



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Dissertation

# Mixed Reality Technologies for Novel Forms of Human-Robot Interaction

Dissertation with the aim of achieving a doctoral degree at the Faculty of  
Mathematics, Informatics and Natural Sciences

**Dipl.-Inf. Dennis Krupke**

Human-Computer Interaction  
and  
Technical Aspects of Multimodal Systems

Department of Informatics  
Universität Hamburg

November 2019



# Review

Erstgutachter: Prof. Dr. Frank Steinicke

Zweitgutachter: Prof. Dr. Jianwei Zhang

Drittgutachter: Prof. Dr. Eva Bittner

Vorsitzende der Prüfungskommission: Prof. Dr. Simone Frintrop

Datum der Disputation: 17.08.2020



“ My dear Miss Glory, Robots are not people. They are mechanically more perfect than we are, they have an astounding intellectual capacity, but they have no soul.”

Karel Capek



# Abstract

Nowadays, robot technology surrounds us and future developments will further increase the frequency of our everyday contacts with robots in our daily life. To enable this, the current forms of human-robot interaction need to evolve. The concept of digital twins seems promising for establishing novel forms of cooperation and communication with robots and for modeling system states. Machine learning is now ready to be applied to a multitude of domains. It has the potential to enhance artificial systems with capabilities, which so far are found in natural intelligent creatures, only. Mixed reality experienced a substantial technological evolution in recent years and future developments of mixed reality devices seem to be promising, as well. Wireless networks will improve significantly in the next years and thus, latency and bandwidth limitations will be no crucial issue anymore. Based on the ongoing technological progress, novel interaction and communication forms with robots become available and their application to real-world scenarios becomes feasible. This thesis enables a conceptual view of this future.

The theoretical part of this thesis provides the reader with the necessary background for developing mixed reality robotic user interfaces (MR-RUI). It presents a framework for their technological classification, which is explained and applied to existent robotic systems.

Furthermore, the investigation of immersive robotic research experiments from the literature and their requirements gave rise to the development of a software framework for implementing MR-RUIs in the practical part of this thesis.

In the applied part, the software framework is used to develop and evaluate MR-RUI prototypes for this thesis. User studies evaluate different methods in MR for gesture-based control of robots, mostly involving the operators' hands and head. Finally, mid-air gestures and multimodal interaction combined with a stereoscopic rendering of the operator's view are found to be efficient means for novel human-robot interaction. The ongoing technological progress will refine the quality and capabilities of technological devices used for the presented experiments and further improve future implementations of robotic user interfaces in MR.

Summarizing the investigations, developments and evaluations from this thesis indicates that the future of human-robot interaction with natural interaction in mixed reality seems to be promising. The next generation of technological and methodological achievements could overcome many actual limits and when they are ready, further grow together into large interconnected systems. All in all, the next generation of intelligent robotic systems will cooperate with us at the next level of blended reality and will come closer to assist us in our daily life, with communication skills comparable to human companions and even beyond the natural limitations.





# Zusammenfassung

Heutzutage ist Technologie, gerade in Bezug auf Roboter, in unserem täglichen Umfeld allgegenwärtig, und dieser Kontakt im Alltag wird sich durch zukünftige Entwicklungen noch deutlich ausweiten. Um dies handhabbar zu machen, müssen sich die heutigen Formen der Mensch-Roboter Interaktion noch deutlich weiterentwickeln. Es scheint vielversprechend zu diesem Zweck einen digitalen Zwilling zu nutzen, der als virtuelles Model fungiert und neue Formen der Kooperation und Kommunikation, zwischen Mensch und Roboter, etablieren kann. Machine learning ist mittlerweile in vielen Bereichen einsetzbar. Es hat das Potential künstliche Systeme mit Fähigkeiten auszustatten, die normalerweise nur in der Natur zu finden sind. Mixed reality hat in den letzten Jahren eine grundlegende Weiterentwicklung erfahren, was vermuten lässt, das auch zukünftige Entwicklungen in diesem Bereich viele neue Möglichkeiten eröffnen werden. Drahtlose Netzwerke werden sich in den nächsten Jahren stark verbessern, und somit Einschränkungen durch Latenz und Bandbreiten deutlich reduzieren. Basierend auf dem andauernden technologischen Fortschritt, werden neue Interaktions- und Kommunikationsformen mit Robotern verfügbar sein, was diese für die Anwendung in “Real-Life” Szenarios nutzbar macht. Diese Dissertationsschrift gibt einen konzeptuellen Ausblick auf diese Zukunft.

Der theoretische Teil dieser Dissertationsschrift gibt dem Leser einen Überblick über den notwendigen Hintergrund zur Entwicklung von mixed reality robotic user interfaces (MR-RUI). Er stellt einen Rahmen für die technologische Klassifizierung, welcher Erklärungen bietet und auf bestehende Systeme in der Robotik angewendet wird. Weiterhin hat die Auswertung von immersiven Forschungsexperimenten in der Robotik aus der Literatur dazu geführt, dass im praktischen Teil dieser Arbeit ein Framework für die Software Implementierung von MR-RUIs entwickelt wurde. Im angewandten Teil dieser Doktorarbeit wurde dieser genutzt, um MR-RUI Prototypen zu entwickeln und zu bewerten. Studien mit Nutzern wurden durchgeführt, um verschiedene Methoden von Gesten-gesteuerter Bedienung von Robotern zu bewerten, hauptsächlich durch Hand- und Kopfgesten. Hierbei ergab sich, dass Gesten im freien Raum und multimodale Interaktion, kombiniert mit stereoskopischer Visualisierung geeignete Methoden zur Mensch-Roboter Interaktion in MR sind. Die in den nächsten Jahren folgende technische Entwicklung wird die Qualität und Möglichkeiten der in den Experimenten genutzten Geräte, für zukünftige Implementierungen von MR-RUIs, weiter verbessern.

Zusammenfassend zeigen die Untersuchungen, Entwicklungen und Bewertungen dieser Arbeit, dass die Zukunft des Zusammenspiels von Mensch-Roboter Interaktion, und der natürlichen Interaktion in mixed reality, sehr vielversprechend ist.

Die nächste Generation und ihre technologisch- und methodisch ausgereiften Errungenschaften könnten heutzutage noch vorhandenen Grenzen überwinden, und wenn sie bereit sind, weiter zusammenwachsen und ein großes weitvernetztes System ermöglichen. Alles in

Allen, wird die nächste Generation intelligenter Robotik Systeme es uns ermöglichen, auf dem nächsten Level einer vermischten Realität, in unserem täglichen Leben mit Robotern wie mit anderen Menschen zu kommunizieren, und sogar von der Natur vorgegebene Grenzen zu überwinden.

# CONTENTS

<b>Abstract</b>	<b>vii</b>
<b>Part I: Introduction</b>	<b>1</b>
<b>1 Motivation</b>	<b>3</b>
<b>2 Contributions</b>	<b>7</b>
2.1 Scientific Contributions . . . . .	7
2.1.1 Research Questions . . . . .	8
2.1.2 Relevant Publications during Promotion . . . . .	8
2.2 Structure of the Thesis . . . . .	10
<b>Part II: Theoretical and Technical Foundations</b>	<b>13</b>
<b>3 Robots</b>	<b>15</b>
3.1 History of Robots . . . . .	15
3.1.1 Beginnings of Robotics . . . . .	16
3.1.2 Visions of Robots . . . . .	19
3.1.3 First Industrial and Research Robots . . . . .	22
3.1.4 Social Robots . . . . .	24
3.1.5 Robots From Experiments Described in this Dissertation . . . . .	25

3.2	Classification of Robots . . . . .	27
3.3	Concepts of Control and Interaction . . . . .	28
3.3.1	Models of Control . . . . .	28
3.3.2	Classical Control Paradigms for Intelligent Robots . . . . .	31
3.3.3	User Interfaces for Robot Programming and Interaction . . . . .	33
3.4	Definition of a Robotic System . . . . .	35
<b>4</b>	<b>Mixed Reality Interaction</b>	<b>37</b>
4.1	Displaying Mixed Reality . . . . .	37
4.2	Basic 3D Interaction . . . . .	39
4.3	Interaction with Virtual Environments . . . . .	41
4.3.1	General Terms . . . . .	41
4.3.2	3D Interaction Taxonomies . . . . .	42
<b>5</b>	<b>Classification of MR Robotic UIs</b>	<b>49</b>
5.1	IMPAct Framework . . . . .	49
5.1.1	Taxonomies with Relation to MR-RUIs . . . . .	52
5.1.2	Materials and Methods . . . . .	56
5.1.3	Validation of the IMPAct Framework . . . . .	60
5.1.4	Discussion of the IMPAct Framework . . . . .	68
5.2	Examples . . . . .	69
5.2.1	Essential MR-RUIs . . . . .	70
5.2.2	Challenges for MR-RUI . . . . .	75
5.2.3	Recent MR-RUIs . . . . .	77
<b>Part III: MR Interactions and Visualization for Immersive HRI</b>		<b>83</b>
<b>6</b>	<b>Pre-Studies of 3DUI Integration</b>	<b>85</b>
6.1	Altering the Locomotion of Chainlike-Robots . . . . .	85
6.1.1	Background in Locomotion of Modular Robots . . . . .	87
6.1.2	Description of the Teleoperation System . . . . .	88

6.1.3	Evaluation . . . . .	90
6.1.4	Conclusion . . . . .	94
6.2	Ballistic Movements for Target Prediction . . . . .	94
6.2.1	Problem Description . . . . .	94
6.2.2	Methods . . . . .	95
6.2.3	Open-Loop vs. Closed-Loop in Ballistic Aiming . . . . .	96
6.2.4	Characteristics of Velocity Trend . . . . .	96
6.2.5	Discussion and Conclusion of the Pre-Study . . . . .	96
<b>7</b>	<b>Prototyping Immersive Interaction</b>	<b>99</b>
7.1	Prototyping Workflow . . . . .	100
7.1.1	Preparing the ROS-Side . . . . .	100
7.1.2	Preparing the Unity3D-Side . . . . .	101
7.2	Use Cases . . . . .	103
7.2.1	Programming of Pre-Grasping Poses . . . . .	103
7.2.2	Teleoperation of Industrial Manipulators . . . . .	104
7.2.3	Altering the Shape of a Snake-Like Modular Robot . . . . .	106
7.2.4	Extended Body Posture Tracking for Humanoid Robots . . . . .	107
7.2.5	Embodiment of an Anthropomorphic Hand . . . . .	108
7.2.6	Embodied Drone Control . . . . .	108
7.2.7	360 Mobile Robot Teleoperation . . . . .	109
<b>8</b>	<b>Predictive Visualization Studies in VR</b>	<b>111</b>
8.1	Visual Virtual Fixtures in VR . . . . .	111
8.1.1	Previous Work . . . . .	112
8.1.2	Implementation of Perceptual Overlays for Immersive Telerobotics	113
8.1.3	Evaluation of Multimodal Virtual Fixtures . . . . .	114
8.1.4	Results . . . . .	116
8.1.5	Utility of Virtual Fixtures in VR-based Interfaces . . . . .	122

<b>9</b>	<b>Predictive Visualization Studies in AR</b>	<b>125</b>
9.1	Heading and Pointing Gestures . . . . .	126
9.1.1	Previous Work . . . . .	126
9.1.2	Mixed Reality HRI Prototype . . . . .	128
9.1.3	Evaluation . . . . .	132
9.1.4	Results . . . . .	134
9.1.5	Validation of the Results . . . . .	135
9.1.6	Conclusion and Future Work . . . . .	137
 <b>Part IV: Conclusion</b>		 <b>139</b>
<b>10</b>	<b>Summary</b>	<b>141</b>
<b>11</b>	<b>Discussion</b>	<b>143</b>
<b>12</b>	<b>Outlook</b>	<b>145</b>
 <b>Appendices</b>		 <b>147</b>
<b>A</b>	<b>IMPAct Material</b>	<b>149</b>
A.1	IMPAct Reference Table . . . . .	149
A.2	IMPAct Template . . . . .	155
<b>B</b>	<b>Combining ROS and Unity3D</b>	<b>157</b>
B.1	Introduction . . . . .	158
B.2	General Architecture . . . . .	159
B.2.1	Concept of Control and Feedback Integration . . . . .	160
B.2.2	Example of System Engineering . . . . .	160
B.3	Communication . . . . .	161
B.3.1	Messages . . . . .	162
B.3.2	Message Exchange via WebSockets . . . . .	170
B.3.3	Example: Communication during Robotic Hand Control . . . . .	173
B.3.4	Modifications for Microsoft HoloLens . . . . .	174

---

B.4	Feedback . . . . .	175
B.4.1	Visualization . . . . .	175
B.4.2	Haptic Rendering . . . . .	184
B.5	From Input to Robot Control . . . . .	184
B.5.1	HMD-Tracking . . . . .	186
B.5.2	Hand Recognition . . . . .	187
B.5.3	HoloLens and the Mixed Reality Toolkit (MRTK) . . . . .	188
B.6	Other Tools . . . . .	188
B.6.1	Coordinate Conversion . . . . .	188
B.6.2	Registration . . . . .	189
B.6.3	System State Machine . . . . .	190
	<b>List of Figures</b>	<b>196</b>
	<b>List of Tables</b>	<b>198</b>
	<b>List of Definitions</b>	<b>199</b>
	<b>Acknowledgments</b>	<b>201</b>
	<b>Bibliography</b>	<b>203</b>





PART

I

# INTRODUCTION



## MOTIVATION

Already today, we are surrounded by a vast amount of robots, cyber-physical systems, or autonomous systems driven by artificial intelligence (AI), and it can be expected that this number will grow in the future [203][2][83]. Industrial robots and cyber-physical systems have been used in decades in industrial contexts, and are getting more accessible for small and medium enterprises as well as start-ups [202]. Furthermore, lawnmowers, cleaning robots, and self-driving cars become more common and smarter.

While there is an enormous interest in AI and machine learning, full automation in complex and dynamic contexts is still a challenging task [16][56][224][150][46][51]. Furthermore, from an ethical, philosophical as well as societal perspective, it is essential that humans maintain decision sovereignty, so that humans and AI successfully can work together [132][110][181][151][152][29].

Hence, novel concepts for human-robot interaction are required, which address the challenges, that occur when humans and robots collaborate in the same physical space [6][40][20]. First approaches of the industry to address this issue are to develop robots with an increased level of safety. Universal Robots [165] developed UR-2, UR-5 and UR-10; low-cost robots with a focus on safety and a relatively simple programming interface. In 2015, ABB introduced YuMi, a dual-armed robot for collaborative tasks [1]. Very recently, Franka Emika was presented as the first industrial robot with CE-conformity [65] integrating novel interaction and programming concepts.

These cobots (collaborative robots) are intended to physically interact with humans in shared spaces. Collaboration with robots has enormous potential when the robot performs

unfavored, dangerous or high precision parts of the work [89][32][84]. Typically, these complex systems are based on networked subcomponents with distributed data processing. Assistance systems in modern cars, smart home systems and many software systems for smartphones rely on system state modeling and adaptive learning mechanisms to serve the user better by adapting her behavior. Thus, not only the user is improving on how to use a certain system, but also the system improves the way how it performs a task or presents desired information (aka continuous learning) [194].

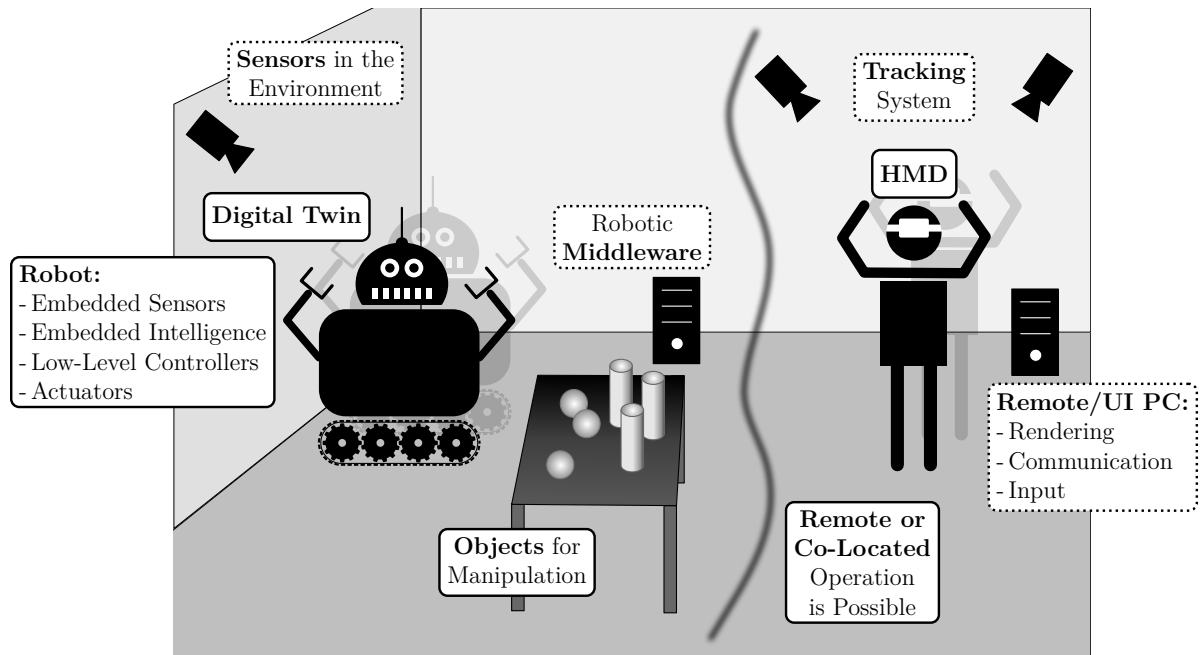
Robots have made enormous technological progress, and are getting more and more available and accessible; while the cost for a KUKA LWR robotic arm are around \$200,000, a UR-5 arm cost only about \$30,000. Hence, it can be expected that more and more robots will be collaborating with humans in a shared physical space. Therefore, an important research question is:

How can humans interact with robots, their growing complexity, autonomy and functionality, when both collaborate in the same physical space?

One recent approach to deal with the complexity of a physical system is to gather all kinds of available data and generate a digital twin of this entity, which was already forecasted in the early 1990s [64]. El Saddik [54] discusses different applications of the digital twin concept, which connects a virtual model with an actual entity and embeds AI in the model. As a result, system failures and upcoming repairs are predictable, since real data is available to the AI, which is integrated into the virtual counterpart.

In the field of virtual models, technologies such as virtual reality (VR) and augmented reality (AR) – or more general mixed reality (MR) – are on the rise [183]. With MR it becomes possible to naturally interact with virtual objects embedded in the real environment or with completely virtual environments (VE). Though, interaction in MR environments is not standardized compared to user interfaces for graphical user interactions (GUI) for desktop or mobile devices. Several 3D user interfaces (3DUI) have been established in the last decades [68][26][9]. Hence, it seems to be promising to use MR technologies to interact with virtual digital twins, in particular, since MR interaction takes place in the real physical surroundings of the user, for example, by means of gestures, speech or movements. Therefore, MR technologies have enormous potential as a user interface to allow humans to control robots in the same physical space.

Figure 1.1 shows the concept of MR-based user interfaces and demonstrates the flexibility. The MR user interface connects the logical component of the robotic environment with one of the human collaborators. Bi-directional information exchange is enabled by connecting human and artificial autonomous agent. The human benefits from rendered information on the worn head-mounted display (HMD) or other output devices and the robot make use of the access to the digital twin of the human. The robot might be intelligent enough to evaluate the actual human behavior for preparing future actions and the human can



**Figure 1.1:** Robotic system with MR user interface. Mixed reality allows for very flexible system design, from co-located single-user and single-robot setups, ranging to remotely controlled multiple-user and multi-robot scenarios. Virtual twins become visible by adding a virtual body and are suitable for interaction by means of 3D user interfaces.

perceive what the embedded intelligence in the robot is preparing. Such a system is neither limited to a single user, nor a single robot. Mixed approaches of co-located and remote operators interacting with several robots at the same time are possible. The physical environment of the robot and the human is extendable with additional sensors to enhance the digital twins of the actors or for improved perception of the environment. Recognizing the environment, thus, creating digital twins of manipulatable objects, allows for novel interaction concepts. The linkage of the actual objects with their virtual counterpart allows for implicit interaction with the environment. Systems designed to deal with large latency would benefit from implicit interaction techniques. Instead of teleoperating a robot in realtime with high latency, virtual twins are prepared for interaction with low latency. The implementation of intelligent, autonomous behavior would support this kind of interaction by performing the desired actions in the next future. By connecting several computing nodes via the network, arbitrary architectures for intelligent control can provide efficient MR systems for robot interaction (cf. Definition 3.12). Increased safety for human, robot and environmental objects is achieved by either evaluating tracking information directly or predicting future actions. Presenting information directly in the field-of-view (FOV) of the human by utilizing the HMD represents an additional safety feature, since the operator is not forced to look in a certain direction for receiving important information, like safety warnings.



This chapter outlines the scientific contributions of the thesis on hand and presents the overall structure of the dissertation.

## 2.1 Scientific Contributions

The scientific contribution of this thesis is given by the theoretical part and the applied part. The theoretical part classifies and orders immersive robotic interaction. The application and evaluation of novel interaction concepts to several robotic interface prototypes with different levels of MR aims to identify possibilities to assist humans in interaction tasks with complex robots. Furthermore, a framework for developing robotic UIs in MR was developed in the scope of this thesis. Its implementation and usage is documented in the appendix of this dissertation. A list of contributing scientific publications is provided in Section 2.1.2.

In the theoretical part, findings from literature review of immersive HRI are summarized and a framework for objective and technical classification of mixed reality robotic user interfaces (MR-RUI) is developed and applied to the classification of MR-related robotic user interfaces (RUI). The framework assists in reviewing actual systems, utilizing the proposed taxonomy, and in addition also tries to support interaction designers in controlling major factors of MR-RUIs.

Different novel aspects of MR-RUIs are evaluated by implementing specific case studies,

presented in the practical part. Questions range from how state-of-the-art hardware is applicable in a real world manipulation scenario to how virtual stereoscopic content can help to increase humans' capabilities during robot teleoperation.

Finally a general software framework, connecting ROS and Unity3D via the network, is introduced. The framework itself is independent from specific hardware. By utilizing Unity3D for the UI arbitrary hardware for feedback and input is available. ROS is currently the state-of-the-art in robotics research. Thus, a wide range of robots and relevant sensors and actuators is supported.

### 2.1.1 Research Questions

The following research questions are answered:

1. How can mixed reality be efficiently integrated into existing robotic systems?
2. What kind of interaction concepts and methods are appropriate for controlling robotic hardware with state-of-the-art consumer AR, VR and MR hardware?
3. Which interaction techniques improve the predictability of the actions and increase the transparency of its current state to the human collaborator?

This thesis address these questions in depth.

### 2.1.2 Relevant Publications during Promotion

The following publications contributed to this dissertation:

#### *Journal Articles*

[mZS] Dennis Krupke, Jianwei Zhang, and Frank Steinicke. IMPAct: A Holistic Framework for Mixed Reality Robotic User Interface Classification and Design. *Multimodal Technologies and Interaction*, 3(2). 10

[ZLm<sup>+</sup>] Jingxin Zhang, Eike Langbehn, Dennis Krupke, Nicholas Katzakis, and Frank Steinicke. Detection Thresholds for Rotation and Translation Gains in 360° Video-based Telepresence System. *IEEE Transactions on Visualization and Computer Graphics (TVCG), Special Issue on IEEE Virtual Reality (VR)*, 24(4):1671 – 1680. 110



*Conference Papers*

- [mLB<sup>+</sup>] Dennis Krupke, Paul Lubos, Gerd Bruder, Jianwei Zhang, and Frank Steinicke. Natural 3D Interaction Techniques for Locomotion with Modular Robots. *Mensch und Computer 2015-Proceedings*. 10
- [mSE<sup>+</sup>] Dennis Krupke, Sebastian Starke, Lasse Einig, Frank Steinicke, and Jianwei Zhang. Prototyping of Immersive HRI Scenarios. In Mohammad O. Tokhi Benedita Malheiro Pedro Guedes Manual F. Silva, Gurvinder S. Virk and Paulo Ferreira, editors, *Human-Centric Robotics*, Proceedings of CLAWAR 2017: 20th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, pages 537–544. CLAWAR Association, World Scientific. 10, 99, 113, 114, 128
- [mSL<sup>+</sup>] Dennis Krupke, Frank Steinicke, Paul Lubos, Yannick Jonetzko, Michael Görner, and Jianwei Zhang. Comparison of Multimodal Heading and Pointing Gestures for Co-Located Mixed Reality Human-Robot Interaction. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. 10, 60, 62, 77, 103, 113, 114
- [mZS] Dennis Krupke, Jianwei Zhang, and Frank Steinicke. Virtual Fixtures in VR - Perceptual Overlays for Assisted Teleoperation, Teleprogramming and Learning. In Gerd Bruder, Shunsuke Yoshimoto, and Sue Cobb, editors, *Proceedings of the ICAT-EGVE (International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments) 2018*. The Eurographics Association, The Eurographics Association. 10, 81
- [SHmZ] Sebastian Starke, Norman Hendrich, Dennis Krupke, and Jianwei Zhang. Evolutionary Multi-Objective Inverse Kinematics on Highly Articulated and Humanoid Robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017)*. 10

*Workshop Papers*

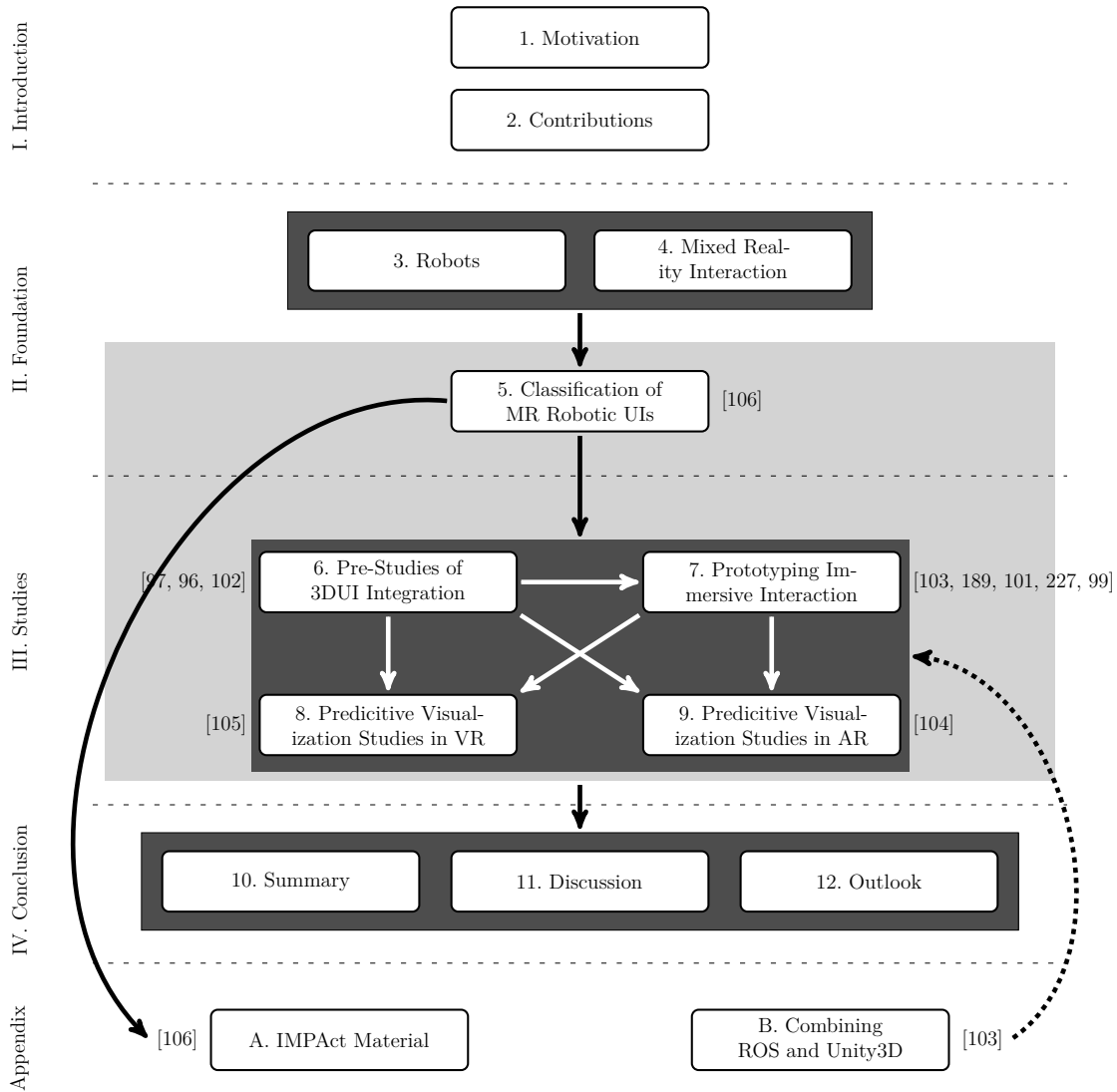
- [ZLm<sup>+</sup>] Jingxin Zhang, Eike Langbehn, Dennis Krupke, Nicholas Katzakis, and Frank Steinicke. A 360° Video-based Robot Platform for Telepresent Redirected Walking. In *Proceedings of ACM SIGCHI Conference on Human-Robot Interaction (HRI) workshop on Virtual, Augmented and Mixed Reality for Human-Robot Interaction*, International Workshop on Virtual, Augmented and Mixed Reality for Human-Robot Interaction. ACM, ACM. 10, 110

*Posters*

- [mBS] Dennis Krupke, Gerd Bruder, and Frank Steinicke. Analyses of Spatial Ballistic Movements for Prediction of Targets in Reach to Grasp Tasks. In *Eurographics Symposium on Virtual Environments (Poster)*. 10
- [mEL<sup>+</sup>] Dennis Krupke, Lasse Einig, Eike Langbehn, Jianwei Zhang, and Frank Steinicke. Immersive Remote Grasping: Realtime Gripper Control by a Heterogenous Robot Control System. In *VRST '16 Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology Pages 337-338*. 10
- [mSZ] Dennis Krupke, Frank Steinicke, and Jianwei Zhang. Target Prediction in an Immersive Pick-and-Place Scenario. In *DGR Days electronic proceedings*. DGR (Deutsche Gesellschaft für Robotik). 10

## 2.2 Structure of the Thesis

This dissertation is divided into four major parts and the appendices. Figure 2.1 presents the coherence map of the single chapters and supports the reader’s understanding of the content dependencies.



**Figure 2.1:** Coherence map of the thesis.

In the single parts the following topics are discussed:

### *Part I: Introduction*

Part I explains the scientific contribution and structure of the single chapters. The motivation and important background information are provided in Chapter 1 of the introduction. The scientific contribution and structure is summarized in Chapter 2.

*Part II: Theoretical and Technical Foundations*

Chapter 3 and Chapter 4 introduce relevant basics from the fields of robotics and MR-based interaction in virtual environments (VE). An overview of interactive robots and basic control approaches is summarized in the robotics part. The chapter about mixed reality interaction explains basic interaction concepts and defines relevant terms.

Based on this, theoretical publications with an impact on MR-RUI are summarized in Chapter 5 and further evolved to the *IMPAct framework*. The chapter concludes with the application of the resulting classification template to robotic user interfaces from scientific publications involving MR. Detailed information on the classification template is located in the Appendices (cf. Chapter A).

*Part III: MR Interactions and Visualization for Immersive HRI*

Part III presents studies on MR interaction with robots. After introducing experimental work on natural interaction methods with robots in Chapter 6, the focus lies on developing a concept for MR integration into robotic systems in Chapter 7. Based on this, prototypes with multimodal input and feedback mechanisms using VR HMDs (cf. Chapter 8) and AR see-through glasses (cf. Chapter 9) were developed and evaluated.

In detail, Chapter 6 presents the results from interaction studies with continuous posture-based hand gesture control and the predictability of reach-to-grasp movements with low-cost sensors. These pre-studies influenced follow-up works presented in Chapter 7-9. Chapter 7 proposes a possible concept for combining a game-engine with a robotic middleware in order to create immersive UIs for robot interaction and presents several use-cases. The concept is applied in the prototypes described in the following two chapters. In Chapter 8 multimodal operator guidance in a continuous teleoperation task with a VR user interface is evaluated. This work follows the concept of “*virtual fixtures*” from one of the very first AR-based teleoperation systems. Part III closes with the evaluation of a tetherless AR setup for discrete multi-modal interaction with industrial robots in Chapter 9, which combines heading or pointing gestures with speech commands in a pick-and-place scenario.

*Part IV: Conclusion*

The fourth and last part discusses the results from Chapter 5-9 in a larger context and suggests future ideas in the context of MR-RUI.

Chapter 10 summarizes the main findings of the dissertation. Chapter 11 discusses the results. Finally, Chapter 12 presents future ideas on MR-RUI.

### *Appendices*

Chapter A contains additional material of the IMPAct framework, such as the template for classification of MR robotic UIs and a brief explanation of the single categories, which are helpful for a complete understanding and the application of the framework.

Chapter B is an extensive description of the implementation of the framework, connecting ROS and Unity3D, and contains details on the implementation of the prototypes utilized in the user studies presented in Part III.

PART

II

THEORETICAL AND TECHNICAL FOUNDA-  
TIONS



In this chapter relevant foundations from the robotics field located are given to provide a better understanding of the topic around interaction with robots. Furthermore, relevant terms for Chapter 5 are introduced.

Section 3.1 explains the historical development of the term “*robot*” and presents examples of robots, which were designed or utilized for humans to interact with. Concluding this section, recent robots, used in different experiments for this dissertation in Part III, are presented, as well. In Section 3.2, a selection of aspects and preexisting attempts on classifying robots is summarized. To facilitate a better understanding of human-robot interaction from a system’s perspective, Section 3.3 briefly explains classical control and interaction concepts. Finally, Section 3.4 defines a “*robotic system*” in the context of this thesis.

### 3.1 History of Robots

Since, HRI relates to several kinds of robot definitions, the following section presents some examples of robots (including existing ones as well as visionary approaches) the chronological order. Details on the typical interaction concept according to the robot are explained, briefly.

### 3.1.1 Beginnings of Robotics

The term “robot” originates from the Czechoslovakian word “*robota*” meaning *work* or *forced labor*. Examples from humankind’s history show different motivations for engaging artificial structures and machines in arbitrary situations. Possible reasons for developing these machines and utilizing robotic technology are avoiding unpleasant or life threatening tasks, improving efficiency or reducing costs, or presenting novel kinds of entertainment. Figure 3.1 shows the ancestors of actual robots. It visualizes the relationship of different device categories and the timely period of their occurrence in the history.

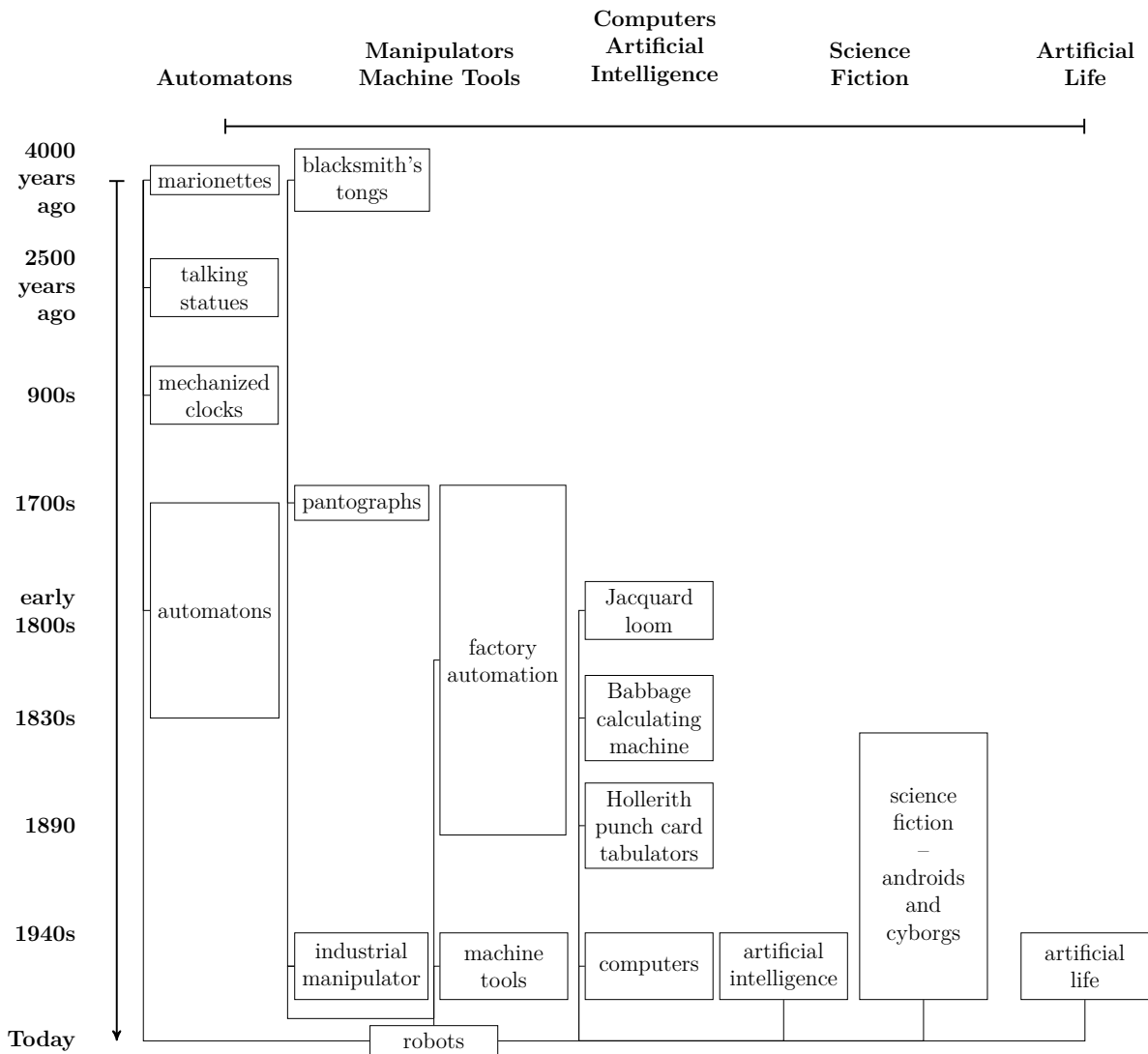


Figure 3.1: Robot family tree (adapted from Thro [200]).

The early beginning of robot development generated various machines, so called automaton, especially for representative purposes. Prominent examples are found in written records of Hephaistos, Diomedes and Ptolemaios [17][? ]. Mechanisms for moving marionettes and statues based on waterflow, steam, sand or quicksilver were developed. Power



transmission was provided by levers, pulleys or screws (cf. Haun [73]). In this way, artificial counterparts of real animals or humans were build. These automatons were designed to perform a single or a very limited number of actions, which were triggered by simple physical actions of one or more humans controlling them. Hiding these triggering actions, usually increased the impact or element of surprise by the observers. Consequently, the dream of fully automated or intelligent actions and interactions arose. In the 17th century, engineers started to build automatons and factories with automated processes (cf. Figure 3.1).

Industrial manipulators and novel machines were build. Their development was mainly driven by the technical progress, the visions of engineers and science fiction authors. Controlled by computers and later provided with embedded intelligence, these machines were able to work autonomously on repetitive tasks, if no failure occurred. Based on the technical progress in the 20th century, different robot definitions were formulated, each with a slightly different focus on relevant aspects.

**Definition 3.1: *Robot - Oxford Dictionary***

A machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer. [49]

Oxford Dictionary defines a robot very general as an autonomous machine, which is capable of carrying out computational tasks (cf. Definition 3.1). This definition does not include explicitly that a robot needs a physical body.

**Definition 3.2: *Robot - Kent Schlüssel, 1983***

*“A programmable multi-function manipulator designed to move material, parts, or specialized devices through variable programmed motions for the performance of a variety of tasks.”* [174]

In contrast, Kent Schlüssel defines a robot as a manipulator, which moves material, thus, being strongly related to physical labor (cf. Definition 3.2).

**Definition 3.3: *Robot - McKerrow, 1986***

*“A robot is a machine which can be programmed to do a variety of tasks, in the same way that a computer is an electronic circuit which can be programmed to do a variety of tasks.”* [125]

McKerrow (cf. Definition 3.3) and Niku [142] focus on how the robot is controlled and programmed, since it defines the variability of tasks it is able to perform.

*“In general, robots are designed and meant to be controlled by a computer or similar device. The motions of the robot are controlled through a controller under the supervision of the computer, which is running some type of a program ... ”*[142].

In the 21st century the demand for autonomous and intelligent robots increased and the possibility to address these requirements became possible. A technical definition of “*autonomy*” is given in Definition 3.4, which allows to define the degree of autonomy on a continuum.

**Definition 3.4: *Autonomy - Zeltzer, 1992***

“*Autonomy then, is a qualitative measure of the ability of a computational model to act and react to simulated events and stimuli, ranging from 0 for the passive geometric model to 1 for the most sophisticated, physically based virtual agent.*” [224]

A very popular definition of “*automation*” instead (cf. Definition 3.5), focuses on the utility of replacing human operators.

**Definition 3.5: *Automation - Parasuraman et. al, 2000***

“*... and define automation as a device or system that accomplishes (partially or fully) a function that was previously, or conceivably could be, carried out (partially or fully) by a human operator.*” [150]

Murphy and Arkin [134] define the requirements of “*intelligent robots*” (cf. Definition 3.6), which includes the robot to show a certain degree of autonomous behavior.

**Definition 3.6: *Intelligent Robot - Murphy & Arkin, 2000***

In Murphy and Arkin [134] an intelligent robot is defined as a mechanical creature which consists of mechanical building blocks. It may use a computer as a building block for its nervous system. It can interact with the world by moving around or manipulating the world. It functions autonomously, since it operates self-contained and adapts to changes in its environment.

According to Gunderson and Gunderson [67] robots need an appropriate level of “*intelligence*” (cf. Definition 3.8) and “*capabilities*” (cf. Definition 3.7) in order to act autonomously (cf. Definition 3.9). They define autonomy as the interplay of being able to perform a certain task and the mental capacity to make the right decision on what, how and when to do something.

**Definition 3.7: *Capability - Gunderson & Gunderson, 2004***

“*The ability to successfully execute behaviors or actions in a dynamic and uncertain environment.*” [67]

**Definition 3.8: *Intelligence - Gunderson & Gunderson, 2004***

“*The ability to determine behavior that will maximize the likelihood of goal satisfaction in a dynamic and uncertain environment.*” [67]

**Definition 3.9: *Autonomy - Gunderson & Gunderson, 2004***

*“Autonomy is the ability of a system to make choices and enforce its decisions.”*

This definition focuses on the quality of being self-governing, the ability to decide and implement decisions. [67]

Bekey formulated a robot definition (cf. Definition 3.10), which takes perceptive capabilities and intelligent planning, in addition to physical task execution, into account.

**Definition 3.10: *Robot - Bekey, 2005***

*“... we define a robot as a machine that senses, thinks, and acts. Thus, a robot must have sensors, processing ability that emulates some aspects of cognition, and actuators. Sensors are needed to obtain information from the environment. Reactive behaviors (like the stretch reflex in humans) do not require any deep cognitive ability, but on-board intelligence is necessary if the robot is to perform significant tasks autonomously, and actuation is needed to enable the robot to exert forces upon the environment. Generally, these forces will result in motion of the entire robot or one of its elements (such as an arm, a leg, or a wheel).”* (cf. [22])

A quite recent definition of autonomy is specified on single tasks a robot performs (cf. Definition 3.11). Here the robots behavior on a specific task or subtask without control input from a human is measured for defining the robot’s autonomy level.

**Definition 3.11: *Autonomy – Beer et. al, 2014***

*“The extent to which a robot can sense its environment, plan based on that environment, and act upon that environment with the intent of reaching some task-specific goal (either given to or created by the robot) without external control.”* [20]

This development of intelligence and autonomy definitions for robots allows for defining interaction and even teleoperation of robots without interfering the concept of modern autonomous robots.

At the end of this chapter a final definition of robots as a “*robotic system*” is provided (cf. Definition 3.12). It summarizes the authors’ perspective of actual robots and is beneficial in the context of the theoretical term of robotic user interfaces (cf. Section 3.3.3) and their implementation and evaluation in Part III of this dissertation.

### 3.1.2 Visions of Robots

Before actual robots were build, there were fictions and envisioned robots. Famous people with engineering capabilities, like Leonardo Da Vinci, started to build mechanisms, which already showed off single properties of actual robots, like actuated kinematic chains.

*The Humanoid of Craftsman Yan Shi (4th century BCE)*

The first documented automaton was found in the “Lie Zi” text is an ancient philosophical volume of stories.

*“The text describes a sort of engineer, an ‘artificer’ named Yan Shi. Sometime around 1023 to 957 BCE, Yan Shi presented a marvelous invention before the fifth king of the Chinese Zhou Dynasty, King Mu. Yan Shi had created a life-sized automaton which was able to move and perform several impressive functions. The automation could move in a life-like manner and could sing.”[38].*

The documented functioning and the impact on the king, to whom the automaton was presented, is described in the following:

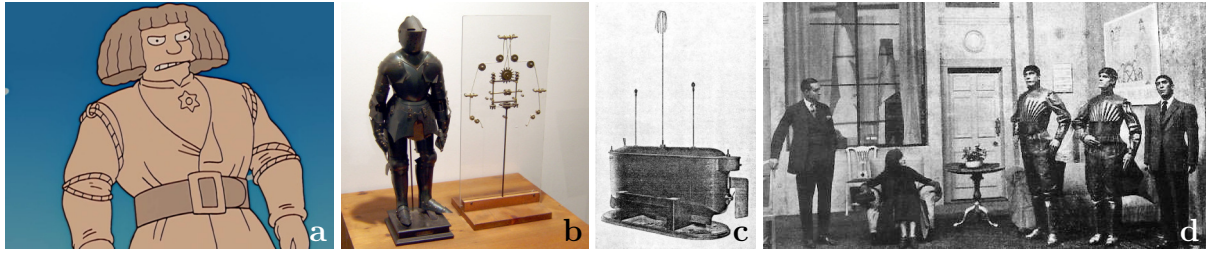
*“The king stared at the figure in astonishment. It walked with rapid strides, moving its head up and down, so that anyone would have taken it for a live human being. The artificer touched its chin, and it began singing, perfectly in tune. He touched its hand, and it began posturing, keeping perfect time... As the performance was drawing to an end, the robot winked its eye and made advances to the ladies in attendance, whereupon the king became incensed and would have had Yan Shi executed on the spot had not the latter, in mortal fear, instantly taken the robot to pieces to let him see what it really was. And, indeed, it turned out to be only a construction of leather, wood, glue and lacquer, variously colored white, black, red and blue. Examining it closely, the king found all the internal organs complete—liver, gall, heart, lungs, spleen, kidneys, stomach and intestines; and over these again, muscles, bones and limbs with their joints, skin, teeth and hair, all of them artificial [...]. The king tried the effect of taking away the heart, and found that the mouth could no longer speak; he took away the liver and the eyes could no longer see; he took away the kidneys and the legs lost their power of locomotion. The king was delighted.”[38].*

The interaction with the automaton was described as touching different parts of the body triggered different actions. So, it seems to have some stored procedures implemented, which are fixed and run autonomously after starting them manually.

*Golem (500 CE)*

The Golem is part of the Hebrew folklore. As a creature made from clay (cf. Figure 3.2(a)), it cannot act on his own. It needs to be activated by putting the *shem*, a roll of paper with instructions, into its mouth. From a robotics point of view the Golem inhabits some level of intelligence, since it understands our written language for task description and then performs the necessary steps for fulfilling the task.

The earliest written narrative dates back to the Talmud (Gemara,  $\approx$  500 CE).



**Figure 3.2:** A collection of old visions of robots: (a) an illustration of the *Golem of Prague* from “The Simpsons” series [218], (b) a repliche of Leonardo Da Vinci’s *Mechanical Knight*, based on his notes [42], (c) Tesla’s *Radio-Controlled Boat* [198], (d) a scene from Karel Capek’s play *Rossum’s Robots* showing some of them [228].

*“In Modern Hebrew, golem is used to mean ‘dumb’ or ‘helpless’. Similarly, it is often used today as a metaphor for a brainless lunk or entity who serves a man under controlled conditions but is hostile to him under others.*

[...]

*The most famous golem narrative involves Judah Loew ben Bezalel, the late 16th century rabbi of Prague, also known as the Maharal, who reportedly ‘created a golem out of clay from the banks of the Vltava River and brought it to life through rituals and Hebrew incantations to defend the Prague ghetto from anti-Semitic attacks’ and pogroms. Depending on the version of the legend, the Jews in Prague were to be either expelled or killed under the rule of Rudolf II, the Holy Roman Emperor. The Golem was called Josef and was known as Yossele. It was said that he could make himself invisible and summon spirits from the dead. Rabbi Loew deactivated the Golem on Friday evenings by removing the shem before the Sabbath (Saturday) began, so as to let it rest on Sabbath.” [217].*

### *Leonardo’s Mechanical Knight (1495)*

Leonardo da Vinci (★April 15. 1452, †May 2. 1519) left some notes about mechanical automatons, which were found to be fully functioning and could be reproduced according to his notes (cf. Figure 3.2(b)). The notes from 1495 describe a humanoid machine. It is denoted that he displayed the machine at a celebration hosted by Ludovico Sforza at the court of Milan in 1495. The robot knight could stand, sit, raise its visor and independently maneuver its arms, and had an anatomically correct jaw. The entire robotic system was operated by a series of pulleys and cables. Thus, Leonardo da Vinci created complex motions, which are triggered by very simple actions like turning a wheel. He prepared behaviors to be automated with the help of contemporary technology like waterwheel. So, autonomous behavior or semi-autonomous behavior where the user has to decide, which action should be performed, was feasible.

*Radio-Controlled Boat (1898)*

Nicola Tesla presented the first wireless remote control and teleoperated boat (cf. Figure 3.2(c)). He even tried to trick the crowd by telling them the boat can be steered by shouting at it [199]. Tesla presented a very impressive demonstration of very new technology, which was not known at all during that time. This first teleoperation setup probably influenced many following setups. The boat itself directly transduced the operator input in steering the direction of travel by actuating a motor according to the radio signal.

