, 36).
- [David WEINTROP, AFZAL, SALAC, FRANCIS, B. LI, David C SHEPHERD, *et al.* 2018a] David WEINTROP, Afsoon AFZAL, Jean SALAC, Patrick FRANCIS, Boyang LI, David C SHEPHERD, *et al.* “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*. 2018, pp. 1–12 (cit. on pp. 16, 36).
- [David WEINTROP, AFZAL, SALAC, FRANCIS, B. LI, David C. SHEPHERD, *et al.* 2018b] David WEINTROP, Afsoon AFZAL, Jean SALAC, Patrick FRANCIS, Boyang LI, David C. SHEPHERD, *et al.* “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, 2018, 366:1–366:12 (cit. on p. 24).

- [David WEINTROP and WILENSKY 2015] David WEINTROP and Uri WILENSKY. “To block or not to block, that is the question: students’ perceptions of blocks-based programming”. In: *Proceedings of the 14th International Conference on Interaction Design and Children*. ACM, 2015, pp. 199–208 (cit. on p. 24).
- [WELLEK 1993] Stefan WELLEK. “A log-rank test for equivalence of two survivor functions”. In: *Biometrics* (1993), pp. 877–881 (cit. on p. 32).
- [WU *et al.* 2023] Tianyu WU *et al.* “A brief overview of chatgpt: the history, status quo and potential future development”. In: *IEEE/CAA Journal of Automatica Sinica* 10.5 (2023), pp. 1122–1136 (cit. on p. 70).
- [YOUNIS *et al.* 2023] Hussain A YOUNIS, Nur Intan Raihana RUHAIYEM, Wad GHABAN, Nadhmi A GAZEM, and Maged NASSER. “A systematic literature review on the applications of robots and natural language processing in education”. In: *Electronics* 12.13 (2023), p. 2864 (cit. on p. 7).
- [ZHANG *et al.* 2023] Ceng ZHANG, Junxin CHEN, Jiatong LI, Yanhong PENG, and Zebing MAO. “Large language models for human-robot interaction: a review”. In: *Biomimetic Intelligence and Robotics* (2023), p. 100131 (cit. on p. 3).
- [ZHOU *et al.* 2020] Tianyu ZHOU, Qi ZHU, and Jing DU. “Intuitive robot teleoperation for civil engineering operations with virtual reality and deep learning scene reconstruction”. In: *Advanced Engineering Informatics* 46 (2020), p. 101170 (cit. on p. 17).
- [ZIAFATI *et al.* 2017] Pouyan ZIAFATI *et al.* “ProcroB architecture for personalized social robotics”. In: *Robots for Learning Workshop@ HRI*. 2017, pp. 6–9 (cit. on p. 16).
- [ZIBRAN 2007] Minhaz Fahim ZIBRAN. “Chi-squared test of independence”. In: *Department of Computer Science, University of Calgary, Alberta, Canada* 1.1 (2007), pp. 1–7 (cit. on p. 30).