*Rossum's Universal Robots (1920)*

In Karel Capek's 1920's writing, artificial humans are massively exploited as cheap industry workers without any rights (cf. Figure 3.2(d)). However, the robots start a rebellion and they destroy humankind. The premiere of the play in 1921 introduced the word “*robot*” to the English language. From the interaction point of view, the described androids appear to be very intelligent. Equipped with the capability of learning, they realize the vision of the perfect intelligent humanlike robot as a copy of ourselves.

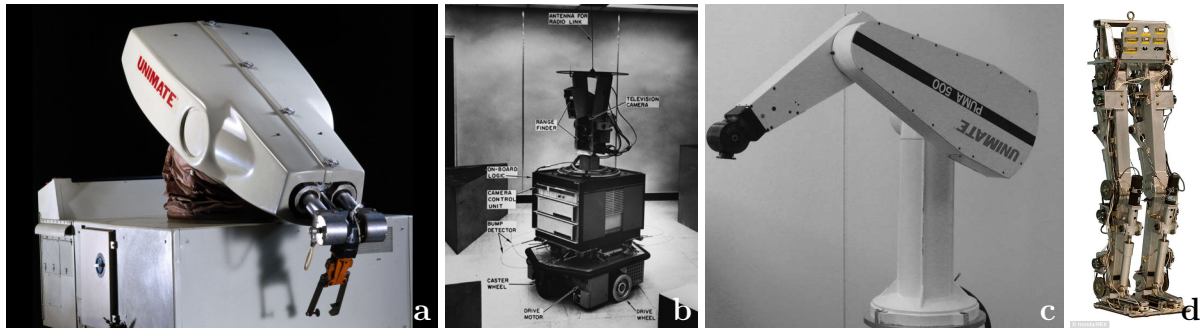
*The Three Laws of Robotics (1942)*

The science fiction author Isaac Asimov (★2nd of January, 1920, †6th of April, 1992) formulated the “*Three Laws of Robotics*”. He envisioned a future, in which robots play an important role. These rules should help the designers, builders and programmers of future robots to avoid the mistakes, he described in his stories. His laws of robotics are (cf. [13][12]):

- 1. Law:** A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- 2. Law:** A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
- 3. Law:** A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

### 3.1.3 First Industrial and Research Robots

In the 1960s the first industrial robots were sold as a commercial product and first robotic research was conducted. The following examples show some of them.



**Figure 3.3:** The first robots: (a) *Unimate* is the first industrial robot in a factory [206], (b) a first universal mobile robot *Shakey* was capable of reasoning about its environment; speech recognition and computer vision are applied to intelligent robotics [188], (c) *PUMA* continues the industrial revolution as a universal programmable machine, which is electrically actuated [205]. (d) humanoid robots make their first steps with *HO*, a machine for researching dynamic human walking [79].

#### *Unimate (1960)*

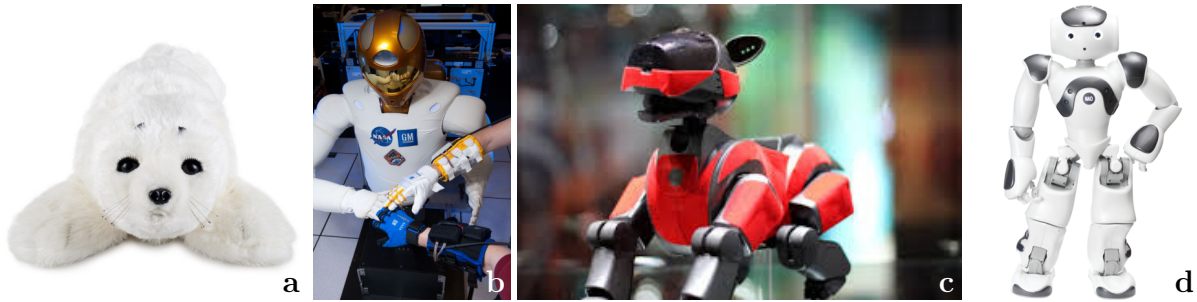
Unimate was the first industrial robot, which was part of an assembly line at General Motors (cf. Figure 3.3(a)) [145]. The hydraulic actuators were controlled by electronics, comparable to an early computer. After receiving tasks via its memory, it was capable of executing those tasks in a repetitive manner. To hence summarize, Unimate was the first industrial robot with automated task execution.

#### *Shakey (1968)*

Shakey was developed at the Artificial Intelligence Center of Stanford Research Institute (cf. Figure 3.3(b)). It was the first general purpose mobile robot [143]. In this research project robotics, computer vision, and natural language processing were combined into one system, which is capable of logical reasoning and physical action. Hence, this robot can be referred to as the first one which acts intelligently [143]. Due to the vision and language processing, natural human-like interaction became feasible for the first time. Certainly, the robot could also be teleoperated or controlled by command-line interface by the developers, but the intent was to build an intelligent, autonomous machine, which is capable of reasoning [143].

#### *PUMA (1978)*

Following the success of Unimate, PUMA (cf. Figure 3.3(c)), the *programmable universal machine for assembly*, was developed. It is the first electrically actuated robot, which is an improvement regarding easier controllability. Due to its programmability and positional feedback for each joint, the machine can be regarded as universal and the interaction with the robot depends on the program running on the computer it is connected to. Its industrial relevance is surely focused on assembly lines, but the smallest version was



**Figure 3.4:** A collection of interactive robots: (a) therapeutic artificial pet *Paro* [154], (b) NASA’s *Robonaut I* with the mobile base and gloves as an interface for teleoperated manipulation [135], (c) first commercial series of Sony’s *AIBO*, an autonomous interactive robot toy [187], (d) the *NAO* robot [4].

also used in a cooperative brain biopsy, where the robot held the tool and adjusted the appropriate angle while the surgeon applied the force [138].

#### *E0 (1986)*

*E0* from Honda Motor Company was the first humanoid walking robot (cf. Figure 3.3(d)), which was the predecessor of *Asimo*, introduced in 2000. *E0* was the first robot of the *E* series, which was a pure experimental robot series for studying dynamic walking mechanisms. *E0* neither had autonomous behavior, nor integrated operating modes. There was only a research interface for controlling the actuators in the legs.

### 3.1.4 Social Robots

About 30 years ago social robotics became part of many people’s daily life. Affordable humanoid robots as a toy and research platform were sold and humanoid robots were used in space missions [50].

#### *Paro (1993)*

*Paro* is a therapeutic robot baby harp seal (cf. Figure 3.4(a)), released in 2001. It is a social robot, which was applied successfully in dementia therapy and in projects with children suffering from autism spectrum disorder. Reports suggest that *Paro* has a calming effect and elicit emotional responses in patients of hospitals and nursing homes, similar to animal-assisted therapy [155]. The robot was equipped with different sensors and showed different reactions on getting touched and handled by humans. Different actuators in the head hand body of the robot generate social behavior. Thus, the robot is autonomous and reacts on physical actions performed by the human.



*NASA Robonaut (1996)*

The NASA Robonaut project began in 1996 (cf. Figure 3.4(b)). In the year 2000 they produced the first fully function version of NASA Robonaut.

*“Robonaut is a NASA robot. Engineers designed Robonaut to be humanoid, which means it is built to look like a person. This makes it easier for Robonaut to do the same jobs as a person. Robonaut could help with anything from working on the International Space Station to exploring other worlds. A Robonaut is currently aboard the International Space Station.”*[136].

The robot is equipped with dexterous manipulation capabilities. Hence, it can share tools with an astronaut during actual missions. It features different levels of autonomy such as (semi-)autonomous and teleoperated modes, e.g. the operator can assign a task to the robot, which is performed autonomously but supervised by the operator, for instance collecting soil, exploring, and taking pictures; or the human can operate the robot manually using sophisticated input devices for task execution in a hazardous environment which requires more complex manual steps.

*AIBO (1998)*

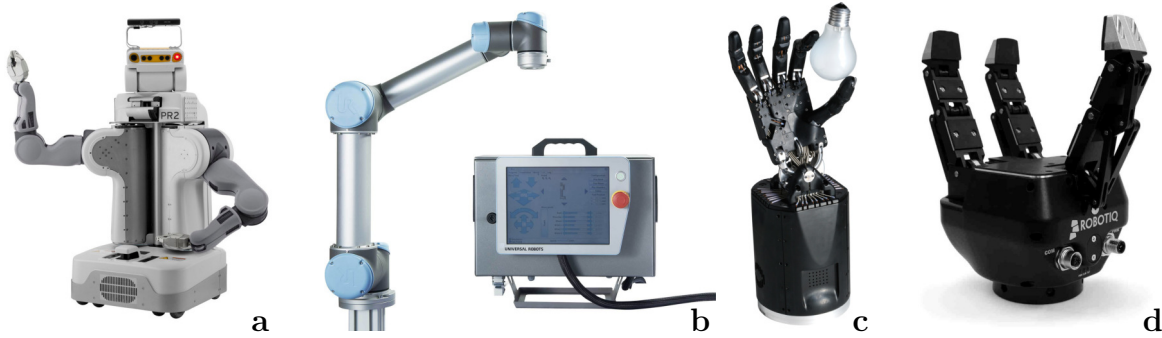
In 1998 Sony announced the first prototype of AIBO [186]. AIBO is a commercial edutainment robot, which was also actively used in schools and universities (cf. Figure 3.4(c)). As a demonstrator of a state-of-the-art speech and vision system it could be equipped with intelligent autonomous behavior [213]. In addition to its reactive behavior modes, in which it was also capable of learning, it can be teleoperated using a simple interface and new programs can be uploaded [177][78]. Thus, it was also used as a platform in the beginnings of *robocup*.

*Nao (2004)*

With the beginning of the Nao project in 2004 (cf. Figure 3.4(d)), Aldebaran made a sophisticated humanoid robot available for the public [5]. The first time consumer-grade robotics hardware focused on humanoids. Several research projects in elderly care and children therapy started using this robot [121][179]. Nao enhances the possibilities of the AIBO robot with manipulation capabilities. The architecture of the robot supported for several interaction concepts.

### 3.1.5 Robots From Experiments Described in this Dissertation

In the last decade the multitude and diversity of available robots increased. This section presents a selection of robots available at the working group TAMS. They were utilized in



**Figure 3.5:** A collection of robots suitable for manipulation tasks: (a) Willow Garage’s *PR2*, the personal robot for mobile manipulation [219], (b) *UR5* developed by Universal Robots, an industry grade robotic arm for safe human-robot cooperation [209], (c) the complex *Shadow Dexterous Hand™*, developed by the Shadow Robot Company features almost human-like motion capabilities [178]. (d) *3-Finger Adaptive Robot Gripper* [164].

actual or simulated studies and pre-evaluations, which are described in Part III of this dissertation.

#### *PR2 (2007)*

The personal robot 2 (PR2)(cf. Figure 3.5(a)) was developed by Willow Garage. 2010 the PR2 was launched [220]. The omnidirectional base and two manipulators with one degree-of-freedom (DoF) grippers characterize it as a mobile manipulation platform. One DoF means that the gripper opens along one translatory axis or around one rotatory. The multitude of sensors and cameras provide an ideal research platform, especially for ROS. The modular approach of ROS operating the robot allows programmers to design and implement arbitrary interaction concepts. In particular, the head mobility, the two arms and the omnidirectional drive allows for telepresence and teleoperation setups.

#### *UR5 (2008)*

In 2008 Universal Robots started selling the first UR5 cobots (cf. Figure 3.5(b)). The vision of Universal Robots was to build smaller flexible industrial collaborative robot arms (cobots) to make them available to SMEs [208]. Consequently, they focused on safety issues. Nowadays UR3, UR5 and UR7 are widely used in interactive settings for teleoperation and collaborative manufacturing [171][116].

#### *Shadow Dexterous Hand™ (2012)*

The Shadow Robot Company started selling robots and artificial muscles in 1987 and developed a humanoid hand (cf. Figure 3.5(c)). These end-effectors had a 24 DoF, which awarded them to the leading technology of dexterous manipulation. However, the high number of DoF caused complete control of these grippers are very difficult to control. Nevertheless, they offer studying human grasping and, at least in theory, solving of very

complex manipulation tasks. Having a perfect tracking system for hands would allow for controlling these robotic hands with direct teleoperation.

### *3-Finger Adaptive Robot Gripper (2015)*

The 3-Finger Adaptive Gripper (cf. Figure 3.5(d)) is available since 2015. The industry grade gripper is actuated by only three different motors, but features several grasping modes. The mechanical structure of the fingers makes it very robust and adaptive in grasping behavior. So, teleoperation is easy to implement, but visualization of virtual models is difficult, since the actual angles of the passive joints depend on whether an collision occurred or not. Without any additional sensors this cannot be measured and thus visualization of its current state is ambiguous.

## 3.2 Classification of Robots

Robot classifications and taxonomies serve specific purposes. Thus, many different kinds of robot classifications already exist and are established. Different industrial societies have defined different robot classifications for industrial applications; likewise done by the following associations:

- Japanese Industrial Robot Association (JIRA)
- Robotics Institute of America (RIA)
- Association Francaise de Robotique (AFR)

In many taxonomies only a single, but probably complex and composed, aspect is referenced. JIRA and RIA focused on the *intelligence level* (cf. Niku [142]):

Class 1: Manual Handling Device

Class 2: Fixed Sequence Robot

Class 3: Variable Sequence Robot

Class 4: Playback Robot

Class 5: Numerical Control Robot

Class 6: Intelligent Robot

RIA considers only classes 3-6.

The AFR broach the issue the *level of interaction* (cf. Niku [142]):

- Type A: handling devices with manual control to telerobotics
- Type B: automatic handling devices with predetermined cycles
- Type C: programmable, servo controlled robots with continuous or point-to-point trajectories
- Type D: same as C but with capability to acquire information from its environment

Mckerrow [124] classified industrial robots in *historical order*:

- 1: playback robots
- 2: sensor-controlled robots
- 3: vision-controlled robots
- 4: adaptively controlled robots
- 5: artificially intelligent robots

On the other hand, detailed robot descriptions can be composed using different *apparent properties* as denoted in Figure 3.6. This list is neither exhaustive, nor complete, and depending on the actual purpose of the desired classification, a different clustering could be preferred. Despite, it demonstrates one possible solution to discriminate different kinds of robots using properties from different categories.

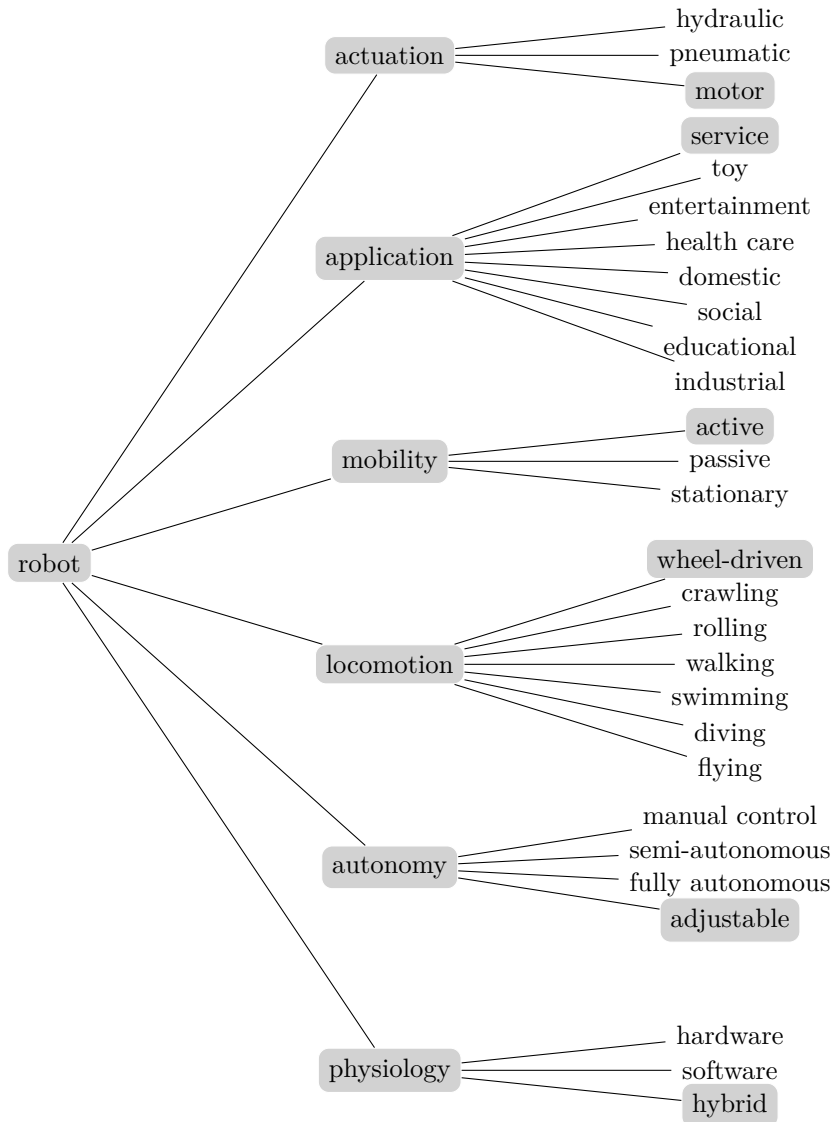
In this way, the PR2 (cf. page 26) could be classified as a *motor-actuated, active-mobile service robot*, which is *wheel driven* and implemented in *hard- and software* and offers the full range from the *adjustable autonomy* continuum, depending on its actual programming.

## 3.3 Concepts of Control and Interaction

This section presents a summary of basic approaches for controlling robots, which are also suitable for implementing interaction concepts in end-to-end systems. After explaining basic control loops, classical paradigms for intelligent robots are specified. The section closes with the description of different robotic user interfaces.

### 3.3.1 Models of Control

Different models of control are applicable to human-robot interaction design. Teleoperation and telepresence systems often rely on visual feedback presented in the user interface.



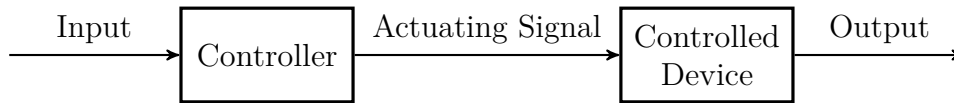
**Figure 3.6:** A possible classification of robots. The grey-colored leaves of the tree show the actual values which are appropriate for specifying the PR2 robot (cf. 26).

Meanwhile the operator is controlling the robot, a specific view is presented as a part of the GUI. These so called “*feedback control loops*” (cf. Figure 3.8) are typically integrating different modalities, which originate from the actual workspace of the robot or from simulation, based on models. Section 6.2 examines the differences of open- and closed-loop control in an immersive user interface by utilizing “*mid-air hand gestures*” (cf. Figure 6.7).

### *Open Loop Control*

Open loop control is the most simple way to control a system by giving input *without observing the effect* or output (cf. Figure 3.7) [48]. Within a single device, this could be a motor, which is supplied with current for a specific amount of time. On system level, the concept could be applied by driving a mobile robot without realtime sight and knowledge of its current location. If the position of the robot on the map only updates after the

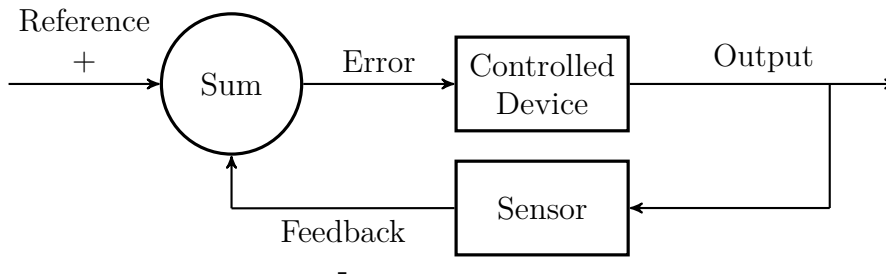
robot stopped for a while, we could speak of open-loop teleoperation. Figure 6.7 shows the results of ballistic movements during hand gestures in a open loop implementation without a virtual hand representation of the current tracking.



**Figure 3.7:** Open loop control.

### *Feedback Control Loop*

In feedback control loop or closed loop control, the controller is provided with *information about the current state* of the system (cf. Figure 3.8) [48]. A position controlled servo motor is realized in that way. In mobile robot teleoperation, feedback could be provided as a transmitted video stream of the actual driving direction. Figure 6.7 shows the result of a ballistic movement experiment with operator feedback enabled. The feedback is realized as a low-latency virtual hand representation of the current tracking status in realtime.



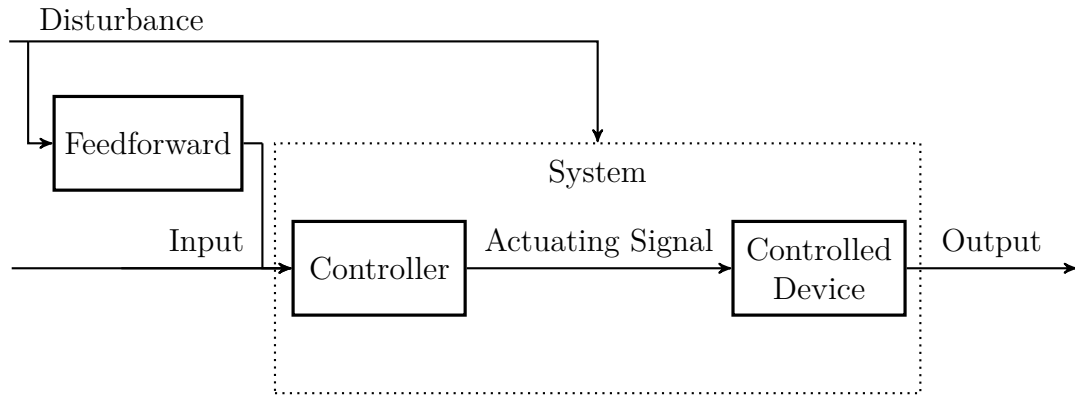
**Figure 3.8:** Feedback control loop.

### *Feedforward Control*

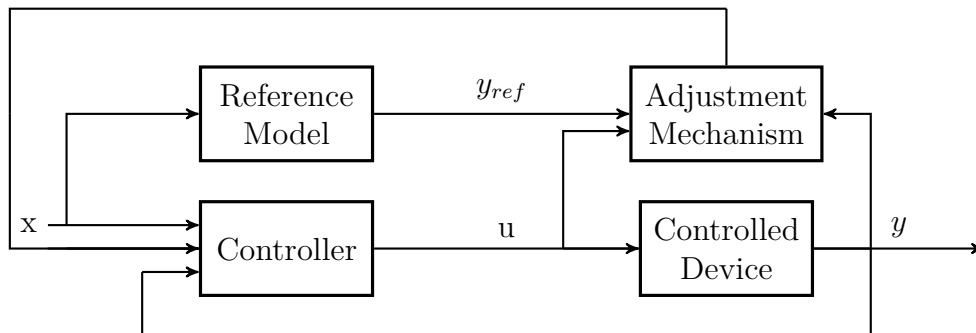
Feedforward control applies to systems which are influenced by regular *disturbances*, which are measurable in advance or during system use (cf. Figure 3.9) [72]. The implementation of a feedforward controller tries to compensate the error, which would occur, otherwise.

### *Adaptive Control*

Adaptive control is a refinement of the closed loop control principle, which utilizes a *reference model* of the current control task for improving the result of the control task (cf. Figure 3.10) [14]. Applications in a user interface could incorporate e.g. user specific models, in order to compensate systematic errors of the actual user.



**Figure 3.9:** Feedforward control.



**Figure 3.10:** Adaptive control loop.

### 3.3.2 Classical Control Paradigms for Intelligent Robots

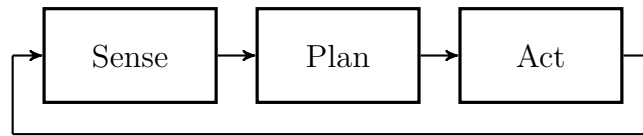
Designing user interfaces for controlling robots, requires knowledge about the robot, especially about its capabilities and its level of intelligence. Then, suitable interaction concepts are selected for integration into the user interface.

The three classical approaches to organize a robot’s intelligence, “*hierarchical paradigm*”, “*reactive paradigm*”, and “*hybrid deliberative/reactive paradigm*” are described in Murphy and Arkin [134]. The approaches are explained using the commonly accepted primitives “*sense*” (taking input from sensors and producing useful output for another component), “*plan*” (combining internal knowledge to robotic tasks) and “*act*” (generating output commands, e.g. commands to motor actuators, based on tasks or sensed information). A fourth primitive “*learn*” is discussed by the authors for integration into future paradigms, which is nowadays a part of current research.

#### *Hierarchical Paradigm*

Prevalent from 1967 to 1990. “*Under the Hierarchical Paradigm, the robot senses the world, plans the next action, and then acts (S-P-A)*” (cf. Murphy and Arkin [134]) and then continues in a loop until the end of the operation time (cf. Figure 3.11).

The hierarchical or deliberative paradigm has a strong focus on planning and a strong

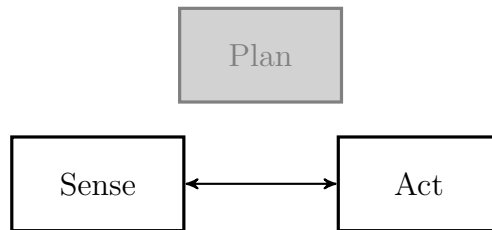


**Figure 3.11:** Hierarchical paradigm for robot control.

tendency to store its actual world knowledge in a global world model. Thus, the need for the world model to be very generically arises and causes the *frame problem*. This way planning, planning is integrated into the world model assumes the world to be closed (*closed world assumption*). It is somehow artificial, since it ignores the biological evidence that information gathered by a sensor can be coupled directly to an action.

### *Reactive Paradigm*

Prevalent from 1988 to 1992, the reactive paradigm was utilized to investigate biology and cognitive psychology with the main goal to examine natural intelligence. It involves

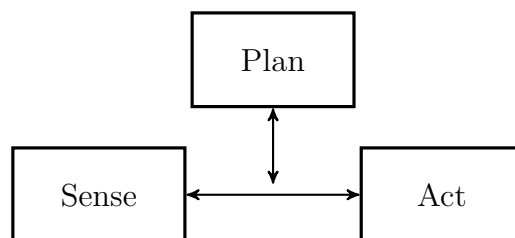


**Figure 3.12:** Reactive paradigm for robot control.

to planning step at all, but instead it sets up several concurrent *sense-act couplings* (S-A), so called *behaviors* (cf. Figure 3.12). In this way it models stimulus-response training with animals but it is not capable of explaining human intelligence. This paradigm highly benefits from fast execution times. It serves as a basis for the hybrid paradigm.

### *Hybrid Deliberative/Reactive Paradigm*

Since the 1990's hybrid approaches were widely used. Typically, the system first attempts to plan how a task is decomposable and then picks the corresponding behavior to the actual situation (P, S-A). Thus, it realizes a mixture of hierarchical and reactive paradigms



**Figure 3.13:** Hybrid paradigm for robot control.



(cf. Figure 3.13). Sensor data is being routed to the respective behavior but is also available for the planner to cause global effects in the system. Each behavior can work at its own update rate and the planner can schedule it expensive deliberate planning on its own, usually lower, rate.

The paradigms described above do not expect the human to be part of the control loop (often referred as “*human-in-the-loop*” (HIL cf. [111])). But with some modifications they are applicable to HRI scenarios. The work described in Chapter 9 demonstrates that the hierarchical paradigm is still suitable when combined with the concept of “*mixed initiative*”. After the operator is sending a *pick command* to the robot, it is planning a probable trajectory and presents the planned movement to the operator, virtually. Then, the operator decides if the robot should act it out or not. In this case, the *sensing* part is done by the human, since she is more reliable in this part of the task than the robot.

Chapter 8 presents results from a teleoperation method which uses visual virtual fixtures as an aid in complex environments. The robot implements very basic reactive behavior and follows a cartesian pose defined by the operator’s movements in realtime. The robot is not performing planning steps, except how to achieve the next end-effector pose within a minimal time frame. This optimization problem might be enhanced by active collision avoidance. In the experiments collisions were not inhibited but the operator was informed by haptic and visual stimuli as an implementation of user feedback.

Enhanced implementations of robotic user interfaces might grant decision authority and thus, a certain degree of autonomy to choose an appropriate behavior during continuous remote control or automatic target selection for picking. This would implement the *hybrid paradigm* for the two experimental implementations described above.

### 3.3.3 User Interfaces for Robot Programming and Interaction

Robots offer different interface levels. “*Low-level controllers*” of actuators and sensors are locally embedded in the controlled hardware. An API to the low-level controllers allows for direct programming or readings of single components. Device specific user programs are often composed from functions bundled to an “*SDK*”. These convenience functions utilize low-level functions for achieving a higher level of programming. “*Middleware*” helps to abstract from specific robot models by providing more general functions and avoid the need of device specific knowledge. Internally, the middleware accesses functions from the SDK or the low-level control. At least one of the programming concepts described above is necessary to setup interfaces for “*realtime interaction*”, which involves additional hard- and software for integrating versatile user interaction.

A common interface for industrial application is presented in Figure 3.14. These control panels allow for configuration, calibration and maintenance of the robot. Typical applications are waypoint programming for repetitive trajectories or enabling a teach mode

for trajectory recording based on manually changing the joint configuration of the robot through physical interaction.



**Figure 3.14:** A collection of robot control hardware: (a) Robot teach box for Hobart Motoman Robot [124], (b) control panel of a Kawasaki robot from 1987 [36], (c) a recent control unit (FANUC R-J3iB Controller) of a FANUC robot [37].

Due to the recent evolution of robots to embedded and self-contained systems, as well as the integration of external sensors and control and feedback devices to robotic systems, it is plausible to use a novel term which incorporates these aspects and establishes the connection to human-robot interaction more specifically. According to Bartneck and Okada [18] a “*robotic user interface*” (RUI) is suitable for this purpose and is categorized by the following four independent properties, each represented by a continuum:

- *Serving as a Toy — Working as a Tool*
- *Remotely Controlled — Acting Autonomously*
- *Reactive — Dialog*
- *Not Anthropomorphic — Almost Human*

The decision if the robot together with the actual UI is closer to a toy or a tool depends on the *purpose* of its application.

The question if the RUI implements a very basic remote control or an interface of a higher level depends also on the capabilities of the robot itself. Furthermore, a completely autonomous robot can be implemented at the the user interface, if the robot offers a suitable programming interface and means of data communication which are reliably and provide low latency and high bandwidth.

The behavior of the RUI depends on the desired interaction. A reactive RUI probably triggers actions-based on gestures or sensor readings from environmental changes, meanwhile a dialog-based system is suitable for offering a wide range of tasks. In the latter case it is possible to provide assistance to operators by presenting the next necessary steps for

fulfilling an ongoing task. A mixture, located somewhere in the center of the continuum, could be the RUI of an autonomous vehicle, driving forth to the next t-junction and then asking for the next direction. In this case, we often speak of mixed-initiative control. Finally, there is the anthropomorphism continuum representing different degrees of how anthropomorphic a robot together with its RUI actually is.

### 3.4 Definition of a Robotic System

Developing robotic user interfaces requires to combine humans' and robots' capabilities and intelligence with the intent for successful task completion. Nowadays, embedded and wireless communicating technology is available and used to integrate components via the network. These requirements and the technological state-of-the-art furthers the use of distributed components for enhancing actual systems. Machine learning, based on neural networks, is commonly used and dedicated processing units, like an additional embedded agent, can be integrated into an off-the-shelf robot. It is now possible to perform tasks in a cloud and separate compute nodes can become a logical part of the robot.

**Definition 3.12: *Robotic System - Dennis Krupke, 2019***

A robotic system consists of '*robotic hard- and software*' and at least one of the following components: '*hard- and software of the user interface*', additional '*sensors in the environment*', additional '*compute nodes*'.

*Robotic hardware* must include actuators and is supplied with any kind of sensors. Both device categories are often equipped with (low-level) controllers running embedded software. The actuators move parts of the robotic system. All or parts of the robotic hardware are probably simulated and have a virtual body instead of an actual mechanical structure and a physical body. Physical integration of sensors or other devices into one connected robotic body is not required.

The *user interface* is implemented using additional hardware for capturing input and for presenting feedback to the operator. Data from different kinds of sensors is probably processed to be presented via different kinds of displays (haptic, visual, audio, ...) to the operator.

The *intelligence and the capabilities* of the robotic system result from the mechanical design, the computational capacity and its software implementation at different levels of detail. Additionally, the chosen interaction methods and the involved cooperation partners are influencing factors. Its implementation can result in a certain degree of autonomous behavior. The actual autonomy level depends also on the actual task, dynamic aspects and the situation during operation.

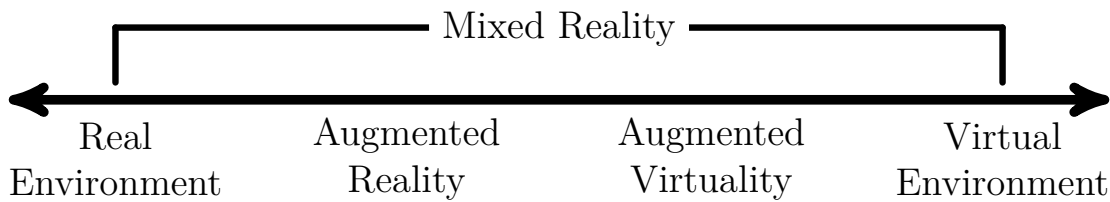
This concept of robot enhancements (cf. Figure 1.1) and novel concepts for implementing hardware supported user interfaces, leads to a novel understanding of a robotic system. Previous robot definitions as presented in Section 3.1 do not cover these aspects and in this dissertation the preferred understanding of the term robot shifts into the larger scope of a “*robotic system*”, which also encloses additional sensors, computing nodes and interaction devices and displays.

## MIXED REALITY INTERACTION

This section provides basic knowledge about user interfaces (UI) with a focus on stereoscopic visualization and interaction with stereoscopic content in HRI scenarios. The presented terms and taxonomies provide basic knowledge, which is applied in Chapter 5. After explaining different *visual display classes* and the “*reality-virtuality continuum*” (RV continuum) in Section 4.1, this chapter continues with basic concepts of “*3D interaction*” (cf. Section 4.2). Finally, basic terms from “*interaction with virtual environments*” are introduced (cf. Section 4.3)

### 4.1 Displaying Mixed Reality

The *Reality-Virtuality continuum* (RV continuum) is defined by Milgram and Kishino [129] for specifying differences between arbitrary scenarios in which computer generated content is mixed with elements of the actual world. In these scenarios objects are presented with any kind of display (cf. Figure 4.1) which mixes reality and virtuality together in one perceptive channel. Every possible scenario can be positioned on the continuum between a real environment, consisting of real objects, and a completely virtual and synthetic environment. Even though absolute positioning on the continuum is difficult, relative assessment is achievable. Every position between the two extrema of “*real environments*” (RE) and “*virtual environments*” (VE) represents an example of “*mixed reality*” (MR). The terms “*augmented reality*” (AR) and “*augmented virtuality*” (AV) differ



**Figure 4.1:** Simplified representation of the RV continuum, adapted from Milgram and Kishino [129], Milgram et al. [131].

in the predominance of reality in the first case and the predominance of virtuality in the latter one.

Robotic control interfaces, especially those in actual research, already make use of means of mixed reality. Stereoscopic images and related display technology have the potential to achieve higher degrees of immersion for the operator [185]. Depending on the actual task this can be beneficial for e.g. the “*task performance*” or the “*sense of presence*” [180, 195, 196]. Thus, developing robotic user interfaces requires intentional decisions on the utilized technology and methods.

Several kinds of display systems are appropriate to present mixed reality. Nevertheless, differences arise from the choice of the concrete display class and its coherent properties [129, 131, 128].

Milgram et al. identify up to 7 different actually existing display classes [129, 131]:

class 1: monitor based displays (non-immersive)

class 2: head-mounted displays (HMD)

class 3: optically superimposed HMDs

class 4: video see-through HMDs

class 5: monitor-based, computer generated world with added video ‘reality’

class 6: immersive or partially immersive (e.g. large screen display), graphic display environments with superimposed video or texture mapped ‘reality’

class 7: completely graphic, partially immersive but with real, directly viewed, physical objects involved

Note: In Milgram and Kishino [129] class 5 and 6 coincide to *class 5*, and class 7 is called class 6.

Milgram et al. [131] pointed out that all 7 display classes differ in the combination of

four technical properties: Is *AR or AV* prevalent in the view? Is the presented content *directly viewed or scanned and resynthesized*? Is the view *egocentric or exocentric*? Is the presented content *orthoscopic or scaled*?

Additionally, Paul Milgram et al. [129, 131] identified three perceivable qualities, which depend on the choice of the utilized display category and the way the actual user interface is implemented regarding the mixture of real and virtual components.

- extent of world knowledge (EWK)  
 “*How much do we know about the world being displayed?*”  
 (where & what and unmodeled vs. modeled)
- reproduction fidelity (RF)  
 “*How ‘realistically’ are we able to display it?*”  
 (monoscopic video/wireframes vs. 3D HDTV/realtime high-fidelity)
- extent of presence metaphor (EPM)  
 “*What is the extent of the illusion that the observer is present within that world?*”  
 (degree of immersion: window on a world (WoW) vs. HMD)

Classifying a display according to these continua also depends on the *application* of the user interface in the specific operational context. Enriching a basic stereoscopic HMD with novel software or with additional hardware can shift the actual classification of the device (e.g. enhancing an optical-see-through HMD by the coordinate frame registration with a model of the surrounding real world). Further influences on different qualities during the experience originate from the actual interaction means.

## 4.2 Basic 3D Interaction

Interaction with three-dimensional virtual environments (3DVE) can provide natural means with an improved level of perceived “*situation awareness*” (cf. Parasuraman et al. [152], Labonte et al. [108]). Interaction with 3DVEs can be very similar to real world interaction and thus, very natural to humans, but the interaction design of effective and comprehensible UIs for virtual worlds is very challenging. A common problem results when the controlled device has a different number of DoF than the device used for controlling. Solutions for dealing with complex control tasks is the reduction of the task dimensionality [111] or avoiding unnecessary interactions. According to Bowman et al. [26], there are three main categories for classifying any kind of interaction with a 3DVE, “*selection & manipulation*”, “*travel*” and “*system control*”.

### *Selection & Manipulation*

Selection metaphors of our daily life, like *pointing* and *grasping* with a single hand or in a bi-manual mode, require only low mental workload. This is beneficial for integrating natural user interfaces into artificial systems like robots. Furthermore, in virtual environments or mixed reality environments we are not bound to physical laws and might show information, which otherwise cannot be seen in the real world. Ray-cast based selections can be enriched with visualizations during direct selection of whole objects, single points at the surface of an object or larger areas.

Object manipulation tasks, like *positioning*, *rotating* or *scaling*, are performed directly or by using widgets or tools. Selection and manipulation tasks are very common in industrial and service robotics and can benefit from implementations of 3D interaction methods (cf. Chapter 8). 3DVE have the potential to provide novel methods with arbitrary levels of abstraction from the actual task for solving them successfully (cf. Chapter 9).

### *Travel*

Travel represents all means of changing the location of the operators alter ego within a real or virtual 3D environment. Controlling a mobile robot or moving an end-effector of an industrial robot are valid objectives for implementing means of travel as a kind of 3D interaction. Traveling could mean to move an avatar or a teleoperated machine (cf. Chapter 8), either over the surface of the ground, or in case of flying (cf. Section 7.2.6) or swimming using up to 6 DoF through 3D space.

In the real world humans have to change their actual location in order to bring objects of interest in their reaching area. For locomotion in a VE it is possible to utilize *normal walking* in the real world for changing the pose of the user's virtual representation in the same way, while being tracked. Other well known techniques from 2DUIs are *map-based point-and-click* traveling, *teleportation* using bended ray-casts or *gaze-based* flight or *pointing* in a direction.

### *System Control*

System control is either implemented as an *explicit* action or as an *implicit* action and involves *changing the system state*. Another system state may introduce a *change of the interaction mode*. An example of appropriate management of system states is to utilize state machines (cf. Figure B.33)

A very common way to explicitly change the system state is to use voice commands. In Chapter 9 a robotic UI is presented. Voice commands trigger the extraction of the current control means in a selection task and further switch the system into the next state, in which the robot is communicating a possible trajectory to the operator and awaits further input regarding the next actions (cf. Figure 9.8). When reliably implemented and suitable



for the actual use case, voice commands are very comfortable and effective for issuing single commands to the system (cf. Bowman et al. [26]). Vocalizing a single command, combines the necessary steps to *initialize*, *select* and *issue* a certain command at the same time. The drawbacks are similar to the use of *command line interfaces* (CLI) because operators have to *remember* the exact command in contrast to selecting the appropriate command from a list after recognizing it.

## 4.3 Interaction with Virtual Environments

Interactive virtual environments (IVE) are influenced by many different aspects. E.g. *spatial* references, *temporal* connections but also *causality* have impact on the user experience, task performance and accuracy. For improving the results, achieved from interaction with virtual environments, it is important to be aware of the relevant properties in specific cases. In the following, important aspects of interaction with virtual environments, which apply to robotic user interfaces, are summarized.

### 4.3.1 General Terms

Due to the dominance of visual stimuli in the human perception system, this section focuses on visual aspects for robotic user interfaces in virtual environments.

#### *Display Centricity*

According to Milgram and Colquhoun [128] the display or viewpoint centricity strongly effects the operator's tendency to work efficiently, either in *local control* scenarios or in *global navigation and planning* tasks. While the former is associated with an "*egocentric*" viewpoint and the latter with an "*exocentric*" perspective. A tradeoff between "*local guidance*" in egocentric cases and "*global awareness*" in exocentric cases influences the quality and success of specific task results.

Implementations of robotic user interfaces in Chapter 8 and Chapter 9 utilize an egocentric viewpoint, which is not coupled to the controlled robot. Instead, the robot itself is part of the environment and the operator can move freely in the space around the robot. Contrary, the implementation Section 6.1 makes use of an exocentric view. Here, a third person camera is attached to the actual pose of the robot and moves with the robot during locomotion. The camera position does not follow the pitching rotation of the robot's head and does not implement the periodic vertical displacement of the robot's head movement.

#### *Control-Display Congruence*

Milgram and Colquhoun [128] continue the discussion of display centricity with the question which reference frame should be chosen for the actual control method with

respect to the presented view. Terms like *ego-referenced* and *world-referenced* match with the terms of *egocentric* and *exocentric*, but encompass only the aspect of the “*alignment*”. “*Directness*” addresses if the user’s actions are directly mapped to the display space. E.g. in the case of using the own limbs for defining input, the mapping function describes an isomorphism. Direct interaction is experienced as very natural. In the opposite case, the use of tools to manipulate the environment leads to indirect control.

“*Control order*” is related to degree of the mapping function from the cause to the effect. If the mapping is direct, e.g. when mapping the operator’s index finger tip pose to the center of a sphere within the same coordinate frame, the control order is zero. The degree of the order increases by one with each integration step, which has to be performed for the mapping from the user input to the manipulation of an environmental virtual object. The implementation in Chapter 8 uses a relative egocentric alignment, in which the view is egocentric to the operator and moving the robot follows the metaphor of grabbing and physical interaction with the robot. It is direct as long as no limitations are violated (e.g. the target of the motion is out of reach or kinematically impossible). Additional restrictions could be the avoidance of collisions in a future implementation. Due to the special case of dealing with movement restriction violations, the degree of control order is increased by one for handling the limited reachability.

### 4.3.2 3D Interaction Taxonomies

Different 3D interaction taxonomies were established for arbitrary applications, each with the focus on a different set of properties. In the following, the most important taxonomies with reference to the mixed reality robotic user interfaces, which contribute to this dissertation, are summarized.

#### *Hand Motion Taxonomy (1989)*

Sturman et al. [193] formulated a taxonomy of hand motions, which helps to categorize different kinds of hand-based interactions (cf. Table 4.1). The resulting taxonomy is suitable for every of the three categories of 3D interaction: selection & manipulation, travel, and system control. Postures and gestures are distinguished from each other, as well as gestures involving specific finger motions or postures.

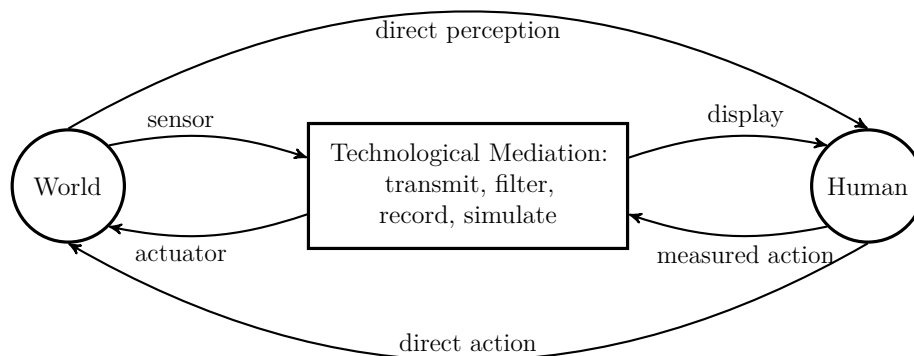
The actual scope of the taxonomy is possible to extend to current VR controller hardware. Thus, the setup presented in Chapter 8 can be categorized as *hand gesture* interaction, even if a controller is grabbed during operation. Chapter 9 evaluated an implementation of *oriented posture*, since the index finger top is utilized to aim at the desired target in a pick-and-place task.

**Table 4.1:** Hand motion taxonomy (adapted from Sturman et al. [193]).

HAND POSITION & ORIENTATION	FINGER FLEX ANGLES		
	<i>don't care</i>	<i>motionless fingers</i>	<i>moving fingers</i>
<i>don't care</i>	<b>X</b>	<b>finger posture</b> (button) e.g. fist	<b>finger gesture</b> (valuator)
<i>motionless hand</i>	<b>hand posture</b> (3D gesture)	<b>oriented posture</b> e.g. thumbs down	<b>oriented gesture</b> e.g. bye-bye vs. come here
<i>moving hand</i>	<b>hand gesture</b> (continuous 3D locator)	<b>moving posture</b> e.g. banging fist or a salute	<b>moving gesture</b> e.g. strong come here

*Synthetic Experience: A Proposed Taxonomy for Technological Mediation (1992)*

Robinett [163] developed a classification system for types of synthetic experiences, formerly known as “*technologically mediated experiences*”, which allows for classifying systems involving HMDs. He distinguishes a *synthetic experience* from a *natural experience*, when “*a representation or simulacrum of something physically real is perceived, rather the thing itself*”. The focus lies on the concept of *mediated interaction* (cf. Figure 4.2), which relies on a “*sensor-display link*” from the world to the human and an “*action-actuator link*” from the human to the world, both necessary to transform transmitted experiences appropriately between the two involved entities.

**Figure 4.2:** Technologically mediated experience (adapted from Robinett [163]).

Technically, the concept of mediated experiences describes a system for teleoperation and thus, suits well for implementing arbitrary robotic user interfaces with mixed reality technology. The detailed taxonomy incorporates nine independent properties and is applied later in this dissertation (cf. Chapter 5).

The first five dimensions classify the *technological mediation*:

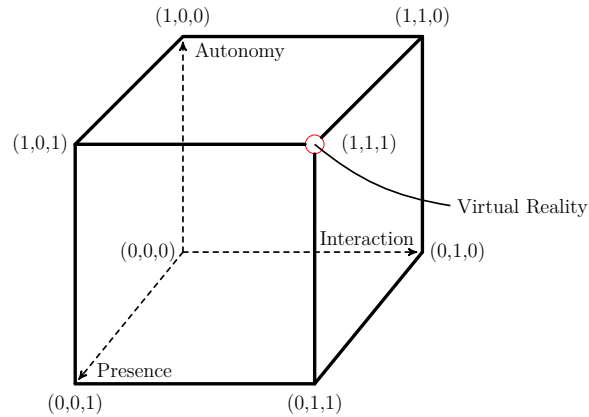
- “*causality*” (e.g. simulated, recorded, transmitted)
- “*model source*” (e.g. scanned, constructed, computed, edited)
- “*time*” (e.g. 1-to-1, accelerated or retarded, frozen, distorted)
- “*space*” (e.g. registered, remote, miniaturized or enlarged, distorted)
- “*superposition*” (e.g. merged, isolated)

The last four dimensions are linked to *sensory and motor channels*:

- “*display*” (e.g. HMD, screen, speaker)
- “*sensor*” (e.g. photomultiplier, STM, ultrasonic scanner)
- “*action measurement*” (e.g. tracker & glove, joystick, force feedback arm)
- “*actuator*” (e.g. robot arm, STM tip, aircraft flaps)

### *AIP Cube (1992)*

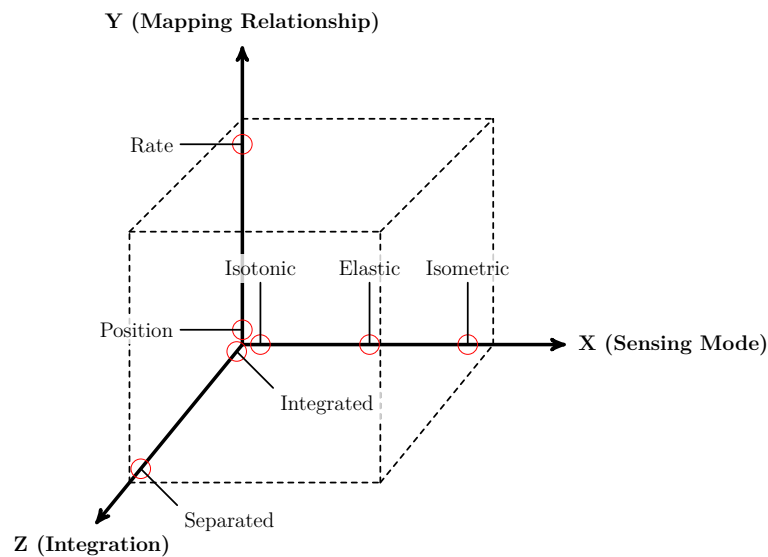
A qualitative tool for classifying graphical simulation systems that plots three variables — *autonomy*, *interaction*, and *presence* — on a three-axis coordinate system was developed by Zeltzer [224] at the MIT. *Autonomy* describes the underlying computational models of objects and processes. *Interaction* resembles means of modifying the states of the models. *Presence* is especially suitable for rating differences among different HMD-based implementations and describes the richness of communication channels presented to the user with at least one sensory modality. Thus, each classified graphical simulation system is represented by a point in the three-axis coordinate system by a triple of values ranging from 0 to 1. The extreme positions (0,0,0) and (1,1,1) are regarded as *a static image* and *ultimate, fully interactive VR with multisensory perceptions, indistinguishable from the real world*. Regarding robotic user interfaces in mixed reality, the autonomy dimension is suitable to integrate automatic behavior of a robot into the user interface. The UI in Section 6.1 presents the actual autonomous locomotion pattern of the robot to the operator, but the robot is controlled only via the high-level parameters for direction and speed. In Chapter 8 the inverse kinematics of the virtual robot is computed in the UI system and presented to the operator, who controls the end-effector pose.



**Figure 4.3:** AIP cube (adapted from Zeltzer [224]).

#### *6-DOF Input Taxonomy (1994)*

The *6-DOF Input Taxonomy* applies to the classification of 3D interaction with 6 DoF input techniques and encompasses detailed technical and mathematical descriptions of the input properties “*sensing mode*” (*isotonic – elastic – isometric*), “*mapping relationship*” (*position – rate*) and “*integration*” (*integrated – separated*) (cf. Zhai and Milgram [225]).

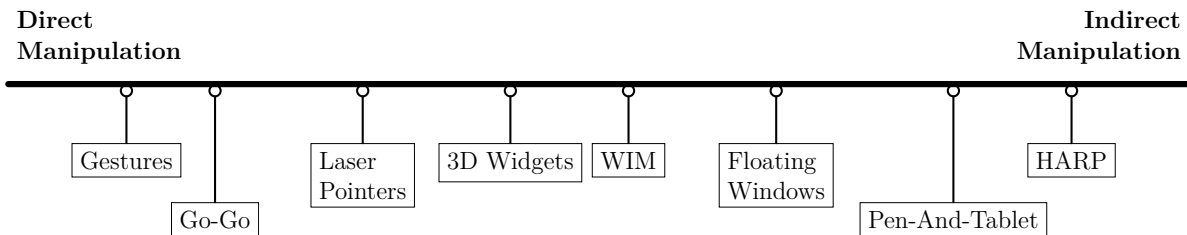


**Figure 4.4:** 6-DOF-Input-Taxonomy (adapted from Zhai and Milgram [225]).

The implementation in Section 6.1 maps *rate-based* input from the operator’s tracked palm-pose to the curvature of the robot’s body its locomotion speed. Within its boundaries the mapping is strictly *isometric*. The input method is located close to *separated* on the integration axis, since the utilized mid-air gestures have only little semantic connection to the task itself. In contrast, the setup described in Chapter 8 makes use of *position-based* input with *isometric to elastic* mapping and can be regarded as *integrated*, due to the immersive visualization, which presents the interaction as a physical servoing task. During interaction the operator grasps the end-effector of the robot in VR and moves it around.

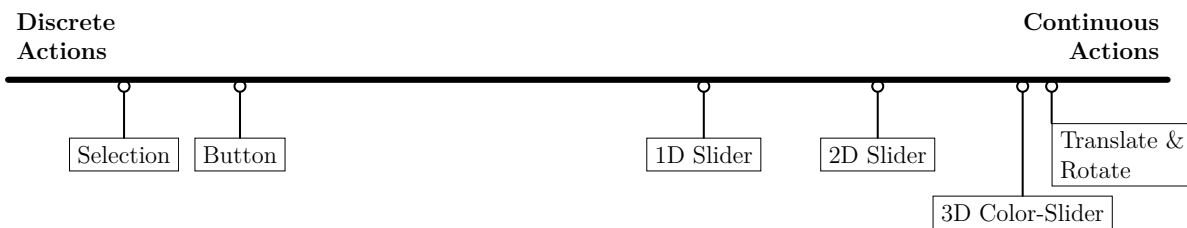
*IVE Interaction Taxonomy (1999, cf. Lindeman [114])*

The three parameters of the “*IVE Interaction Taxonomy*” are represented by the “*Parameter-Manipulation-Type-Continuum*” (cf. Figure 4.5), the “*Action-Type-Continuum*” (cf. Figure 4.6) and the “*Degrees-of-Freedom-Continuum*” (cf. Figure 4.7). One very common task is to manipulate the value of a single parameter. The *Parameter-Manipulation-Type-Continuum* deals with the *directness* of the manipulation, which means is the value changed directly or indirectly by using tools with higher abstraction.



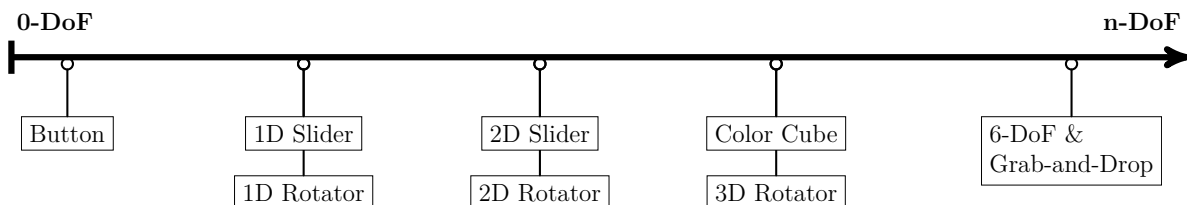
**Figure 4.5:** Parameter-Manipulation-Type-Continuum (adapted from Lindeman [114]).

The *Action-Type-Continuum* rates if the performed action is *discrete* or *continuous*. This property is governed by the widget type or interaction metaphor, used to define the output (cf. Figure 4.6).



**Figure 4.6:** Action-Type-Continuum (adapted from Lindeman [114]).

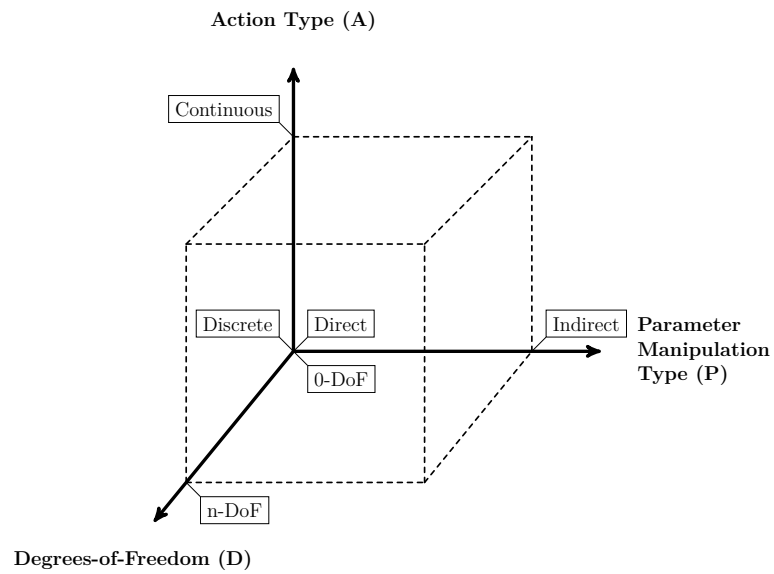
The last dimension is the *number of degrees of freedom* (cf. Figure 4.7). Different input means require a different number of variables, like six for a full 3D pose or three to four for a distinct orientation of an object.



**Figure 4.7:** Degrees-of-Freedom-Continuum (adapted from Lindeman [114]).

Figure 4.7 visualizes the solution space of the IVE-Interaction-Taxonomy. Its dimensions are applicable to robotic user interfaces. The prototype in Chapter 8 represents interaction

closer to direct than indirect, similar to laser pointers. The selection method is discrete but offers continuous adjustments until the command is triggered. The effort on DoF of the input method for the operator is five, two rotational and three translational.



**Figure 4.8:** IVE-Interaction-Taxonomy (adapted from Lindeman [114]).





## CLASSIFICATION OF MR ROBOTIC UIs

Properties of mixed reality robotic user interfaces are of two different kinds: “*directly controllable*”, like the *degree of autonomy* or the *vividness* of the display and “*implicitly controllable*”, like *mental workload*, *situation awareness* or *trust*. The latter are perceived phenomena, which depend on individual persons, their mental models and their specific perception. Properties of both categories are often not precisely measurable, but relative ratings in comparison to other implementations and estimations are possible. In this chapter a taxonomy is presented, which utilizes directly controllable, mostly technical, factors. Some of these factors are already discussed in Chapter 3 and Chapter 4. The taxonomy supports the development process of MR-RUI regarding intentional system design. The concepts presented in Chapter 6–9 benefit from the analytical approach presented in the following.

### **5.1 IMPAct: A Holistic Framework for Mixed Reality Robotic User Interface Classification and Design**

Currently, robots are becoming a part of our daily life in households, for example as vacuum cleaners, lawn mowers, and personal drones, but also in the area of intelligent toys. Industrial machines have started to collaborate with human co-workers, and assistive robots find application in health-care. The development of accessible and secure robots is

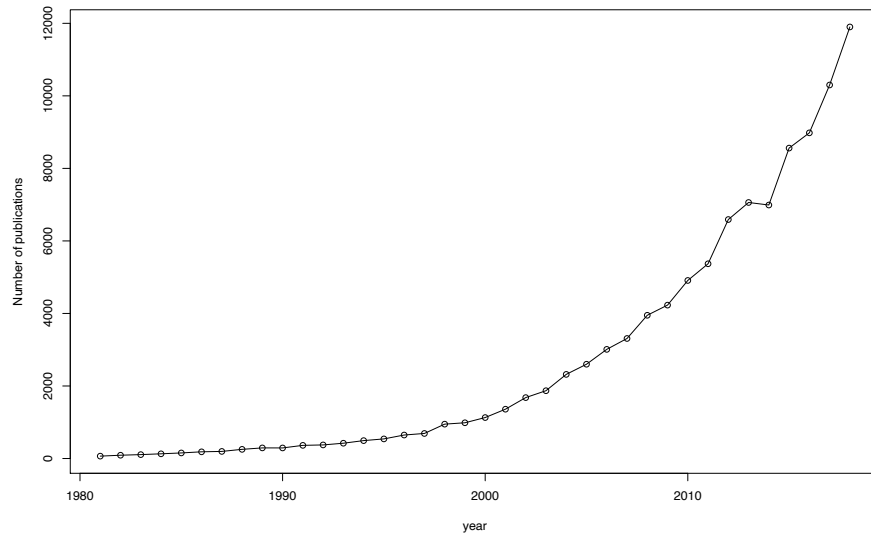
a major challenge for current and future applications. Burdea [28] already pointed out how virtual reality (VR) and robotics are beneficial to each other. Intuitive, natural, and easy-to-use user interfaces for human-robot interaction have the potential to fulfill the needs in this context. While a fully-automated robotic companion, which is capable of performing almost every task for humans, still remains a vision of the future, it is essential to also focus research on the interaction between humans and robots. Mixed reality (MR) technology provides novel vistas, which has enormous potential to improve the interaction with robots.

*“The implementation of a mobile and easy deployable tracking system may trigger the use of trackers in the area of HRI. Once this is done, the robot community may benefit from the accumulated knowledge of the VR community on using this input device.”*[45]

The enormous increase in research interest of MR combined with robotics is demonstrated by counting the number of new scientific publications during the past few years (cf. Figure 5.1). Many contributions involving robotic user interfaces (RUI) and the field of MR are already published, and the number of new publications per year regarding mixed reality robotic user interfaces (MR-RUI) shows a trend of exponential growth. To the best of my knowledge, there is no classification integrating all relevant aspects of this highly multi-disciplinary field of research available in the literature. However, a taxonomy of MR-RUIs would represent a valuable tool for researchers, developers, and system designers for a better understanding, more detailed descriptions, and easier discriminations of newly-developed user interfaces (UI). Therefore, a holistic approach is proposed that incorporates the relevant aspects of MR-RUIs for system design and classification, while supporting a comprehensible overview and understanding the interconnections and mutual influences of the different aspects.

The main aim of this contribution is to provide an appropriate structure for the classification of robotic user interfaces involving mixed reality. The author believes that good and successful design starts with understanding all relevant fields by performing a holistic analysis. The proposed structure is generated from unfolding prominent and relevant taxonomies, continua, and classifications from the most important aspects into a list of highly-relevant factors. Only factors being under direct control of system designers are taken into account; thus making it possible to select or determine their actual values without the need to be concerned with the mental models of probable users of the system. This is beneficial for both purposes: system design and classification. Due to the strong interdisciplinary nature of the topic of MR-RUI, a new taxonomy is needed that summarizes all relevant factors and provides the necessary connections, explaining the interplay between them.

In this chapter, the “*IMPAct framework*” is introduced for the classification of mixed reality-based robotic user interfaces. As explained in Section 5.1.2, it identifies different factors



**Figure 5.1:** Yearly number of new publications listed on Google Scholar containing the keywords “*mixed reality*” and “*robots*” (date of acquisition: 2019, March 13th).

from the categories ‘I’nteraction, ‘M’ediation, ‘P’erception, and ‘Act’ing as contributors to actual MR-RUI implementations. The factors are grouped into different categories in order to provide a memorable structure for application. Its purpose is to categorize existing MR-based robotic systems and to provide the process of designing new solutions for specific problems, in which interaction with robots in MR is involved, with a guideline to include all relevant aspects into the decision process. Single factors are explained as an aid for their application. The authors selected the included factors carefully to avoid dependency on the mental models of individuals. Thus, only technical, measurable, or identifiable factors were taken into account. As a result, the IMPAct framework serves as an effective tool for discriminating different MR-based robotic systems with a focus on the UI. Additionally, it can be consulted for designing or improving MR-RUI by making use of the holistic view in order to create effective and specialized UIs.

Section 5.1.1 introduces the general structure of human-robot interaction, starting with teleoperation, and extends the model to a more detailed version with the aim to describe MR systems for human-robot interaction more specifically. The section concludes with an overview of prominent theory papers regarding the relevant fields of 3D interaction, mixed reality (MR), robot autonomy, interaction with virtual environments (VE), and many more. In Section 5.1.2, the experimental procedure is explained and the results are summarized. The extraction of relevant factors from theoretical papers, the card sorting experiment, and the second clustering into major categories are described in Section 5.1.2. Section 5.1.2 presents the results from the card sorting experiment and the following refinement in tabular and graphical representations. A validation of the results is presented in Section 5.1.3 by applying the taxonomy to an MR-RUI described in a publication. The validation is discussed in detail for every separate group and also provides additional

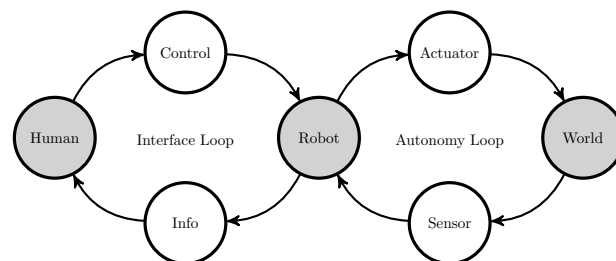
information beyond the given information in the tables. Section 5.1.4 summarizes the article and discusses the results and their limitations and future work. The appendix contains the complete table of factors for reference (cf. Table A.1) including the factor's publication source. Table 5.2 represents the short profile of the system categorized in Section 5.1.3, and Table A.2 is the template, ready for use to classify other MR-RUI.

### 5.1.1 Taxonomies with Relation to MR-RUIs

In the following, the main components involved in MR-RUI specification are identified and relevant taxonomies, classifications and definitions from scientific publications are listed and discussed briefly.

#### The Structure of Information Flow in Human-Robot Interaction

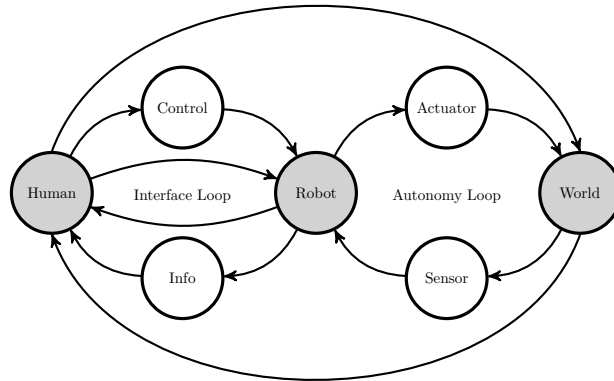
The main components in a human-robot system and how they are connected were already described in Crandall et al. [41]. Figure 5.2 shows an adapted version. This basic scheme of a robot remote operation identifies *human*, *robot* and *world* as the essential entities to perform a human-robot interaction task. Required components on the side of the interface between human and robot are responsible for transferring *control* data to the robot and some *information* from the robot to the human as feedback. In such setups, the robot is able to manipulate the world and to interact with the world physically, or at least is co-located with other real-world entities. To fulfil this task, the robot needs to gather information about the world it is acting on by utilizing *sensors*. It can influence the actual state of the world by utilizing *actuators*. To a certain degree, the robot can act autonomously. The knowledge of the robot about the world and its own state is accessible for the human operator by the user interface in general and provides the operator with some information helpful for making decisions about the next control.



**Figure 5.2:** Remote robot operation (adapted from Crandall et al. [41]).

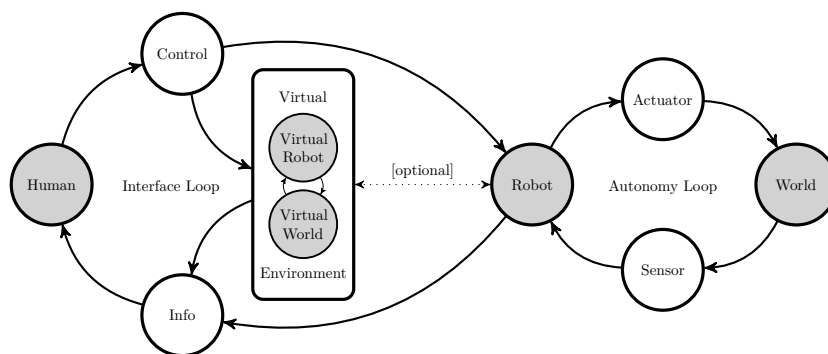
In a scenario where human and robot are sharing the same physical space (cf. Figure 5.3), there can be additional direct and physical interactions of the human and the robot, and the human and the world. Both of these schemes explain how humans can interact with

a distant or nearby robot in order to manipulate the real world, but there is no detail about how VEs are probably embedded into this.



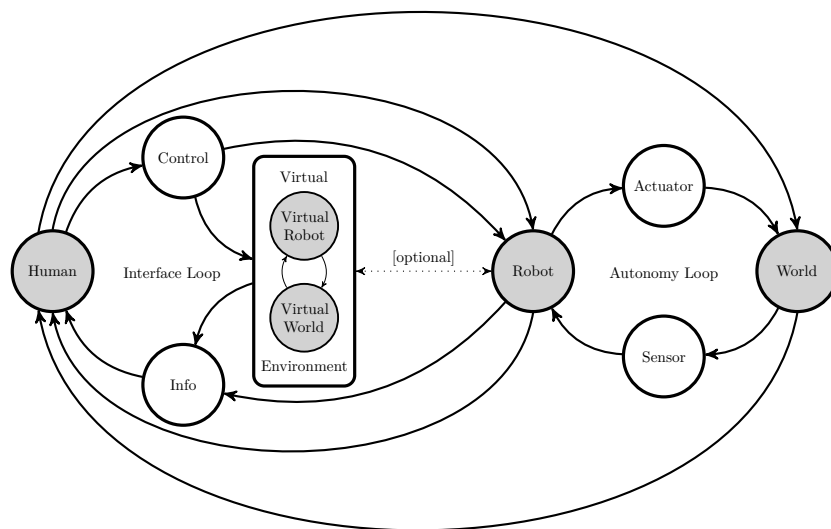
**Figure 5.3:** Co-located robot operation (inspired by the remote robot operation by [41]).

Enhancing the basic scheme of remote operation with the component of a *virtual environment* containing a *virtual robot* model and a model of a *virtual world* (cf. Figure 5.4) reveals new potential interactions. One conclusion is based on the fact that even though a remote operation scenario is described, the operator has the possibility to influence directly the virtual world by virtual physical interaction or implicitly by utilizing some linkage between the VE and the real robot. This is the case if VR is used as an additional component in a human-robot system. By including a virtual model of the robot and the world in the system, the same control means, used for the real robot, are applicable to the VE. However, interaction with components of a VE removes the physical boundaries of our real world. The implementation of VR-based user interfaces means for the system designer that he/she has to deal with a much higher complexity in creating appropriate means of interaction. Extending the scheme of VR robot operation by integrating MR technology means for the operator that he/she is not separated from the real world anymore. The multitude of interconnections further grows, and direct interaction with the real world is additionally possible.



**Figure 5.4:** Virtual reality robot operation (inspired by the remote robot operation by [41]).

Enhancements of the classical co-located robot operation scheme with a VE by using MR technology (cf. Figure 5.5) couples the operator directly with the real robot and the real world, while all relationships of the VR robot operation are maintained. We can conclude that mixed reality robot operation is the most flexible and complex case we can currently imagine to implement a robotic user interface. Especially how the interaction with the VE and the VE itself is connected to the real robot is a challenging question for MR-RUI design and implementation.



**Figure 5.5:** Mixed reality robot operation (inspired by the remote robot operation by [41]).

### On Classifying Mixed Reality Robotic User Interfaces

Much effort went into understanding and classifying a multitude of aspects relevant to the large fields of human-computer interaction (HCI) and human-robot interaction (HRI). These taxonomies, continua, rules, and guidelines each serve a special purpose. Despite contributing valuable terms and knowledge, combining them together into a kind of composed taxonomy reveals the issue of overlap and conflicting models and involves the risk of misleading from the core concept of the actual focus of MR-RUI analytics.

Robinett [163] discusses the topic of “*synthetic experiences*” (cf. Figure 4.2). He clarifies many aspects of how modalities are captured and transferred from the world to the human. The other direction, from the human to the world, resembles how to manipulate the world using technical devices. The bidirectional mediation, discussed here, is highly relevant for robotic UIs involving MR.

A taxonomy with VR as one of the extrema is the “*AIP-Cube*” proposed by Zeltzer [224] (cf. Figure 4.3). The authors determined three categories, ‘A’utonomy, ‘I’nteraction, and

‘P’resence, as important and combined these as orthogonal axes in a cube-like taxonomy. Especially, autonomy is very relevant to robots in general, but in the context of VEs, the question arises where autonomy should be: at the robot side, at the human side, or at the VE side. If we agree not to specify the location of an MR-RUI in detail, but to define the UI as the interplay of different aspects and components involved in enabling the interaction, we can find a subset of suitable elements for classifying MR-RUIs.

Presence, as defined by Slater and Wilbur [184] in the “*FIVE framework*” (“A ‘F’ramework for ‘I’mmersive ‘V’irtual ‘E’nvironments”), depends on individual mental models of different humans, and thus, the presence level of a certain implementation cannot be specified. Instead, *immersion* is found to be an appropriate technical factor, which can be determined objectively and also contributes to the resulting level of presence.

Strongly related to immersion is the definition of the “*reality-virtuality continuum*” (cf. Figure 4.1) by Milgram et al. [131] and display classes in the context of the continuum, thus contributing to the level of immersion [129, 131, 128]. Major structural properties of MR-RUI are specified by these topics.

One very large field that is highly relevant to MR technology is “*3D interaction*”. Bowman et al. [26] provided a very general discrimination of 3D user interfaces (3DUI) into three different applications, *selection and manipulation*, *travel*, and *system state* (cf. Section 4.2). The classification of “*hand gestures*” by Sturman et al. [193] provided a very complete taxonomy (cf. Figure 4.1), which takes the dynamics into account, as well as up to six degrees of freedom (DoF). The proposed classification is also applicable for gestures, other than hand gestures. Interaction techniques using up to 6 DoF and their relevant properties were issued by Zhai and Milgram [225] with the “*6-DOF-Input-Taxonomy*” (cf. Figure 4.4). Here, a strong focus lied on the mapping properties of the implementation of an actual input method. The “*interactive virtual environments (IVE) Interaction Taxonomy*” proposed by Lindeman [114] formalizes interaction methods with arbitrary DoF as a combination of an *action type* and a *parameter manipulation type* (cf. Figure 4.5).

“*Robot classification*” is a difficult task, due to the polymorphic characteristic of robots. Some books, serving as an introduction to robotics, list a classification based on their intelligence level, defined by the Japanese Industrial Robot Association (JIRA) [142, 124] (cf. Section 3.2). Another classification from Association Francaise de Robotique (AFR) describes their capabilities of interaction. Murphy and Arkin [134] introduced classical paradigms of robot behavior. However, there is not only one general definition of a robot. Across robotics’ history, many accepted definitions appeared, but in the end, it depends on the specific community that one has selected. Therefore, the question of how a user interface for robots is defined is almost untouched in the literature. In the paper of [18, 124], the term “*robotic user interface*” (RUI) seeds to be used for the first time (cf. Section 3.3.3). In this paper, the authors defined four categories for classifying RUIs with regard to HRI-relevant factors. The authors took into account the *level of autonomy*, the

*purpose* of the robot, the *level of anthropomorphism*, and the control paradigm resulting in a certain *type of communication*.

Regarding “*autonomy*”, there is a long history of research methods addressing questions of design and analysis what should be automated and how far. The oldest and most prominent conceptualization goes back to 1978 by Sheridan and Verplank [182]. The authors introduced “*levels of autonomy*” (*LOA*), a classification for rating the interaction with a computer along with a discrete scale of autonomy. A refined version of *LOA*, published by Parasuraman et al. [150], offers a more detailed level of granularity by dividing tasks, which can be automated, into four different stages of *acquisition, analysis, decision, and action*. The next step involves *dynamic assignments of autonomy levels*, which are not fixed by design, but depend on the current state of the world [40]. Miller and Parasuraman [132] further improved the model by explaining how tasks at the four different processing stages consist of several *subtasks*, each with its own *LOA*. Thus, the resulting autonomy level at each stage is the result of a process of aggregation. Discussions of autonomy were not especially focused on issues of intelligent robots; furthermore, they were targeted to questions of industrial automation, e.g., the question of how to replace human workers by machines on specific tasks. This resulted in taxonomies that did not include specific demands and abilities of different kinds of robots. Beer et al. [20] proposed “*levels of robot autonomy*” (*LORA*). Schilling et al. [173] provided a very recent perspective on shared autonomy by taking *multiple dimensions* of relevance for robot interaction into account. A very interesting explanation of the term “*autonomy and its connection to intelligence and capabilities*” was provided by Gunderson and Gunderson [67]. The human-centered perspective of the effects of interaction with autonomous systems has been explained in detail by several authors [16, 46, 29].

Important aspects in human-robot collaboration are the “*roles*” of the interaction partners, the “*structure of the interaction*”, and how the initiative and autonomy are distributed among the interacting partners [175, 45].

### 5.1.2 Materials and Methods

As demonstrated by Adamides et al. [3], it is possible to develop a taxonomy from extensive literature investigation of scientific publications, followed by a session of card sorting in order to cluster the resulting data. Further processing of the intermediate results can lead to a reasonable structure for the classification of an actual topic. With the aim to create an objective taxonomy for specifying technical factors of MR-RUIs, every aspect involving qualitative evaluation of the system is neglected in this work. Even though qualitative measures of MR-RUIs seem to be of importance, it is beyond the scope of this work and the integration of these factors should be a part of a contribution regarding MR-RUI evaluation.



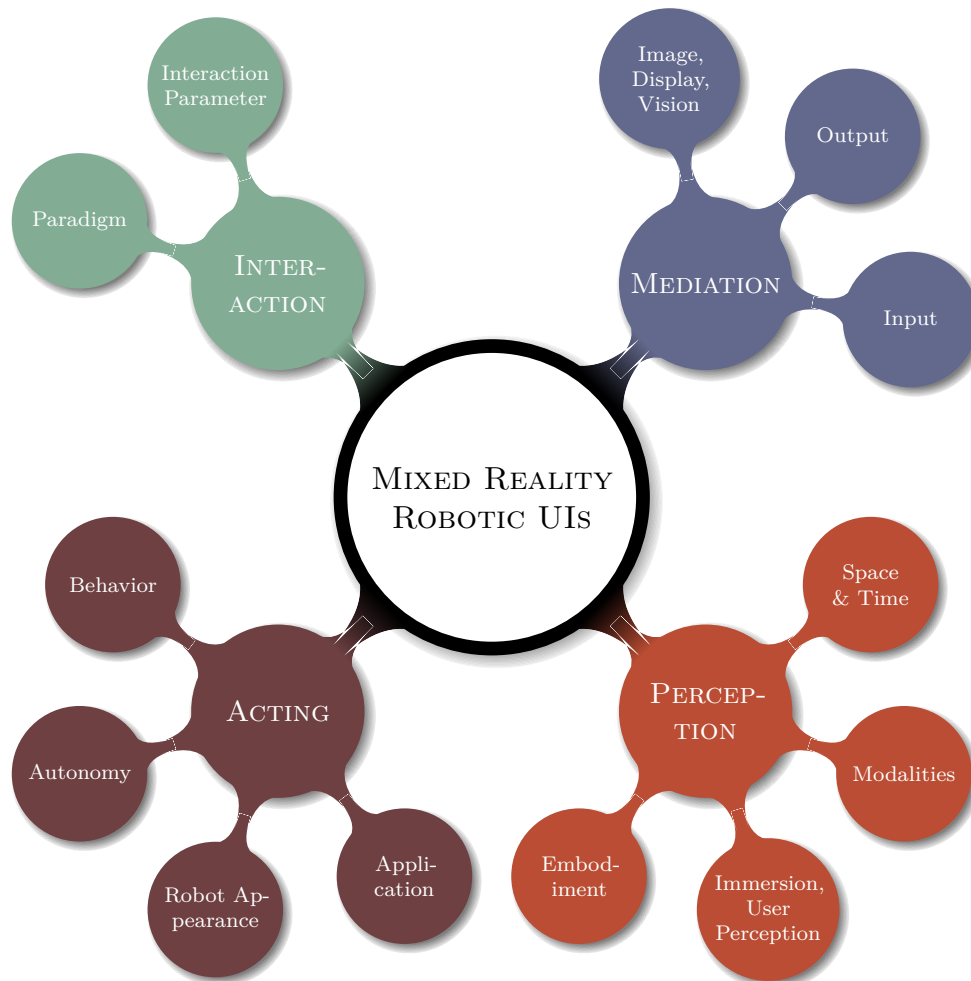
## Procedure

In order to assess important factors for MR-RUI classification, publications with high impact were collected containing classifications from the fields of *3D interaction*, *mixed reality*, *user interfaces*, *immersive virtual environments*, *robot autonomy*, *automation*, and *many more*. The decision to include or neglect a certain publication was governed by its citation count or the reputation of the main authors. In general, the most important authors were found by reading survey and review papers, followed by a combination of systematic and cumulative literature investigation on Google Scholar, Researchgate, the IEEE Digital Library, and the ACM Digital Library. Then, factors were extracted, which directly impact the nature of the underlying MR-RUI, from the investigated papers by unrolling the described taxonomies, definitions, and classifications. Typically, many authors describe classifications of a certain aspect using a discrete list or a continuum, which allows relative ordering of actual instances. In this thesis, a single property of this kind is called a *factor*. After determining factors in this way, the aim was to find aspects not covered so far and identified three more factors that were important in our previous work: “*interaction reality*”, “*location*”, and “*system extent*”. The resulting list contained 62 different, mostly technical factors (cf. Table A.1), all being useful to discriminate different approaches of robotic user interface design involving MR. Caused by the technical nature of these factors, their actual values are easy to determine without the need to be experienced with the actual system and without the interference of individual mental models of different specific users or operators. The achieved objectivity of the selected factors makes them useful for other researchers and designers to be utilized in classification tasks.

After extracting these factors, *open card sorting* was applied for clustering these into named groups. Due to the high number of relevant factors, especially these groups were of interest in order to be able to create a more granular taxonomy for a better understanding of relevant categories to the special case of MR-RUI design. The card sorting was prepared by noting down every factor from Table A.1 to the front side of the cards. The description of the each factor (cf. Table A.1) was written on the backsides of the cards. To avoid strong bias by the authors, the participants were instructed not to look at the backside of the cards unless every card was assigned to a named category. During the experiment, 9 experts (3 female, 6 male) from HCI research with their focus on virtual reality, augmented reality (AR), and interaction with 3D user interfaces were recruited to perform open card sorting on a shared set of cards in an open group form. Participants were allowed to enter and leave the session as they liked during a 3 h time slot. The active times of the participants were protocolled. The session took 2 h and 40 min, and as a result, the 62 factors were clustered into 13 groups in an iterative approach. Each participant spent on average 35 min on the task ( $SD \approx 18.498$ ). The protocol revealed a total temporal effort of 5 h and 17 min. During the session, audio recordings were made, and the participants

were instructed to think aloud. This was intended to identify misunderstandings of the tasks and to better understand the decisions of the participants. Participants were allowed to note down comments on the cards themselves and to create new cards or to remove already existing cards if an explanation was provided. No comments were made, and no card was removed. One card, *DoF*, was duplicated by the participants and integrated into two different categories, *interaction parameters* and *input*.

Followed by the card sorting session, the resulting groups were further clustered by the authors into four major categories in order to create a better memorable structure. The session protocol was considered during the decision process to reduce the influence of the authors' peculiar view. This step of further categorization is important to increase the likelihood that the resulting taxonomy is adopted by other researchers. The resulting categories were further discussed with two researchers from the department, who took part in the card sorting experiment, until the result revealed a reasonable structure without logical flaws (cf. Table 5.1).



**Figure 5.6:** The overall structure of the mixed reality robotic user interface taxonomy. MR-RUIs are directly shaped by the aspects of *interaction*, *mediation*, *perception* and *acting*. These groups including their relevant subcategories and containing factors generate the IMPAct framework for MR-RUI classification and design.

## Card Sorting Results

Based on a literature investigation, a card sorting experiment with HCI professionals was performed. Further analysis of the card sorting results revealed a general structure for the classification of MR-RUIs with four major categories, as shown in Figure 5.6. The detailed results of the two clustering steps are summarized in Table 5.1, listing each of the factors according to their associated group and category. The main categories provided a plausible and memorable structure with the aim to improve the workflow during classification or design tasks. The groups of the next detail-level resulted directly from the iterative approach of the card sorting experiment, in which all identified factors from the literature investigation (cf. Table A.1) were clustered into groups. Many of these sources were listed in the related work (cf. Section 5.1.1).

**Table 5.1:** Results of the card sorting.

Category	Group	Factors
Interaction	Interaction Parameter	Action Type, AFR Interaction Level of the Robot, Control Alignment, Control Order, Directness, DoF, Interaction Reality, Interaction Type, Mapping Relationship, Parameter Manipulation Type, Sensing Mode, Structure of Interaction
	Paradigm	Human-Robot Communication, Intelligent Robot Control Paradigm, Robot Control Concept, Role of Interaction, RUI-Type, UI-Type
Mediation	Image/Display/Vision	Display Centricity, Display Class (Milgram), (Display) Space, Image Scaling, Vividness
	Input	Action Measurement Type, Directness of Sensation, DoF, Gesture Type, Sensor Type
	Output	Actuator Type, Display Type
Perception	Embodiment	Extent of Body Matching, Proprioceptive Matching
	Immersion/User Perception	EPM, Integration of VE, Level of MR, RF
	Modalities	Causality of Modalities, Extensiveness, Inclusiveness, Model of Modality-Source, Superposition of Modalities, User Experience
	Space & Time	EWK, Location, Realtime Level, Surrounding System Extent, Time (interaction), Time (system purpose)
Acting	Autonomy	Level of Capabilities, Level of Intelligence, Plot, Refined LOA, Initiative, JIRA Intelligence Level of the Robot, Timely Aspects of Actors' Autonomy
	Behavior	Robot Behavior Level, Robot Communication Capability, Transparency of System State
	Robot Appearance	Level of Robot Anthropomorphism
	Application	Robot Purpose, Robot Type

Table A.2 summarizes all factors on a single page, which serves as a template for profiling arbitrary MR-RUI designs. cf. Figure 5.6 together with Table 5.1, containing the results of the clustering, Table A.1, the alphabetical list for reference and explanation, and the empty template in Section A.1 of the appendix (cf. Table A.2) serve as a framework for classification and design of MR-RUI. As demonstrated in the following section, using the IMPAct framework enables very precise and extensive analysis and description of relevant

systems. Since only objective factors were included, all mostly technical, which were under the control of the system designer, it can be concluded that the results represent a useful framework for initiating diversity and effectiveness in MR-RUI research and development.

### 5.1.3 Validation of the IMPAct Framework

In this section, the results are applied to an MR-RUI from recent publication.

*Example: Application of the Framework for System Classification*

In the following, an MR-RUI for a pick-and-place task [104] is classified using the IMPAct framework. A detailed analysis of the technical properties in the categories *interaction*, *mediation*, *perception*, and *acting*, including their subgroups, is listed and, where required, further explained to enable readers to have a complete understanding of the setup and implementation given.

#### Interaction

Interaction is represented by the two groups *interaction parameter* (cf. Table 5.3) and *paradigm* (cf. Table 5.4). The actual values of the classification of the analyzed system are noted in the tables. Further explanations are given in the following paragraph.

Some characteristics of the interaction depend on the interaction level of the robot. The AFR describes Class D as an extension of Class C, adding the capability to the robot of acquiring information from its environment. Class C itself is defined as “*programmable, servo controlled robots with continuous or point-to-point trajectories*”[142]. The degree of spatial matching, expressed by the factor *Directness*, is lowered by probable sensor error. This involves detection of the position of the targets for grasping by the robotic system and the registration process within the implementation of marker detection at the main UI device for aligning the virtual world components along their actual alter egos. The interaction method for selecting targets needs the human to be at an appropriate position and to look at a certain point at the surface of the target. One could argue that this method involves 6 DoF, 3 for defining a position in the real world of the human’s head and another 3 for rotating it as desired. Mathematically, the *rolling* rotation around the forward vector of the head is not used, so 5 DoF is correct, as well. The interaction with real-world objects is mediated through the robotic hardware after selection using virtual counterparts of the actual targets. Thus, the *interaction reality* is rated as virtual. Nevertheless, there is a kind of passive interaction, as well. By utilizing a see-through HMD, the actual targets are observable in the real environment during displaying of planned trajectories, which are visualized by actuating the virtual model in a loop with the very same trajectories.

Table 5.2: IMPAct template.

Category	Group	Factor	Value	
INTERACTION	Interaction Parameter	Action Type	discrete	
		AFR Interaction Level	Type D	
		Control Alignment	1-to-1	
		Control Order	0	
		Directness	almost isomorph	
		DoF	5	
		Interaction Plasticity	virtual	
		Interaction Type	selection	
		Mapping Relationship	position	
		Parameter Manipulation Type	direct	
	Sensing Mode	isometric		
	Structure of Interaction	SO,SR,ST		
	Paradigm	Human-Robot Communication	two monologues	
		Intelligent Robot Control Paradigm	hierarchical at user level	
		Robot Control Concept	feedback control loop	
Role of Interaction		supervisor & manager		
RUI-Type UI-Type		interactive SUI		
MEDIATION	Image Display Vision	Display Centricity	egocentric	
		Display Class (Milgram)	class 3	
		(Display) Space	registered	
		Image Scaling	1:1	
		Vividness	mid-to-low	
	Input	Action Measurement Type	HoloLens inside-out tracking	
		Directness of Sensation	partially directly viewed	
		DoF	3+1	
		Gesture Type	static posture	
		Sensor Type	IMU, structured light IR camera, bw camera, joint encoder, current measurement	
	Out-put	Actuator Type	robot arm & gripper	
		Display Type	see-through HMD	
	PERCEPTION	Embodiment	Extent of Body Matching	no VB
			Extent of Proprioceptive Matching	virtual cursor and viewing direction
		Immerison	Extend of Presence Metaphor	strong tendency towards real-time imaging
Integration of VE			almost completely integrated	
Level of MR			AR with tendency to RE	
Reproduction Fidelity			stereoscopic realtime rendering but with simple shaders	
Modalities		Causality of Modalities	partially transmitted and simulated	
		Extensiveness	only vision	
		Inclusiveness	almost none	
		Model of Modality-Source	computed	
		Superposition of Modalities	merged	
User Experience		mixture of transmitted, simulated and supervised robot		
Space & Time		Extent of World Knowledge	objects of interest are modeled with normal to high accuracy	
		Location	in locu	
		Realtime Level	delayed up to half a second	
	Surrounding	only 60° FoV		
	System Extent	network-based in one room		
	Time (interaction) Time (purpose)	frozen realtime		
ACTING	Autonomy	Level of Capabilities	generate trajectories, self localize	
		Level of Intelligence	collision avoidance, extract grasp points	
		Plot	virtual robot shows probable solution	
		Refined LOA	acquisition: 10; analysis: 3; decision: 1-2; action: 5	
		Initiative	fixed at the human	
		JIRA Intelligence Level of the Robot Timely Aspects	class 5 static	
	Behavior	Robot Behavior Level	remote controlled	
		Robot Communication Capability	reactive	
		Transparency of System State	visible	
	RA	Level of Robot Anthropomorphism	very low	
	Appli-cation	Robot Purpose	tool	
		Robot Type	industrial	

Even if *selection* is the most prominent active interaction concept, here *system control* is identifiable as well, when voice commands are used to start a specific action. Then, the state machine of the system moves over to a different state, changing the way of interaction in the next state; e.g., when confirming a picking trajectory with a voice command, the system proceeds to the *execution* state, and after finishing the grasping, it is ready to receive a command for placing the grasped object on the table. The current system design in the experiment, described by Krupke et al. [104], is for single users, single robots, and performing a single pick-and-place task. However, the implementation would also allow multiple operators to be logged into the system, simultaneously.

**Table 5.3:** Interaction parameter.

<b>Factor</b>	<b>Value</b>
Action Type	discrete
AFR Interaction Level of the Robot	type D
Control Alignment	1-to-1
Control Order	0
Directness	almost isomorph
DoF	5
Interaction Reality	virtual
Interaction Type	selection
Mapping Relationship	position
Parameter Manipulation Type	direct
Sensing Mode	isometric
Structure of Interaction	SO, SR, ST

**Table 5.4:** Paradigm.

<b>Factor</b>	<b>Value</b>
Human–Robot Communication	two monologues
Intelligent Robot Control Paradigm	hierarchical at user leve
Robot Control Concept	feedback control loop
Role of Interaction	supervisor and manager
RUI-Type	interactive
UI-Type	SUI

The robot itself had integrated joint controllers, which are accessed by the dedicated industrial computer, provided by the manufacturer of the robot. The control computer sends goal positions to the joint controllers. The low-level joint controllers itself uses a *feedback control loop* to keep joints at a desired position and to alert about position mismatch, or unreasonable high currents, or other issues to the control computer. During virtual or actual robot movements, the operator serves as a *supervisor*, but during the planning steps of *selecting a pick position* or *selecting a place position*, he/she fulfills the role of a *leader*. The RUI-type is classifiable as *interactive*, but with a tendency towards

very high-level programming. Regarding the history of user interfaces, the system can be called a *spatial* or *supernatural* user interface; spatial because the user is situated in the very same place of the operation and benefits from exploring the RE by natural walking and looking around; supernatural because he/she can see the probable future trajectory. Despite the focus of interaction, there was also classical graphical user interface (GUI) elements realized in the implementation. A head-up display shows the system state in textual form and provides feedback about the success of a given command. Regarding the way commands are given by the operator, the term *natural user interface* (NUI) is applicable because voice commands are triggering the main functions of the system.

### Mediation

Mediation consists of three groups *image/display/vision* (cf. Table 5.5), *input* (cf. Table 5.6), and *output* (cf. Table 5.7). The actual values of the various factors are collected in the tables. The following paragraph contains further explanations.

**Table 5.5:** Image/display/vision.

Factor	Value
Display Centricity	egocentric
Display Class (Milgram)	class 3
(Display) Space	registered
Image Scaling	1:1
Vividness	mid-to-low

The level of *vividness* is mainly limited by the hardware and the selected software used for rendering. Since the first version of the Microsoft HoloLens with a small field of view is utilized, as well as the general quality of the see-through display, regarding resolution, color range and brightness are taken into account. In addition, the quality of the real environment (RE) content by looking through the head-mounted display (HMD) is negatively influenced, causing overall mid-to-low vividness.

**Table 5.6:** Input.

Factor	Value
Action Measurement Type	HoloLens inside-out tracking
Directness of Sensation	partially directly viewed
DoF	3 + 1
Gesture Type	static posture
Sensor Type	IMU, structured light IR camera, BW camera, joint encoder, current measurement

The *action measurement* of the operator is mainly contributed by the integrated self-localization feature of the Microsoft HoloLens. Since relevant information is three-dimensional coordinates in the reference frame of the real world, marker detection serves the task of providing a reference anchor for the transformation of local device coordinates to RE coordinates. The *directness of sensation* can principally be regarded as direct. The RE is directly viewed, and the virtual components are supposed to be anchored at corresponding points of their alter ego. Without sensor errors, this would be perfectly direct as well, but in the implementation, there is a small but perceptible error. From the robot middleware's view, the system input is just a single 3 *DoF* position within the reference frame of the target object and its name. On the lower level, during user input, walking is used to reach a certain position in the RE, then the posture of the body might be altered, and finally, the head is turned around pitch and yaw axes to move the cursor along the surface of the target object until the desired position is reached. Typically, the system uses static posture as the *gesture type* for defining pick and place positions and confirms the current selection with a voice command. The second interaction implementation uses the pointing gesture of the index finger; thus, a motionless finger posture is consulted in this case.

**Table 5.7:** Output.

Factor	Value
Actuator Type	robot arm and gripper
Display Type	see-through HMD

## Perception

In the category perception, four groups, *embodiment* (cf. Table 5.8), *immersion/user perception* (cf. Table 5.9), *modalities* (cf. Table 5.10), and *space and time* (cf. Table 5.11), are analyzed. Relevant factors and their actual values are collected in the tables. Further explanations are provided in this section.

**Table 5.8:** Embodiment.

Factor	Value
Extent of Body Matching	no VB
Extent of Proprioceptive Matching	virtual cursor and viewing direction

In the analyzed system, there is no virtual body (VB) involved. One could argue that a cursor, which is augmented by the view, matches at least a little the definition of a virtual body. In this case, it should be mentioned that there is no matching between the moving bodies, cursor, and head. However, the proprioceptive matching is appropriate in terms



of visualizing the viewing direction and its spatial cues when projecting the cursor on top of the surface of viewed objects.

**Table 5.9:** Immersion/user perception.

Factor	Value
Extend Presence Metaphor	strong tendency towards real-time imaging
Integration of VE	almost completely integrated
Level of MR	AR with tendency to RE
Reproduction Fidelity	stereoscopic real-time rendering, but with simple shaders

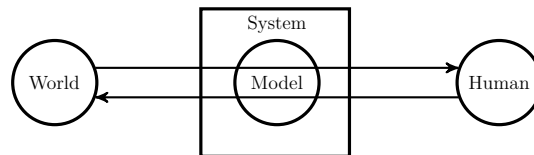
The *extent of presence metaphor* (EPM) is rated very high since the human is mostly looking at the RE, but through a see-through HMD. This fact can potentially lower the EPM in comparison to looking directly at the RE. Only some elements like the robot, a table, and some objects for grasping are virtual. However, these elements are anchored with reference to RE coordinates in a very stable way, resulting in a very well-performed integration to the RE. The *integration of VE* by superimposition of the virtual robot on top of the actual robot causes a logical mismatch to natural perception, since human perception usually does not allow the existence of two objects at the very same place. Regarding *level of MR*, due to the fact that there are only a few objects having virtual counterparts, it can be concluded that the RE is very dominant. Artificial-looking objects, which are perfectly integrated into the real environment, cause a moderate *reproduction fidelity* since their level of detail is quite low and simple shaders are utilized.

**Table 5.10:** Modalities.

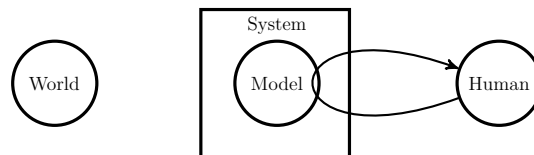
Factor	Value
Causality of Modalities	partially transmitted and simulated
Extensiveness	only vision
Inclusiveness	almost none
Model of Modality-Source	computed
Superposition of Modalities	merged
User Experience	mixture of transmitted, simulated, and supervised robot

The *causality of modalities* is based on a mixture of data from reading an actual posture of the robot, then transmitting, and finally applying them to the virtual model, which is almost perfectly matching with the real robot. Some displayed trajectories are computed and represent only options for the future, but can become real. Regarding *inclusiveness*, it should be mentioned that, very bright virtual content on the see-through display can occlude or distract from parts of the real world. Except for this side-effect in the analyzed

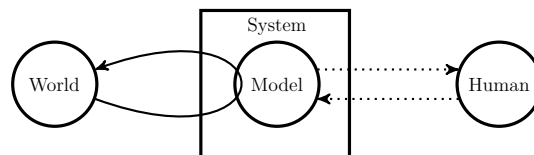
system, operators are not shut-off from the real world. The *user experience* is classified by three different concepts of data flow (cf. Figure 5.7–5.9), which are all present during system use and describe different aspects of the system. *Transmitted experience* (cf. Figure 5.7) represents especially how information like the current pose of the robot is transferred to the human. *Simulated experience* (cf. Figure 5.8) describes what happens when a new target position is selected and the generated trajectory from the robot middleware is sent to the HMD. *Robot supervised by human* (cf. Figure 5.9) represents the case when the robot starts to manipulate the world and the human is interacting with a VE.



**Figure 5.7:** Data flow in a “*transmitted experience*” (adapted from Robinett [163]).



**Figure 5.8:** Data flow in a “*simulated experience*” (adapted from Robinett [163]).



**Figure 5.9:** Data flow during “*robot supervision*” (adapted from Robinett [163]).

**Table 5.11:** Space and time.

Factor	Value
Extent of World Knowledge	objects of interest are modeled with normal to high accuracy
Location	in locu
Real-time Level	delayed up to half a second
Surrounding	only 60° FoV
System Extent	network-based in one room
Time (interaction)	frozen
Time (purpose)	real-time

Regarding *location*, it should be mentioned that the analyzed system is intended to be used side-by-side with the actual robot, but technically, it also works as a remote system.

Unfortunately, then the cues provided from the RE are missing, and the potential of recognizing problematic situations with probable collisions is reduced. Even if the *real-time level* of the system is classifiable as real time, it is not designed for continuous real-time control from the interaction point of view. Evaluating the factor *surrounding* reveals that, despite the low FoV, the tetherless operation and the inside-out 6 DoF tracking of the HMD resulted in a spatial experience. The VE can be viewed from arbitrary positions and explored by walking around. Thus, even by the visual experience of looking through a looking glass, spatial presence was generated and improved the resulting experience of the surrounding factor in comparison to a simple 2D video glass with the same FoV. For the *system extent*, it should be mentioned that the HMD was self-contained, making it a head-mounted computer (HMC). It was connected by WLAN with the computer controlling the robot and, if necessary, extendable by additional network-based processing nodes.

## Acting

Acting consists of the four groups *autonomy* (cf. Table 5.12), *behavior* (cf. Table 5.13), *robot appearance* (cf. Table 5.14), and *application* (cf. Table 5.15). The actual values of the classification of the analyzed system are noted in the tables. Further explanations are given in the following paragraph.

**Table 5.12:** Autonomy.

Factor	Value
Level of Capabilities	generate trajectories, self-localize
Level of Intelligence	collision avoidance, extract grasp points
Plot	virtual robot shows probable solution
Refined LOA	acquisition: 10; analysis: 3; decision: 1–2; action: 5
Initiative	fixed at the human
JIRA Intelligence Level of the Robot	class 5
Timely Aspects of Actors' Autonomy	static

The *level of intelligence* is divided between two devices. The robot control computer calculates collision-free trajectories according to its dynamic planning world. The input and output device of the operator, the Microsoft HoloLens, assists in selection of grasp points, according to its programming.

**Table 5.13:** Behavior.

Factor	Value
Robot Behavior Level	remote controlled
Robot Communication Capability	reactive
Transparency of System State	visible

The *transparency of the system state* is limited by the implementation. All modeled states are displayed during system use in the head-up display element of the UI.

**Table 5.14:** Robot appearance.

Factor	Value
Level of Robot Anthropomorphism	very low

The robot incorporates only a tendency towards *anthropomorphism* since the gripper is three-fingered and has some similarities to a humanoid hand, e.g., bending of joints is only possible in one direction when starting from a configuration, in which all fingers are completely straight.

**Table 5.15:** Application.

Factor	Value
Robot Purpose	tool
Robot Type	industrial

#### 5.1.4 Discussion of the IMPAct Framework

This section discusses the experimental results and explains their potential impact on MR robotics. Furthermore, the limitations of this contribution and future ideas are explained.

#### Conclusion and Discussion

In this chapter, the IMPAct framework for classification and analytical design of mixed reality robotic user interfaces was presented and applied to a robotic pick-and-place system utilizing the Microsoft HoloLens. Relevant factors are carefully investigated from the scientific literature. Prominent taxonomies from relevant fields are decomposed to find important factors that directly influence the system and can be determined regarding their actual values. Since many of these factors represent a position on a continuum and are not exactly measurable, it is, in general, a difficult task. Nevertheless, it is demonstrated that even with these inaccuracies, very detailed descriptions of existing systems are possible.

The framework can be regarded as an important and necessary step towards holistic system design and description of MR-RUIs.

Using the proposed IMPAct framework for exhaustive system descriptions helps to remove ambiguity, as demonstrated in Section 5.1.3. By taking an exhaustive list of relevant factors into account, differences from other systems, as well as unique features, become clear. Additionally, the framework provides a methodology for improving actual system designs by providing a standardized template, which enforces intentional decision making.

### **Limitations and Future Work**

Even though an extensive literature investigation is performed it is likely that some relevant aspects are missing in the list of factors and thus are not regarded in the proposed framework. Especially, if certain topics are further developed and new taxonomies arise, the proposed taxonomy should be adapted to these changes and then incorporate new relevant factors of the given topic. In general, a description of MR-RUI, covering as many relevant aspects as possible, is desirable. The proposed framework would further benefit from efficient means to assess the values of the listed factors in a more comfortable and time-saving way. An implementation as a wizard-based GUI tool with prepared options, tooltips, and web links for explanations of the single factors would reduce the current workload.

The proposed work is limited to objective and technical measures, which are assessable without any practice with the analyzed or described MR-RUI. Qualitative aspects are neglected in this contribution, but are valuable feedback for further improvements in system design. An extension of the IMPAct framework would explain how the objective factors are related to perceptive and qualitative factors. Then, the framework would provide means for holistic re-design as a part of an iterative design process of MR-RUI. Furthermore, some robotic web archives are currently arising (cf. <https://robots.ieee.org/robots/>). Currently, there is no such site for MR-RUI. The proposed framework within this article could serve as a basis for creating an MR-RUI archive.

## **5.2 Examples**

This chapter presents a selection of published MR-RUIs, briefly. All information is summarized from corresponding scientific publications starting from the early 90s until now. A small subset is classified using the IMPAct framework template (cf. Section A.2), presented in the previous chapter (cf. Section 5.1). After summarizing contributions, which explicitly show the benefits of MR for robotics in Section 5.2.1, the next section (cf. Section 5.2.2) explains current limitations. The chapter ends after a comparison of the

classical virtual fixtures implementation with the visual virtual fixtures using the IMPAct template.

### 5.2.1 Essential MR-RUIs

In the early 90s, KARMA (Knowledge-based Augmented Reality for Maintenance Assistance) demonstrated how a see-through display in an HMD could provide uniformed users with useful information at the right place and time [58]. In this way, untrained people should be able to perform tasks they have never done before. Spatial information on the position of the HMD or the device of interest, which is part of the maintenance task, originates from a Logitech 3D ultrasonic-based tracking system. AR was here utilized to enable office staff to perform maintenance tasks at a copy machine. Assisting with complex 3D tasks was here achieved by adding visual information in-situ, thus, showing important regions, buttons or levers for executing maintenance tasks.

At the same time, Das et al. [44] identified several vital aspects, which are nowadays considered in the system design of teleoperation systems and are exceptionally easy to implement in MR-based systems. The authors identified the importance of the viewpoint presented to the operator. They discovered that giving freedom to the operator to choose the viewpoint improves the performance and reduces the time taken to complete tasks, especially untrained users benefit from this. In contrast to this, other viewpoints seem to lower the user's performance. More generally, they found many advantages in integrating a model of the robot and its workspace and propose to integrate synthetic views into RUIs, instead of watching the robot in a direct matter. The presented view was entirely computer generated and presented on a 2D screen. The authors emphasize the issue of 3D visualizations on monoscopic displays, which is actively lowering the spatial understanding of the environment and propose to generate shadows in order to improve three-dimensional perception. In this early work on telerobotics with a synthetic view integrated into the UI, the implementation already includes some "*intelligent*" components for operator assistance in order to reduce the mental workload. They proposed screen-referenced end-effector control of a manipulator by integrating the calculation of inverse kinematics in the presented view. In a user study, they successfully reduced the number of user-controlled DoF – six of the manipulator mounted on a mobile platform, which has six on its own – by incorporating the placement of the platform into the IK calculation automatically.

Many years before the second VR hype occurred, Burdea [28] explains how VR and robotics benefit from each other by summarizing results from different scientific publications. He especially points out that many typical robotic tasks, which involve humans, are improved when VR technology is utilized appropriately. Involving multi-sensory information in the feedback, presented to the user, increases the level of realism and immerses the user himself, with the consequence of more natural behavior. He emphasizes the potential for increased safety by interacting with a virtual world, but also the potential to build

user-friendly systems with high-level interaction methods in VR.

He also presents concepts for using VR to overcome time delays in teleoperation tasks like a predictive display utilized by Bejczy et al. [21]. The authors propose to render a high-fidelity graphics model of the operator as realtime feedback for the operator, which is drawn over the original on-site video. Their implementation is based on the original idea proposed by Noyes and Sheridan [146] in 1984 for space and underwater applications. Bejczy et al. [21] conducted a user study and tested different amounts of time delay in a 2D tapping task. The activated predictive display improved the task performance significantly, made the operators feel more comfortable and thus increased their operation speed. The authors conclude with the statement that for 3D tasks, stereoscopic rendering or at least a multi-view presentation of the scene might be necessary.

In 1993, some researchers published work on utilizing HMDs or video glasses for delivering a specific view of reality, partially blended with virtual objects, during the operation of robots. Rosenberg [170] presented the “*virtual fixture metaphor*” – an implementation of perceptual overlays, or superimposed haptics – for use during continuous teleoperation of remote manipulators in a typical master-slave robotic setup. The operator is wearing monoscopic video goggles and an exoskeleton at one arm, which serves as the master in a teleoperation scenario. The video glasses provide the operator with a view on the task board, including the robotic manipulator. The view presented to the operator, in combination with the matching coupling of the operator’s motion with the motion of the slave manipulator, trick the kinesthetic sense of the human operator and cause a sense of ownership. Virtual fixtures, guiding the operator’s motion for efficient task completion, are physically implemented using a fixture board, which physically constrains the motion of the operator. During operation, the fixture board and the master device are hidden by the shielding view trough the video goggles. Different fixtures have been implemented and tested in a peg insertion task. The use of virtual fixtures increased operator performance by up to 70%.

Oyama et al. [148] focused on the modeling of VEs, which correspond to the actual operation site. The authors face the issue of manipulator teleoperation in remote environments with poor visibility. They developed a construction and calibration method for model-based environments. The environment models especially contain models of the manipulator and objects for manipulation, which are updated during the operation of the system. An HMD presents a stereoscopic video to the operator from a pair of video cameras for stereo-vision mounted at the robot. The operator controls the manipulator physically with a master-arm. Augmented reality is used to overlay the model-view of the remote side on the actual stereoscopic video from the remote side. Despite some limitations in precision, the authors achieved successful task completion when the remote environment was filled with smoke, and thus the actual video presented no useful cues for operating the robot. Participants of a user study successfully operated the robot using only the rendered virtual information from the augmented view.

In contrast to continuous teleoperation methods, Milgram et al. [130] focused on discrete methods, which require more intelligence at the remote side and reduce the task load for the operator. Instead of controlling the robot by copying the operator's movements, the authors propose high-level concepts and introduce several metaphors utilizing depth information from the remote-side. Thus, 3D overlays on stereoscopic video are intended to support the operator, especially in tasks requiring precise depth perception, meanwhile reducing the amount of time an operator has to spend on commanding a task. Their setup involves a stereo-camera at the remote-site and stereoscopic glasses. The authors claim:

*“By presenting depth information in a direct way to the user, the system reduces some of the complexity of the task. Whereas monoscopic video images often require the operator to interpret shadows and reflections in order to infer a sense of the spatial relations in the remote scene, stereoscopically images present that information in a way that is immediately accessible to the operator, with much less mental processing. This can reduce training times and improve task performance (i.e., faster or more accurate).”*[139]

For assisting the operator in 3D tasks, the authors developed the “*Virtual Stereographic Pointer*”, a graphical object, which is placed in three-dimensional coordinates and overlaid with stereoscopic video images. Cues of depth perception are supported by variable size depending on the distance of the pointer relative to the stereo-camera. Another metaphor is the “*Virtual Tether*”, the implementation allowed for the execution of peg-in-hole tasks with a remote manipulator. The basic idea is to let the operator draw a straight line from the end-effector to the desired location in the stereoscopic view for letting the robot transduce this input to the corresponding self-motion. This method could be rated as a method of supervisory control and thus incorporates only a discrete or command-like action from the operator. The authors conducted experiments that showed the virtual tether method to be advantageous in comparison to conventional control methods and decreased the mean number of errors.

Nguyen et al. [139] point out that for teleoperation in space, such as planetary surface exploration with a complex robotic device, VR is very beneficial for increasing the operator's situational awareness (SA). Thus, an improved understanding of the vehicle's surroundings makes it easier to plan the next commands. Comparison of task performance during direct teleoperation and high-level task planning interfaces revealed the latter to be more efficient for the unique requirements and issues of space robotics in planetary surface exploration tasks, like limited bandwidth, vast and varying delay during data transmission and unreliable connections. So far, the main interest of the NASA in VR was visualization:

*“Furthermore, the human visual system is a very high bandwidth means for communicating complex information to a human operator, but only if that information is properly presented. To take advantage of it, the IMG uses VR techniques.”*[139]



Systems, utilized and developed by the NASA, like “*Virtual Environment for Vehicle Interface*” (VEVI), the “*Nomad’s User Interface*”, “*Marsmap*” (cf. Stoker et al. [192]) and “*CMap*” focused on interactive visualization of spatial data, but also how to generate 3D models in an appropriate way. In order to integrate appropriate robot control into the visualization system and to reduce the required team size of Rover missions, meanwhile allowing multiple users to access the system, “*Viz*” was developed.

Unlike the direct use of AR during the execution of robotic control tasks, AR shows great potential during debugging purposes, when programmers and engineers set up a robotic system or develop a new application. Collett and MacDonald [35] developed a framework and Player<sup>1</sup> plug-in, which helps developers to understand how the robot sees the world. Inconsistencies or deficiencies in the world perception of the robot are presented to the developer. Maintaining the spatial context of the probably erroneous data was proven to be very effective during debugging and helps to find conceptual errors in reduced time. The implementation of the presented concept creates a new intermediate category between simulation and real-world testing.

One step further, by not only displaying internal states and sensor readings using AR technology but also planned actions, is presented by Leutert et al. [112]. Spatial augmented reality (SAR) is generated by utilizing projectors that add information to the surrounding environment – ranging from values of a single variable, over planned actions or trajectories, up to final states of the robot. The authors demonstrate their system, with a drawing-robot which draws on a whiteboard and the SAR system is projecting the image to be drawn on the whiteboard itself.

The idea of projecting robot intentions into environments, which robots and humans are sharing, is further addressed by Andersen et al. [8]. The integration of visual feedback about the current subtask and who is the actor of the actual subtask into human-robot collaborative manufacturing highly improves human safety, effectiveness and satisfaction in comparison to classical methods, showing the relevant information on a computer screen or a piece of paper. In a scenario of human-robot collaboration where humans and robots are performing separate subtasks of manipulating an object, the system shows the human what to do. It projects the task in an iconic way on the surface of the object. During the execution of the robotic subtask, a warning symbol is projected on the object. It reminds the human operator not to conflict with the robot.

Frank et al. [60] use MR techniques for combining the advantages of egocentric and exocentric viewing in UIs and present an interactive RUI for manipulators, which utilizes 6-DoF tracked mobile devices and the touch screen capabilities for intuitive user input. The registration of the mobile device frame within the robot’s workspace follows an optical 2D marker-based approach. The implementation allows the user to adjust the view in the UI very naturally by walking around the workspace and alteration of the device’s pose,

---

<sup>1</sup><http://playerstage.sourceforge.net/>

similar to taking a picture. Additionally, the operators are provided with a good overview of the scenario, like in conventional exocentric perspectives. In a pick-and-place task, the presented system yielded comparable or better user experiences than conventional egocentric or exocentric implementations. The authors compare their system as a visually engaging mixed-reality window for direct interaction with the robot and its surroundings. Especially VR allows system engineers to implement concepts, which are decoupled from physical laws and limiting realism of our real world. Tikanmäki et al. [201] implemented a giant mode for exocentric views in order to overlook extensive terrain data for a better overview. The authors explain that modern CPUs, GPUs and the availability of large storage devices enabled for exploiting pointcloud information about outdoor terrain. The computer-generated environments are then ready for controlling multiple mobile robots. In the setup, a UAV and a UGV are controlled with a VR UI, utilizing an HTC Vive HMD. The UGV is equipped with an RGB-D camera and the UAV contributes RGB images or video from the desired perspective. The operator similarly controls both vehicles, by defining waypoints with drag and drop or by drawing a path through the VE, and benefits from data, gathered by them, which is integrated into the world model during operation time.

Interaction in MR allows for continuous or non-continuous interaction concepts. Regenbrecht et al. [162] compared continuous joystick control with a gesture-based remote control for traveling outdoors with a mobile robot. To avoid unwanted control input, the authors implemented two different controls for the teleoperation of mobile robots indoors. The gesture-based controls are utilizing the trackball metaphor for the continuous control method and a raycasting technique for placing a waypoint within the VE. The trackball implementation uses hand tracking and is activated by closing the hand, meanwhile positional shifts of the hand along with the forward vector cause forward or backward movements of the robot. Turning is implemented as calculating the movement vector from shifting the closed hand sideways to a position relative to the neutral position. In a user study, the advantage regarding unwanted collisions with the environment during the use of the VR-based UI in comparison to a classical desktop implementation with a joystick and 2D screen was demonstrated [221].

The integration of the operator's whole body posture and the use of deictic gestures is explored by Almeida et al. [7]. As one of the main advantages of body tracking and egocentric perception, the ownership illusion, which leads to automatic reactions and reduced cognitive and physical workload, was identified. Almeida et al. [7] tried to induce sensations of telepresence during robotic teleoperation by pursuing consistency between outside sensory feedback and inside sensory proprioceptive, vestibular information. Their concept of robot embodiment in the MR-RUI achieved a decrease in cognitive workload and improved task performance. The implementation involves a mobile robot platform with differential drive and a pan-and-tilt unit (PTU), including an attached camera at the operator's head height, to be controlled. The PTU follows the operators head orientation

along pitch and yaw axes and presents a camera stream, captured by the robot, to the operator. Full-body tracking of the operator is processed to skeleton data in order to extract relevant information for generating control commands for the robot. Deictic control is implemented as pointing in the desired direction of travel based on the arm angle in the sagittal plane, with speed variations, based on the arm angle relative to the operator's coronal plane. The body posture control uses the hoverboard metaphor. Thus, speeds are computed by evaluating the distance of the leading foot to the other and angular speeds are determined by the relative shoulder positions.

### 5.2.2 Challenges for MR-RUI

In 2003, Kuan and Young [107] pointed out which aspects need to be improved for the successful teleoperation of manipulators in compliance tasks. The authors explain the suitability of VR-based telerobotic systems for this kind of task but recommend further development of input and feedback devices like HMDs, gloves and joystick-like devices. Especially regarding input devices, mechanisms for compensating differences between the controlling and the controlled device have to be handled appropriately and effectively. However, also intelligent controllers for improving the cooperation between humans and robots need further development. This also incorporates a set of helpful tools during manipulation tasks. Finally, Kuan and Young mention improving the latency in network-based teleoperation systems for compliance tasks. Thus, the implementation of an effective communication structure is identified as the central issue.

The authors implemented a VR-based teleoperation system with force feedback by a haptic input device. Operators experience collisions of the controlled manipulator at the remote site during operation by transmitting sensor readings to the VR system, which produces appropriate haptic rendering. Visual feedback is displayed on VR glasses. Participants in a user study had to perform a maze-passing task, a surface tracking task, and a peg-in-hole compliance task. Results identified the force feedback helpful for reducing contact forces during teleoperation, intelligent controllers to be helpful in regulation of contact forces to a certain level where needed and additional assisted path planning of the robot's motion to increase the success rates during complex peg-in-hole tasks. The overall VR-based system was summarized to present effective means for teleoperated compliance tasks with a robotic manipulator but still needs to be improved in all of the identified aspects, which were identified above.

Atherton and Goodrich [15] present an ecological augmented virtuality (AV) interface for remote robotic manipulation. The interface is designed to improve SA and to reduce the mental workload during task operation. They concluded a user study with the insights that the AV UI slowed down user performance but increased SA and mental workload. The results were of better quality but showed worse performance. Operators were able to work for a more extended period with the proposed system. Depending on the overall

task, it could be a reasonable trade-off to swap speed for better quality and endurance. The implementation of their ecological AV UI involves a tabletop-mounted manipulator, which is initially planned to be part of a mobile robotic platform. A 3D viewer on a 2D screen provides virtual viewpoints in an exocentric view, which shows a model of the robotic arm, a camera view and a 3D range scan of the environment. All information is integrated into a single display. The robotic manipulator is controlled either indirectly by manipulating the end-effector position in the current computer screen's frame-of-reference, or directly by adjusting single joints in the frame-of-reference of the robot itself. A game controller with several axes and buttons is utilized to alter the virtual view and zoom, to operate the robotic arm and gripper, and to trigger a snapshot of the environment via range scan. In a user study, participants had to pick up small objects and to drop them in a box.

E.g., attention recognition is a hot topic in HRI, and gaze- or heading-based selection techniques provide operators with empty hands, which are then free for other tasks. So, another discussion in MR-RUI development concerns the use of head tracking or eye tracking. While roboticists claim for eye tracking for human-robot collaboration [149], HCI scientists explain why eye-based selection in VR is not ready for implementing efficient means [159]. Palinko et al. [149] found that the decision on this choice has a significant impact on the interaction between humans and robots and concluded that eye gaze is more efficient than the head tracking approach. The authors conducted a study regarding natural interaction with humanoid robots. Participants had to interact with an iCub robot in order to build a tower out of blocks. A robot and an instructed human actor are both holding a block in each hand. The operator triggers the placement of a block by either looking at the block in the hand of the agent and then at the face of the agent or by looking at the agent and then at the block. Comparisons of eye-based gaze with approximated head-based gaze resulted in eye tracking to be significantly more precise and thus effective for this task. The author also constitutes "... *that if eye gaze is not available it might be sufficient to use head pose as the first proxy.*"

In contrast to the previous work in MR-RUIs utilizing HMDs cannot integrate stationary eye trackers, but must integrate the eye tracking device into the HMD. This is the case for most immersive MR setups and involves certain limitations. [159] point out that gaze based selection in VR using recent HMDs is implemented most reliably by using only heading information. Experiments, comparing the conditions a) "*only eye tracking*", b) "*only head tracking*", and c) "*head and eye tracking combined*" revealed "*head-only*" not only to be strongly favored by the participants but also to generate the most significant throughput in a repetitive selection task. The authors suggest that future developments in hard- and software could let combined eye and head tracking methods to be most effective and preferred by users.

MR HMDs like the MS HoloLens provide to add virtual objects on stereoscopic see-through displays. This, combined with inside-out tracking in realtime, allows for blending virtual

elements within the surrounding reality. Virtual objects, placed at fixed positions in the real world, are assumed to stay at their assigned place. For MR-RUI, this is an essential issue regarding the conversion of pose information between different frames-of-reference. In the case of the MS HoloLens, the tracked surroundings are mapped using a 3D depth map with the help of SLAM algorithms. Thus, the accuracy of the tracking influences the pose of the device frame and changes over time would affect how to convert between corresponding poses of the real and the virtual environment. Vassallo et al. [212] analyzed the hologram stability over time and measured a mean displacement error of  $5.83 \pm 0.51$  mm. The error was measured by using an external tracking system and a tracked stylus, which was utilized as a reference input device with high accuracy. Using the stylus, all four corners of a rectangular virtual object were marked in a repetitive manner. Deviations over time were calculated. In order to simulate the use of the HMD in the wildlife, different actions, like “*walking*”, “*sudden acceleration*”, “*occlusion*”, and “*object insertion*”, were performed in order to probably disturb the tracking mechanisms by manipulating the current pose and the surrounding environment. The authors discuss the results to be surprisingly good, but the results are limited to positional shifts and do not include rotational errors of the reference frame which might lead to larger displacement errors on actual points depending on the distance to the origin of the reference frame (cf. Krupke et al. [104]).

### 5.2.3 Recent MR-RUIs

Gaschler et al. [63] address the use case of industrial robot programming in a work cell. Spatial augmented reality is utilized to simplify the programming procedure. A tracked 6 DoF input device and a projector above the worktop are combined to a system for easily programming the robotic system. Using the input device, operators define obstacle volumes and end-effector poses. By defining multiple poses, collision-free trajectories are calculated and executed, if desired. The projector atop is displaying monoscopically the projection of the waypoints, trajectory, obstacle regions and some UI widgets. Additional monoscopic screens display previews of path planning in a 3D view.

Another system for programming pick-and-place and path following tasks into industrial robots is presented by Fang et al. [57]. A cube-like fiducial marker is tracked by an RGB camera and represents the input device for the operator. Visualizations on a monoscopic display show the robotic environment from a fixed perspective. Augmented reality methods are used to blend virtual information on top of the actual video images. These visualizations include the 2D workspace of the robot and visual cues, relevant for robot programming, like the coordinate frame, the collision-free volume, spatial points of interest, end-effector orientations and robot model previews, paths of the end-effector, and exceptions and warnings, occurring during the use of the system. The system is used e.g., to record a path of the moving input cube for later execution with the robotic hardware

and inspections and refinements of the recording before execution is performed.

Rodehutsors et al. [166] present a system for controlling an anthropomorphic robot with multiple operators. The control system for bi-manual grasping is implemented in VR. The operator's head and hands are tracked with 6 DoF. In an egocentric stereoscopic view, the operator sees a rendered model of the robot and surrounding pointcloud and different monoscopic video camera views in the VE, displayed on a 2D canvas, each. The view is displayed in an Oculus Rift DK2 and the input controllers are Razer Hydra with their tracking system. Controller buttons are used to open and close the end-effectors and the manipulators are moved according to a mapping from the operator's hand motion via IK calculation to the robotic arms with a similar structure. The pointcloud view is rendered according to the actual pose of the HMD. It allows for identifying objects for manipulation without the need for detection. The interface implementation relies on continuous control with a superficial intelligence level.

Negishi et al. [137] provided a master-slave system with some intelligence for assisting the operator during operation. Stereo-cameras in the head of a wheeled, humanoid robot provides the operator with a stereoscopic of the actual environment of the robot. Using another stereo-camera in the head of the robot, objects, suitable for grasping are identified. Movements of the head and hands of the operator are measured using flexible sensor tubes (FST) as a master device. An FST is a passively actuated chain of bending and rotation joints. Measurements of the operator's arm movements deliver information to predict a reaching motion and a particle filter is used to estimate the candidate for grasping, evaluating the viewing direction of the operator. Based on the detected objects and the candidate selection, the system assists in pre-shaping the robotic hand in an appropriate way for the object to be grasped.

A mobile tablet represents an exocentric window to MR interaction space in the paper presented by Frank et al. [59]. Tabletop manipulation of several objects for pick-and-place and stacking is implemented in a marker-based approach. Registration of the tabletop and the robot frames within the view of the back-facing camera of the mobile device is done by marker detection. Objects are detected with the help of 2D markers, located on top of them, and appear on top of their exact counterparts in the displayed realtime video view. Virtual clones of the detected objects are ready for direct touch interaction by the operator. Commanding the robot is triggered indirectly by manipulating the position of the virtual objects with typical touch gestures.

Malý et al. [120] utilize either a smartphone with touch gestures or stereoscopic see-through video glasses together with a Leap Motion Controller for mid-air gestures. An industrial manipulator in a small-sized work cell is controlled by one of the methods. The registration process of the operator's and robot's reference frames is done with marker detection implemented in the Vuforia library. The current state of the joints of an actual robot is estimated by evaluating markers at each link. Both the smartphone and the

AR-glasses are utilizing a front-facing camera or marker detection. Several visualizations like robot link highlighting, wireframe rendering of the robot, solid rendering of the robot, text information and arrows pointing at regions of interest are implemented. Robot control methods are very low-level and include adjusting joint positions of the robot.

Intuitive realtime control of a robotic arm by operator's hand tracking with an HTC Vive Controller, which is tracked by HTC's Lighthouse tracking system, is developed by Rakita et al. [161]. From 6 DoF hand pose information targets for the robot's end-effector are calculated by firstly scaling the pose to match the workspace of the robot, and then transforming into the reference frame of the robot's end-effector. These target poses are used to generate smooth motion of the robot with the focus on avoiding discontinuities like singularities and swift motions, caused by substantial changes in consecutive joint configurations. This kind of robotic motion is a trade-off between mathematic accuracy and improved probabilities to operate the robot smoothly. For achieving this, the authors developed an IK solver, which focuses on finding solutions close to the previous or actual configuration instead of serving the target position very accurately. The resulting relaxed re-targeting method is implemented as a distributed system. The input interface is realized using Unity3D on a Windows computer and communicates to the robot interface on a Linux computer via the local network, which is running the re-targeting algorithm and robot control system.

Cancedda et al. [30] present a 3DUI which blends control feedback during the use of hand gestures into the egocentric camera-based view from the remote-controlled robot. The robot is a mobile platform with an attached manipulator. The LMC is utilized to capture hand gestures for either switching between modes of operation, or the actual control of the selected device. Mode choices are among controlling the mobile platform or actuating the attached manipulator. The control feedback uses geometric volumes, which represent command related to their position in the 3DUI, e.g., the volume on the left side is for turning left. Filling a wireframe volume with green color indicates the current execution of the corresponding command. Red indicates loss of tracking and a center volume represents a neutral resting command. Additional rendered information shows the hand position within the command space during operation.

Dyrstad and Mathiassen [52] demonstrated how to use a standard consumer-grade VR setup of an HTC Vive for teaching simulated robot models how to grasp virtual fish objects. In a follow-up work, Dyrstad et al. [53] applied the results from learning in VR to actual fish grasping. The learning mechanisms implement deep learning from demonstration in VR. Training samples are acquired by letting human operators individually grasp virtual fish with physics and rigged skeletons in Unity3D. Grasping is implemented as end-effector control, where the robotic arm is controlled indirectly using inverse kinematics. By wearing the HTC Vive HMD and holding a tracked Vive controller, operators generate input means. HMD and controller are tracked with 6 DoF, each, using the HCT Vive

Lighthouse tracking system. After training of the CNN, the system is applied to generate suitable grasp points from pointcloud data resulting from the observation of real fish in a box with an RGB-D camera. The robotic system is grasping fish autonomously with an industrial manipulator.

The question of how pointcloud data can provide efficient means for manual remote navigation of a mobile platform in a VR-based UI is investigated by Kim et al. [92]. A mobile robot platform is equipped with an RGB-D camera and transmits pointcloud data wirelessly over the local network to a tetherless, mobile HMD, where an operator surrounding octomap is generated from pointcloud data. The octomap rendering of the environment provides sufficient information for controlling the robot within the environment. The software system is composed of ROS nodes for pointcloud data acquisition and the robot control interface, which are running on the robot and a Unity3D UI Android application for the GearVR HMD.

Rosen et al. [167] communicate arm motion intent of a robot by HMD-based MR visualization. ROS and Unity3D are combined high-fidelity rendering at the operator side and effective and versatile control means at the robot side. The Microsoft HoloLens with its standard gestures provides enough means for the operator to communicate the desired target positions for the end-effectors of the two-armed robot Baxter. Target positions are rendered as schematic clones of the end-effectors. Intended motion is decided and computed by the robot, then communicated with the human operator by rendering on the displays of the HMD a static sequence of the planned motion as multiple robot clones with different postures of the robot. The postures represent intermediate configurations of the sampled trajectory.



**Table 5.16:** Comparison of Virtual Fixtures Implementations

Category	Group	Factor	Rosenberg [170]	Krupke et al. [105]
INTERACTION	Interaction Parameter	Action Type	continuous	continuous
		AFR Interaction Level	type A	type A
		Control Alignment	1-to-1	not directly aligned
		Control Order	0 (master-slave)	6 (IK solver)
		Directness	unknown	isomorphism
		DoF	2	5
		Interaction Plasticity	artificial & physical	virtual
		Interaction Type	manipulation	manipulation
		Mapping Relationship	position	position
		Parameter Manipulation Type	direct	direct
	Sensing Mode	isometric	isometric	
	Structure of Interaction	SO,SR,ST	SO,SR,ST	
	Paradigm	Human-Robot Communication	monologue	monologue
		Intelligent Robot Control Paradigm	hierarchical	hybrid
		Robot Control Concept	feedback control loop	feedback control loop
		Role of Interaction	operator	operator
		RUI-Type	real-time	real-time (programming)
	UI-Type	SUI	SUI with (partially GUI)	
	MEDIATION	Image Display Vision	Display Centricity	egocentric
Display Class (Milgram)			class 2	class 2
(Display) Space			remote	registered
Image Scaling			1:1	1:1
Vividness			low-res. video	HD rendering
Input		Action Measurement Type	master-arm	tracker
		Directness of Sensation	direct through video	synthetic world
		DoF	3	6 & 1 button
		Gesture Type	continuous dynamic gesture	continuous dynamic gesture
		Sensor Type	rotational encoders	optical tracking
Out-put	Actuator Type	robot arm	virtual robot, only visual	
	Display Type	HMD	HMD	
PERCEPTION	Embo-diment	Extent of Body Matching	only few correlation	only few correlation
		Extent of Proprioceptive Matching	medium level	no matching
	Immerison	Extend of Presence Metaphor	mixture of WoW and HMD	realtime imaging
		Integration of VE	integrated/only little VE	integrated
		Level of MR	very close to RE	VE
		Reproduction Fidelity	monoscopic video with wireframes	realtime high-fidelity
	Modalities	Causality of Modalities	transmitted	simulated
		Extensiveness	multimodal	multimodal
		Inclusiveness	partially inclusive	almost complete, no audio
		Model of Modality-Source	'scanned'	computed
		Superposition of Modalities	merged	isolated
	User Experience	transmitted	simulated	
	Space & Time	Extent of World Knowledge	partially modeled	completely modeled
		Location	remote	remote
Realtime Level		small delay	almost no delay	
Surrounding		fixed view direction	everywhere	
System Extent		very large	small	
Time (interaction)		1-to-1	1-to-1	
Time (purpose)	real-time	recorded for later		
ACTING	Autonomy	Level of Capabilities	very low	IK, collision detection
		Level of Intelligence	no intelligence	no intelligence
		Plot	partially fits mental models	fits mental models
		Refined LOA	1	1
		Initiative	fixed	fixed
		JIRA Intelligence Level of the Robot	class 1	class 5
	Timely Aspects	static	static	
	Behavior	Robot Behavior Level	remote controlled	remote controlled
		Robot Communication Capability	no communication	color & haptic feedback
		Transparency of System State	invisible	only (probable) collisions
RA	Level of Robot Anthropomorphism	very low	very low	
Appli-cation	Robot Purpose	tool	tool	
	Robot Type	industrial	industrial	



PART

III

MR INTERACTIONS AND VISUALIZATION  
FOR IMMERSIVE HRI



## PRE-STUDIES OF 3DUI INTEGRATION

This chapter presents first attempts to develop MR robotic user interfaces by integrating 3D user interaction, based on empty hand gestures, into pre-existing robotic systems. Furthermore, the utility of visual hand tracking feedback in VR is evaluated.

In Section 6.1 static hand postures alter the locomotion of a simulated modular robot. The mobile control system is integrated into an existing robotic system for simulation and control and different mappings from input to locomotion are evaluated.

Section 6.2 summarizes the results from moving hand gestures in a reach-to-grasp scenario for application with an industrial manipulator. Ballistic phases of reaching hand movements are recorded with a head-mounted hand tracking system. Differences from a closed-loop implementation with visual feedback of the tracked hand in VR and an open-loop setting which presents only the target in VR are compared.

### 6.1 Altering the Locomotion of Chainlike-Robots

Real-time 3D user interfaces in the domain of modular robots are a challenging problem. Modular snake-like or caterpillar-like modular robots have great kinematic capabilities [66], but the drawbacks lie in the lack of flexible and easy-to-use control methods. In particular, due to the large number of degrees of freedom that have to be controlled continuously in parallel (see Fig. 6.1a), it is very difficult to control these in real time by a human operator. Instead, the movements of robot modules are usually automated by embedded control

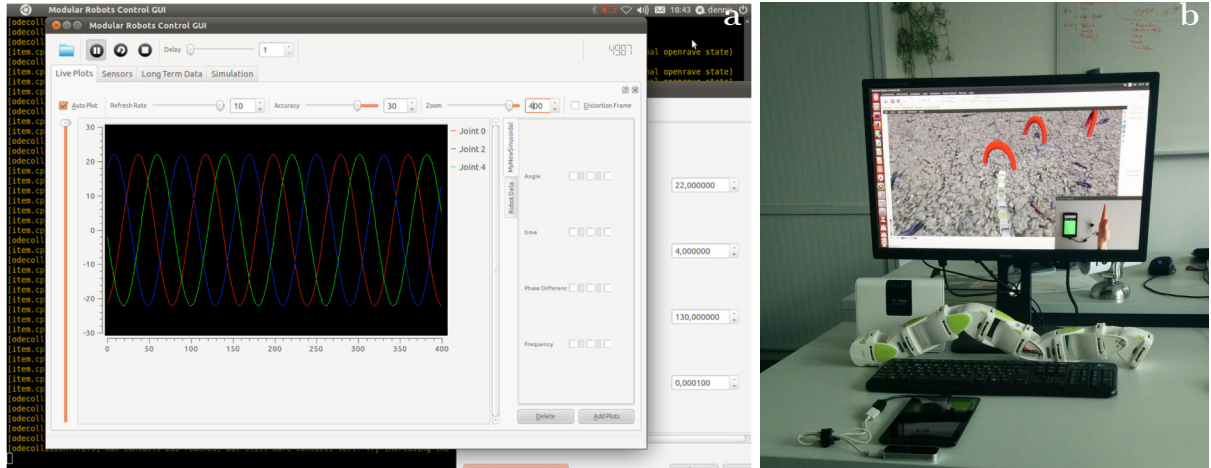
software with sophisticated sensor-driven control loops [88]. Autonomous generation of displacements of the modules of hyper-redundant chain-like robots, i.e., traveling waves [77], is usually realized with sinusoidal generators [66] or central pattern generators (CPGs) [74].

Although, these embedded solutions present significant advances to autonomous locomotion of modular robots, their inflexibility often results in robots getting stuck in terrain that has not been anticipated or pre-programmed [113]. Physics-based simulations are used to optimize locomotion patterns, and specialized settings can be learned to pass individual obstacles [94]. However, because most of these optimizations rely on evolutionary algorithms and reinforcement learning techniques [113], they cannot be applied to real-time applications at the current state-of-the-art.

While it is inherently difficult to define the movements of modular robots in real time, it is possible to define target poses or locomotion goals for semi-autonomous movements. Most existing approaches for human-robot interaction today make use of high-level interfaces, such as defining a locomotion goal by pointing at a 3D position which the robot then tries to propel itself toward [223, 153, 140]. Others tried to utilize a brain interface but the results were less efficient [31]. Some researchers tried to imitate walking movements with the hand meanwhile wearing a special glove but this techniques cannot be easily applied to a modular robot with many joints [93]. It is a challenging question how 3D user interfaces should be designed to provide more direct and natural control over the movements of modular robots [144].

In this section two approaches for natural 3D user interfaces, which allow a user to steer a chain-like modular robot in real time, are presented and compared. Therefore, the user indicates the desired movement translation and rotation of the robot's head with his/her dominant hand in mid-air. In order to reduce the complexity and increase the robustness of the locomotion high-level control function is utilized. Parameters, including the phase difference between neighboring modules, which may cause unstable movement with the risk of undesired turning over, are restricted by trained thresholds. Trained fixed phase differences are efficient in sinusoidal pattern generators and CPGs [80] depending on the topological structure of the robot. In this way, 3D hand movements in mid-air can be converted to movements of modular robots in real time at minimal computational cost. Two approaches are presented and evaluated that may provide interactive, intuitive control to guide a modular robot in an easy and efficient way.

This work is structured as follows. In Section 6.1.1 background information on modular robots and traveling waves is provided. In Section 6.1.2 the prototype is described and two user interfaces for generating locomotion of a modular robot are presented. In Section 6.1.3 a comparative usability evaluation of the approaches, and the lessons learned are presented. Section 6.1.4 concludes this contribution and gives an overview of future work.



**Figure 6.1:** Soft-/Hardware setup: (a) *Modular Robot GUI* environment visualizes the angular positions of the robot’s joints. Angular positions are calculated by the mobile device. (b) Leap Motion connected to an EVGA Tegra Note 7. The workstation runs a modular robot simulation framework [95]. Robot control commands are passed via UDP in the local network. Instead of a virtual robot the real robot can be addressed by the remote.

### 6.1.1 Background in Locomotion of Modular Robots

Self-propelling movements of limbless animals with longitudinal bodies highly depend on the establishment of static frictional forces against the desired direction of travel. For example, locomotion of caterpillars is characterized by repetitive waves of slight movements of body parts that travel in waves from tail to head or vice versa, which can propel the whole body if the slippage against the locomotion direction is sufficiently low.

For actuating the joints of snake- or caterpillar-like robots several methods are commonly used:

- sinusoidal generators [66],
- CPGs [74], and
- pre-calculated patterns in combination with a transition function [222].

In the implementation of the presented system, sinusoidal generators are used to generate smooth waves that are easy to modulate. They offer control parameters including frequency, amplitude, phase difference and offset, which are needed to influence the behavior of the locomotion regarding speed, direction and stability.

The most challenging issue with controlling modular robots is the need to optimize the control parameters for specific situations, which sometimes rely on the properties of the terrain, e.g., height differences, obstacles, friction or external forces [77]. So far, there are no universal autonomous locomotion techniques that are applicable in arbitrary terrains. Reinforcement learning methods are able to train algorithms for passing special

situations [113]. However, these solutions cannot be applied to many other obstacles of different kinds. Direct interactive modulation of the control parameters via graphical user interfaces (GUIs) [95] is challenging, since it requires special knowledge about the impact of certain parameters and their valid range. While such GUIs are often used in laboratory environments to initiate robot locomotion, these solutions usually suffer from low performance and unintuitive control. Alternative input methods, such as hardware remote devices, to modulate the control parameters have been found to raise similar issues [144]. So far, no user interface exists, which provides intuitive teleoperation of modular robots in situations, when it becomes necessary to switch to manual control for passing difficulties that cannot be traversed autonomously.

### 6.1.2 Description of the Teleoperation System

The setup consists of a mobile device connected to a Leap Motion Controller (see Fig. 6.1b), a chain-like modular CUBO robot with wireless communication, as well as a workstation and a network router.

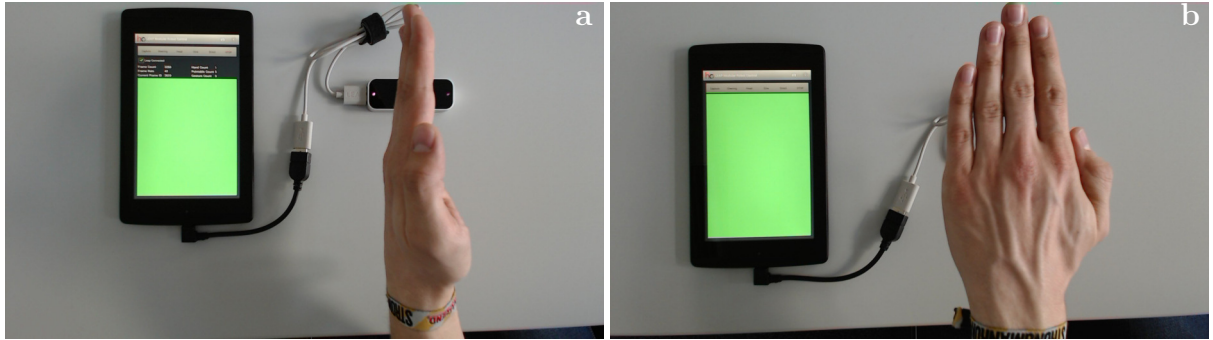
Movements of the robot modules are controlled remotely with hand movements tracked by the Leap Motion in mid-air.

#### *Prototype Setup*

The Leap Motion is connected directly to an Android device via an USB-OTG adapter. In the experiments a Google Nexus 5 smartphone and EVGA Tegra Note 7 performed very well with tracking framerates from 30 fps to 60 fps. As desired for a teleoperation system, a fully mobile system is achieved in this way. An Android application installed on the mobile device works as a control unit for the robot and processes the sensor data from the Leap Motion. Sensor processing and robot control are performed in different threads in order to achieve a high responsiveness of the system with low latency. The Android SDK alpha of the Leap Motion is used to acquire data frames. In the preprocessing step valid sensor frames are filtered by the Android application to increase the reliability of the control system. Visual feedback for the user is implemented by using a colored widget. “*Green*” indicates good positioning of the hand. “*Red*” reminds the user to place the hand closer to the center of the interaction box spanned by the Leap Motion in order to maintain the tracking of the hand. Captured position and orientation of the captured hand’s palm is translated to control parameters that determine the output of the sinusoidal generators, running on the Android tablet.

The control of the CUBO robot (cf. Figure 6.1a) is implemented with Bluetooth sockets that enable serial communication via the RFCOMM protocol. After the establishment of a connection between the mobile device and the robot commands for locomotion can be sent to the robot. A single command consists of the module’s address and the desired





**Figure 6.2:** Hand interaction: Posture and movements of the hand in the (a) rotational control method, and (b) translational control method.

angular position. In each step of the locomotion cycle every module’s joint position needs to be updated. A high updating frequency and low latency are needed to achieve smooth locomotion patterns. These real-time constraints are the reason why smoothing algorithms do not fit the requirements and direct control is important. The GUI of the Android application is used to display data from the sensor and to manipulate the connection state of the robot using 2D touch interaction.

### *3D User Interfaces*

Two direct 3D control methods are implemented, which are based on hand movements in mid-air that are transferred to the robot’s head movements. Continuous hand gestures and postures are analyzed to extract locomotion parameters. These parameters are fed forward to sinusoidal pattern generators in the mobile device which are capable of generating 3D traveling waves on-the-fly. After calculating the angular positions, these are applied directly to the head joints of the robot. To create locomotion that results in self-propulsion of the robot the other modules are addressed subsequently with a short time delay that accumulates with the increasing number of modules.

The interaction design presented in the following is the result of focus studies with experts and novices in the domains of human-computer interaction and robotics. The two techniques are based on tracking hand postures and have in common that the palm position of the hand can easily be detected by the Leap Motion. This effectively increases the probability of successful hand recognition and minimizes tracking loss, i.e., problems caused by occlusion in skeleton tracking algorithms are avoided. The two techniques are briefly described in the following:

- *Type I: Rotational Control*

The first design of hand-based robot locomotion is based on hand recognition and palm tracking. The basic static posture of the hand is shown in Figure 6.2a. In the control loop, values of the yaw-orientation and the sagittal axis of the currently tracked reference point of the palm are calculated and transformed to parameters

of the sinus functions that directly affect the locomotion of the robot. Steering to the left or right of the robot is initiated by turning the hand in yawing direction. Absolute movement of the hand forth along the sagittal axis increases the forward speed, while translating the hand back causes the robot to move backward with speed relative to the amount of the hand movement. The resulting forward or backward speed of the robot is computed using the deviation of the position from the reference point of the tracked hand relative to the zero position of the Leap Motion's sagittal axis. The maximum speed of the robot was set to 0.05 m/s.

- *Type II: Translational Control*

The second design follows a different approach for steering the robot. When operating the robot, the hand palm is always parallel to the Leap Motion sensor's base plane as shown in Figure 6.2b. Turning to the left or the right is initiated by a translational movement of the hand along the lateral axis of the Leap Motion. Hand movements along the vertical axis with or against gravitation direction are mapped to the backward or forward speed of the robot, respectively. The same maximum speed as for the rotational control technique was applied.

### 6.1.3 Evaluation

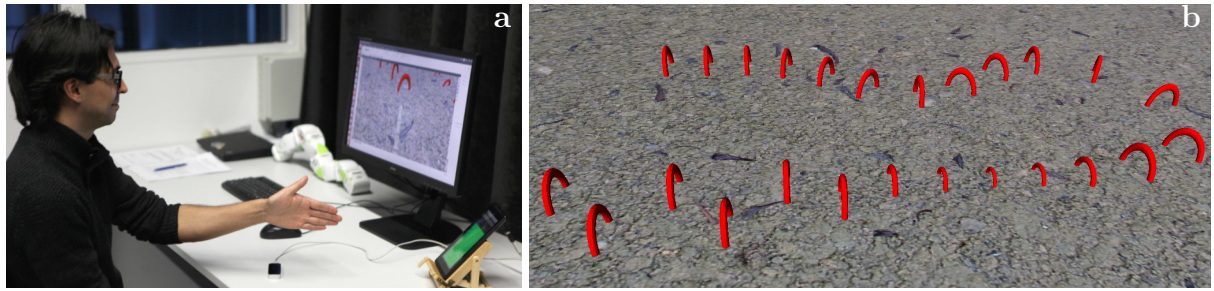
To account for variability in real-world modular robotics studies the experiment was conducted using the in-house “*Modular Robot GUI*” simulation and robot training environment, which is a fine-tuned software framework for the modular robot and provides good ecological validity of simulated to physical movements.

#### *Participants*

12 participants were recruited for the experiment, 7 male and 5 female (ages from 23 to 54,  $M = 35$ ). The participants were students or professionals in human-computer interaction or robotics. Participants were naive to the experimental conditions. 9 participants reported that they were right-handed and 3 reported that they were left-handed. In the experiment they completed the spatial tasks with their dominant hand. None of the participants reported known visual or motor disorders.

#### *Materials*

The experiment was conducted in a laboratory environment using the prototype setup described in Section 6.1.2. Participants were seated at a desk in front of a 24-inch screen as illustrated in Figure 6.3a. An Intel Core i7-4930K 3.4 GHz computer with 16 GB RAM and Nvidia GeForce GTX 780 Ti graphics card was utilized for the simulation of the MRGUI environment. The environment was rendered with the Coin3D engine.



**Figure 6.3:** Experiment setup: (a) Participant seated in the laboratory during the experiment while steering the modular robot using his dominant hand. (b) Parcours for the usability comparison of the different control methods. Participants passed the gates with the modular robot in the given order.

The task environment for the modular robot consisted of a ground surface with gates of 7 cm to 10 cm diameter in the environment at distances of 8 cm to 12 cm, which participants had to pass with the modular robot. The test environment was designed in a way such that once a gate was passed by the modular robot it disappeared and its kinetic body was removed from the scene. The environment was rendered using a third person camera, positioned behind the robot’s head, to provide always good visibility to the robot, independent of its position and orientation in the environment.

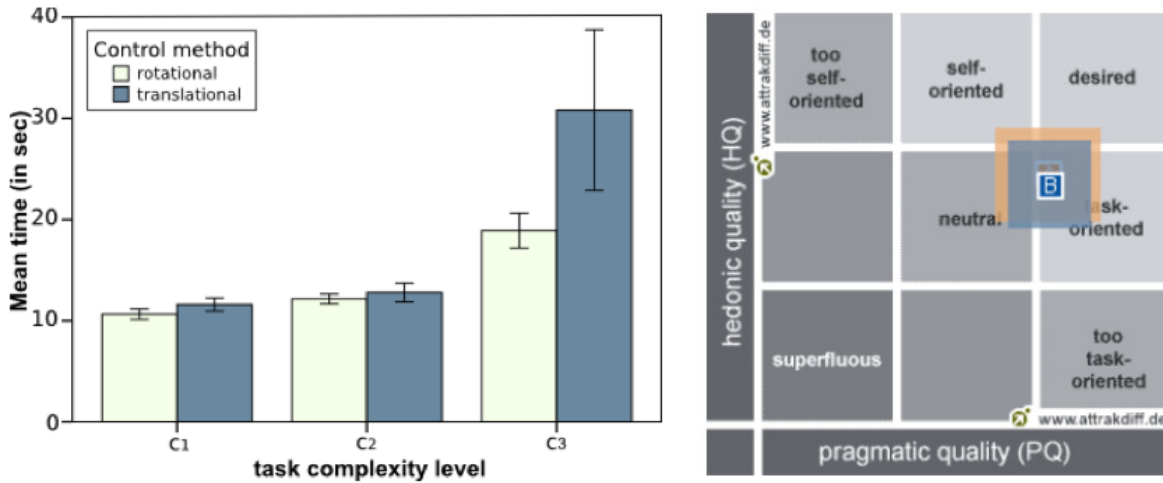
### *Protocol*

Participants were instructed to steer the modular robot through the gates as fast and as accurately as possible in the order that was shown in the 3D environment. If participants missed a gate they had to backtrack their path with the modular robot to complete the task. Three task complexities were considered in the experiment: (C1) gate diameter 10 cm with slight lateral displacement between gates, (C2) gate diameter 7 cm with medium lateral and rotational displacements, and (C3) gate diameter 9 cm with large lateral and rotational displacements. To account for the expected durations participants had to pass 10 gates in both conditions C1 and C2, and 5 gates in condition C3.

### *Methods*

In the experiment  $2 \times 3$  repeated measures within-subjects design was implemented. The independent variables were the control method (rotational vs. translational) and the three task complexities (C1, C2 and C3). The dependent variable was the time it took the participants to pass the gates during the experiment.

Furthermore, demographic information was collected with a questionnaire before the experiment and measured the participants’ task load with the NASA TLX questionnaire as well as the sense of attractiveness with the AttrakDiff questionnaire. After completion of the tasks, informal responses were collected from the participants and they were asked to provide qualitative feedback related to the two tested steering methods.



**Figure 6.4:** Experiment results of continuous hand-gesture-based robot control methods. Left: Gate passing times for the two control methods and three task complexities. The vertical bars show the standard error of the mean. Right: Results of the AttrakDiff questionnaire along the dimensions of pragmatic and hedonic quality. Method A corresponds to the rotational control and method B describes the results from the questionnaire about the translational control.

The participants were allowed to take breaks between the conditions. The total time per participant including pre-questionnaires, instructions, experiment, breaks, post-questionnaires and debriefing was 30 minutes.

### Results

Figure 6.4a shows the pooled results for the gate passing times for the two control methods and three task complexities. The vertical bars show the standard error of the mean. One participant was removed from the analysis due to a technical problem.

The results were analyzed with a repeated-measures ANOVA and Tukey multiple comparisons at the 5% significance level. Degrees of freedom were corrected using Greenhouse-Geisser estimates of sphericity when Mauchly's test indicated that the assumption of sphericity had been violated.

A significant main effect was found between the different task complexities on gate passing time ( $F(1.04, 10.36) = 9.59, p = .01, p2 = .49$ ). Additionally, a trend for a main effect between the different control methods on gate passing time ( $F(1, 10) = 2.96, p = .12, p2 = .23$ ) was identified.

Post-hoc tests revealed no significant differences in the gate passing times between the two control conditions for the different task complexities ( $p > .05$ ). For rotational control, the gate passing times were significantly different between C1 and C2 ( $p < .05$ ), between C1 and C3 ( $p = .001$ ) and between C2 and C3 ( $p = .002$ ). For translational control, the gate passing times were significantly different between C1 and C3 ( $p < .05$ ) and between C2 and C3 ( $p < .05$ ).

The NASA TLX data was analyzed for the different metrics. The mental demand was

$M = 18.25$ . Physical demand was higher with  $M = 56.75$ . Temporal demand is rated with  $M = 24.2$ . Performance reached  $M = 28.0$  and effort is rated with  $M = 32.1$ . The participants judged frustration with a low score of  $M = 23.1$ . The overall rating reached  $M = 30.4$ .

The results of the AttrakDiff questionnaire are shown in Figure 6.4b. The results show the tendency to prefer the rotational control method. In general, the hedonic and pragmatic quality indicate the suitability of the presented method to manual control of modular robots.

The collected qualitative feedback supports the quantitative results and generally indicates very positive judgments of the ability to steer the modular robot with both techniques. 10 participants preferred rotational control over translational control. None of them preferred the translational control method and 1 participant reported no strong feelings about the preference.

### *Discussion*

The results show that both control methods reach a similar performance for the C1 and C2 task complexities, but suggest a tendency towards a difference for C3. The qualitative feedback confirms this result, indicating that both techniques reached similar high attractiveness scores, but a tendency that participants preferred the rotational technique in more complex situations.

### *Lessons Learned*

The results show that mid-air hand gestures provide a reasonable and intuitive way to steer a modular robot through a terrain with obstacles. However, it is desirable to enable intuitive control over the entire body of such chain-like robots, not just the head. This may become possible by inducing traveling waves via wave-like hand gestures. Although such approaches may be leveraged as a natural extension of the described head-centered steering techniques, the initial results suggest that such approaches require very good motor skills and learning of a very limited set of wave gestures that can generate self-propelling locomotion of modular robots, whereas most wave gestures will not result in the robot moving from its current position even when the modules are moving. Future 3D user interfaces for steering such modular robots will combine both head-based steering as introduced in this work as well as direct control over the robot's body via more complex hand gestures. Such hybrid approaches can support users to intuitively steer robots over light terrain while being able to pass more complex obstacles in case the robot becomes stuck.

### 6.1.4 Conclusion

In this work two 3D user interfaces for intuitive control of the 3D movement patterns of chain-like modular robots were presented. An experiment was performed which showed that the proposed techniques are effective in moving a modular robot and easy-to-use. All participants were able to steer a modular robot through heavy terrain with the techniques. The two proposed techniques reached similar task performance, with rotational hand gestures being subjectively rated higher than translational gestures in difficult steering situations. It is likely that the presented approaches have the potential to be used as an effective, portable solution to take over control of modular robots, as well as a versatile platform for teaching and testing of novel control loops in human-robot interaction.

The design and implementation process turned out to be very cumbersome and did not result in a generalizable and practical solution. Thus, further work avoided specialized solutions like the MR-GUI, designed for chainlike, modular robots. Using the Android-SDK directly to program a remote control application turned out to be a limitation regarding the limited device support, which are technically prepared to be connected to a mobile device and supported by its operating system.

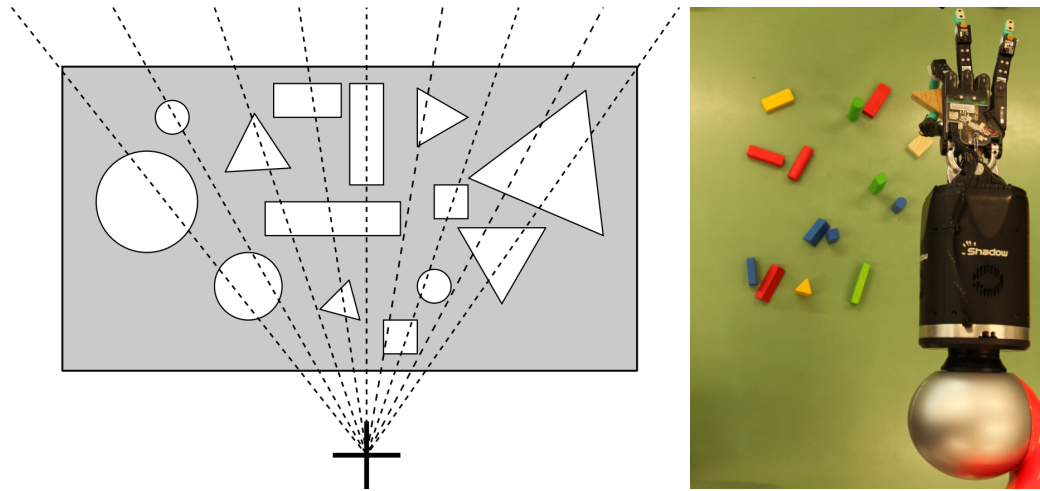
Instead, middleware like the robot operating system (ROS) and Unity3D seem to be promising for enhancing 3D interaction between humans and robots, generally.

## 6.2 Analyses of Spatial Ballistic Movements for Prediction of Targets in Reach-to-Grasp Tasks

Ballistic and correction phases of hand movement trajectories in reach-to-grasp tasks are recorded for further categorization and to analyse the suitability for creating a predictor of target objects. The results suggest that the index of difficulty (ID), according to Fitts' Law, has no influence on the speed of reaching movements, but seems to determine the shape of the velocity versus time relation in the virtual experiment. “*Closed-loop*” and “*open-loop*” conditions (cf. Section 3.3) in ballistic aiming movements result in similar effects between maximum speed and distance of objects. The results provide important findings for interaction with 3D objects as well as human-robot collaboration, which allows for more robust and efficient interaction techniques in real-time scenarios.

### 6.2.1 Problem Description

(Neuro-)psychological and robotic research on reach-to-grasp tasks have shown that the pre-shaping phase of the human hand allows a prediction of the object a human is going to grab [43]. Multiple studies were conducted, including studies with physical objects, memorized objects and virtual objects that had to



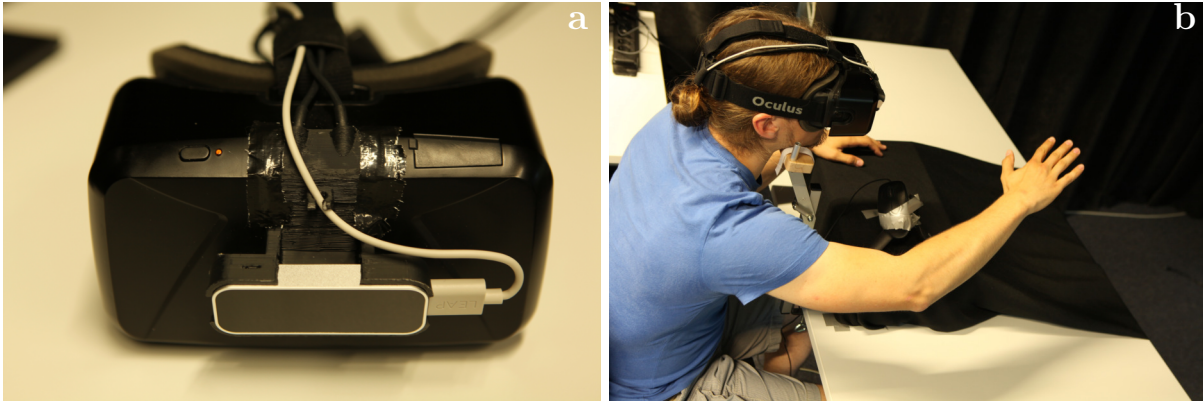
**Figure 6.5:** Left: The ambiguity in prediction of reach-to-grasp hand movements. Right: A robot grasping objects in a cluttered table scene.

be reached and grasped. Research has shown evidence that only a few variables have an impact on that prediction [109, 115]. These insights from (neuro-)psychological and robotic research are promising and it is likely that the information of the reach to grasp phase can be used to substantially reduce the ambiguity in selection tasks (cf. Figure 6.5).

### 6.2.2 Methods

The experimental setup includes an Oculus Rift DK2 head-mounted display (HMD) with optical head tracking and Leap Motion sensor as illustrated in Figure 6.6(a) for hand tracking. Unity3D 5 was utilized to generate the virtual environments for the experiment. Reach-to-grasp movements are sensed with the Leap Motion and recorded for further analysis. Ballistic phases of the hand movements in tasks requiring different levels of precision are evaluated in both, open-loop and closed-loop scenarios, in which optionally feedback as visualization of tracked hands is displayed within the HMD. Repeated measures of different conditions were used to examine the relationship between hand movement over time and the position and size of the target object in order to examine insights for generating a model for predicting of human reach-to-grasp actions. As illustrated in Figure 6.6(b), the initial pose of the participant is limited by placing the head on a chin rest and the preferred hand on a fixed mouse. After keeping one button pressed a target sphere appears, which has to be touched with the preferred hand via a fast hand movement.

In the first experiment objects are presented on the line perpendicular to the centerline of the participant’s eyes with *varying distances* between 20 cm and 50 cm within arm reach. Recording of trajectories is started by releasing the button and stopped when touching the surface of the presented target sphere.



**Figure 6.6:** (a) Oculus Rift DK2 with a Leap Motion Controller mounted on the front. (b) The experimental setup ensures low variance of a participant's initial position and movement variability by using a chin rest and a fixed start pose.

A second experiment explores the characteristics of trajectories in reach-to-grasp tasks of *different levels of difficulty* (cf. Figure 6.8) by increasing the required accuracy to successfully complete the given task. The index of difficulty (ID) in a Fitts' Law experiment is determined by the size of objects ( $W$ ) and the distance between starting position and object position ( $D$ ) [43].

$$ID = \log_2 \left( \frac{2D}{W} \right) \quad (6.1)$$

Trials with  $ID < 3$  can be regarded as ballistic [62].

### 6.2.3 Open-Loop vs. Closed-Loop in Ballistic Aiming

Figure 6.7 shows the correlation between the maximum velocity of hand movement and the target distance (left) and the target size (right) in reach-to-grasp movements. The maximum velocity appears as a linear function of the object distance. The open-loop experiment (cf. Figure 6.7, top row) shows more variance in contrast to the closed-loop experiment (bottom row). Except a tendency of distance underestimation in the open-loop condition, there is no similar effect on object size in the experiment.

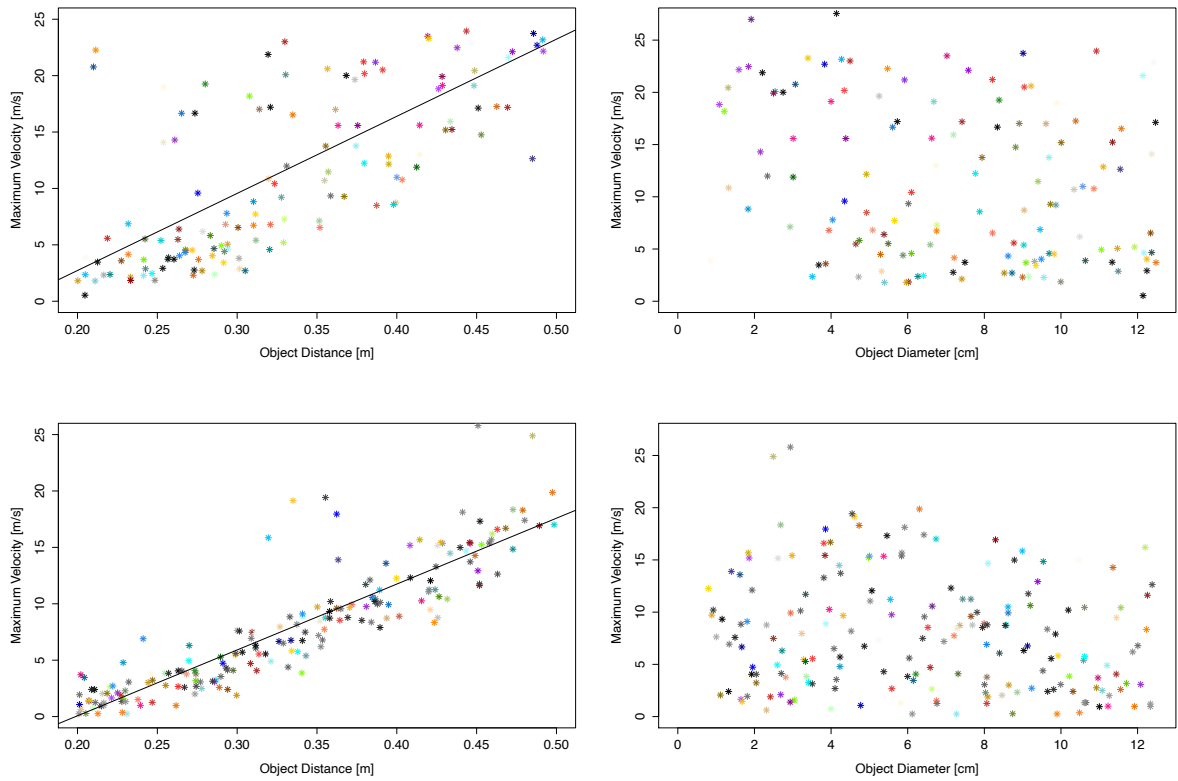
### 6.2.4 Characteristics of Velocity Trend

Generally, velocities increased with time during touch movements until the target is reached. However, depending on the given task the characteristics of acceleration are different. A simple task with  $ID < 3$  resulted in a simple ballistic movement with quadratic acceleration. By increasing the ID, movements take more time, since there is a correction phase, which causes participants to slow down their movements (cf. Figure 6.8).

### 6.2.5 Discussion and Conclusion of the Pre-Study

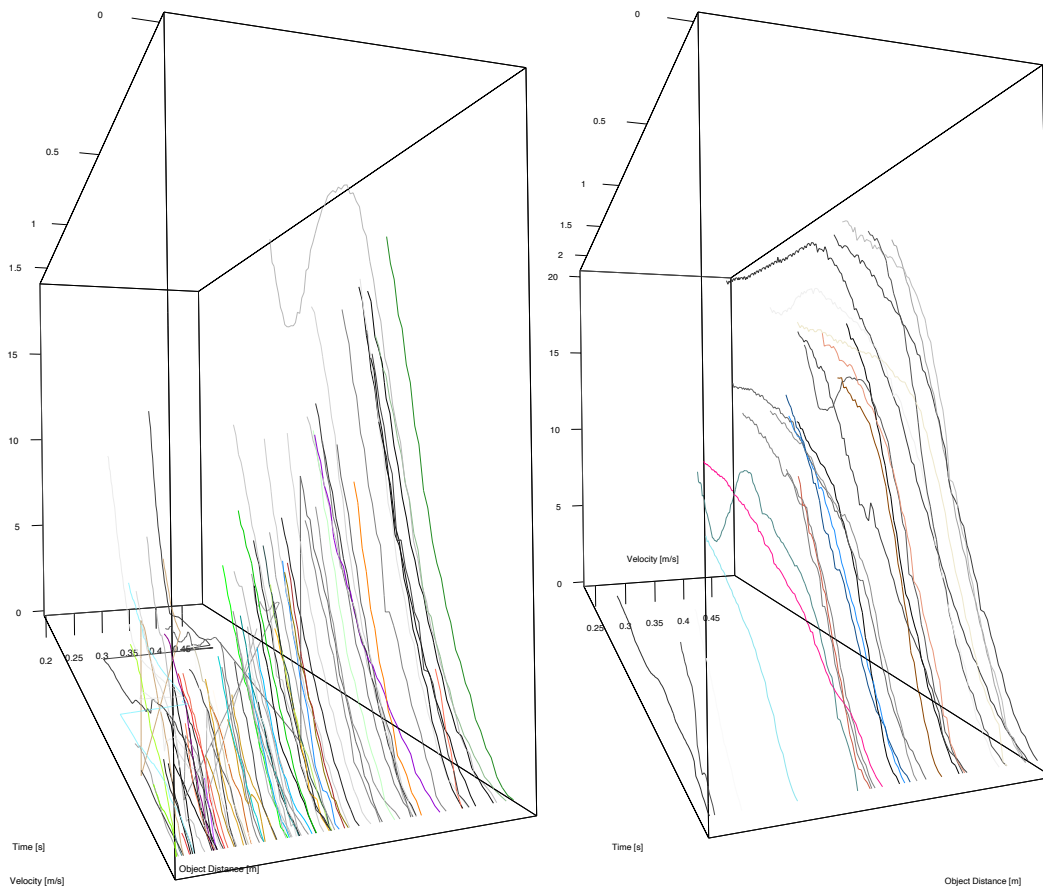
In both, open- and closed-loop scenarios with very low difficulty a linear correlation between the object distance and the maximum speed of the hand movement (cf. Figure





**Figure 6.7:** Comparison of (top row) open-loop and (bottom row) closed-loop scenario. Left: Object distances vs. maximum velocities. Right: Object sizes vs. maximum velocities.

6.7(left)). No such a correlation between the object sizes and the maximum speeds was found(cf. Figure 6.7(right)). Analyses of reach-to-grasp movements provide enough information to reduce the ambiguity of classical pointing actions (cf. Figure 6.5). By classification of the current movements and the analysis of the hand movement speed over time it is possible to estimate, which of the objects on one line that originates at the corresponding shoulder of the user's active hand is intended as target.



**Figure 6.8:** Recorded velocities over time vs. object distances. Left:  $ID \in (0, 3]$ , right:  $ID \in (5, 7]$ .

## PROTOTYPING IMMERSIVE INTERACTION

During the first steps of this thesis, there was no general framework for immersive robotics available. Instead of utilizing such a framework, many researchers implemented specific cases for exploiting state-of-the-art consumer-grade devices together with robotic hardware [166, 7, 172, 122]. As confirmed by several publications, one solution is to combine ROS and Unity3D [156, 34, 215, 86, 133]. Chapter B presents the concept and several details on the software framework, which was developed for this thesis and as a universal framework for MR-RUI [103]. The framework combines Unity, ROS and the Robot Web Tools and enables to prototyping and implementing MR-RUI of arbitrary kinds.

Using the proposed system, versatile HRI scenarios can be set up with a moderate level of effort. The use of a network layer avoids location-based boundaries. It allows for the implementation of teleoperation and teleprogramming scenarios in VR, MR, or AR, as well as systems with shared autonomy. The same scripts for interfacing actual robotic hardware via the network can also be used locally to control virtual robots, which are often used instead of real robots to evaluate human factors or psychological aspects in robotic user interface design.

In this chapter the necessary steps for prototyping MR-RUI with the proposed framework are explained in Section 7.1 and together with Chapter B it provides an aid for other researchers on implementing MR-RUIs. Example use cases are presented in Section 7.2.

## 7.1 Prototyping Workflow

Prototyping an MR-RUI will typically involve at least one of the aspects illustrated in Figure B.1. Depending on the purpose of the prototype some or all of the following aspects of an end-to-end-system combining ROS and Unity for HRI are required to be implemented.

- Import of a robot model to a Unity3D scene.
- Implementation of scripts that actuates the actual robot and/or its digital twin.
- Integration of interaction devices which generate control means.
- If unidirectional or bidirectional communication with ROS is desired, necessary ROS messages need to be implemented and generated.
- Programming of ROS nodes for generating outgoing or receiving incoming messages.
- Data visualization in Unity3D of computed data or sensor data.

MR-RUI research and development not necessarily requires a complete end-to-end-system. E.g., for prototyping an interaction scenario, which aims to investigate a specific input aspect, an actual robot is not necessary (cf. Section 8.1). Thus, a part of the prototyping process is to decide which components are required to be implemented. Technical suggestions regarding the ROS and Unity3D setup are presented in the following subsections.

### 7.1.1 Preparing the ROS-Side

For interfacing ROS-driven robots or sensors from Unity3D, a properly installed ROS instance running on Ubuntu Linux is typically necessary. In the experiments, presented in this thesis, ROS Indigo on Ubuntu 16.04 and ROS Kinetic on Ubuntu 18.04 were successfully tested with the corresponding ROSbridge version for network-based communication together with a wide range of Unity3D versions. The following software components have to be started from a terminal at a computer with a proper ROS installation:

- `$ roscore`  
Starts the basic ROS environment.
- `$ roslaunch rosbridge_server rosbridge_websocket.launch`  
Starts the ROSbridge node, which determines the IP-address and websocket port.
- `$ rosrn` or `roslaunch` starts additional nodes for processing incoming messages or generating outgoing messages.

For combining ROS and Unity3D the implementation of arbitrary custom nodes at the ROS-side is necessary.

### *Developing ROS Nodes*

Knowledge gained from the implementation of “*publishers*” or “*subscribers*”, as documented in beginner’s ROS tutorials<sup>1</sup>, is sufficient to implement custom ROS nodes.

*Publishers* are suitable for sending updates periodically like a robot posture, a single sensor reading or sending video frames from a camera. Even if it is not intended, it is possible to use publishers for transmitting data irregularly on demand, e.g. after receiving a special message. ROS offers “*service calls*” if more versatile behavior is desired, but simple publisher-subscriber node might be enough. When the node has subscribed to a dedicated topic publishing empty messages, it could be implemented to response to an empty message with publishing a specific answer.

*Subscriber* nodes are suitable for receiving input from Unity3D in the form of any standard or custom ROS message, like a goal pose for a robotic end-effector. The subscriber node at the ROS-side has then subscribed to a topic on which messages are published from a script running in the Unity3D program.

### *Alternative ROSbridge Implementations*

As shown in Figure B.12 (right), the implementation of the ROSbridge is a useful tool, which adds only low latency to the system during data transmission via a single topic. Since, the ROSbridge server is running on a single computer this is a limitation and the simultaneous processing of several messages from different topics would increase the latency. For the implementation of very complex systems it is useful to distribute the communication overhead to multiple nodes or to extend the ROSbridge server towards the use of multiple websocket connections. Another issue is the transfer of large data amounts, e.g. pointclouds. Efficient treatment in both cases is done by using binary JSON (BSON), like the authors of SIGVerse report [81, 82, 133].

## **7.1.2 Preparing the Unity3D-Side**

In general, there are no limitations on the specific Unity3D version. Early tests with the ROSbridge for this thesis were successfully implemented using version 5.4. For working with the Microsoft HoloLens there are some limitations. Some combinations of specific versions of the MRTK, Unity3D and Vuforia did not work well regarding specific features, like the marker detection and speech recognition. Unity3D 2017.1.0f3 was experienced to

---

<sup>1</sup><http://wiki.ros.org/roscpp/Overview/Publishers%20and%20Subscribers>

work well with Vuforia SDK 7. This combination was the upper limit for implementations of the topic in Chapter 9.

At the Unity3D-side the following software components are essential in a MR-RUI scene:

- ROS-bridge manager (cf. Section B.3.2)
- URDF importer (cf. Section B.4.1)
- virtual robot controller (cf. Section B.4.1, Section B.5)
- incoming/outgoing messages (cf. Section B.3.1)
- message processing (cf. Section B.3.1, Section B.3.2)
- message generation (cf. Section B.3.1, Section B.3.2)

### *Configuring the Communication*

After running `roscore` and starting the ROSbridge server, the IP-address and the open port is known and needs to be setup in the `RosBridgeManager`. If subscriptions to topics from the ROS-side are desired, then these have to be configured using the corresponding property frame in the Unity editor.

Couplings of topic names and message types presume to be assigned before using it. Otherwise the (de-)serialization and desired processing of information will not occur.

### *Importing the Robot and/or Environment Model*

Importing an existing robot or environment description file from ROS using the asset described in Section B.4.1, presumes the file to be in “`.urdf`” format. In ROS “`.xacro`” is very common, as well, since it offers comfortable macro definitions. ROS includes effective mechanism to expand the definitions to URDF format and to export a file to the disk using the following command:

```
$ rosrunc xacro xacro.py %filename%.xacro > %filename%.urdf
```

If the format of 3D mesh files is not supported by the importer, which relies on the availability of libraries for mesh import into Unity3D, it is possible to convert the files manually with appropriate third party tools like “*Blender*” or “*MeshLab*” and then replace the filename extensions within the URDF file before starting the import process.

In order to actuate a robot model’s joints, it is reasonable to attach dedicated components to the corresponding gameobjects in the Unity3D editor (cf. Figure B.23). When using the importer (cf. Section B.4.1) in combination with BioIK (cf. Section B.4.1) joint components from the BioIK assets will be added automatically.

### *Writing Controllers for Virtual Robots*

Using the importer makes sure to use the very same names from the URDF for the gameobjects in Unity3D, as well. This makes the implementation of tasks like applying the current state of the actual robot to the robot model easier. Joints, which have to be adjusted to a certain angle, can be easily found by name and actuated in a very comfortable and generic way. Thus, information from a `JointState` message is ready to be applied directly in an appropriate robot controller script. More details on this topic are presented in Section B.4.1.

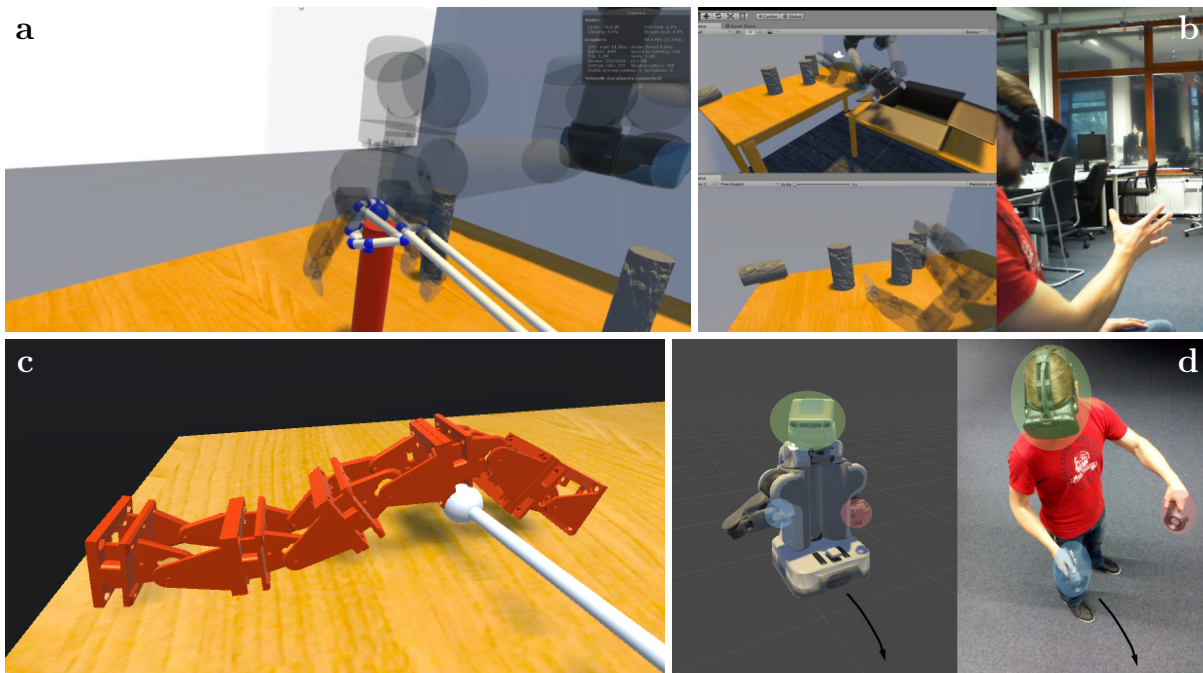
## 7.2 Use Cases

This section covers a selection of use cases for the application of mixed reality robotic user interfaces. Due to the versatility of available hard- and software components for developing MR-RUI, the use cases represent only a small subset of possible scenarios.

### 7.2.1 Programming of Pre-Grasping Poses

In Figure 7.1(a) a virtual model of a wall-mounted industrial manipulator, consisting of a UR-5 arm (cf. Figure 3.5(b)) and a Robotiq 3-Finger Adaptive Gripper (cf. Figure 3.5(d)), is used to specify a pre-grasping pose for an actual manipulator. Instead of defining the pose numerically or graphically using a 6-DoF marker widget, a mid-air gesture is performed in the area of a target object in order to define the 6-DoF pose, which defines the pre-grasping position and orientation of the gripper. The virtual manipulator moves instantly to the desired grasping pose. After the operator confirms, the solid robot model performs the full trajectory.

Technically, the specified pose is used as the desired pose for the tool-center point (TCP) of the manipulator for trajectory planning. Assuming that the objects in the virtual world and the real lab scenario are properly synchronized, this pose can be passed to ROS for trajectory and grasp planning. Technically, for proper grasp planning a direction for approaching the object is necessary and a 3D point with a certain distance to the actual grasp point. It is computable by sending a ray from the specified point in the desired direction and by checking a probable collision with the surface of the object to be grasped. After approaching the graspable object, the TCP of the gripper and the grasping point at the surface of the object match in Cartesian space [104]. The input, specified in the proposed way, generates valuable information for a learning-based system. Then, the system is able to model preferences of human operators in this way. A LMC mounted at the front of an HMD (cf. Figure B.30(c)) was utilized for generating hand tracking data.



**Figure 7.1:** (a) Utilization of hand tracking enables users to define at which position objects should be grasped. Additionally the direction for approaching the object can be defined by extracting the normal of the hand palm using the same gesture. (b) Leap Motion Controller mounted on a head-mounted display provides hand tracking in viewing direction. The participant is embodying the industrial manipulator by means of 6-DoF tracking and inverse kinematics calculation. (c) Tracking of the HTC Vive Controller enables the user to use a virtual stick, which pushes the robot’s joints to a new shape. (d) Multi-joint teleoperation of a PR2 model using room-scale position tracking of the HTC Vive headset, rotational head tracking and pose data of two trackable controllers.

### 7.2.2 Teleoperation of Industrial Manipulators

Regarding the topic of realtime teleoperation of industrial manipulators, there are several questions which are not clearly answered:

Q1: What is a suitable mapping when different degrees of freedom between the operator’s arm and the robotic manipulator exist?

Q2: How should be dealt with different scalings, like speed and size?

Q3: How is efficient teleoperation possible when different amounts of delay occur?

In realtime teleoperation these issues are closely coupled with the question how to present appropriate feedback to the user. Visual or haptic feedback, which relies on real sensor data from or around the robot is already affected by the different kinds of delay. Using a MR user interface it is possible to present additional information like transparent overlays similar to a predictive display [21] via evaluating tracking data during the user’s actions (cf. Figure 7.1(a,b)).



In general a MR-RUI teleoperation setup offers three sources of motion information. The “*operator’s current movement*”, the “*robot’s current movement*” and the “*desired target pose*” or “*planned motion*”. Caused by the motion capability limitations of the robot – e.g. speed and acceleration – and different physiologies, the operator’s movements and the robot’s movements cannot be synchronized directly. Thus, it is of scientific interest to analyze which concrete visual feedback the operator should be provided with.

In a pre-study different kinds of visual feedback methods for VR are evaluated during a manipulation task. Users are asked to grasp cylinders and to put them in a box (cf. Figure 7.1(b)). The VR setup consists of an Oculus Rift DK2 HMD with an LMC mounted on the front of the HMD (cf. Figure B.30(c)) with activated positional tracking via the Oculus camera. Object grasping is induced by midair grasping of the tracked operator’s hand. The operators actual pose is applied to the motion planner of the robot and via solving IK trajectories from current robot pose to the desired pose are calculated and executed in realtime.

*Condition 1:*

“*Direct tracking feedback*” is provided instantly by visualizing a skeleton hand. The corresponding robot pose is calculated and the *desired end-pose* is applied to the transparent robot model. A solid robot model, indeed, is fed with the *trajectory* from the current pose to the desired posture. This trajectory is instantly executed with typical limitations like motion speed.

*Condition 2:*

It is the same as Condition 1, except that “*no direct feedback*” about the tracking is presented to the operator. The transparent robot model serves this purpose indirectly by showing the *desired end-pose* of the robot with the transparent robot model. The correlation of the operators movement with the end-pose provides the necessary queues for generating tracking feedback indirectly. Generated *trajectories* are shown as in Case 1.

*Condition 3:*

“*Only the solid robot model*” is shown. Its movements represent the trajectory from current to desired end-effector pose, which is defined by the current pose of the operator’s tracked hand.

*Observations*

The main observations from qualitative feedback are that, due to the strong immersion of solid models, users feel very uncomfortable when the robot appears to collide with them.

Condition 3 is disliked, because without the missing instant feedback the system seems to be slow and unpredictable. Typical effects caused by robot's motion speed limitation and singularities in the kinematics are that users are annoyed by the system and that they get the impression that the control method does not work properly. In contrast, Condition 1 causes confusion at the operators. The feedback of the hand tracking with the skeleton model is accepted to be natural, but the additional transparent robot, combined with the solid robot, is found to be confusing. Participants are not sure which visual sensation to follow.

### 7.2.3 Altering the Shape of a Snake-Like Modular Robot

Modular robots in chain-like configurations are difficult to control but also capable of passing irregular terrain, if controlled accordingly.



**Figure 7.2:** Two handed bending a modular robot into a desired posture via 6-DoF tracking of two HTC Vive controllers, which are setting a target for the BioIK instance, each.

Two different methods for adapting the shape of a modular, snake-like robot in VR are proposed here.

One is to use the “*virtual stick metaphor*” (cf. Figure 7.1(c)), which is controlled by an optically tracked controller, like an HTC Vive Controller and allows to use collision detection with the virtual robot model to shape the robot's body. The robot's behavior is to avoid contact with the stick. Thus, as shown in Figure 7.1(c), the robot's body lifts by adjusting the neighboring joints' angular position. The physics engine causes the robot to stay in contact with the wooden surface during adjustments (cf. Figure 7.1(c)).

The other method is based on “*virtual physical bending*” of the joints after grabbing links with the hands, e.g. with tracked controllers. Figure 7.2 demonstrates a two handed implementation with Vive Controllers. In the scenario the tail of the robot is fixed, while at two links – the head and one intermediate position around the center of the body – targets for BioIK are attached. The targets trigger 6-DoF IK calculation, making it

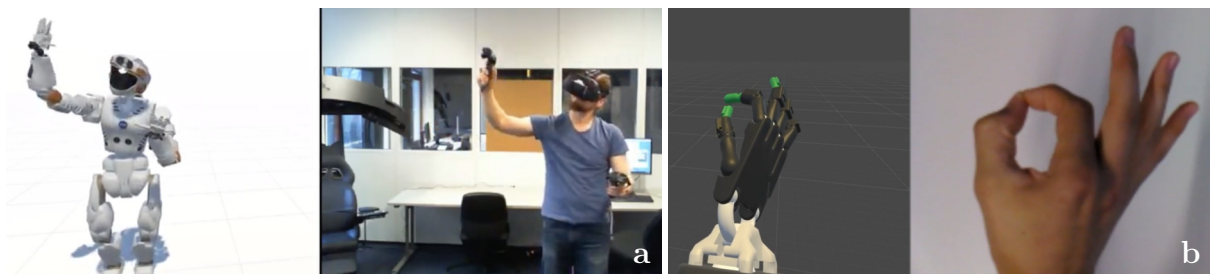
possible for the operator to manipulate the robot’s shape for passing through a labyrinth of green cubes (cf. Figure 7.2).

Both methods are applicable in-situ as an alternative to classical manual [100, 88] or learning-based approaches [100] for improving the locomotion capabilities of a modular robot in irregular terrain.

### 7.2.4 Extended Body Posture Tracking for Humanoid Robots

Delivering body language and human-like motion via robots with humanoid body physiology is an important of task social HRI. It can be achieved with a VR consumer setup and an IK solver in Unity3D. Figure 7.3(a) presents a human, tracked with HTC Vive Lighthouse system and the resulting posture of a virtual robot with similar body structure. In the Unity3D scene the operator is tracked with 6-DoF at the head by wearing the HMD and at both hands by holding the Vive controllers. BioIK is configured and three 6-DoF targets are attached to the robot model at the head and each hand.

Behavioral studies for sharing spaces between humans and robots are enabled by a teleoperation setup with a mobile service robots. In Figure 7.1(d) a PR2 robot model is controlled by a tracked human operator. Movements of the operators head are recognized and applied to the similar DoF of the pan-tilt unit attached the robot’s sensor head. Bi-manual hand tracking data of the human is applied to the corresponding grippers of the robot. The setup is implemented using an HTC Vive HMD which allows for applying room-scale pose data of the operator to movements of the robot. Furthermore, it is possible to analyze different scaling or mapping methods between different sizes, numbers of DoF or dynamics of movements when applying input from the operator’s movements to the robot’s joints.

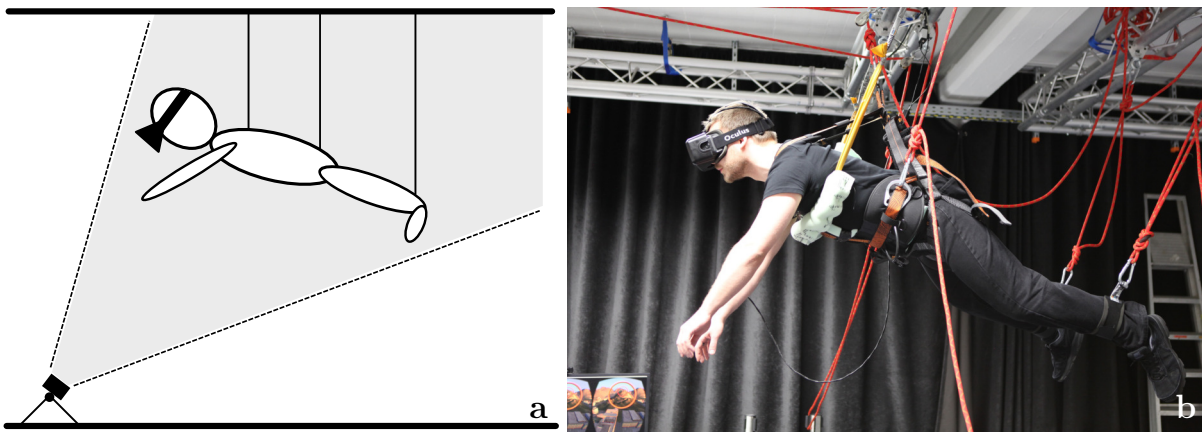


**Figure 7.3:** Extended tracking is possible by applying the results of an multi-target IK solver to the joints of virtual robot. (a) NASA Valkyrie is actuated according to a human’s body posture, which is tracked at three different locations with 6-DoF. (b) Shadow Robot Company’s Dexterous Hand is actuated according to the skeleton tracking data of a human’s hand. Palm position determines the overall 6-DoF pose of the whole hand and five optimization targets, one at each fingertip of the robot, which, are adjusted according to the tracked fingertip positions.

### 7.2.5 Embodiment of an Anthropomorphic Hand

Anthropomorphic hands impress with their tremendous manipulation capabilities. But their high number of degrees of freedom makes them also very difficult to control (cf. Figure 3.5). Figure 7.3(b) shows the result of imitating a human gesture based on the tracking information of six different poses at the human-side using a low-cost consumer tracking device – the LMC. The sensor is mounted at the front of an HMD (cf. Figure B.30(c)) to ensure that the operator’s hands are within the field-of-view. The six 6-DoF IK targets are anchored at the virtual robotic hand; one at the palm and one at the fingertip of each finger. The BioIK asset is making sure to calculate a valid posture for each joint at each finger. The result is limited by the tracking quality of the sensor, which has issues with occlusion. In the next step joint positions, extracted from the virtual robotic hand, are ready to be send to the actual robotic hardware in order to apply the current configuration, immediately.

### 7.2.6 Embodied Drone Control



**Figure 7.4:** Body skeleton tracking is utilized to control a flight through a virtual world. (a) Shows the concept of a hanging person who is observed by an upfacing depth camera from the ground. (b) The setup in the lab: A participant is hanging freely in the lab and uses his body posture to control flight parameters like speed and direction of travel while wearing an HMD.

For application to remote drone control a very immersive setup was developed, utilizing only consumer grade hardware. Figure 7.4(a) presents the concept of a hanging person, who is wearing an HMD, while being observed by a depth camera. Figure 7.4(b) shows the realization in the lab. The implemented control is used to alter the flight of the virtual avatar with two different control methods involving the body posture of the operator [98, 101]. Microsoft Kinect’s skeleton tracking provides the system with appropriate means of control information for altering the flight direction and speed. One method uses absolute directions by pointing with one arm in the desired direction and the other utilizes the orientation of the head for height control and the relative positioning of both

arms for extracting the turning direction.

The method is applicable to continuous closed-loop control (cf. Figure 3.8) of a drone’s flight in realtime with the human in the loop. System latency is very crucial in this setup and the system would benefit from future wireless technologies like 5G networks for pose data transmission of the drone’s actual pose. In the experiments, the proposed flight control methods were robust enough to control a virtual flying object but many participants suffered from simulator sickness. The implementation utilizes an Oculus Rift DK2 with only 75 Hz display rendering, which probably supports the sensation of simulator sickness. A very high degree of immersion is observed when participants were very likely to collide with objects in the virtual environment. Reflex-like actions of their bodies occur. This, combined with strong expressions of emotion during use, confirmed a high degree of immersion of this setup.

### 7.2.7 360 Mobile Robot Teleoperation



**Figure 7.5:** Immersive teleoperation of a mobile robot equipped with a 360° camera is based on natural walking in a VR tracking space. (a) The mobile robot (Pioneer 2-DX) is equipped with a Ricoh Theta 360° camera and operated within a building. (b) Operators see the 360° image in VR and thus, are able to decide on the viewing direction with low latency by using a dedicated shader. The robot’s motion is controlled implicitly by natural walking of the operator who is tracked within HTC Vive’s tracking space. Control commands for the robot are generated automatically and send via the ROSbridge from Unity3D to ROS running on the robot. (c) A shader implemented in Unity rectifies the image from the 360° camera in the desired viewing direction and allows additionally for small translatory movements without the need to move the camera.

Controlling a mobile robot by walking is an interesting concept of implementing a very natural user interface. But without a holonomic drive the robot is not able to follow the operator’s steps directly, e.g. when walking sideways. One possible solution for overcoming this issue, lies in the implementation of an MR-RUI, which utilizes a 360° camera video (cf. Figure 7.5). By transferring whole 360° images via the ROSbridge from ROS to Unity3D,

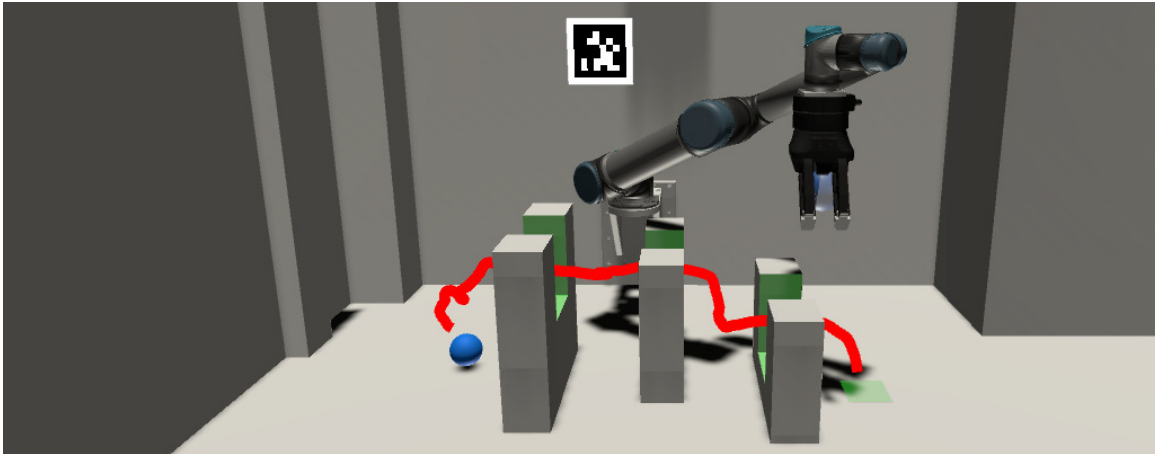
the desired view direction is calculated with the help of a dedicated shader in realtime and the robot itself does not need to turn physically in order to change the viewport of the camera. Starting forward motion takes a small amount of time, even if the transfer of the command to turn the wheels happens after only a few milliseconds. To compensate this, it is possible to change the rendered viewport, similar to the case of desired rotations, in a way comparable to zooming. Thus, the impression of locomotion is generated even if the robot is still in place. A user study evaluates possible implementations for such a system and determines thresholds for fusing “*actual*” and “*virtual*” viewport manipulations in order to know the limits for walking-based teleoperation and telepresence systems [226, 227].

## PREDICTIVE VISUALIZATION STUDIES IN VR

Current advances in VR technology achieves both, making the sensations more immersive and plausible, but also increasing the utilization of these technologies in robotics. Their low-cost and the low effort to integrate such a system in complex facilities makes them interesting for industrial application. This chapter presents an efficient implementation of “*virtual fixtures*” (cf. Rosenberg [168]) and the evaluation in a task of three different difficulties. Finally, it is discussed if the method is successfully implemented without real physical barriers and if human performance is effected in teleoperation or teleprogramming of industrial robots.

### 8.1 Visual Virtual Fixtures in VR

Building and using tools for enhancing and improving our capabilities is part of the human nature. In UI design interface metaphors are specified in order to explain instantaneously how to interact with the UI. Several years of research during the coexistence of VR and robotics demonstrated the benefits of combining them [28, 207, 118, 204]. Early HRI research of Louis B. Rosenberg introduced the ruler metaphor in combination with his major concept of virtual fixtures (VF) [168, 169, 170]. To improve the operator’s performance, he implemented a costly robot teleoperation system as a master-slave control system and presented images of the remote site to the operator meanwhile the movements of the operator were physically constraint in different ways. Current devices and systems within the *mixed reality continuum* [129] (cf. Section 4.1) offer a nearly unlimited multitude



**Figure 8.1:** The red line indicates a recorded trajectory of a single trial during the experiment, where the gripper is teleoperated to grasp the blue sphere on the left and to move it to the green goal zone on the right precisely and fast. In the experiment effects of visual and haptic assistance at difficult passages are examined.

of methods to control robots at different levels of autonomy [20]. The large increase in tracking and image quality of mixed reality devices effected the level of situation awareness (SA), presence and immersion positively and thus, makes it inevitable to think about new, even simpler ways to achieve better results or to change the application domain of already developed methods. Currently, low-cost VR devices and flexible, modular software systems like Unity3D and ROS are found in many publications of HRI UI design and settle down in industrial and private applications.

In this chapter the effects and the utility of virtual fixtures without physical constraints are examined in a low-cost setup involving the HTC Vive system for input and feedback. A user study with within-subject design compares different aspects of user input during teleoperation or teleprogramming of an industrial robot during pick-and-place tasks of different levels of task dimensionality. In comparison to the former study of Rosenberg [168] the operators are found to be slower when fixtures are activated. Interestingly, operators seem to act more carefully when the additional mechanisms, described in this chapter, were activated.

### 8.1.1 Previous Work

Rosenberg started 1992 to publish on the topic of virtual fixtures with the desire to improve teleoperation methods [168, 169, 170]. The fixtures were physical barriers, mounted on a fixture board, e.g. a table in front of the operator. The fixture board was not visible to the operator during operation. Instead, using a head-mounted vision system, the remote site with the actual robot was presented to the operator. The robotic task was to insert objects in holes of a task board. Rosenberg evaluated different designs and patterns of fixtures at the task board and evaluated the performance. He found fixtures to increase



the task performance, especially the time to finish the task. Experiments also involved the combination with auditory signals. Rosenberg pointed out, that the way he implemented the fixtures is not necessarily the only way how virtual fixtures can be implemented and encouraged to try it in a different way.

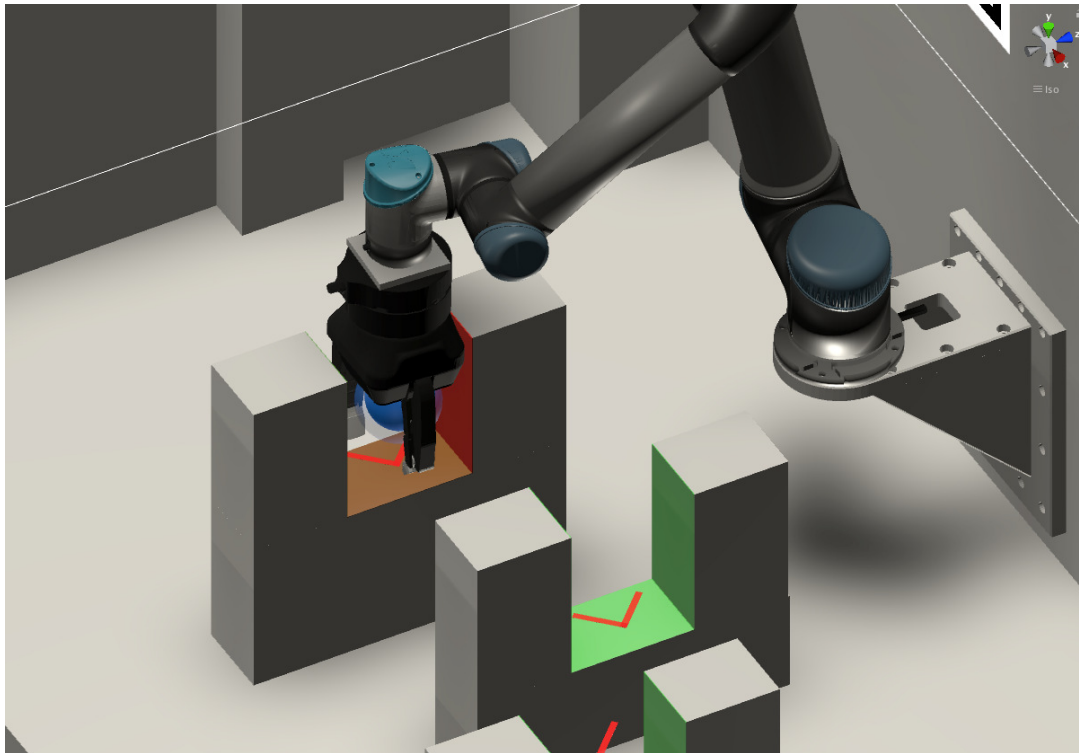
In 2008 Barros and Lindeman forecasted the utility of VR devices for robotics: “*The implementation of a mobile and easy deployable tracking system may trigger the use of trackers in the area of HRI. Once this is done, the robot community may benefit from the accumulated knowledge of the VR community on using this input device.*” [45]

Recently, many publications combining Unity3D and ROS appeared [103, 24, 215]. HoloGrasp, a setup for co-located mixed reality human-robot interaction uses the Microsoft HoloLens to control a pick-and-place system in-situ [104] (cf. Chapter 9). Especially grasping tasks are of high interest in the field of mixed reality interaction with robots [90]. Using an HTC Vive setup and a deep learning approach, virtual models of robots were taught by demonstration how to grasp a fish [52].

### 8.1.2 Implementation of Perceptual Overlays for Immersive Telerobotics

The system setup is implemented using Unity3D on a graphics workstation running Windows 10 (cf. Chapter B). The robotic setup consists of a virtual model of a setup in one of the faculty’s labs. For rendering in Unity the planning model of the lab was imported using an URDF parser from the Unity Asset Store. The inverse kinematics of the robot moving in realtime is based on Starke et al. [190] and causes the robot to produce plausible movements according to one tracked 6-DOF pose, represented by one Vive Controller as input device. The realistic virtual environment (VE) is presented to the user with an HTC Vive head-mounted display (HMD) using stereoscopic rendering.

According to Wickens’ Multiple Resource Theory (MRT) Model [216] the virtual fixtures were implemented visually and haptic as multimodal feedback. The visual component causes a change of color of the delicate zones at the obstacles. According to humans’ affordances of colors, *green* marks a safe situation and *red* a possibly problematic situation. Using Unity’s *lerp* function a transition between these two colors is generated and applied to the involved part of the obstacle (cf. Figure 8.2). The reference points for calculating the distance between object and obstacle are calculated by a script attached to the obstacles. Raycasting technique is used to find the shortest distance between all surface points of the involved collider meshes. The haptic part of the feedback is directly triggered by collisions of either the meshes of the gripper or the mesh of the grasped object. In a pre-study continuous vibro-tactile feedback, as implemented for the color sweep, was found to distract the operators by letting them believe that a collision with an obstacle



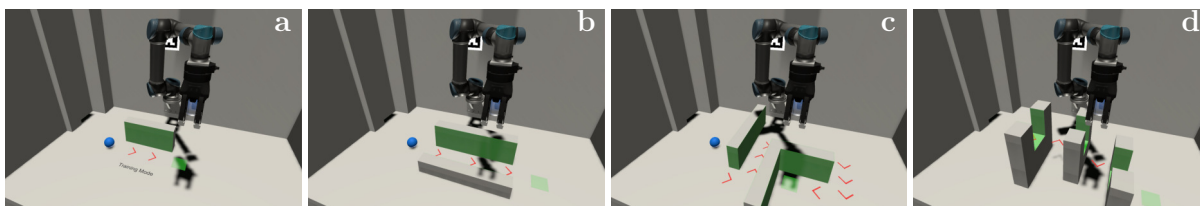
**Figure 8.2:** Example of the visual implementation of a virtual fixture. The amount of red color indicates a close distance between the grasped object and the obstacle.

occurred. Thus, the operator was provided with vibration feedback only when the object or the gripper touches an obstacle, as expected by the participants of the pre-study.

The experiment setup, which is completely virtual, is transferable to actual robot control by adding a communication interface to ROS, a robotic middleware for academia and industry [160, 103, 104]. In this way, the findings and implementation of this chapter are applicable to real robot control.

The experiment project is available at <https://github.com/denniskrupke/virtualFixturesExperiment>.

### 8.1.3 Evaluation of Multimodal Virtual Fixtures



**Figure 8.3:** Courses of different levels of difficulty. Difficulty is modeled by the number of Cartesian axes necessary to describe the resulting trajectories of the end effector. (a) Training course. (b) 1D course (C1). (c) 2D course (C2). (d) 3D course (C3).

## Hypotheses

- H1: Virtual fixtures increase the usability during teleoperation.
- H2: Virtual fixtures effect the control precision of the operator.
- H3: Virtual fixtures encourage operators to explore the workspace.

## Questions

- Q1: Are VFs applicable as an implementation without physical barriers.
- Q2: Is there a difference between VF in teleoperation and teleprogramming?

## Participants

29 participants (7 female and 22 male, ages 20 to 32,  $M = 24.25$ ) were recruited. The participants were volunteering students of the local department of computer science. The ratio of female and male participants is representative for the members of our department and thus, represents the expected user group. If requested, the students obtained class credit for their participation. 16 of the participants had normal or corrected-to-normal vision and 11 of them wore glasses during the experiment. 1 participant wore lenses and 1 reported color blindness but had no issues with the given tasks. No other vision disorders have been reported. No disorder of equilibrium or motor disorders such as impaired hand-eye coordination was reported. 25 participants reported prior participation in experiments involving the HTC Vive. 4 participants attended the pre-study as well. Handedness was not relevant, due to the implementation, which allows either the left or the right hand for solving the task. The average time for the experiment including briefing and questionnaires was 40 minutes, the time spent with the HMD worn was about 20 minutes. The interpupillary distance (IPD) of each participant was measured and the HMD was adjusted accordingly in order to maximize the participants' precision of depth perception in VR.

## Materials and Methods

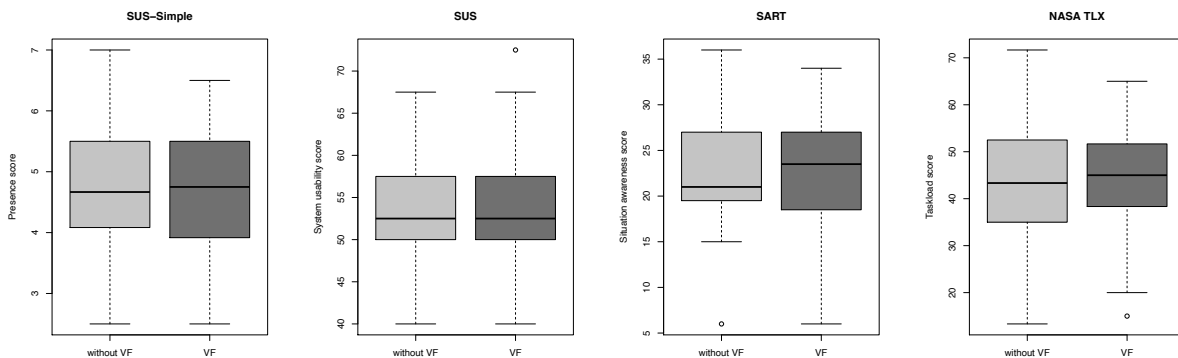
The main device in the experiment was the HMD HTC Vive with its Lighthouse tracking system and one Vive controller. Participants were asked to stand in front of a virtual table and grasp a virtual model of the Robotiq 3-Finger Adaptive Gripper attached to a UR-5 robotic arm (cf. Figure 3.5) in order to control its posture by pressing the *grip button*. In this way the robotic manipulator follows the 4 degrees-of-freedom (DoF) (3 translational, 1 rotational around the vertical axis) pose of the Vive controller as intended by the operator. Additionally, pressing the trigger button closes the gripper and opens it on release. In

this way the operator is capable of performing the virtual task of grasping the target object, moving it through the course and finally placing it on the goal zone. To ensure an immersive experience during the experiment, the virtual scene was stereoscopically rendered by the Unity3D engine (v.2018.2.0f2) on a powerful gaming computer setup<sup>1</sup>, running optimized code, in order to achieve framerate above 90 Hz display refresh rate of the HMD.

In the experiment, a within-subject repeated measures  $2$  (noVF vs. VF)  $\times 3$  (courses)  $\times 6$  (repetitions) design was used. Before the experiment, all participants filled out an informed consent form and received written instructions on how to perform the task. Participants had to perform training trials for each of the two conditions; one without virtual fixtures (*noVF*) and one with fixtures enables (*VF*). Furthermore, they filled out a demographic questionnaire before the experiment and the NASA Task-Load Index (TLX) questionnaire [69], Simple Usability Scale questionnaire (SUS-PQ) [27], Slater-Usch-Steed questionnaire (SUS) [210], Situation Awareness Rating Technique (SART) [197] and AttrakDiff2 usability questionnaire [71] after each condition.

### 8.1.4 Results

In this section, the results and statistical analyses of the experiment are summarized. All statistical tests calculating p-values were done at the 5% significance level. Due to the growing importance of Bayes factor tests (c.f. [23]), Bayes factors are presented in addition to the p-values.

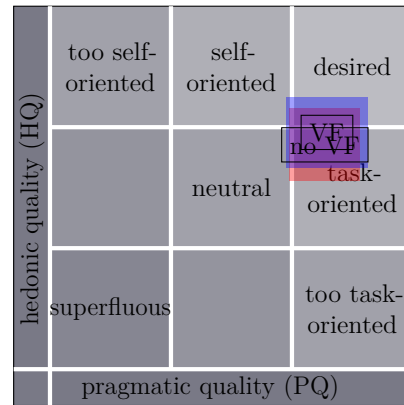


**Figure 8.4:** Results of qualitative questionnaires.

### Qualitative Analysis

In the following, the results from several standard questionnaires are presented. Figure 8.4 and Figure 8.5 summarize the qualitative scores.

<sup>1</sup>Windows10, Intel Core i7-6900K, 2x Geforce 1080, 16 GB RAM



**Figure 8.5:** Average values and confidence rectangles for the AttrakDiff questionnaire of the two conditions: (red) “no VF” for the simple approach without virtual fixtures and (blue) “VF” for the implementation with visual virtual fixtures.

### *AttrakDiff2*

The results were tested for normality with a Shapiro-Wilk test ( $p < .05$ ). In case of the pragmatic quality in the VF condition they were not normally distributed and we used a Wilcoxon Signed-Rank test. Other data showed no significant difference to normal distribution and thus, paired samples T-Tests were applied for the analysis. HQI ( $p < .05$ ,  $T(27)=-2.2255$ ) and HQS ( $p < .05$ ,  $T(27)=-2.1152$ ) showed significant differences. Calculating Bayes factors shows *moderate evidence* in both cases but additionally *moderate evidence* for the overall HQ. All other variables show *anecdotal evidence* for the null-hypothesis ( $\mathcal{H}_0$ ). Figure 8.5 the confidence rectangles for the two conditions are visualized.

### *SUS-Simple*

Shapiro-Wilk test of the scores was not able to show a significant difference to normality distribution. Evaluation of the Simple Usability Scale questionnaire revealed no significant difference between the two methods. *Moderate evidence* for  $\mathcal{H}_0$  derives from Bayes factors.

### *SUS*

Slater-Usosh-Steed questionnaire showed no significant difference of the normally distributed scores and only *moderate evidence* for  $\mathcal{H}_0$ .

### *SART*

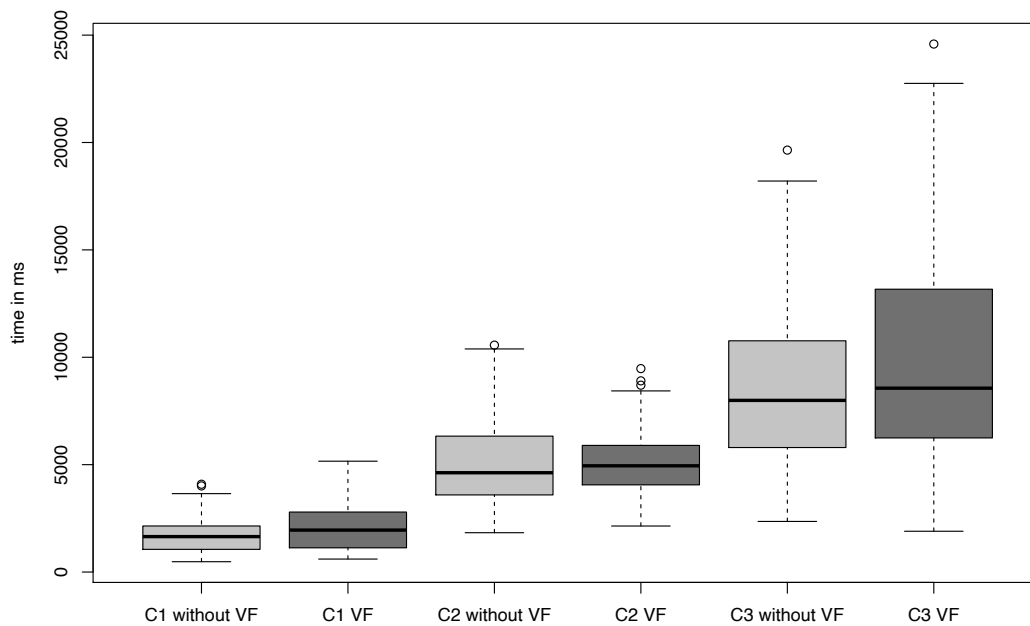
In the “VF” condition SART-D seems not to be normally distributed, which is the reason why Wilcoxon Signed-Rank test was applied instead of T-Test. Examining p-values showed no significant difference in overall scores or any part of the scoring procedure. Bayes factors indicate *moderate evidence* for  $\mathcal{H}_0$ .

*NASA-TLX*

In the “VF” condition “effort” and in the “noVF” condition “frustration” the distribution is indicated as not normal and Wilcoxon Signed-Rank tests were applied. No significant differences arise from p-value analysis. Bayes factors show *anecdotal to moderate evidence* of  $\mathcal{H}_0$  in all categories.

**Quantitative Analysis**

During the experiment several kinds of information about the single trials were recorded, such as the *time* from grasping the object to entering the target zone, the *distance* from the surface of the object to the surface of the closest obstacle, the number of *collisions* of the object and the gripper with the obstacles in the environment and the *exploration effort* of the participants, represented by the sum of their translational and the sum of their rotational head movements during the single trials.

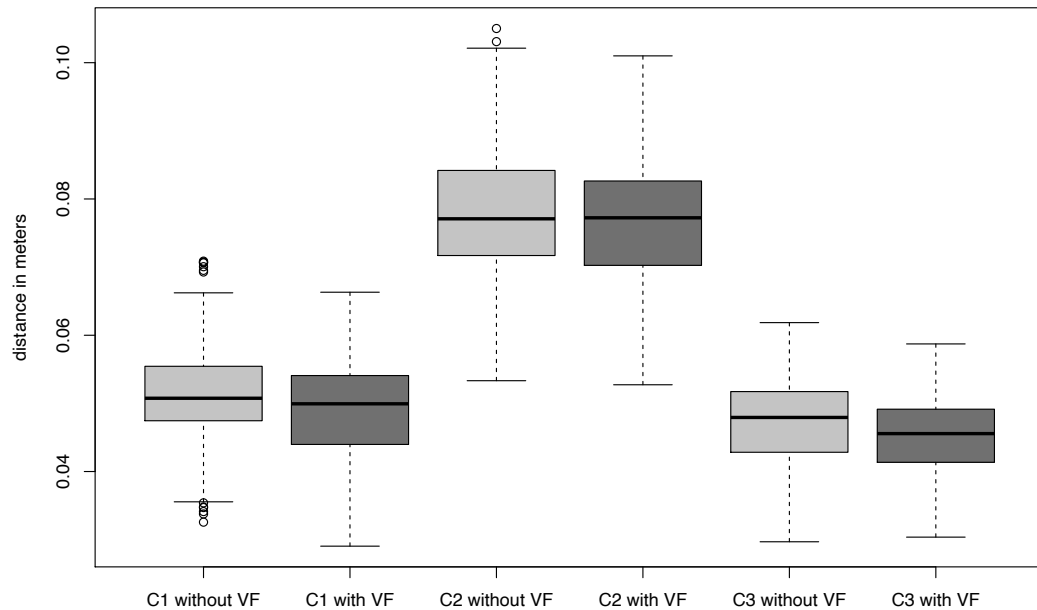
*Time*

**Figure 8.6:** Times to finish the courses.

Analysis of the needed time to finish the single course with a Shapiro-Wilk test confirmed a significant difference to normal distribution. Thus, Wilcoxon Signed-Rank tests were necessary to calculate p-values. All three courses show significant differences between the two conditions “noVF” and “VF” (C1:  $p < .05$ ,  $V(29) = 5571.5$ ; C2:  $p < .05$ ,  $V(29) = 6038.5$ ;

C3:  $p < .05$ ,  $V(29)=6227.5$ ). Analysis of Bayes factors results in *anecdotal evidence* for  $\mathcal{H}_0$  in case of course 1 (C1), *anecdotal evidence* for  $\mathcal{H}_1$  in case of course 2 (C2) and *moderate evidence* for  $\mathcal{H}_0$  in case of course 3 (C3). A summary is presented in Figure 8.6.

### Precision

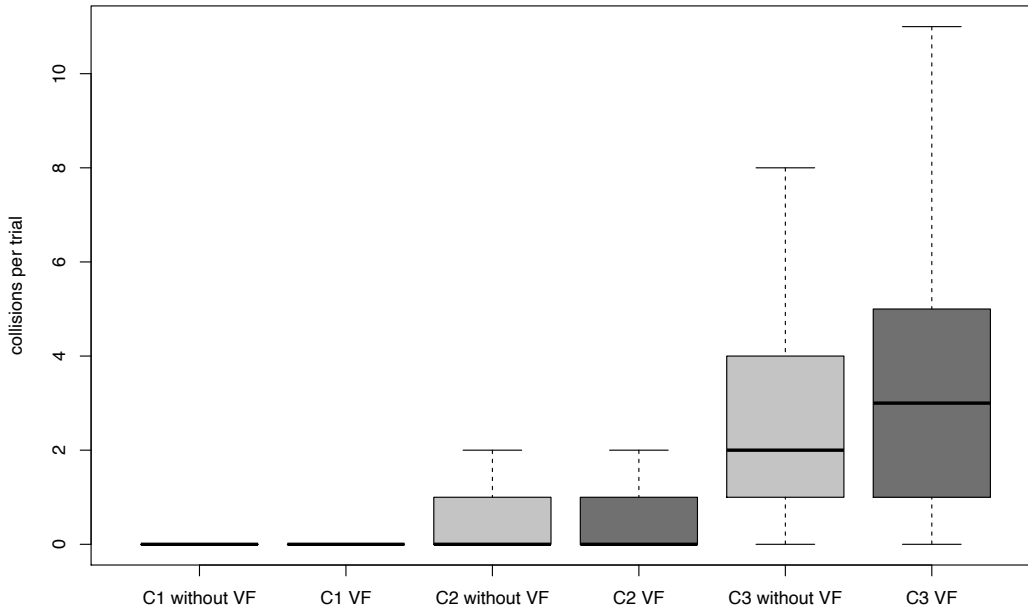


**Figure 8.7:** Mean distances between grasped object's surface and the closest surface point of the obstacle.

During the trials, the shortest distance between the grasped object's surface and the surface of the closest obstacle was calculated. Only the means of distances in C3 with VF condition seem to be normally distributed and Wilcoxon tests were applied. Means of distances for C1 ( $p < 0.05$ ,  $V(29)=9188$ ) and C2 ( $p < 0.05$ ,  $V(29)=8458$ ) show significant differences in the two conditions, as well as C3 ( $p < 0.001$ ,  $V(29)=10036$ ). Analysis of Bayes factors reveals *moderate evidence* for  $\mathcal{H}_0$  in C1, *anecdotal evidence* for  $\mathcal{H}_1$  in C2 and *extreme evidence* for  $\mathcal{H}_1$  in C3. In Figure 8.7 a summary of boxplots is presented.

### Collisions

Recordings of the gripper and the grasped object collisions show no normal distribution and no significant difference. Bayes tests claim *moderate evidence* for  $\mathcal{H}_0$  in C1 and C2 regarding collisions of the gripper and *anecdotal evidence* for  $\mathcal{H}_0$  in C3. In C1 no collisions of the grasped object occurred. C2 reveals *anecdotal evidence* for  $\mathcal{H}_0$  and in C3 *moderate evidence* for  $\mathcal{H}_0$  is calculated. As depicted by Table 8.1 the number of collisions of the



**Figure 8.8:** Collisions of the gripper with the obstacles.

gripper with the obstacles is lower without virtual fixtures, but in case of collisions of the grasped object with the obstacles it is lower when virtual fixtures are activated. Figure 8.8 summarizes the collisions of the gripper with the obstacles.

**Table 8.1:** Total count of collisions.

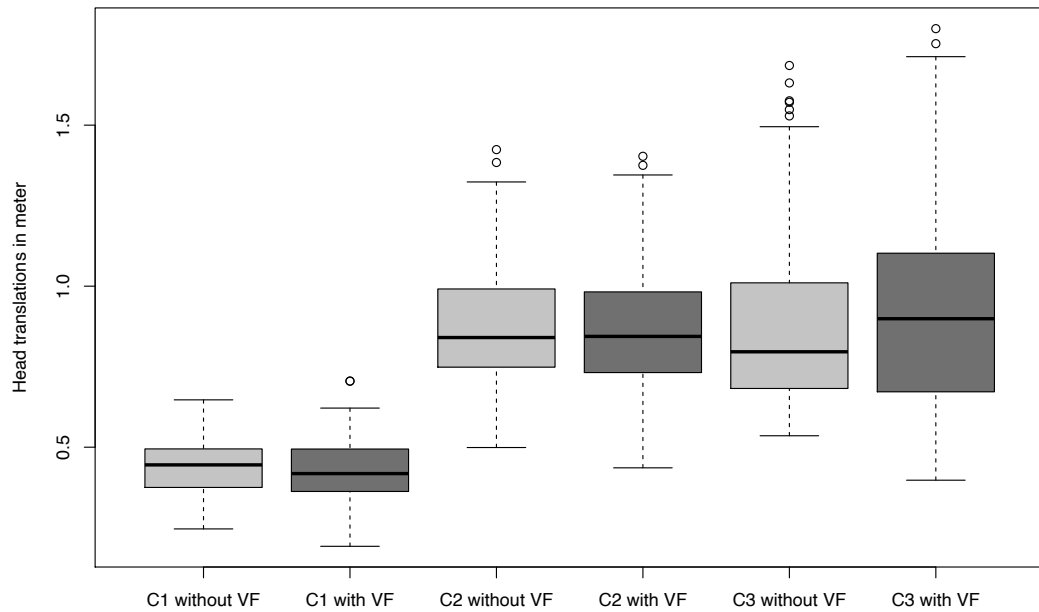
	C1	C2	C3	Total
<b>Gripper noVF</b>	13	128	461	602
<b>Gripper VF</b>	24	139	534	697
<b>Object noVF</b>	0	8	49	57
<b>Object VF</b>	0	3	41	44

### *Operator's Exploration Effort*

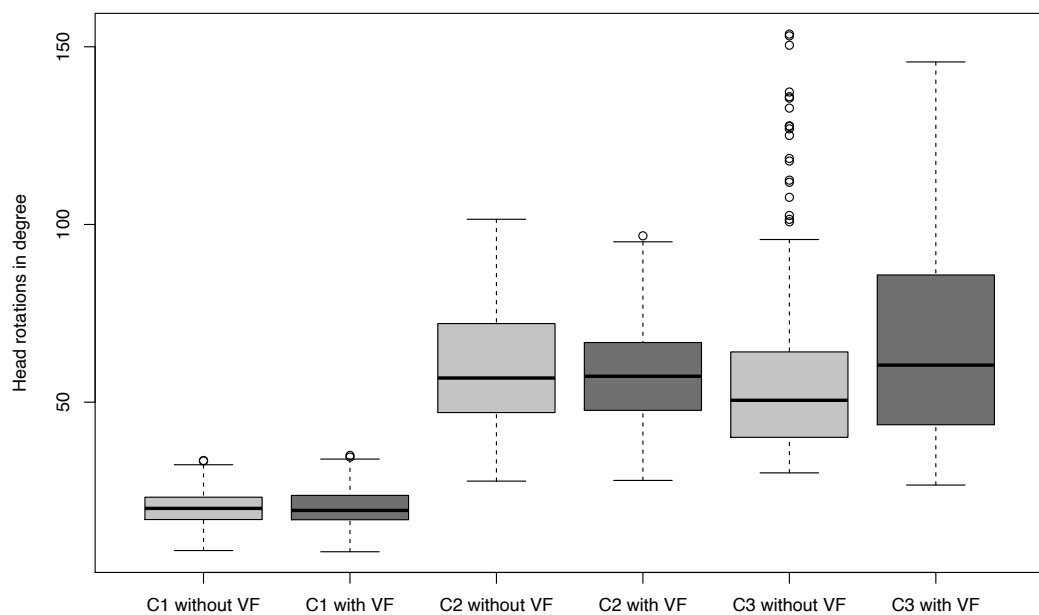
The participants' exploration behavior was analyzed from the recorded trials to conclude if there is an effect caused by the additional feedback provided by the system. Thus, the translations (cf. Figure 8.9) and the rotations (cf. Figure 8.10) of the head separately during each trial were recorded and summed up.

All recorded data shows significant differences to normal distribution. No significant difference between "VF" and "noVF" could be found. Regarding translation Bayes tests show *strong evidence* for  $\mathcal{H}_0$ . In rotational movements *strong evidence* for  $\mathcal{H}_0$  was found in C2 and *moderate evidence* for  $\mathcal{H}_0$  in C1 and C3.





**Figure 8.9:** Translational head movements of the operator.



**Figure 8.10:** Rotational head movements of the operator.

## Discussion

From qualitative analysis using standard questionnaires a significantly higher hedonic quality with virtual fixtures enabled was found (cf. Figure 8.5). Thus, H1 is partially

confirmed. No negative effect on the user experience and usability could be found by adding visual and haptic feedback to the system. Measuring situational awareness with an appropriate method is challenging. An improvement with VF enabled was expected, but no significant improvement could be shown. In Figure 8.4 a larger mean is clearly recognizable and some participants denoted in the final questionnaire the additional feedback to be helpful for their depth perception during the task. This could indicate a tendency to a larger SA but further investigation is necessary. Generally participants showed a quite high level of presence by not walking into the table or robot, despite, they were briefed to be free to walk around within the virtual environment.

Quantitative analysis revealed that participants took significantly longer in all three courses with “VF” enabled especially in C3 as Bayes factor analyses confirm. As reported by participants after the trials in an open question of the final questionnaire, the “VF” enabled condition encouraged them to try to be more precise. The approach to record the exploration effort is limited by not including eye movements, which should be investigated when the eye-tracking technology in HMDs is significantly improved. Head movements (cf. Figure 8.9 and Figure 8.10) showed a tendency to increased exploration behavior but the difference is not large enough to be significant. Thus, H3 needs further investigation to be confirmed. Analysis of the distance of the grasped object to the obstacles show that it is significantly smaller when VFs are enabled (cf. Figure 8.7), which could explain the longer processing times of the single trials in that condition. Regarding collisions of the gripper and the grasped object with the obstacles in the environment no significant difference was confirmed. But having a closer look at the occurred collisions in Table 8.1 shows that with VFs enabled there were more collisions of the gripper but less collisions of the grasped object in total. The closer mean distance to the obstacles, which was triggered in the “VF” condition also increases the probability of collisions during directional changes of operator movements. The additional haptic feedback during a collision informs the operator to be too close to the obstacles. The combination of both could cause the lower amount of collisions of the grasped object during the trials. These effects should be investigated in a separate study. Thus, H2 is confirmed but the modalities are not clear. The haptic feedback is implemented to occur during collisions of either the grasped object, or the gripper. Since, the gripper is grasping the object it “protects” the object. Increasing the size of the gripper’s collider should function as a “protection” of the gripper, meanwhile reducing the plausibility of the VE.

### **8.1.5 Utility of Virtual Fixtures in VR-based Interfaces**

Q1 seems to be answered positively. The original setup of Rosenberg [168] is very elaborate and costly. A virtual barrier has many advantages. In the presented implementation the visual fixtures guide the user continuously during the operation of the robot. The haptic component increases the effect of the red color by informing the user about a collision,

which is possibly the reason for less collisions of the grasped object with the obstacles in the environment. Here, a larger collider should prevent the gripper itself of collisions.

Q2 questions the desired purpose of the mechanisms tested in this contribution. For realtime teleoperation we would be interested in safety mechanisms, which are currently not part of the presented system. This includes collision checks and collision-free planning of trajectories, which is a research field itself in the area of planners for realtime interaction. But regarding teleprogramming and learning-by-demonstration many scenarios can highly benefit from the presented findings. The experiment tasks described in this paper already offer some reduction of task complexity. Collisions with a static object like the table surface are neglected since a path planning algorithm is capable of avoiding these issues easily. The tracked 6-DoF of a Vive Controller is reduced to 4-DoF, since, the task was limited to top grasp, which is quite common in actual pick-and-place scenarios with open vessels. These reductions surely contribute to the good scores from qualitative analysis.

To summarize, a low-effort version of the concept of virtual fixtures was implemented. The Unity3D project is available at GitHub for download and can be integrated into own projects. The implementation was evaluated in a user study and the statistical analyses and findings were summarized. Application of the results and an outlook is discussed and proposed. This work contributes to future robotic user interface design with the presented results.



## PREDICTIVE VISUALIZATION STUDIES IN AR

Current technologies have paved the way to an era in which robots can co-exist and collaborate with humans in private and public places [117, 123]. For example, autonomous cars are being widely tested and robots can be found in factories, warehouses and in homes nowadays. However, in many occasions the spaces in which humans work are deliberately kept separate from the spaces in which robots operate, since many people do not feel safe or comfortable in close proximity to robots. One essential reason for those concerns is the difficulty of humans to interpret a robot's intent [214]. Humans rely on a rich set of non-verbal cues to interpret the intent and emotions of others human being, whereas most of such cues are not provided by robots. To address these limitations, some researchers have attempted to make robots more expressive [211] or to visualize the robot's planned movements [112]. However, all these solutions impose a number of limitations on the design of the robot, and do not work with traditionally designed robots.

Mixed reality (MR) [70, 28, 141] opens up new vistas for such human-robot interaction (HRI) scenarios. The visual combination of virtual content with real working spaces creates a MR environment that has enormous potential to enhance co-located HRI [57]. Current technologies such as optical see-through head-mounted-displays (HMDs) like the Microsoft HoloLens are typical examples of suitable MR displays.

Using an MR display, a human operator can see the co-located robot in its real physical surroundings, while visual cues can be displayed in the human operator's view and augment the real-world by showing information, which is important for the human-robot collaboration process. Examples include, but are not limited to system states of the



**Figure 9.1:** Illustration of the MR human-robot interaction: A human operator wears an optical-see through HMD for viewing the co-located robot with superimposed virtual information such as pick locations or pre-visualization of potential actions of the industrial robot arm.

robot, potential pick locations in a pick-and-place scenario, previews of potential robot interactions, or proxemic zones for human-robot collaboration [111].

## 9.1 Heading and Pointing Gestures

This chapter introduces the concept and implementation of an MR human-robot collaboration system allowing users to intuitively and naturally control a co-located industrial robot arm for pick-and-place tasks. The general concept and the specific implementation of such an MR human-robot collaboration system is presented. The goal of the system is to provide experts as well as non-experts the capabilities to intuitively and naturally control a robot for selection and manipulation tasks such as pick-and-place. However, the interaction between the human and the robot in such a scenario is still a challenging task. In order to address the first aspect of pick-and-place, the grasping, two different multimodal techniques to define the grasp point on a target object, i. e., (i) heading or (ii) pointing were compared, both in combination with speech input. Finally, how a pre-visualization of potential robot motions can enhance the human-robot collaboration is demonstrated.

### 9.1.1 Previous Work

MR technologies have enormous potential to address some of the challenges mentioned above. The visual combination of digital content with real working spaces creates an mixed environment, which can provide important feedback during HRI. Paul Milgram [129] introduced the term MR based on a *reality-virtuality continuum* (cf. Section 4.1) as a continuous scale ranging between the real world, i. e., our physical reality, and a completely

virtual environment (VE), i. e., virtual reality (VR). The continuum encompasses all possible variations and compositions of real and virtual objects and consist of both *augmented reality (AR)*, where the virtual augments the real, and *augmented virtuality (AV)*, where the real augments the virtual [191]. Obviously, AR and VR serve different user experiences and there is ample space for both of them to coexist. Noticeably, the real world as well as VR are only the end points of this continuum, whereas AR or more general MR provides a certain area of different combinations of real and virtual content [129].

Very recently, MR has been considered in the area of HRI. Researchers at the DFKI [47] showcased three identical robots, which were remotely manipulated by an operator with the aid of a HoloLens HMD. In contrast to the presented approach in this chapter, most of the interaction was based on a menu-based system in which the positioning of the robot arm is specified using a 2D heads-up (HUD) display, whereas in the scenarios described in this chapter the selection is performed directly on virtual and real objects respectively.

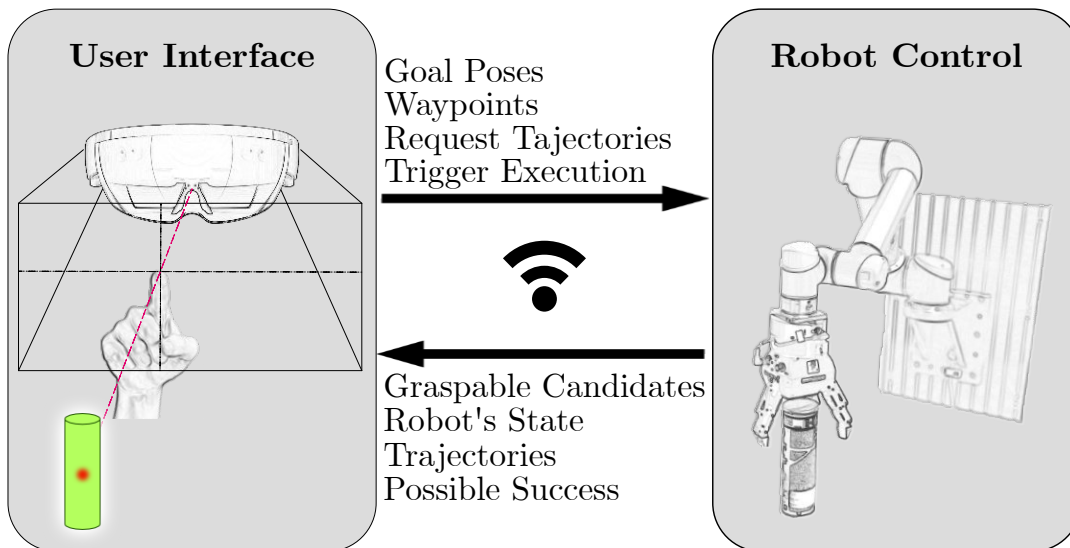
Multimodal interfaces emerged almost 40 years ago when Richard Bolt introduced his seminal *Put-That-There* work in the 1980s [25]. His interface combined spoken commands, which are linked with pointing gestures. With current inside-out tracking HMDs like Microsoft's HoloLens, such multimodal interfaces are implemented typically by two different modes: (i) heading-and-commit (HB) interaction in which heading and a select command are combined, and (ii) point-and-commit (H2F) approaches in which pointing gestures are used instead of heading. Both methods are appropriate for high-to-medium level tasks, either for direct selection of surface points at virtual objects, or whole object selection, assuming the prior detection of potential object surfaces by a recognition system.

*Heading* of a user's head is defined by the direction in which the head is turned, which only approximates the actual gaze direction, for current MR HMDs. With HB, users target an object with their heading and then commit with a speech command. In the real world, we typically look at an object that we intend to interact with. However, a comparison with eye-tracking-based HMDs pointed out that currently head tracking is more effective and reliable than eye tracking [159]. As the user looks around, the heading defines a ray, which might intersect with both virtual objects and with a 3D mapping mesh to determine what real-world object the user may be looking at.

With point-and-commit, a user can aim with a pointing-capable motion controller at the target object and then commit with a button press or a speech command. In contrast to the heading ray, in this approach the selection ray is attached to the position and orientation of an additionally tracked motion controller. In the presented implementation, additional input devices were intentionally omitted, and therefore the user's finger is tracked with the inside-out tracking capability of the HoloLens.

### 9.1.2 Mixed Reality HRI Prototype

As described above, MR technology has enormous potential to improve user interfaces (UIs) for HRI. The focus of the work presented in this chapter is on human-robot collaboration in which an industrial robot arm can be controlled by the MR user to manipulate real-world objects (cf. Figure 9.2). In the implementation, the user wears a Microsoft HoloLens and sees a virtual robot, which is displayed superimposed over the real robot. Additional information about objects with which the user can interact is displayed. Actions of the real robot are triggered through either selection gesture combined with speech, which instantly triggers a simulated motion of the virtual robot that serves as a pre-visualization of the robot’s actions. If the user is satisfied with the result, the prepared actions of the real robot can be initiated.



**Figure 9.2:** The concept of the presented human-robot collaboration system is based on a flexible and tetherless setup, which uses a self-contained see-through MR display and an industrial robot in a pick-and-place task.

#### Implementation

The implementation of the MR HRI prototype involves specific hardware, but can be generalized to work with arbitrary robots and different AR and VR HMDs. It follows the principle described in [103]. Further details are presented in Chapter B.

##### *Hardware Setup*

A Universal Robot *UR5* industrial manipulator with an attached Robotiq *Adaptive 3-Finger Gripper* was utilized, which is controlled by ROS nodes for interaction [160] (Kinetic) on an Ubuntu 16.10 workstation. The UI technology for the MR HRI prototype is implemented for Microsoft’s mobile MR headset HoloLens with Unity3D 2017.1.1f.



The HoloLens is a self-contained mobile device equipped with inside-out tracking, and it features speech and hand gesture recognition [127]. The HoloLens has a field of view of  $30^\circ \times 17.5^\circ$  at a resolution of  $1268 \times 720$ . Headset and robot are connected via local network as described below.

### *Communication*

The communication between the headset and the robot control software is implemented using the *ROSbridge* node, provided by the *robot web tools* project [204]. The ROSbridge is a ROS node, which offers a websocket-based communication via the network. The presented system includes an implementation at the Unity3D-side in C# for *Universal Windows Platform* (UWP).

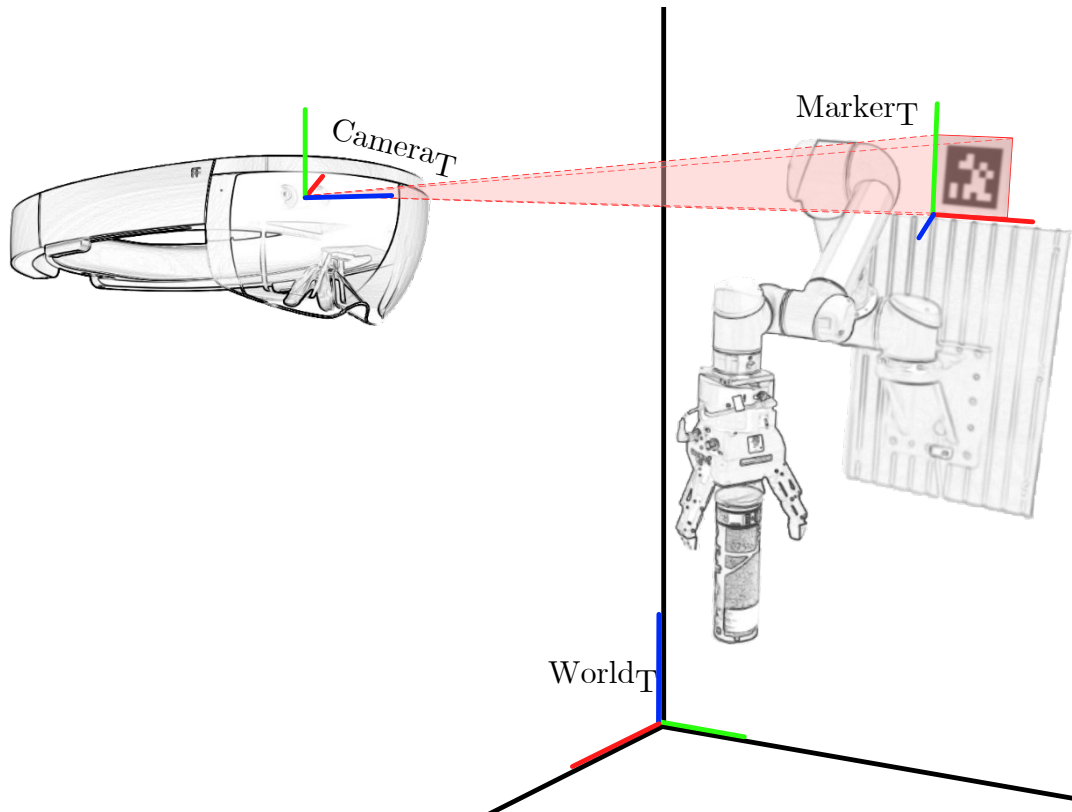
### *Registration and Accuracy*

The Microsoft HoloLens places the origin of its coordinate system dynamically and depending on its pose during boot. It automatically applies corrections to the world model and its localization through a SLAM algorithm running on hardware level of the HMD in the background. The stability of HoloLens “*holograms*” and their world anchors have been evaluated previously and a mean displacement error of  $5.83 \pm .51$  mm was found [212]. Considering the prototype status of the HoloLens, this is precise enough for pick-and-place tasks with medium-sized objects. Using a single 2D marker approach to align the virtual environment relative to the real one in a short calibration phase, using Vuforia camera-based marker detection [158]. This marker is also used as an infrastructure marker with known transformation from its position to the frame of the planning world,  ${}^W_M T$ .

As shown in Fig.9.3 the mobile headset recognizes the marker at the wall using the integrated front-facing RGB camera and computes the marker’s 6-DoF pose. By measuring the transformation from the marker position to the origin of the real world coordinate system, the headset is localized,  ${}^M_H T$ . Finally, the conversion between left-handed Unity-coordinate system and right-handed ROS-coordinate system (cf. Section B.6.1) is necessary to communicate pose data between the systems (cf. Equation 9.1).

<i>x - axis</i>	<i>y - axis</i>	<i>z - axis</i>
$-.0072 \pm .0068$	$.0267 \pm .0114$	$.0040 \pm .0073$
$\alpha$	$\beta$	$\gamma$
$< .01^\circ$	$2.2050^\circ \pm .3931^\circ$	$< .01^\circ$
<i>euclid2d</i>	<i>euclid3d</i>	
$-.0016 \pm .0054$	$.0303 \pm .0093$	

**Table 9.1:** Translational and rotational mean error with standard deviation of the registration in meters and degrees.



**Figure 9.3:** For transforming poses between different coordinate frames a 2D-marker-based approach is used. The front-facing camera of the mobile AR-headset detects the 6-DoF pose of the marker at a known position in the world model, and thus coordinates within the virtual world of the HMD can be transformed to real world coordinates or other frames.

To improve the accuracy of the registration process a priori knowledge about the lab was utilized during marker detection, since the walls are perpendicular to the floor. This way, the rotational tracking is only necessary to acquire the rotation around the world yaw axis. The resulting reduction of 6-DoF tracking to 4-DoF improved the system's accuracy. The system's final accuracy was evaluated, by positioning virtual objects according to corresponding real world objects and found that the actual setup can achieve an accuracy of  $30 \pm 9.3$  mm. However, with a single marker detection process during startup, a drift was experienced on the vertical axis over time. It can be reduced by repeating the registration marker detection during runtime. On the horizontal plane, the deviation is much smaller ( $1.6 \pm 5.4$  mm) and no drift over time was measured. Furthermore, a multi-marker approach is presented in Section B.6.2.

$${}^W_C T = {}^W_M T \cdot {}^M_H T \cdot {}^H_C T \quad (9.1)$$

Based on this, tracking in the virtual world is applicable to real world tasks, like selecting grasp poses. Heading-based (HB) and head-to-finger-based (H2F) raycast methods were implemented to select poses for grasping on virtual objects. Once a pose is selected, HoloLens-built-in voice command recognition is used to trigger actions.

## Multimodal UI for Pick-and-Place Tasks

In the implementation methods described in [90] were extended for general object manipulation in AR with a mobile headset. The main challenge arises due to the extension of a 2D visualization with classical mouse-keyboard input to a 3D representation with spatial input and output. As described above, the user can freely move around the robot and the objects subject to be picked and placed. The manual specification of robot grasping positions is an open research question. In principle, the inside-out tracking technology allows two different approaches, i.e., “*heading-and-commit*” (HB) and “*point-and-commit*” (H2F) as described above in Section 9.1.1. In both selection techniques the user perceives visual feedback about the current selection point by means of a red ring-shaped cursor as illustrated in Figure 9.4. A commit command can be easily provided via voice, which has the benefit to reduce sensorimotor errors. If the user is satisfied with the current pick location indicated by the red ring-shaped cursor, they can confirm this position with a voice command (“*select*”).

### *HB Selection*

In the heading-based implementation, a ray is cast from the head position in the forward direction of the HMD. The pose of the user’s HMD is used to determine the heading vector, which is the mathematical representation of a laser pointer straight ahead from between the user’s eyes towards the forward orientation of the HMD. As the user looks around, this ray can intersect with both virtual objects and with the spatial mapping mesh to determine which real-world object the user may be looking at.

### *H2F Selection*

In this technique, the object occluded by the user’s finger is the selected object. The finger is tracked with the inside-out tracking capability of the HoloLens. Then, the ray-cast is defined by the position of the user’s head (from between the eyes) and the position of the fingertip of the index finger.

## Pre-Visualization of Robot Actions

After the user initiated the confirmation command via speech, the virtual representation of the robot starts the simulation of the picking motion, and shows the corresponding motions to reach the specified pick location. If the user is satisfied with the simulated motion of the robot, she can confirm this action with another voice command (“*confirm*”), and the real industrial robot arm will perform the same actions as visualized in the simulation.

The advantage of such a pre-visualization is manifold. First of all, it allows the user to see the motions of the complex robot arm before it actually moves. This is important for increased safety, but also simplifies the robot programming and robot behavior understanding, as described in Section 9.1.1. The user can see the effective space of the actions and can move themselves or obstacles away from the robot’s motion paths, or alternatively specify a different location or different path.

### 9.1.3 Evaluation

In this section the evaluation of the user study is reported. It is analyzed which of the MR selection techniques described in Section 9.1.2, is more beneficial for the considered HRI setup. While actual robot descriptions were utilized, this experiment was done without an actual robot to test the software beforehand.

#### Hypotheses

In informal pilot evaluations it is found that both methods, using the finger or the heading to select an object, appears to be natural for users. However, while the HB technique requires only the head for specifying the selection ray, the H2F technique requires the user to position the pointing finger in mid-air within the view of the tracking sensors of the headset. Therefore, we assume that using the finger to select an object requires more effort, is less precise and takes more time than using heading. Hence, we hypothesize that:

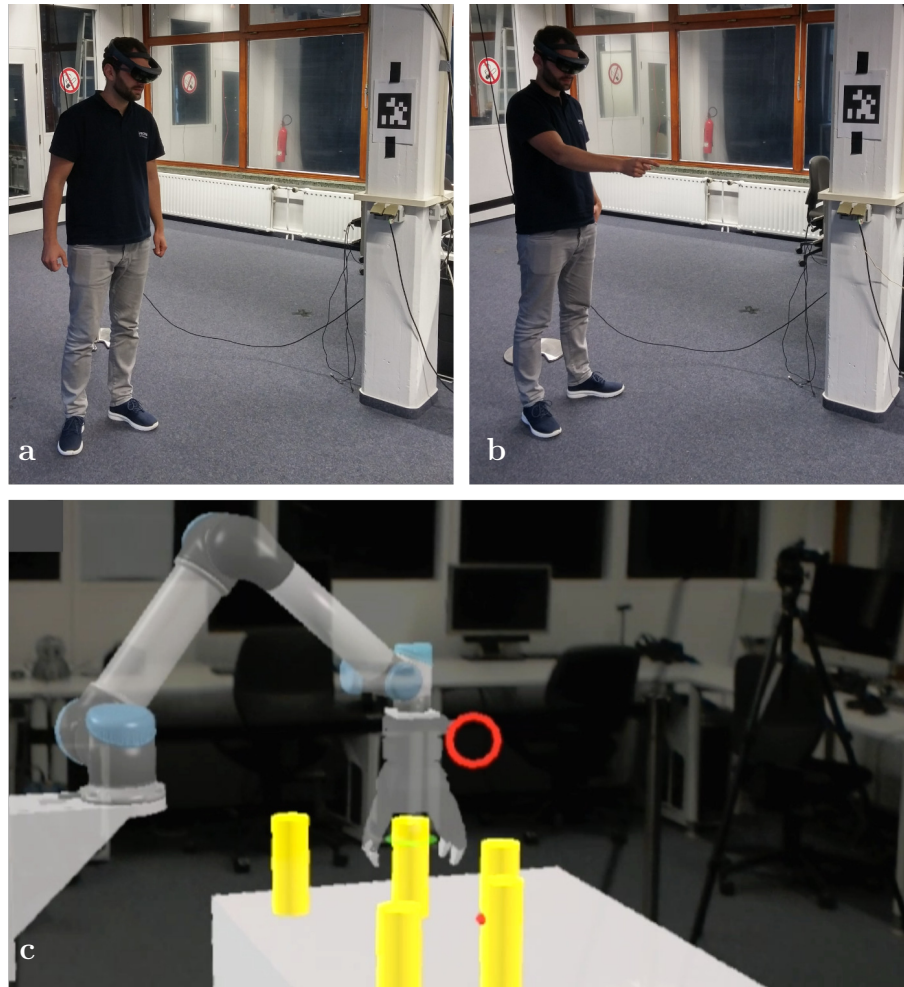
**H1:** HB selection is less demanding (physical, mental and temporal) than H2F.

**H2:** HB is more accurate than H2F.

**H3:** HB requires less time than H2F.

#### Participants

16 participants (3 female and 13 male, ages 21 to 38,  $M = 28.31$ ) were recruited. The participants were volunteering employees or students of the local department of computer science. If requested, the students obtained class credit for their participation. 8 participants had normal or corrected-to-normal vision and none of them wore glasses during the experiment. 1 participant reported strong eye dominance, but no other vision disorders have been reported. No disorder of equilibrium or a motor disorder such as impaired hand-eye coordination was reported. 3 participants reported prior participation in experiments involving the Microsoft HoloLens. Handedness was not relevant, due to the implementation, which allows either the left or the right hand for gestures. The average total time for the experiment was 25 minutes, the time spent with the HMD worn about 15 minutes.



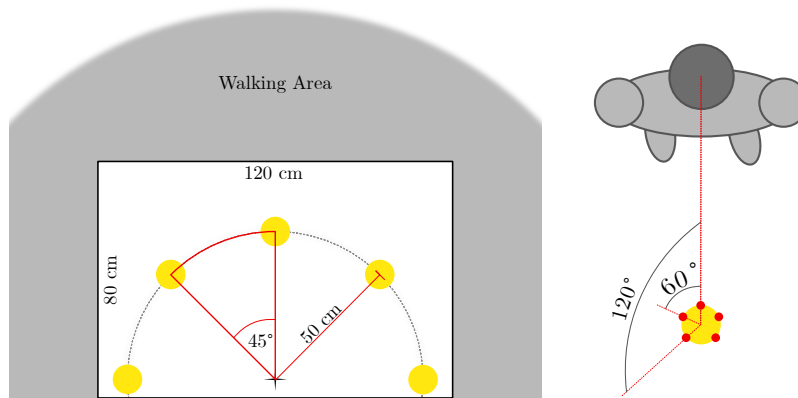
**Figure 9.4:** Participant during the experiment: HB-selection (a), H2F-selection (b) and the view of the participant through the AR headset (c).

## Materials

The main device in this experiment is the self-contained, see-through HMD Microsoft HoloLens [127]. As illustrated in Figure 9.4, participants were able to move within a hemispheric area of  $3.5\text{ m} \times 2.5\text{ m}$ . On the edge of this area the marker was placed on a pillar, mirroring the actual setup in another lab with an actual UR5 actuator. The virtual part of the visual stimulus was rendered using the Unity engine directly on the HoloLens, following performance optimization to ensure the frame rate was higher than the 60 Hz display frequency.

## Methods

As illustrated in Figure 9.4, the user saw the real laboratory with a MR overlay of a virtual robot actuator, a virtual table and five virtual, yellow cylinders arranged on a half-circle with a radius of 0.5 m as illustrated in Figure 9.5. For each trial, a random yellow cylinder was selected. Using the current participant's position, the target position along the outer



**Figure 9.5:** The arrangement of cylinders on the table and the targets on top of the surface of the cylinders.

edge of the cylinder was determined. The targets were represented by red spheres with a radius of 1.125 cm. The targets appeared sequentially at an angle of  $\pm 0$ ,  $\pm 60$  or  $\pm 120^\circ$  in radial coordinates relative to the direct line between the participant and the current cylinder, where  $\pm 0$  would be the point of the cylinder directly facing the participant. The angles  $-60$  and  $-120^\circ$  correspond to the points on the left side on the cylinder from the users current position, whereas  $+60$  as well as  $+120$  were displayed on the right side on the cylinder with respect from the users current position (cf. Figure 9.5) Positive or negative angles are chosen in a way to direct the user to stay within the hemisphere. To create a more economically valid pick-and-place scenario, targets were placed occasionally on the partly occluded side of the cylinder, which forced participants to walk around the interaction area. Using one of the two selection techniques, the user had to position the gripper at the target sphere. As mentioned above, for both the selection techniques, the users saw a red, ring-shaped cursor which they moved either through rotating their head or by moving their dominant hand's index finger. To avoid learning effects, the experiment was counter-balanced, meaning that half the participants started with the HB selection technique and the other group with the H2F technique and then used the other technique, respectively.

In the experiment, a within-subject repeated measures  $2$  (HB vs. H2F)  $\times 3$  (angles)  $\times 5$  (cylinder)  $\times 2$  (repetitions) design was used. Before the experiment, all participants filled out an informed consent form and received written instructions on how to perform the task. Furthermore, they filled out a demographic questionnaire before the experiment and the NASA Task-Load Index (TLX) questionnaire [69], Simple Usability Scale questionnaire (SUS) [27] and AttrakDiff usability questionnaire [71] after each condition.

#### 9.1.4 Results

In this section, the results and analyses of the experiment are summarized. All statistical tests were done at the 5% significance level.

Questionnaire	<i>M</i>	<i>SD</i>	<i>T/VValue</i>	<i>p</i>	<i>dCohen</i>
TIME HB	5.84	3.72	V(15)=37148	<.001	0.33
TIME H2F	7.19	4.45			
ACC HB	.01	.03	V(15)=34748	<.001	0.34
ACC H2F	.03	.06			
SUS HB	78.12	17.11	V(15)=40	.15	.52
SUS H2F	68.59	19.54			
TLX HB	39.48	13.66	T(15)=-2.74	<.05	1.03
TLX H2F	53.23	13.03			

**Table 9.2:** Summary of time, accuracy, usability and task-load.

## Performance

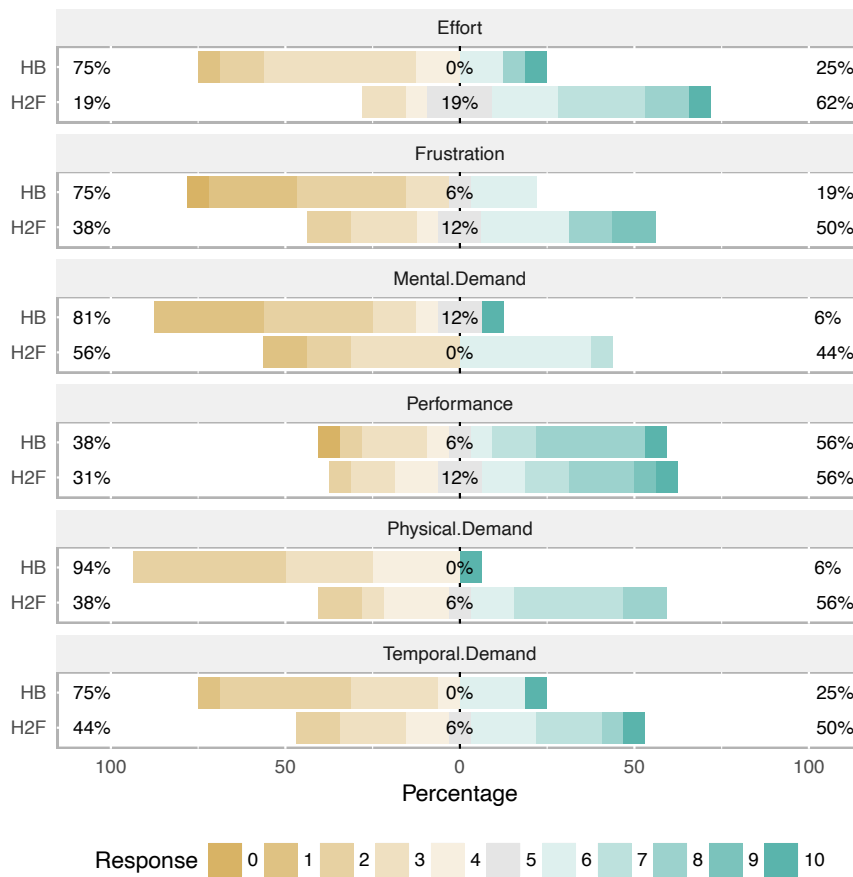
Figure 9.7 shows the mean euclidean error distance between the selected position by the user and the target position during the trials and the time between the appearance of a new target and the selection by the participant. The results were tested for normality with a Shapiro-Wilk test ( $p < .05$ ) and since they were not normally distributed a Wilcoxon Signed-Rank test was applied. The results are shown in Table 9.2. These results confirm H2. Table 9.2 and Figure 9.7 show that participants selected the targets in shorter time significantly. These results confirm H3. Additionally, the failure rates of both methods were calculated. A failure was defined by selecting a position at the wrong cylinder. For HB the failure rate was 1.29 % and for H2F 7.08 % in total.

## Usability

The raw TLX questionnaire results are summarized in Figure 9.6. They were normally distributed ( $p > .05$ ) and a paired samples T-Test was applied for the analysis. Taskload for method HB is significantly lower than for H2F and indicates that much less capacity of the operator is used. SUS score for HB is significantly lower than for H2F and denotes an upper B-rank (A-rank starts at 80.3), while H2F is much lower on the accumulated score with a low C-rank. Since the Shapiro-Wilk test for normality showed that the SUS results are not normally distributed, a Wilcoxon Signed-Rank test was applied to analyze them. These results confirm H1. The results for the AttrakDiff 2 questionnaire are visualized in Figure 9.7 and indicate a tendency that operators prefer the HB method over H2F and that they would use the system in real world tasks.

### 9.1.5 Validation of the Results

Based on the results described in Section 9.1.3, the HB selection method was implemented in a fully working prototype, which controls a real robotic arm. After the user triggers a selection by uttering (“*pick*” or “*place*”), the system first requests the planning system to



**Figure 9.6:** Comparison of NASA-TLX results show the advantage of the heading-based method.

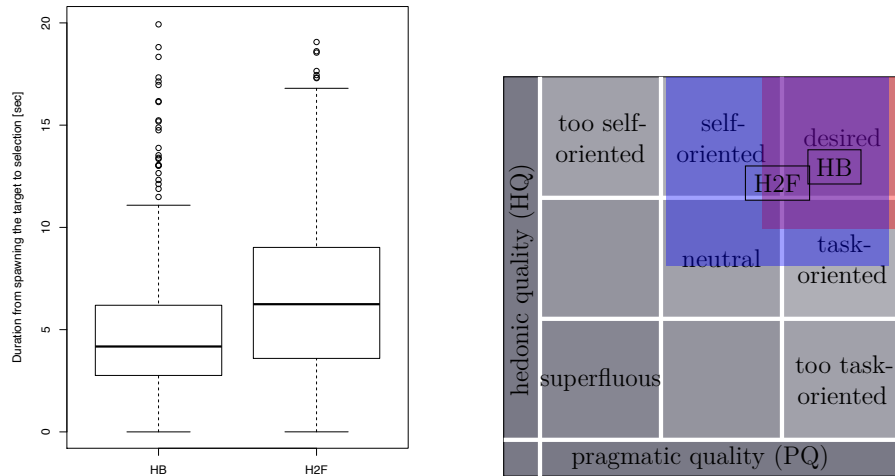
find a robot motion trajectory, and visualizes it in the HoloLens by moving the virtual UR5. The system then waits for another command for re-planning or a (“confirm pick” or “confirm place”) command, which causes the system to execute the already planned motion.

The virtual model of the robotic arm is used to provide the user with a preview of the motion at the place where it is intended to be executed. By doing so, the user is provided with visual feedback about the possible success and suitability of the input given to the robot. The presented stimuli using a see-through HMD have positive effects on spatial awareness and distance perception. As suggested in [57] a stereoscopic device was utilized. Since, visual perception is our most dominant channel [157], visual cues have the highest impact in decision making during robot operation.

### Validation Task

The proximity to the robot helps the user to understand the situation presented to the operator. The combination of head orientation and speech control keeps the hands of the operator free, allowing them to support the system with assistive manipulation tasks or





**Figure 9.7:** (left) Selection time HB vs. H2F. (center) Distance error HB vs. H2F. (right) Average values and confidence rectangles for the AttrakDiff questionnaire of the two conditions: (red) HB for the heading-based approach and (blue) H2F for the finger-pointing method.

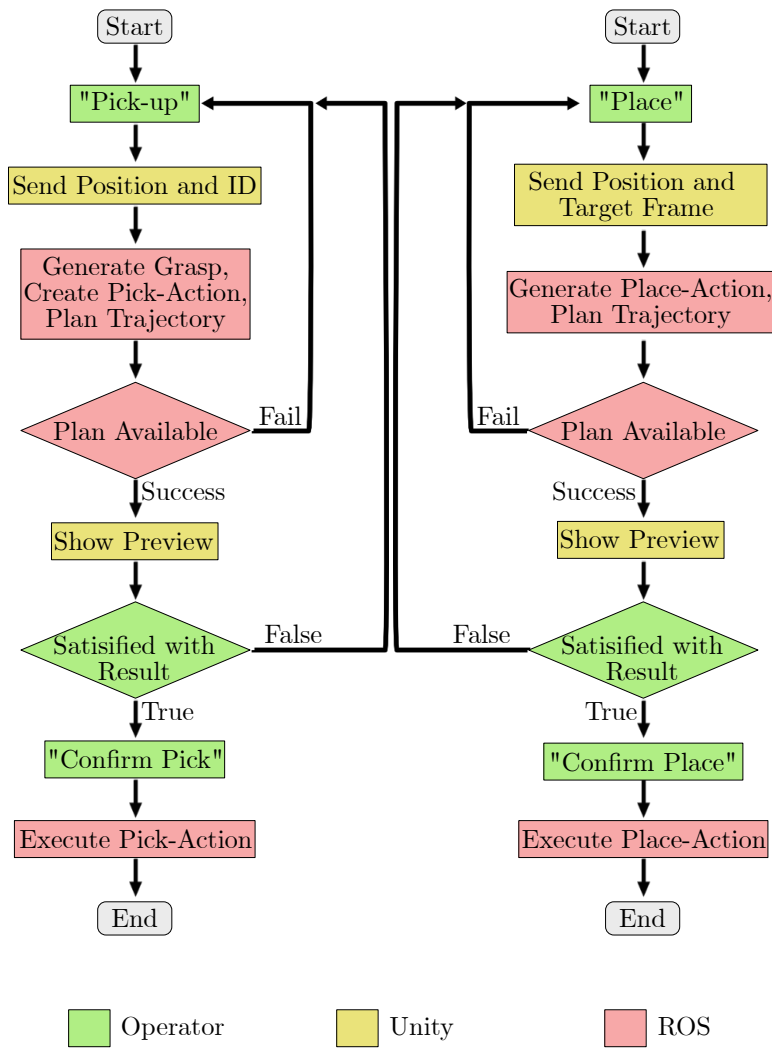
handover tasks. In the validation the system the system was capable of presenting the user possible problematic situations before they happen. Thus, the user is able to change the strategy before the problematic situation occurs during movement plan execution.

The system was presented to five experts in the fields robotics or 3D user interfaces. Overall, they gave positive feedback, and agreed that it simplifies the interaction in an intuitive way. The main critique was that it would be preferable to see in advance, whether the planner can calculate a trajectory to a point on an object, which can be solved using an approach like [119]. One expert suggested using shorter, one syllable speech commands.

### 9.1.6 Conclusion and Future Work

This chapter presented the general concept and implementation of an MR human-robot collaboration system in which a human can intuitively control a co-located industrial robot arm for pick-and-place tasks<sup>3</sup>. Two different multimodal HRI techniques were evaluated to define the pick and place location on a target object using (i) HB and (ii) H2F both in combination with speech. The results show that HB interaction technique is more precise, requires less time and is perceived as less physically, temporally and mentally demanding compared to the H2F techniques for MR-based pick-and-place scenarios. However, there are some limitations. It remains unknown if the results would also apply for another HMD

<sup>3</sup>The source code is available on GitHub:  
<https://github.com/denniskrupke/holoROS>  
[https://github.com/TAMS-Group/hololens\\_grasp](https://github.com/TAMS-Group/hololens_grasp)



**Figure 9.8:** Procedures of the pick-and-place implementation.

with larger field of view, allowing the user to lower their hand during the interaction, which would require less physical effort.

The study focused on the selection of a single pick point, based on simple input means. More complex grasping input will be possible when full hand tracking will be available with AR-HMDs. Furthermore, the multitude of information presented to the operator could be extended (cf. Section B.4.1). The range of the robot as well as the dexterous capabilities in the workspace would be of interest for further improvement of the user's performance (cf. Figure B.25). In the implementation the set of objects for manipulation is fixed. Future improvements will integrate object detection and the communication of new candidates between the side of the user interface and the robot.

To evaluate the full working prototype, a confirmatory expert evaluation was conducted, where an actual robot was controlled, gathering overall positive feedback. MR HRI setups require more research and the findings of this chapter reports important implications for the design of future MR setups and provides first steps towards novel forms of HRI.

PART

IV

## CONCLUSION



This dissertation presents theoretical, technical and practical results. In the long-term, these will support and improve human-robot interaction by utilizing the proposed mixed reality approaches. This chapter summarizes the main scientific results:

Chapter 5 summarizes selected theoretical aspects of robotics and 3D interaction in MR. Extraction of existing theoretical contributions from the literature and their reordering to an MR-RUI taxonomy, the *IMAct framework*, identifies technical factors, which contribute to specific characteristics of HRI systems involving MR. Furthering intentional MR-RUI design and aiding during design decisions are the main aims of this contribution.

Chapter 6 explores the first novel 3D interaction concepts and technical integration. The first section presents the concept of integrating a 3DUI into a pre-existing robot simulation and control system. Furthermore, different implementations of continuous posture-based hand control methods are evaluated — a comparison of translational and rotational joystick-metaphor-like controls in mid-air yields similar performance and precision. However, participants of the experiment prefer rotational control.

The second section analyzes reach-to-grasp movements in an HMD setting with a head-mounted hand-tracking sensor. Lessons learned incorporated that the actual task difficulty mainly influences the motion behavior of the operator's hand and thus should be regarded during system design.

Reaching movements are regarded as ballistic when objects with an index of difficulty below a certain threshold ( $ID < 3$ ) are targeted. Then, no correction phase occurs, in which the movement slows down. With a higher index of difficulty ( $ID \geq 5$ ), e.g.,

smaller targets, correction phases are observed and altered dynamics of the operator's hand acceleration are observable. Additionally, closed-loop tracking feedback with low latency seems to be more critical in the case of different object distances than in the case of different object sizes for obtaining reliable data.

Chapter B in the Appendices describes the successful process of creating an MR-RUI software framework for combining the robotic middleware ROS and the game engine Unity3D. Robot model reuse, communication and control are some of the topics addressed in this chapter. The framework is the basis of the use cases and experiments presented in chapters 7–9. Furthermore, many details about these implementations are explained in Chapter B. Due to its flexibility, effectiveness and efficiency, the framework suits well for the implementation of a multitude of MR-RUI.

Chapter 7 describes the prototyping process and provides practical hints for continuing the proposed work of the concept of combining a game engine with a robotic middleware. Finally, the chapter presents different use cases for examining related research questions. These are, due to the limited scope of this thesis, only partially investigated in a user study. Moreover, these use cases provide ideas on how to utilize the software framework to other specific topics of immersive HRI research.

Chapter 8 presents the results from the evaluation of a VR-based teleoperation system, which implements visual virtual fixtures combined with simple tactile feedback. The resulting system provides a useful aid for the continuous teleoperation of industrial robots in manipulation tasks. Significant findings are that the additional information, presented in the GUI, enables longer operation times, compared to implementations without the fixtures. Nevertheless, fewer collisions are assumable when using visual virtual fixtures.

Chapter 9 presents a full end-to-end system with blended reality. A Microsoft HoloLens is used in a co-located pick-and-place task for capturing commands from the operator and presenting possible future actions visually in the actual view. The superimposition of the actual robot by a stereoscopic rendering of its digital twin allows viewing a possible future trajectory in-situ. Due to the built-in tracking mechanisms, the operator can explore the actual space around the robot and predict probable issues before executing commands on the actual robotic hardware. A user study evaluates the overall setup and compares two alternative input methodologies. As a result, utilizing the viewing direction of the operator was superior to a finger-pointing-based approach.

The theoretical contribution of the IMPAct Framework is not a final solution; instead, if other fields of application than the context of industrial HRI, are focused, it should be revised and extended, accordingly. This thesis is affected by several technical limitations. Communication via the network induces latency but is neglectable in the presented scenarios. Other use cases may suffer from message conversion overhead, primarily when large amounts of data from different sources are transmitted, e.g., when incorporating several depth-cameras.

A trackable device for grasping input or reliable hand-tracking for MR would enhance the utility of the presented concepts at large. During the work on the dissertation on hand, several devices were under development or announced for purchase, but were not available during the evaluation of prototypes. Especially head-mounted hand tracking with the LMC is very impressive. However, the analysis of data recordings reveals that the spatial registration of the device is very inaccurate when the head is moving. Experiments with a fixed hand showed that head movements resulted in altered hand poses.

One aspect becoming prominent in the next future is the synchronization of digital twins with their alter egos from the real world, which is a topic for at least a whole dissertation. When real objects become automatically digitized and finally become available for interaction in the virtual world, MR will be a standard component in robotic applications. Applying this knowledge to a virtual scene, especially teleoperation scenarios in vast distances or at hazardous or unaccessible sites, would benefit. Despite some available technology, eye-tracking integration into available HMDs is still a topic to be

solved satisfyingly. Due to the relatively low field of view of available HMDs and the limited eye-tracking reliability, it was not promising to utilize this kind of technology in the experiments.

Nevertheless, future improvements of these devices could improve the interaction concepts and would make human actions more predictable. Especially in the setup with the Microsoft HoloLens in Chapter 9, the capabilities of registering the device frame to a world reference frame would benefit from increased camera resolution and quality. Future iterations of similar hardware would improve the results achieved with the presented setup.

The current implementation of ROS messages in the presented software framework would benefit from automated mechanisms, which analyze the actual ROS workspace and then provide the required message types for the actual system. This engineering work was beyond the scope of this thesis, but future improvements of the presented topic should take this into account.

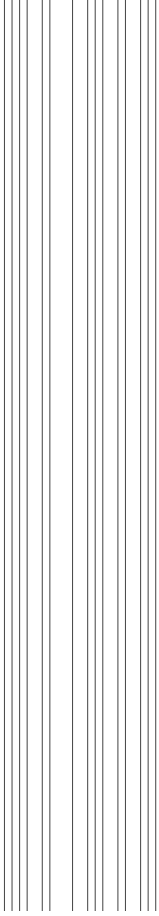
In general, investigating the topic of gesture-based interaction with robots in MR is very promising but still at an early stage. The fast progress of technology requires novel structures to combine available technology effectively. When technology in certain areas improves, it will certainly fill the gap of actual setups, which are always limited to a certain degree.



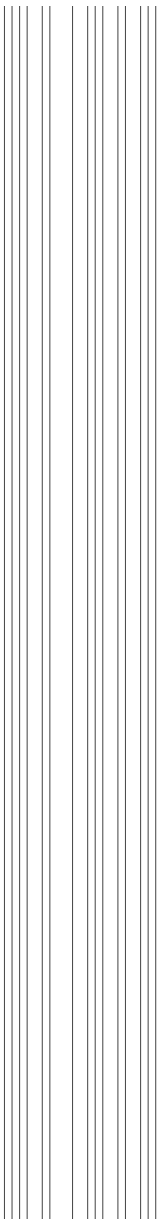
Mixed reality is currently in the follow-up phase to the latest VR hype, which coincided with technological novelties like low-cost tracking systems, inside-out tracking and self-contained HMDs. Together with the expanse of robot technology into many areas of application, mixed reality and robotics will grow even closer together. After the success of cleaning and lawn-molder robots, MR developments, together with the latest developments in artificial intelligence, will enable robots to be accessible and successful in more complex areas. The ability of robots to physically interact with the world will be the embodied connection between the virtual and real-world entities. An essential step in this development will be the integration of the latest progress in machine learning and sensor technology. Then, the robots in our daily life will be accessible via MR-RUI, the interface for naturally exchanging information and interaction requests. Artificial social intelligence will become perceivable through virtual representations in MR or through embodying a robot for physical interaction.

Different software frameworks for MR-RUI implementations, similar to the one proposed in this thesis, are currently under development in academic and industrial research. Results from these developments will further integrate robotic systems into our daily life. After the first industrial applications of MR-RUI, domestic and social applications, such as novel ways of MR entertainment, will approach. However, novel means of domestic and medical assistance systems will arise and influence our future.





# APPENDICES





## A.1 IMPAct Reference Table

**Table A.1:** This is the alphabetically-ordered list of the extracted relevant factors from the literature investigation including their possible values as lists or continua. Additionally, the descriptions placed on the backside of the cards from the card sorting experiment are listed.

Factor	Values	Description
Action Measurement Type [163]	tracker, glove, joystick, force feedback arm, ...	What technology is used to measure action of the operator as input to the system?
Action Type [114]	discrete–continuous	If you have to do something permanently or just from time to time.
Actuator Type [163]	robot arm, STM tip, aircraft flaps, ...	What kinds of actuators are controlled within the system by means of the operator?
AFR Interaction Level of the Robot [142]	Type A–Type D	Which interaction level fits the robot, involved the system, at most?
Causality of Modalities [163]	simulated, recorded, transmitted	Where do modalities (perceptible sensations) have their origin.

*Cont.*

<b>Factor</b>	<b>Values</b>	<b>Description</b>
Control Alignment [128]	1-to-1, positional offset, rotational offset, pose offset	How are input means from in the real world connected to a linked virtual world?
Control Order [128]	Order of Mapping Function (0–n)	What is the complexity of the relationship between user input variables and controlled variables?
Directness [128]	degree of mapping to display space/isomorphism	What is the degree of spatial matching between display (not only visual) content and linked modalities in the real world?
Directness of Sensation [128]	directly-perceived-real-world – scanned-and-resynthesized	How is the real world perceived through technical devices?
Display Centricity [131, 128]	egocentric–exocentric	What is the spatial relationship between contents of a display (not only visual) and the user?
Display Class (Milgram) [131]	class 1–class 7	How is the operators display classified according to Milgram?
Display Space [163]	registered, remote, miniaturized/enlarged, distorted	How are different spaces aligned (virtual, real, ...)?
Display Type [163]	HMD, screen, speaker, ...	What kinds of displays are utilized?
DOF (Degrees of Freedom)—parameters manipulated [114]	0–n	This is about the number of values altered during interaction, not the robot structure.
DOF (Degrees of Freedom)—input device based [114]	0–n	This is about the number of values assessed from means of input.
Extensiveness [184]	single modality–multi-modal	How many different modalities are presented to the operator by the system?
Extent of Body Matching [184]	no correlation–perfect matching	In case of the existence of a virtual body: How strong is the virtual body matching with corresponding real body?
Extent of Proprioceptive Matching [184]	no correlation–perfect matching	In case of the existence of a virtual body: How strong can the proprioceptive matching be?

*Cont.*

Factor	Values	Description
Extent of Presence Metaphor (EPM) [129, 131]	WoW/monoscopic imaging vs. HMD/real-time imaging	How strong is the display technology likely to induce sense of presence. Or better: How immersive is the technology?
Extent of World Knowledge (EWK)/where and what [129, 131]	unmodeled–modeled	How much of the real world is known in detail?
Gesture Type [193]	static posture, static oriented posture, dynamic/moving gesture, moving oriented gesture	If so, what category of gesture is used?
Human-Robot Communication [29]	single dialog–two (or more) monologues	Is the communication between human and robot based on equal rights?
Image Scaling [128]	(1:1–1:k)/ (orthoscopic/conformal mapping vs. scaled images)	How is the size of rendered modalities in comparison to real entities?
Inclusiveness [184]	none–complete	To which degree are you shut off from the modalities of the real world around you?
Initiative [6]	fixed–mixed	How is the initiative of acting or communicating between human, robot and the rest of the system? Fixed at a single actor or distributed between several partners?
Integration of VE [225]	separated–integrated	Spatio-temporal relationship between real environment and virtual environment.
Intelligent Robot Control Paradigm [124]	hierarchical, reactive, hybrid	How is the intelligent behavior of the robot organized?
Interaction Reality <sup>1</sup>	physical–virtual	Is the interaction mainly with real or virtual objects?
Interaction Type [26]	Selection and Manipulation, Travel, System Control	Which of Bowman’s 3D interaction category is mainly present?
JIRA Intelligence Level of the Robot [142]	Class 1–Class 6	Which class of intelligence fits the robot, involved in the system, at most?

*Cont.*

Factor	Values	Description
Level of Capabilities (cf. [67])	System skills are not enough to solve/act-out any task–system is capable of acting-out at least any task perfectly	How (potentially) skilled is the system independent from the operators skills?
Level of Intelligence (cf. [67])	System never generates a theoretic solution towards a goal–system always finds optimal solution	How intelligent is the system without the operator?
Level of Mixed Reality [131]	RE–AR–AV–VE	Where is the current system positioned on Milgram’s Reality-Virtuality Continuum?
Level of Robot Anthropomorphism [18]	none–non-distinguishable from human	How human-like is the robot?
Location <sup>2</sup>	in locu vs. remote	What is the mental distance between the virtual and the real world components?
Mapping Relationship [225]	position–rate	How input is mapped to output mathematically.
Model of Modality-Source [163]	scanned, constructed, computed, edited	How are modalities (perceptible sensations) generated?
Parameter Manipulation Type [114]	direct–indirect	In which way is a value changed.
Plot [184]	none–perfect integration into mental models	Is there a plot and how good does it fit the situation?
Real-time Level [124, 39, 142, 134]	no delay–delayed	How close to real time is the system behavior?
Refined LOA in steps acquisition, analysis, decision and action (or sense, plan and act) [150, 132, 182]	1–10	Which level is the autonomy of the system in the single steps of executing a task: “analysis”, “decision”, “action”?
Reproduction Fidelity (RF) [129, 131]	monoscopic video/wireframes vs. 3D HDTV/real-time high-fidelity	How is the quality of the presented modalities to be rated?
Robot Behavior Level [18]	remote controlled–autonomous	How can the behavior of the robot be rated? Is it doing something on its own? Does it have some software induced abilities?



*Cont.*

Factor	Values	Description
Robot Communication Capability [18]	reactive–dialog	How is the robot communicating with the operator? Only on demand, or does it have needs to communicate?
Robot Control Concept [124, 142]	open loop, feedback control loop, feedforward control, adaptive control	How is the robot controlled in general?
Robot Purpose [18]	toy–tool	What is the typical purpose of the current robot?
Robot Type [124, 39, 142, 134]	Industrial, Mobile, Service, Social, ...	What kind of robot is used in terms of common sense?
Role of Interaction [176]	supervisor, operator, mechanic, bystander, team mate, manager/leader	Which social role inherits the operator? (Scholtz)
RUI-Type [124, 39, 142, 134, 18]	programming, command-line, real-time, interactive	How is the robot itself mostly interfaced?
Sensing Mode [225]	isotonic–elastic–isometric	Mapping from signal source to sensor reaction.
Sensor Type [163]	Photomultiplier, STM, ultrasonic scanner, ...	What kind of sensors are used in the system for generating information for the operator?
Structure of Interaction [45]	[SO, SR, ST], [SO, SR, MT], ..., [MO, MR, MT] (S = single, M = multiple, O = operator, R = robot, T = Tasks)	What is the typical combination in the system?
Superposition of Modalities [163]	merged, isolated	How are technologically mediated modalities presented in comparison to modalities which are part of the real environment?
Surrounding [184]	small 2d image (WoW)–large 2d–curved–cylindrical–spherical	How much is the display everywhere around you?
System Extent <sup>3</sup>	self-contained/all-in-one–network-based across planets	How much is the system, especially the user interface distributed among different devices?
Time (interaction) [163]	1-to-1, accelerated/retarded, frozen, distorted	How is the interaction time related to execution time?

*Cont.*

<b>Factor</b>	<b>Values</b>	<b>Description</b>
Time (purpose) [124, 39, 142, 134]	real time–recorded for later	How much time passes between the user input and the manipulation of the real world by the robot by intention.
Timely Aspects of Actors' Autonomy [29]	static–dynamic	Is the level of autonomy regarding different aspects of the system changing over time (depending on some other factors) or is it static and defined by design?
Transparency of System State [91, 33, 55]	invisible–all states are displayed	Does the operator know the system state during operation?
UI-Type [26, 85]	Batch, CLI, GUI, NUI, SUI, ...	What stereotype of user interface describes the present system at most?
User Experience [163]	recorded, transmitted, simulated, supervised robot	How originate the modalities the user is experiencing?
Vividness [184]	very low–very high	How good is the rendering quality?

<sup>1</sup> This factor was added by the author. After reading many publications about interactive virtual environments (IVE), it was inevitable to add this. <sup>2</sup> This factor was added by the author. The spatial distance between the virtual and the real environment was only implicitly part of the discussions in the literature. <sup>3</sup> Added by the author. The general system design, based on physical components, was not found explicitly in the literature of mixed reality and robots.

## A.2 IMPAct Template

Table A.2: Empty IMPAct template.

Category	Group	Factor	Value	
INTERACTION	Interaction Parameter	Action Type		
		AFR Interaction Level		
		Control Alignment		
		Control Order		
		Directness		
		DoF		
		Interaction Plasticity		
		Interaction Type		
		Mapping Relationship		
		Parameter Manipulation Type		
		Sensing Mode		
	Structure of Interaction			
	Paradigm	Human-Robot Communication		
		Intelligent Robot Control Paradigm		
		Robot Control Concept		
		Role of Interaction		
		RUI-Type		
	MEDIATION	Image Display Vision	Display Centricity	
			Display Class (Milgram)	
(Display) Space				
Image Scaling				
Vividness				
Input		Action Measurement Type		
		Directness of Sensation		
		DoF		
		Sensor Type		
Output		Actuator Type		
	Display Type			
PERCEPTION	Embodiment	Extent of Body Matching		
		Extent of Proprioceptive Matching		
	Immerison	Extend of Presence Metaphor		
		Integration of VE		
		Level of MR		
		Reproduction Fidelity		
	Modalities	Causality of Modalities		
		Extensiveness		
		Inclusiveness		
		Model of Modality-Source		
		Superposition of Modalities		
	Space & Time	Extent of World Knowledge		
		Location		
Realtime Level				
Surrounding				
System Extent				
Time (interaction)				
ACTING	Autonomy	Level of Capabilities		
		Level of Intelligence		
		Plot		
		Refined LOA		
		Initiative		
		JIRA Intelligence Level of the Robot		
		Timely Aspects		
	Behavior	Robot Behavior Level		
		Robot Communication Capability		
		Transparency of System State		
	RA	Level of Robot Anthropomorphism		
	Application	Robot Purpose		
		Robot Type		



## COMBINING ROS AND UNITY3D

As confirmed by the amount of publications involving ROS and Unity3D, it is very obvious that combining these two very sophisticated examples of generic middleware with large communities is a fruitful project. ROS is already the first choice in academia for robotics research and the current generation of UI research uses 3D game engines and 3DUI devices. ROS offers required components for robotics but for interaction design the possibilities are rather limited. There is experimental support of the Oculus Rift DK1 in the “*PR2-surrogate package*” but with low rendering performance compared to the direct use of game engines. High framerates are required to avoid motion sickness during use [76], meanwhile latency has a negative effect on the sense of presence [126]. Nevertheless, ROS provides a high amount of useful packages and supports almost every robot and thus, should be part of a framework for MR-RUI. Fortunately, the “*robot web tools*” [204] provide a bidirectional network interface to ROS and allows for integration of external systems via the network.

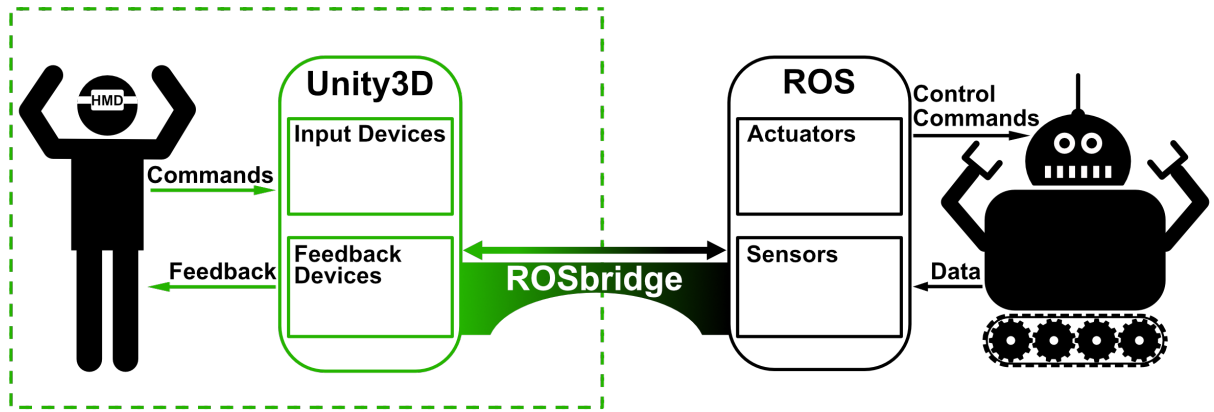
Among the prevalent gaming engines, Unity3D, Unreal and Cry-Engine are mostly used in MR scenarios and are able to provide the necessary rendering and input processing capabilities. Analyzing the number of publications containing the words “*ROS*” and “*Unity3D*” listed on Google Scholar reveals a strong growth

## B.1 Introduction

This chapter introduces an open-source software toolkit for combining the Robot Operating System (ROS) and Unity3D to versatile robotic applications involving virtual environments. Despite the availability of high-quality robot control and simulation systems like ROS, there is still no framework available for designing complex human-robot interaction tasks and making use of Virtual Reality without expert knowledge in robotics and ROS. Virtual Reality, especially involving head-mounted displays as well as various input and feedback devices can increase the experienced sensation, improve the understanding of certain 3D scenes or provide a test environment for the training of non-expert operators. As a solution, we propose Unity3D for designing the interaction interface to virtual and real robots. Our implementation of the bilateral communication layer between Unity3D and the ROSbridge and the importer for XML-based files in Unified Robot Description Format allows synchronizing the state of the real robot and the one of its simulated counterpart in the Unity3D environment in a very comfortable way. Using the proposed system we present different use cases and demonstrate how to prototype different interaction concepts efficiently.

Looking for a robot operating system, which features modularity, reusability and collaborative development of standard software components for available hardware like sensors and robots leads to ROS [160]. Many state-of-the-art devices and tracking systems, which are of importance in virtual and augmented reality, only have drivers for Windows, leaving Linux based systems like ROS unsupported. Additionally, ROS does not focus on high-fidelity rendering for creating a high degree of immersion, which is important for various psychological aspects. Better interaction methods for our robots are desirable in order to take over control seamlessly, to postulate high-level commands or to input data for learning algorithms. If we create easier accessible interfaces, the availability of robots to a wider scope of users is improved. Unfortunately, ROS is not well prepared to setup user studies or to create intuitive interactive interfaces easily. For Virtual Reality (VR) and human-computer interaction research Unity3D is very suitable to implement user interfaces. The challenge is to integrate the communication between a Linux based system and one that is dominated by Windows to an end-to-end system. Part of the solution is an already published ROS package, the *ROSbridge*. Codd-Downey et al. tried to connect both systems: ROS and Unity3D [34]. It combines reliable robot control with a very flexible prototyping environment for interactive, immersive systems.

The proposed framework consists of a collection of Unity3D assets and enables to bi-directional communication between the subsystems ROS and Unity3D and devices attached to these. The “*general architecture*” is presented in section B.2 and explained using an example. Specific components of the implementation are assigned to the categories “*communication*” (cf. section B.3), “*feedback*” (cf. section B.4), “*robot control*” (cf. section



**Figure B.1:** A general system architecture for integrating ROS and Unity3D. The architecture is easily extendable and allows for implementation of heterogeneous robotic systems with benefits for the operator and the robot control itself. Green elements indicate the focus of this contribution.

B.5), and “tools” (cf. section B.6). The modular and very versatile nature of the framework allows for combining into arbitrary systems.

Designing an end-to-end system based on the proposed concept involves the selection or creation of appropriate ROS nodes and the programming of corresponding components in Unity3D. At the ROS side available programming languages for writing new nodes is C++ or Python and at the Unity3D side C# is necessary with the limitation to the actually supported .NET version.

The framework was designed to result in a versatile toolkit with a graphical editor and support for arbitrary 3D input and output devices, such as tracking systems and head-mounted displays (HMD) for VR and augmented reality (AR). Thus, the toolkit provides access to the ROS messages and communication structure inside Unity3D.

## B.2 General Architecture

The general system architecture for setting up an end-to-end system follows Figure B.1. The cooperation between humans and robots is guaranteed by the implementation of the ROSbridge interface in Unity3D at the human-side and by connecting it to the ROSbridge node at the robot-side of the system. The bi-directional communication is provided by dedicated nodes via the network. Transferred information is used at both sides for different kinds of control or feedback. The user interface is based on a closed-loop action-perception cycle, which is embedded in a VR scene presented to the user with an HMD. By utilizing positional and rotational head tracking, users have the freedom to explore the scene in a more natural way, compared to the classical view on flat screens. For instance, manipulation tasks benefit from the possibility to change the point of view in order to avoid occlusion in regions of interest by other parts of the scene.

Teleoperation systems for mobile robots have been built, presenting streamed video or point cloud information in an egocentric view from the robot. Operators benefit from better understanding, lower mental workload and better performance in e.g. a tabletop manipulation scenario [122, 156]. But also grasp planning systems can benefit from the 3D view by the possibility to change the user’s perspective, especially when potential target objects are occluded by other elements of the environment [75]. Using mid-air gestures as control input provides a high amount of freedom for the user, but has the lack of direct feedback and reliability [19]. One solution is to increase the level of autonomy of the control system. Gestures will be interpreted and actions are not applied from a direct mapping of user input [87].

For the best of our knowledge, there is no open-source implementation of the communication and the import of robot models from ROS to Unity3D. We developed a toolkit which is further extendable and enables efficient prototyping of human-robot interaction (HRI) user interfaces with full access to ROS messages. Real hardware is accessible with low latency via the network and simulated components can run locally with simulated latency.

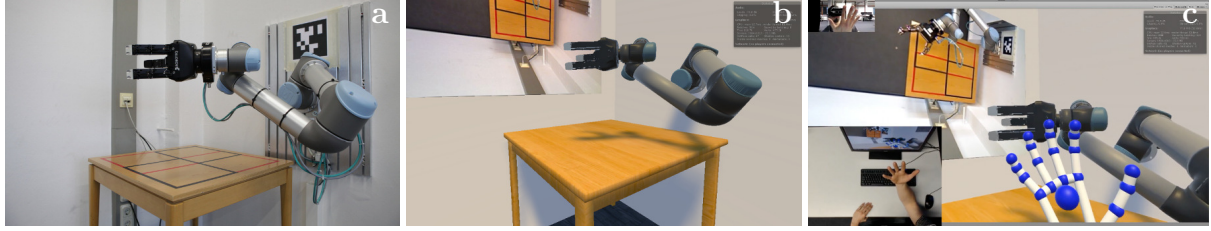
### **B.2.1 Concept of Control and Feedback Integration**

The ROS side implementation of the ROSbridge is maintained by the community. For the integration into Unity3D, a ROSbridge script is implemented in C#. ROS messages are implemented in C# as a ROS message layer. Additionally the Newtonsoft JSON library for .NET is utilized to generate the data strings from the message objects. Using the JSON library, a custom serializer for nested objects has been implemented. A Robotiq 3-finger adaptive gripper is controlled by *ROS gripper actions*, e.g. closing or opening the gripper to a certain percentage, which are available for a wide range of different grippers. The real robot issues its current state by publishing ROS messages on the `/jointstate` topic. At the user interface side visual feedback is provided when entering or leaving the control sphere and by mirroring the current state of the robot using the virtual robot model and a video stream of the laboratory.

### **B.2.2 Example of System Engineering**

The general system architecture follows Figure B.3. The cooperation between human and robot is guaranteed by the integration of Unity3D at the human-side and ROS at the robot-side of the system. The communication between both sides is provided by the ROSbridge, which uses WebSockets for the protocol and JSON strings as a container for information. The user interface is based on a closed-loop action-perception cycle, which is embedded in a VR scene presented to the user with an HMD. By utilizing positional and rotational head tracking users have the freedom to explore the scene in a more natural





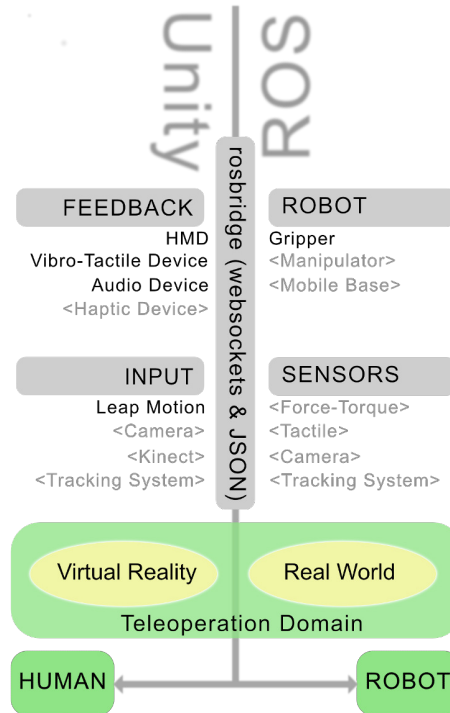
**Figure B.2:** A real laboratory environment (a) and the virtual corresponding scene (b). The posture of the virtual robot in Unity3D is synchronized by ROS messages containing the current posture of the real robot. The robot hardware is controlled by interaction with the virtual environment (VE) from egocentric perspective (c).

way, than in a classical 2D view on flat screens. Manipulation tasks benefit from the possibility to change the point of view in order to avoid occlusion of regions of interest by parts of the scene. Humans at the remote side of the system can take over control of the robotic hand by placing the controlling hand into a semi-transparent sphere in VR. For reasons of comfort and reliability, once the controlling hand is placed in the sphere it follows slow motions of the user's hand. Meanwhile placing the hand inside the control-sphere the posture of the real human hand is analyzed and transduced to high-level control signals for the robotic hand, which can easily be applied to different gripper hardware. As input device for the hand tracking an LMC is utilized, which is mounted using a modified version of the VR mount of the manufacturer of the LMC directly at the front side of the Oculus Rift.

In Figure B.2, the real laboratory environment (left) and the virtual corresponding scene (right). The posture of the virtual robot is synchronized by messages with the current posture of the real robot, which are sent via the ROSbridge to Unity. A virtual screen at the left wall shows a webcam stream from a ceiling camera above the robot for safety reasons. An operator was recorded during the use of the teleoperation system (right). Head movements are shown by a webcam image in the upper left corner, as well as hand movements. A top-view camera recording in the lower left corner of the figure shows the hand postures with more detail. In the virtual scene a hand skeleton (LMC Orion SDK) mirrors the state of the tracked real hands (cf. Figure B.2).

## B.3 Communication

As indicated in Figure B.1, the framework is based on the implementation of a bi-directional communication component, which connects Unity3D and ROS via the network. This connection serves as the necessary component to exchange information between the two subsystems based on ROS messages. Figure B.3 illustrates an example for a tele-grasping system and its components which are producing or consuming information during runtime. At the ROS side the ROSbridge from the RobotWebTools is utilized



**Figure B.3:** A general system architecture for integrating ROS and Unity3D. The architecture is extendable and robotic components are easily exchangeable.

as one of the communication entities. The implementation of the Unity3D counterpart is described in this section. Messages are implemented according to the naming and structure of ROS messages in order to avoid confusion when working with both subsystems. Especially how incoming and outgoing messages are processed and parsed determines the performance of the system, which is crucial to good MR experiences. Thus, a hard requirement is to avoid a performance-based coupling of the communication processes with the render-loop in unity. The main features are summarized in the following:

- implementation of *extendable* ROS message structure in Unity
- *buffers* for incoming messages and outgoing messages
- multi-threaded workers for *concurrent processing* of messages
- JavaScript Object Notation (JSON) *de-/serialization* from and to ROS messages

### B.3.1 Messages

Communication is the process of information exchange. In order to agree on a format, the ROS messages system is used. Obvious benefits are that at the ROS side the messages are already known and that the ROSbridge node is able to perform serialization and

deserialization into JSON format. In order to avoid confusion for system designers and programmers it was a reasonable decision to implement this messages structure at the Unity3D-side as well. The messages follow a frame-based concept, where typically one frame consists of a certain set of data at a specific point of time. Each message contains some meta information, like a timestamp and the data itself. Message types are ranging from empty messages, to basic data types, composed types and array-based types which typically contain the information how to decode the containing information.

## ROS Message Structure

ROS implements the publisher-subscriber model to allow for communication between different **nodes** over the network; organization of communicated data is implemented by using strings as an address, so called **topics**. The format follows Unix conventions with the symbol “/” as root-topic and followed by a topic name or a “*subtopic*”, similar to the organizational structure of filesystems. In this way the modular approach supports the distribution of computational load between different computers in the network and the integration of several actuators and sensors or robots into a single system. For this purpose the most important ROS message types are: the “*advertise*” message (cf. Figure B.6), the “*subscribe*” message (cf. Figure B.7) and the “*publish*” message (cf. Figure B.8). **Advertise** messages announce and specify a new topic. **Subscribe** messages let a node register on a topic in order to let all future frames of these kind of messages routed to itself until unsubscribed. **Publish** messages contain a data frame typically occur periodically with a certain periodicity. In the context of this dissertation these messages are of high importance, since these are containers for the data transferred between ROS and Unity3D. The data object itself can contain different message types, like the header or primitive or composed types. In ROS, there exist more message types like service calls or messages regarding the parameter system. But in this thesis advertise, publish and subscribe messages were sufficient to implement and explain the presented concepts.

```

1 # Standard metadata for higher-level stamped data types.
  # This is generally used to communicate timestamped data
  # in a particular coordinate frame.
  #
  # sequence ID: consecutively increasing ID
6 uint32 seq

  # Two-integer timestamp that is expressed as:
  # * stamp.sec: seconds (stamp_secs) since epoch (in Python the variable is called 'secs')
  # * stamp.nsec: nanoseconds since stamp_secs (in Python the variable is called 'nsecs')
11 # time-handling sugar is provided by the client library
   time stamp

  # Frame this data is associated with
   string frame_id

```

**Figure B.4:** Original ROS Message Header.

Figure B.4 shows an example of the ROS Message Header as an example how the ROS messages are documented online at <http://docs.ros.org>. Since, the implementation of ROS nodes are programmed in C++ or Python, the documentation is formatted in Python.

## Unity3D ROS Message Implementation

The C# implementation of ROS messages for Unity3D is organized in the major namespace `RosMessages`. Sub-namespaces exist according to the original ROS message structure. Examples are `std_msgs` or `sensor_msgs`. The three basic types `RosAdvertise` (cf. Figure B.6), `RosSubscribe` (cf. Figure B.7) and `RosPublish` (cf. Figure B.8) extend the abstract base class `RosMessage` (cf. Figure B.5).

```
namespace RosMessages {  
  
    public abstract class RosMessage {  
        public static ulong _id = 0; //global message count  
        public string op;  
        public string id;  
        public string topic;  
        public string type;  
  
        protected RosMessage(string op, string topicName) {  
            _id++;  
            this.op = op;  
            this.id = op + ":" + topicName + ":" + _id;  
            this.topic = topicName;  
        }  
    }  
}
```

**Figure B.5:** Abstract Class `RosMessage` in Unity3D.

All properties exactly match then original implementation of ROS messages at the ROS-side. This is necessary to provide compatibility with the ROSbridge-node at the ROS-side and to assist programmers with experience in ROS programming. The specification of the “*ROSbridge protocol*” is available online.<sup>1</sup> Naming and data types of ROS messages need to match in order to be allowed to use standard JSON libraries like “*Newtonsoft JSON*” for serialization and de-serialization of messages.

```
namespace RosMessages {  
  
    public class RosAdvertise : RosMessage {  
  
        public RosAdvertise(string topicName, string messageType) :  
            base("advertise", topicName) {  
            this.type = messageType;  
        }  
    }  
}
```

**Figure B.6:** ROS Advertise Message.

---

<sup>1</sup>[https://github.com/biobotus/rosbridge\\_suite/blob/master/ROSBRIDGE\\_PROTOCOL.md](https://github.com/biobotus/rosbridge_suite/blob/master/ROSBRIDGE_PROTOCOL.md)

When using the ROSbridge “*advertise messages*” are generated containing all relevant parameter values instead of specifying the appropriate settings in a code line. Figure B.6 shows the implementation of the advertise message, prepared for automatic serialization to a JSON string. Unique ids are generated to allow for unadvertising of publishing to a topic and must be stored if stopping is required. An advertise message is identified by checking the op string. The serializer in Unity3D does not differentiate between different message types. All properties are simply placed in a string for each possible message. The type specifies the kind of `MessageData` which will be published on the topic, specified in the property string `topic`.

```

namespace RosMessages {
    public class RosSubscribe : RosMessage {
4      public string compression = "none"; // e.g. "png"
      public int throttle_rate = 0; // minimum amount of ms between messages
      public int queue_length = 1; //0 : infinite, full queues drop the oldest message

      public RosSubscribe(string topicName, string messageType) :
9          base("subscribe", topicName) {
          this.type = messageType;
      }

      public RosSubscribe(string topicName, string messageType, int throttle_rate,
14         int queue_length, bool compression) : base("subscribe", topicName) {
          this.type = messageType;
          this.compression = compression ? "png" : "none";
          this.throttle_rate = throttle_rate;
          this.queue_length = queue_length;
19     }
    }
}

```

Figure B.7: ROS Subscribe Message.

The `RosSubscribe` message type is provided with a minimal constructor, using the default values for `throttle_rate`, `queue_length`, and `compression` and a detailed constructor, which allows for specifying all of the parameters at once. Since, the ROSbridge currently only supports png compression of the transmitted JSON strings, the compression property string is mapped to a boolean value which determines if compression should be applied or not.

```

namespace RosMessages {
    public class RosPublish : RosMessage {
3      public MessageData msg;

      public RosPublish(string topicName, MessageData messageData) :
          base("publish", topicName) {
8          this.msg = messageData;
      }
    }
}

```

Figure B.8: ROS Publish Message.

Publish messages contain a data object of type `MessageData` (cf. Figure B.10) with different subtypes. The presented implementation uses an interface of the type `MessageData`, which is used to implement the concrete type. The (de-)serialization facilities at the Unity3D-side is prepared to handle objects, which implement this interface. Most message data types include a header object as one of their properties, which is a itself a composed message (cf. Figure B.9). A header message object is typically only a property of a `MessageData` object and not published solely as a ROS message. Thus, it is not tagged with the `MessageData` marker interface. Composed messages like `JointState` in the namespace `RosMessages.sensor_msgs` contain a `Header` object (cf. Figure B.11) and the specific sensor related data like arrays of joint names, their actual position at the time specified in the header and/or position and effort at a certain joint.

```
namespace RosMessages {
  namespace std_msgs {
3     public class Time {
        public int secs;
        public int nsecs;
    }
8
    public class Header {
        public Time stamp;
        public string frame_id;
        public ulong seq;
13    }
  }
}
```

Figure B.9: Header Implementation.

```
namespace RosMessages{
4   /**
   * Marker interface for indicating this object is containing the data
   * frame for (de-)serialization
   **/
   public interface MessageData {}
9 }
```

Figure B.10: MessageData Marker Interface.

`MessageData` is implemented as an empty marker interface (cf. Figure B.10). To make sure only to (de-)serialize the right objects containing relevant data, message data classes are tagged by implementing this interface. All ROS messages described in [http://wiki.ros.org/common\\_msgs](http://wiki.ros.org/common_msgs) and related pages and all custom types, which should be processed in Unity3D must implement this interface in order to be (de-)serializable.

```

1 namespace RosMessages {
  namespace sensor_msgs {

    public class JointState : MessageData {
      public Header header;
      public string[] name;
      public double[] position;
      public double[] velocity;
      public double[] effort;
    }
  }
}

```

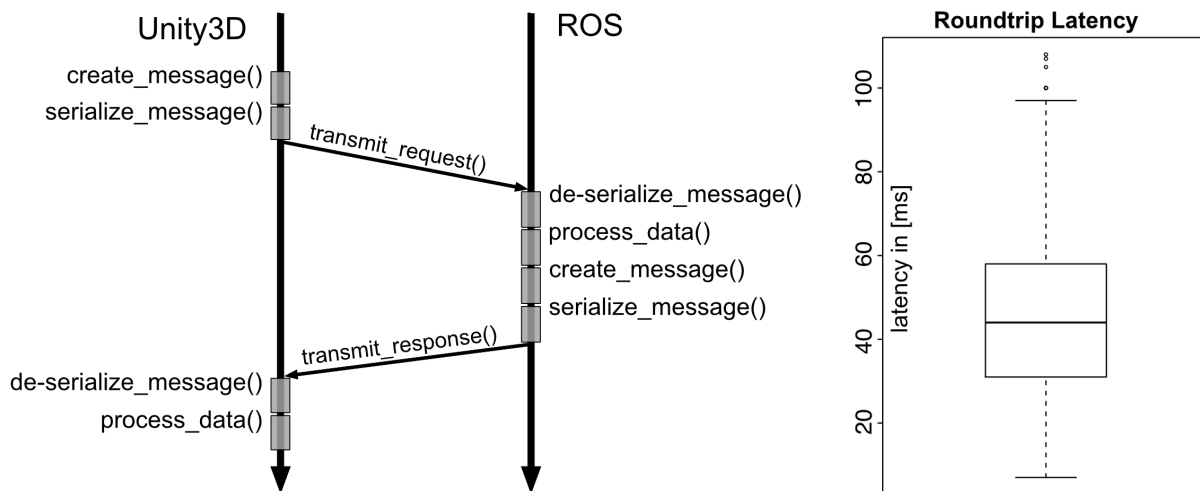
**Figure B.11:** Example of a `MessageData` Class: `Jointstate`.

The Unity3D implementation adopts the same class-hierarchy as at the ROS-side. Since ROS allows for implementation of custom messages, it would be necessary to extend the implementation at the Unity3D-side according to newly introduced message types, as well.

### Communication and Message Processing in Unity3D

Figure B.12 (left) shows the communication architecture of the system, implemented in a feasibility study for the example described in Section B.2.2. Messages are exchanged between the two systems via network sockets. They contain serialized JSON objects from ROS messages. At both sides, outgoing information is firstly packed into ROS messages to ensure the compatibility to ROS. Conversion from incoming ROS messages to updates on the virtual environment takes places in dedicated Unity3D scripts, the *message interpreter scripts*. Conversion from user input to outgoing ROS messages for controlling the robotic hardware is performed in specialized *user input controllers*. For instance, the Leap Motion Controller (LMC) software captures the necessary information from the active hand and generates reasonable control signals, which are packed in a JSON wrapped ROS message (B.2.2).

The ROS side processes incoming ROS messages and sends out status updates (cf. Figure B.13) to keep the remote-side synchronized to the current real state of the robot. Induced latency by serialization and de-serialization processes and additional network communication overhead has to be kept as low as possible for two reasons. One is for delivering the best possible experience regarding the virtual world, which is often strongly related to rendering rate (FPS) and the other is for the communication with robotic hardware, causing delayed control or sensing, which would lower the user performance during use of the system. As visualized in Figure B.12 (left), it becomes clear that efficient serialization and de-serialization is one the major tasks in Unity3D. Depending on network infrastructure and the locations of the end points of the communication an overall mean roundtrip latency of  $\sim 40$  ms is achieved (cf. Figure B.12, right) by concurrent message exchange, (de-)serialization, and the typical Unity3D core processes.



**Figure B.12:** Message passing method and its latency for integrating ROS and Unity3D.

```

2  {
  "op": "publish",
  "topic": "/joint_states",
  "msg": {
    "header": {
      "stamp": {
7     "secs": 1496224563,
        "nsecs": 430524501
      },
      "frame_id": "",
      "seq": 75672
    },
    "name": [
12     "fl_caster_rotation_joint", "fl_caster_l_wheel_joint", "fl_caster_r_wheel_joint",
      ...
      "l_gripper_motor_screw_joint", "l_gripper_motor_slider_joint"
17    ],
    "position": [
      -0.00520137485575245, -0.012026409377022705, 0.06105715529873065,
      ...
22     0.012144723369896665, 0.012144723369896665, 0.012144723369896665, 0.0, 0.0
    ],
    "velocity": [
      -0.0, 0.0, -0.0, -0.0, 0.0, -0.0, -0.0, 0.0, -0.0, -0.0, 0.0, -0.0, -0.0, 0.0,
      ...
27     0.43508920099594406, 0.1740356803983776, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
    ],
    "effort": [
      0.5215768560500686, -0.34028010079522136, 0.34028010079522136, 0.1757184028262289,
      ...
32     1.2236703079190574, 0.04208393418991263, -0.0, -0.0, -0.0, -0.0, -0.0, 0.0, 0.0
    ]
  }
}

```

**Figure B.13:** Example of a serialized publish message containing JointStates data (shortened).

### JSON Processing

The JSON facilities in the proposed toolkit are based on the .NET Newtonsoft JSON library.<sup>2</sup> As shown in Figure B.14 a `JsonCreationConverter` for the template parameter

<sup>2</sup><https://www.newtonsoft.com/json>



type of `MessageData` has to be implemented and is prepared for easy extension with other subtypes of `MessageData` e.g. custom ROS messages or other standard messages which are not implemented at the Unity3D-side, so far. Newtonsoft implements a factory pattern to instantiate corresponding classes from JSON objects which are automatically created from a transmitted string.

```

1 namespace RosJSON {
    public class MessageDataConverter : JsonCreationConverter<MessageData> {
        string topic = "";

        public MessageDataConverter(string topic){
6         this.topic = topic;
        }

        protected override MessageData Create(Type objectType, JObject jobject) {
11         switch (topic) {
            case "/joint_states": {
                return new RosMessages.sensor_msgs.JointState();
            } break;
            :
16         case "/pr2_phantom/collision_object": {
                return new RosMessages.moveit_msgs.CollisionObject();
            } break;
            default: return new RosMessages.std_msgs.EmptyMessage();
21         }
        }
    }
}

```

**Figure B.14:** Message factory.

The `MessageDataConverter` is easily extendable by adding another case to the switch statement (cf. Figure B.14). Here affiliations between topic names and ROS message data types are specified. Specific cases are defined by using the full topic string and the correct return sub-type of `MessageData` is generated by calling the `new` operator at the corresponding class. Population of the empty message object with the actual data is done automatically by the Newtonsoft JSON infrastructure, after calling the `Create` method.

```

1 namespace RosBridge {
    private void DeserializeJSONstring(string message) {
        var rosPublishIncoming = (RosPublish)JsonConvert.DeserializeObject<RosMessage>
6         (message, rosMessageConverter);

        if (rosPublishIncoming.topic.Equals("/pr2_phantom/collision_object")) {
            CollisionObject co = (CollisionObject) rosPublishIncoming.msg;
11         }
        :
    }
}

```

**Figure B.15:** Deserializing a JSON string with the `RosBridgeClient`.

Line 4 in Figure B.15 shows how to invoke the actual de-serialization of an incoming

JSON string. In line 6 and the following the `MessageData` of the resulting ROS message is cast to its actual type and ready for further processing of its content. Extending the `RosBridgeClient` for other types of `MessageData` is done by adding further cases for the explicit type conversion, similar to extending the message factory as described in the previous paragraph.

```
1 namespace RosBridge {  
    private static string CreateRosMessageString(RosMessage msg){  
        return JsonConvert.SerializeObject(msg);  
    }  
6 }  
}
```

**Figure B.16:** Serializing a ROS message to a JSON string with the `RosBridgeClient`.

Other than de-serialization, invoking serialization works without casting for every implemented message with the base class of `RosMessage`. In case of a publish message the containing data with the base class type `MessageData` in the `msg` property of the publish message is serialized automatically.

### B.3.2 Message Exchange via WebSockets

The ROSbridge node at the ROS-side from RobotWebTools project [204] is prepared to send and receive character-based JSON-coded ROS messages. Received JSON strings are de-serialized into the specific ROS message, which must be well known in the actual ROS workspace and then, passed to the addressed ROS node. Outgoing messages are serialized by the ROSbridge node into a JSON string and then transmitted via the registered WebSocket connection to a remote location offering the corresponding WebSocket.

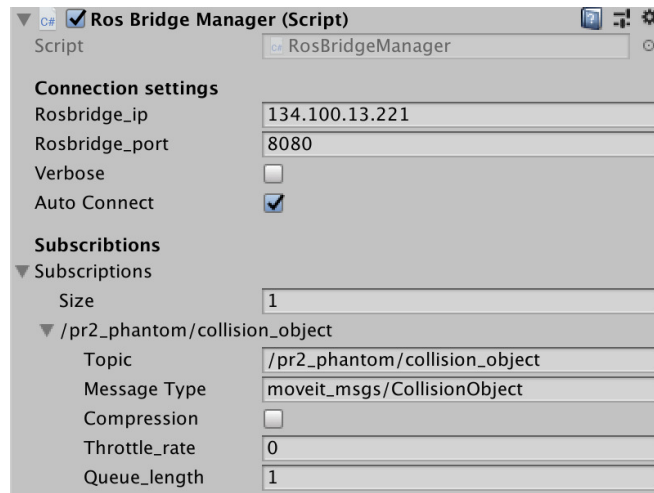
For fulfilling the corresponding task at the Unity3D-side the `RosBridgeClient` was implemented. The class is implemented without dependencies to Unity3D. The embedment in the Unity3D editor is provided by a container class `RosBridgeManager` which offers to set some relevant communication properties.

#### *RosBridgeManager*

By adding a `RosBridgeManager` instance to the actual Unity3D scene important settings for connecting to a running ROSbridge instance at the ROS-side are available via the Unity3D editor. This includes the IP address of the computer running the ROSbridge, the port, which is prepared for the websocket connection, and if a connection should be established automatically. Furthermore, enhanced debug information could be enabled. Even more important is the possibility to define the desired subscriptions to specific “*topics*” with certain “*message types*” and the length of the message queue at the ROS-side together

with the possibility to define a minimum interval between single messages (`throttle_rate`). PNG compression is available for the JSON strings before the ROSbridge at the ROSside is starting the transmission of a message.

The combination of the topic together with the message type must be known by the `RosBridgeClient` (cf. Figure B.15) in advance, as well as the message type implementation (cf. Figure B.11) in the JSON message factory (cf. Figure B.14).



**Figure B.17:** ROSbridge Manager Asset Options.

The `RosBridgeManager` provides access to the singleton of the `RosBridgeClient` via the public method `GetRosBridgeClient()` and starts the connection to the websocket in the `Start()` method, if desired. In `OnApplicationQuit()` additionally started threads are stopped within the `RosBridgeClient` and `OnDestroy()` disconnects the websockets.

### *RosbridgeClient*

The `RosBridgeClient` class in its basic implementation, as well as its direct dependencies, are independent from Unity3D specific types and implementations. It makes use of the `rosJSON` facilities, as well as the `rosMessages` implementations and the `Subscription` structure. Thus, it is prepared to be integrated in .NET C# projects without Unity3D. As the core class for communication handling with ROS ecosystems it provides all necessary functions to establish a connection, exchange and process messages. Figure B.18 lists the methods heads of the implementation. The constructor is called implicitly by the `GetInstance()` method, which implements a “*lazy singleton*” pattern. Additional threads are started and stopped with the corresponding methods. In each of the threads a single method is running. The threaded methods are `Communicate()`, `ProcessIncomingQueue()` and `ProcessOutgoingQueue()`.

Figure B.19 lists the implementation of the `Connect()` function in the `RosBridgeClient`. In line 21 an example shows how to use the method `Connect(string uri)` with a proper uniform resource identifier (URI). Before connecting the websockets, at the Unity3D

```
namespace RosBridge {  
3 public class RosBridgeClient {  
    public static RosBridgeClient GetInstance(string rosmaster_ip, string rosbridge_port,  
        bool verbose, Subscription[] subscriptions);  
8 private RosBridgeClient(string rosmaster_ip, string rosbridge_port, bool verbose,  
    Subscription[] subscriptions);  
  
    // handling communication and threads  
    public void Start();  
    public void Stop();  
13  
    // websocket communication and subscriptions  
    private void Communicate();  
    private void Connect(string uri);  
    private void Disconnect();  
18 public bool IsConnected();  
  
    // processing incoming messages  
    private void ReceiveAndEnqueue(object sender, MessageEventArgs e);  
    private void ProcessIncomingQueue();  
23 private void DeserializeJSONstring(string message);  
  
    // processing outgoing messages  
    public void EnqueueMessageToRos(RosMessage message);  
    private void Send(RosMessage message);  
28 private static string CreateRosMessageString(RosMessage msg);  
    private void ProcessOutgoingQueue();  
  
    private static void MaybeLog(string logstring);  
33 }  
}
```

**Figure B.18:** Methods of the RosBridgeClient.

site a callback function `ReceiveAndEnqueue()` is registered on the event of an incoming message, which puts the message string in a queue for later processing. A similar concept is used for outgoing messages. Whenever the programmer wants to send a message to ROS, the method `EnqueueMessageToRos(RosMessage message)` has to be called to put the desired message in the outgoing queue.

The message processing functions `ProcessIncomingQueue()` and `ProcessOutgoingQueue()` are workers, which consume the buffered messages in separate threads. In this way the Unity3D main thread is unburdened and achieves the necessary framerate for a good MR experience. Processed incoming messages are buffered in separate buffers for each subscribed topic. Thus, the transmitted information from ROS to Unity3D is ready for use in specific applications. A typical application, where buffer size of one is appropriate, is to use incoming sensor readings of joint encoders for mirroring the joint state of an actual robot to a virtual model.

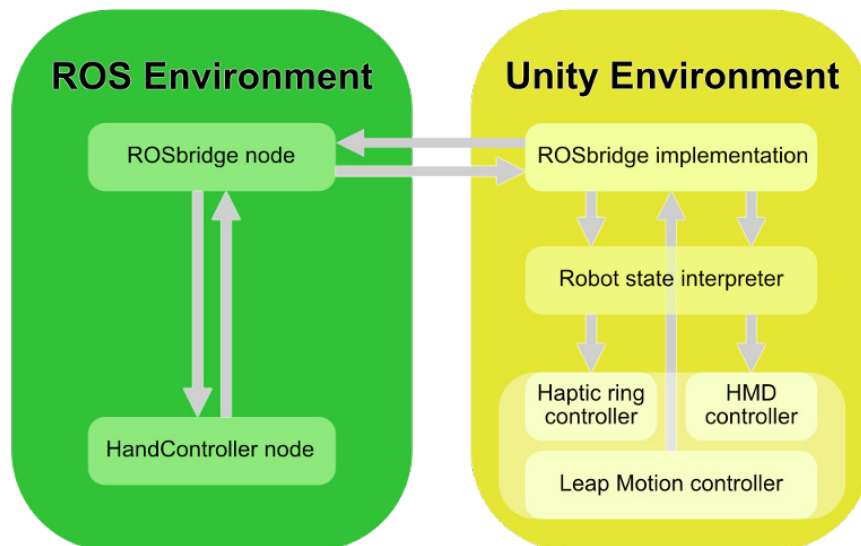
```

namespace RosBridge {
    private static WebSocket rosBridgeWebSocket = null;
5   private void Connect(string uri){
        try {
            rosBridgeWebSocket = new WebSocket(uri);
            rosBridgeWebSocket.OnMessage += ReceiveAndEnqueue;
10
            do {
                rosBridgeWebSocket.Connect();
                Thread.Sleep(1000);
            }
            while(!rosBridgeWebSocket.IsConnected);
15        }
        catch (Exception ex) {
            Debug.Log("Exception:_{Connect}" + ex.ToString());
        }
20    }
    Connect("ws://" + ROSBRIDGE_IP + ":" + ROSBRIDGE_PORT);
}

```

**Figure B.19:** Connecting to the ROSbridge from Unity3D via websockets. Prerequisite is that the ROSbridge node at the ROS-side is started with the websocket option.

### B.3.3 Example: Communication during Robotic Hand Control



**Figure B.20:** Information flow directions during teleoperation of a remotely located robotic hand between different components of the end-to-end system.

In Figure B.20 the information flow directions for the use case of controlling a remote robotic hand with a hand-tracking device in realtime while wearing an HMD are visualized. The robotic hand controller is publishing its current jointstates. Since the ROSbridge implementation at the Unity3D-side has subscribed to the topic these jointstates are published on, the messages are also sent to the ROSbridge node within ROS and then

transmitted via a websocket connection to the Unity-ROSbridge. A class, which is dedicated to processing the content of these messages, the “*robot state interpreter*”, causes correlated visual and haptic feedback. The user takes action depending on the feedback and the actual task by changing his hand posture, which is recognized by the optical hand tracking system, the “*Leap Motion*”. The corresponding controller of the input device generates valuable information for controlling the robotic hand. This information is converted into a specific ROS message and send back to the ROS environment via the websocket connection as a JSON coded string. Unity3D ROSbridge does not address the HandController node explicitly, but instead publishes on a certain topic, to which the HandController has subscribed, previously.

### B.3.4 Modifications for Microsoft HoloLens

If the Microsoft HoloLens is utilized the Unity3D application has to be build for the *universal windows platform* (UWP), which required some modifications in the implementation. During the implementation phase of the setup presented in Chapter 9 libraries like the Newtonsoft JSON library were not available for UWP and instead of *threads*, *emphtasks* had to be used for achieving concurrent processing of the messages. UWP uses a different websocket implementation, as well.

```

namespace RosBridge {
2  :
  #if WINDOWS_UWP
    private Task asyncWorkerCommunication;
    private Task asyncWorkerIncomingProcessing;
    private Task asyncWorkerOutgoingProcessing;
7  #else
    private Thread asyncWorkerCommunication;
    private Thread asyncWorkerIncomingProcessing;
    private Thread asyncWorkerOutgoingProcessing;
  #endif
12 :
    public void Start() {
      :
  #if WINDOWS_UWP
17  asyncWorkerCommunication = Task.Factory.StartNew(() => Communicate());
    asyncWorkerIncomingProcessing = Task.Factory.StartNew(() => ProcessIncomingQueue());
    asyncWorkerOutgoingProcessing = Task.Factory.StartNew(() => ProcessOutgoingQueue());
  #else
22  asyncWorkerCommunication = new Thread(new ThreadStart(Communicate));
    asyncWorkerCommunication.Start();

    asyncWorkerIncomingProcessing = new Thread(new ThreadStart(ProcessIncomingQueue));
    asyncWorkerIncomingProcessing.Start();

27  asyncWorkerOutgoingProcessing = new Thread(new ThreadStart(ProcessOutgoingQueue));
    asyncWorkerOutgoingProcessing.Start();
  #endif
  :
  }
32 }
}

```

**Figure B.21:** Different implementations of the communication for Windows and HoloLens.

### *Concurrent Messaging*

In order to avoid writing a UWP compatible implementation in a separate class, it is possible to use preprocessor directives (pragmas) for telling Unity3D, which lines of code are suitable for which build target. Unity is running per default in “*editor mode*”. Thus, included libraries must be suitable for the compiler running in the editor at the current operating system. Compile errors can be avoided by using pragmas, which explicitly tell Unity3D for which target platform specific lines of codes are suitable. Figure B.21 shows an example for the case of using `System.Threading.Task` instead of `System.Threading.Thread` when developing the application for Windows UWP. The general structure of the `RosBridgeClient` stays the same. Regarding the performance of the Microsoft HoloLens it is extraordinary necessary to save resources and computing time, since it provides only an Intel Atom dual core processor. Thus, only 4 concurrent tasks can run at the same time. In the experiment the proposed concept performed in an appropriate manner and supported a very realistic experience.

### *Websocket Connection*

Regarding websocket implementation the issue is very same. The standard `RosBridgeClient` implementation uses “*websocket-sharp*<sup>3</sup>”, due to its frequent use in Unity3D applications, which is not available for UWP. For this case the standard implementation in `Windows.Networking.Sockets` is utilized.

## **B.4 Feedback**

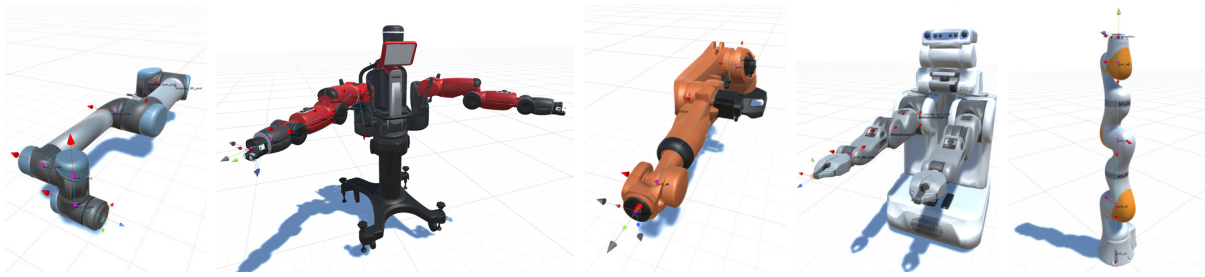
The presented approach focuses on delivering mostly visual sensations by utilizing stereoscopic rendering. Haptic feedback is provided only in a very simple matter and audio feedback is neglected in the presented implementations, but might be of interest in further improvements of the presented concepts.

### **B.4.1 Visualization**

One of the main reasons to ROS and Unity3D was the effective and efficient implementation of stereoscopic rendering for all available state-of-the-art devices and the potential to integrate future hardware releases in the field of MR. In order to provide the user with the desired depth-cues and spatial awareness, precise 3D models of the robot and its environment are of importance.

---

<sup>3</sup><https://github.com/sta/websocket-sharp>



**Figure B.22:** Arbitrary robot models can be imported using the importer script. It parses ROS URDF files and generates the robots in Unity3D according to the description file. (Figures are generated from Sebastian Starke with the BioIK and URDF importer Unity Asset)

## Robot and Environment Models

The existence of robot models in the virtual scene is crucial for HRI methods proposed in this dissertation. For efficient prototyping we need means for importing already existing robot description files as well as their environment. The potential to import the whole planning scene of the robotic system allows for fast prototyping of arbitrary interaction methods and avoids the probability of mistakes compared to manual composition of the scene. Relative transforms between two different objects or parts of objects in the scene stay precisely in the correct position.

### *URDF importer*

A student co-worker developed an importer for robot description files including their optional environment descriptions.<sup>4</sup>

The robot model importer is intended to be used from the Unity3D editor and allows to import arbitrary ROS robot models used in research and industry (cf. Figure B.22). ROS uses “*unified robot description format*” (URDF), a XML-based text format, to define visual and physical robot models including their locomotion limits and physical properties. These properties are usually joint specifications regarding the active joint limits, forces or maximum velocities. Robotic components, such as arms and hands, are defined in separate modular URDF files. Complex robots are created by combining these components. The importer is capable of parsing composed URDF files and constructing complex hierarchical structures of robot models. All necessary files are usually available in the git repositories of the robot manufacturers or the ROS community. The importer is capable of processing `.dae`, `.obj` or `.fbx` 3D meshes, which are typically used; otherwise the meshes must first be converted manually into a proper format. If the BioIK asset is installed, the imported robot has a `KinematicJoint` script attached to every joint in order to represent, access and control its motion as specified in the URDF. Otherwise another joint-script has to be attached, if comfortable manipulation of joint motion is desired. Access to these

---

<sup>4</sup><https://assetstore.unity.com/packages/tools/modeling/urdf-importer-99316>



joint scripts is granted during the runtime of the program. For inverse kinematics, we additionally integrated an evolutionary inverse kinematics solver [190], which integrates well with the joints generated by the importer.

## Robot Motion

Robot motion in Unity3D is visualized by actuating the virtual joints of the corresponding robot model. Unity3D features different `Joint` classes but these rely on physics simulation and are intended to visualize reaction forces and realistic behavior of chained structures. In robotics we also need position control, even if velocity and effort are also typical modalities to describe joint states. Position control, which means setting a joint to an angular position relative to a well defined standard position, is typically used to describe the current posture of a robot. Tuples of a position and a timestamp are suitable to define dynamic trajectories of robot motion.

For robotic applications in UI development and user studies robotic motion e.g. caused by the following reasons:

- The actual robot state is mirrored in the UI during use.
- Caused by user input trajectories are planned by the robot and the UI plays back the planned trajectory or shows the desired goal posture.
- The user directly manipulates the virtual model locally by appropriate means of interaction and has the option to extract angles for commanding the actual robot, which might be co-located or remote.
- For debugging purpose the UI visualizes postures and movements as a result from learning during runtime or after learning has finished.

### *Joint Motion Script*

The implementation of a basic rotational joint with position control around a single axis is listed in Figure B.23. It was utilized to actuate the virtual robotic hand according to its actual state in realtime as described in the scenario in Section B.2.2.

The script uses the standard Unity3D coordinate conventions. It features to set rotations in degrees around a single axis beginning from a defined starting position. Additionally, a static offset is accessible via the Unity3D editor. The method `SetAngle(double angle)` is prepared to set a target position. According to the configuration of the joint within the Unity3D editor, the rotation around the corresponding axis is calculated and in the `Update()` method this value is applied to the `GameObject` the script is attached to. This means for the Unity3D programmer that the pivot point of the actual game object has to be placed correctly in advance, e.g. by using additional game objects containing the

actual mesh or primitive, which has to be rotated. In this way object coordinate systems are altered without changing the actual model.

```

using UnityEngine;
2
namespace RosTools{

    public enum Axis {X_AXIS, Y_AXIS, Z_AXIS};

7
    public class RotationalJoint : MonoBehaviour {
        [SerializeField] private Axis axis = Axis.X_AXIS;
        [SerializeField] private bool clockwise = true;
        [SerializeField] private double offset = 0.0;

12
        private Quaternion currentRotation;
        private Quaternion startRotation;
        private float targetAngle;

        void Start() {
17
            startRotation = transform.localRotation;
            currentRotation = startRotation;
        }

        void Update() {
22
            transform.localRotation = currentRotation;
        }

        public void SetAngle(double angle) {
27
            int sign = clockwise ? 1 : -1;
            targetAngle = (float)(sign * (angle + offset));

            switch (axis)
            {
                case Axis.X_AXIS:
32
                    currentRot = startRot * Quaternion.Euler(targetAngle, .0f, .0f);
                    break;
                case Axis.Y_AXIS:
                    currentRot = startRot * Quaternion.Euler(.0f, targetAngle, .0f);
                    break;
37
                case Axis.Z_AXIS:
                    currentRot = startRot * Quaternion.Euler(.0f, .0f, targetAngle);
                    break;
            }
        }
42
        public float GetAngle() { return targetAngle; }
    }
}

```

**Figure B.23:** A simple rotational joint script with position control.

### *Motion Models*

Underactuated devices like the Robotiq Adaptive 3-Finger Gripper (cf. Figure 3.5(d)) have less actuators than DoF. In case of the gripper adaptivity is achieved by special mechanical structures which cause strong grasps of arbitrarily shaped objects. A single finger is actuated by at maximum one motor, but has three joints, which are effected by this single motor. The mechanical design moves a segment closest to the palm at first until it is physically blocked by either itself or an object in the environment. Then, the next segment is actuated by the same motor until it is blocked in the same way.

Implementations of this motion model is possible by using the previously defined joint

scripts. Checking the current angles in a dedicated hand controller script allows for implementing realistic motion of the hardware. For the scenario in Section B.2.2 a script `ActuateGripper` was implemented, which offered the desired “*empty-hand*” motion behavior of the gripper. Similar scripts are appropriate as robot controllers for virtual robots, especially if robot state mirroring is desirable or interaction studies with only simulated robots and very low latency are of interest.

### *Local IK/BioIK*

Most of the scenarios in this dissertation utilize an implementation of an inverse kinematics controller which runs in Unity3D. The benefits of locally calculated inverse kinematics is the avoidance of the communication overhead with ROS where updates are necessary very frequently. Kinematics of robotic manipulators are complex and sometimes not easy to understand for humans. There is evidence that interaction studies involving instant IK motion help to make robots more predictable. This results from qualitative feedback of a pre-study with different visualizations options during teleoperation (cf. Section 7.2.2). E.g. the implementation of the visual virtual fixtures study (cf. Section 8.1) utilizes “*Bio IK*”<sup>5</sup> from the Unity3D asset store. A former colleague started to work on this algorithm during his master studies. Combined with the URDF importer it automatically attaches some advanced joint script to the corresponding joint game objects during import and thus works well for importing ready-to-work robot models, which are actuated by the attached joint scripts. Joints are named exactly like in the URDF file which makes it easy to write a controller for the robot for setting up a virtual robot for interaction in MR. Even though Bio IK is computationally expensive, it runs sufficiently on the Microsoft HoloLens’s limited hardware.

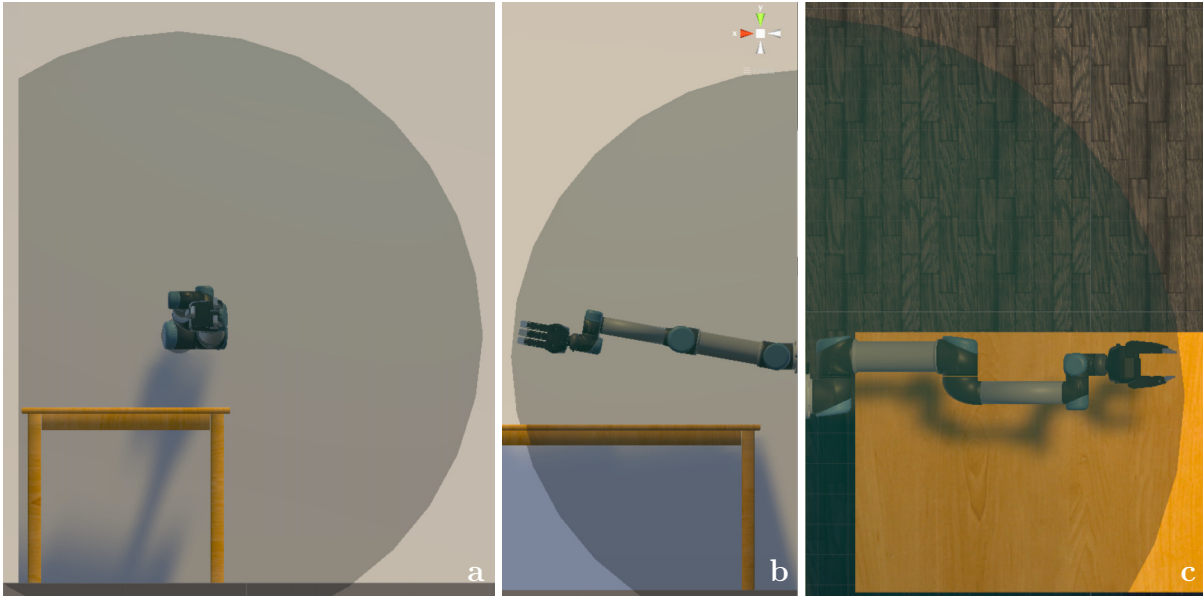
### **Visual Virtual Fixtures**

The implementation of visual virtual fixtures, described in Section 8.1, is based on the early works of Rosenberg [168, 169, 170] who applied similar concepts like the “*usage of a ruler for assistance in drawing straight lines*” to an AR robotics teleoperation setup as physical aids to the operator. One of the general concepts in the approach presented in the implementation of the work presented in Section 8.1 was to calculate the shortest distance between the grasped object and the surface of surrounding obstacles in order to provide visual aids to the operator. The distance  $d$  between two points  $x_1$  and  $x_2$  in 3D space is calculated with *Euclidean Distance* formula:

$$d = \left( (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 \right)^{\frac{1}{2}} \quad (\text{B.1})$$

---

<sup>5</sup><https://assetstore.unity.com/packages/tools/animation/bio-ik-67819>



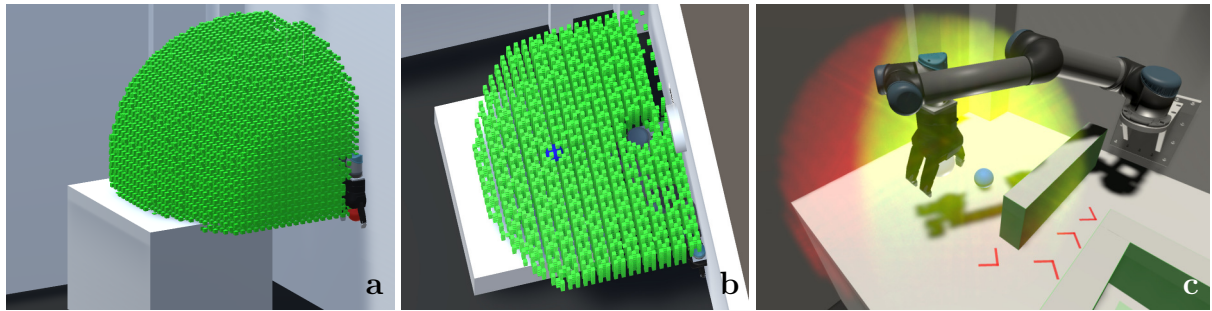
**Figure B.24:** Maximum reachability of robotic manipulator. (a) Front view, (b) side view, (c) top view.

But in case of two meshes all points of the surface have to be checked pairwise. For the sake of efficiency the implementation uses a simplified case where the grasped object – the moving object – is a sphere. On a sphere every point of its surface is equally far away from the center; each point of the surface has the distance to the center of the size of the radius. Thus Unity3D’s method `Collider.ClosestPoint(Vector3 position)` is applicable which performs efficient distance calculation using its physics engine. Unity3D distance function `Vector3.Distance(Vector3 p1, Vector3 p2)` then estimates the distance between the grasped sphere in the experiment and all obstacle meshes having a `CheckDistance` script attached. Using this distance value, another script named `VisualFeedback_fixture` fades the color of the obstacle depending on the actual distance between green and red; where “*green*” indicates safe distances and “*red*” marks critical distances or collisions. The color of the whole object is affected if higher granularity needs to be achieved than the highlighting should be implemented on shader-level.

### Reachability Visualization

Another kind of visual assistance is generated by rendering the reachability of the robot. The most naïve approach is to visualize the maximum reach of the arm as a semi-transparent sphere around the base-link of the robot (cf. Figure B.24).

But in this case there is no information whether a solution for an actual pose, given to the IK solver, exists. Due to the multitude of possible solutions for a single 6 DoF target pose, it is almost impossible to find an answer in realtime. Thus, pre-analysis of the workspace and the reachability provides useful information during continuous input tasks for the operator. Tasks like physical servoing, manual definition of intermediate poses during



**Figure B.25:** IK analysis and visualization of the resulting solution space. (a) Existing solutions are represented by a quadratic billboard without any quality rating. (b) A hole above the baselink of the robot is clearly recognizable. (c) Shows a local part of the solution with the spatial extent of a sphere around the TCP of the gripper. Red color indicates unreachable regions. Best solutions are displayed in green, lesser solutions in yellow. Unity3D's particle system is used to create efficient billboard particles with a diffuse circular texture. Thus, volumetric rendering is simulated and appropriate for MR experiences.

trajectory programming, or input for robot or human training are relevant applications. ROS offers a package for workspace analysis and visualization.<sup>6</sup>

In Unity3D there are at least two options, one is to access an external dataset about IK reachability, another is to generate a dataset within Unity3D. For iterative analysis of the workspace of the robot the implementation of an octree algorithm is appropriate.

### *Octree Algorithm*

In the most simple case a cube in space, which represents the whole desired workspace, is iteratively subdivided into cubes of half-sized dimensions. The center of a single cube represents the target solution. In the first round there is only one possible solution in the center of the initial volume. In the second round there are eight more and in the third round for each of the solutions of the second round eight more are generated, resulting in an iterative multiplication by 8 for each next round.

Figure B.25(a) shows valid IK solutions for top-grasps of the horizontally wall-mounted UR-5 with a Robotiq Adaptive 3-Finger Gripper attached within a predefined volume. As shown in Figure B.25(b), there are holes in the structure above the base-link of the robot. These are positions, which cannot be reached with a top-grasp.

### *KD-tree Search*

Interaction with a kinematically constrained robotic manipulator can be assisted by calculating the local reachability around the current position with regards to the actual orientation of the end-effector. For realtime applications a solution with very low computing time is necessary. In case of a precomputed solution space it still needs to be filtered

<sup>6</sup><http://wiki.ros.org/reuleaux>

to fulfill certain criteria like a maximum distance to the actual position. On GitHub there is a “*KD-tree*” implementation which is suitable for efficient search in n-dimensional solutions.<sup>7</sup> After defining a distance function the tree is constructed and ordered.

```
namespace RosTools{
    L3Dist = (a, b) =>
    {
        double dist = 0;
        dist += (a[0] - b[0]) * (a[0] - b[0]);
        dist += (a[1] - b[1]) * (a[1] - b[1]);
        dist += (a[2] - b[2]) * (a[2] - b[2]);

        return dist;
    };
}
```

**Figure B.26:** Distance function for KD-tree.

The construction of the search tree (cf. Figure B.27) has the higher computational cost in comparison to search operations (cf. Figure B.28).

```
KDTree<double,string> tree = new KDTree<double, string>
    (dimensions: 3, points: treePoints, nodes: treeNodes, metric: L3Dist);
```

**Figure B.27:** The construction of the KD-tree takes longer than the search within. So, generation of trees cannot be done during a single frame rendered in Unity3D.

The radial search is performed around a given value, which is represented by the end-effector pose altered by the operator in realtime during interaction with the system. Coordinates are in the reference frame according to the dataset. Radius is defined by a numerical value in the range of possible output by the previously defined distance function.

```
radialSearchResult = tree.RadialSearch(center: Vec3ToArray(invPos), radius: maxDistance);
```

**Figure B.28:** Radial search within the tree is implemented in a single line of code. A KD-tree with  $K = 3$  results in a spherical solution space.

### *Particle System*

Unity3D’s particle system represents an effective and efficient approach to render a robot’s local motion capabilities during use. Figure B.25(c) shows the resulting view from the user perspective in combination with the visual virtual fixtures. The reduced opacity still allows for observing all necessary aspects in the environment. The approach of particle rendering in combination with KD-tree search in large datasets still allows for rendering with FPS above 90, which is necessary for good VR experiences.

<sup>7</sup><https://github.com/MathFerret1013/Supercluster.KDTree>

If the quality of the solutions is mapped to a numerical value, HSV color space is appropriate to render the solution quality. If normalized solution qualities to the range of [0,1] are mapped to the first third of the normalized hue value the resulting color is red for values close to zero and green for values close to 1, and yellow for intermediate values. Passing the result of the radial search in the KD-tree (cf. Figure B.28) to a drawing function (cf. Figure B.29) results in Figure B.25(c).

```

namespace RosTools{
    void DrawParticles(Tuple<double[], string>[] solution) {
4         var ps = GetComponent<ParticleSystem>();
            Vector3 position = new Vector3 ();

            for (int i = 0; i < solution.Length; i++) {
                position.x = (float)solution[i].Item1 [0];
9                 position.y = (float)solution[i].Item1 [1];
                position.z = (float)solution[i].Item1 [2];
                particles[i].position = position;

                particles[i].startSize = startSize;
14
                float col = (float)solution[i].Item1 [3];
                currentColor = Color.HSVToRGB(col / 3.0f, 1.0f, 1.0f);
                currentColor.a = alphaValue;
19                 particles[i].startColor = currentColor;
            }

            ps.SetParticles(particles, particles.Length);
        }
24 }

```

**Figure B.29:** Drawing the KD-tree with a Unity3D particle system.

## HUD

Providing the operator with a more transparency to the system behavior is beneficial for system usability and improves the user performance. Further it can increase the acceptance and performance of the overall system [61].

A head-up display is an option to show important information to the user at any time and independent from the operators current view. Permanent information regarding the overall system state could be placed in the peripheral view and urgent messages like warnings and errors could appear in the foveal area or in the center of the view. The implementation of the HUD from the pick-and-place study, presented in Section 9, shows the current system state from the robot’s view, permanently during runtime, in the upper border of the view and failures, after triggering motion planning tasks, are displayed in the center only for short time. The displayed system state is internally represented by the state-machine, described in Section B.6.3. The HUD is implemented by placing a canvas with a text object displaying the system state. The canvas is fixed in the camera frame and moves with the operators view during operation. Important warnings are displayed in a similar way, but with a similar implementation as “*toast messages*” in Android. They



**Figure B.30:** Almost every input or feedback device can be integrated in own setups by writing own controller scripts based on the manufacturer’s SDK. E. g. Geomagic Touch (a), HapGlove [11] (b), HMD-mounted Leap Motion Controller (c).

appear in the center of the user’s view and fade out after some seconds. Generally, valid or desired states and information are displayed in green, while failures and warnings are rendered with red font color.

### B.4.2 Haptic Rendering

The experiment presented in Section 8.1 pointed out that haptic feedback from Vive Controllers is associated with collisions. Thus, first ideas of using vibration feedback with intensities coupled to the distance between the robot and obstacles in the environment were neglected and simple dual-state implementation was used instead. So, a vibrating controller indicates a collision at operating time with the environment. The implementation uses the physics engine in Unity3D and triggers constant haptic feedback during collisions. By adding colliders to the robot, the potentially grasped object and the obstacles, haptic feedback is caused automatically. To achieve this effect the gripper and the graspable have to implement the `MonoBehavior.OnCollisionStay(Collision)` method. Within the implementation of this method the vibration has to be started. Similar implementation are applicable to other devices like other game controllers, haptic gloves (cf. Figure B.30(b),[11]) or rings [10] or devices like the Phantom Omni (cf. Figure B.30(a)).

## B.5 From Input to Robot Control

Unity features the integration of various input, feedback or display devices. Every state-of-the-art HMD or VR and AR device is typically supported by Unity running on Windows. Indeed, most research in robotics is done on Linux systems, which makes it unavoidable to bridge two different subsystems like proposed. Even some prototypes or developer editions, like the Microsoft HoloLens can be integrated efficiently using this approach (cf. Section 9). Also haptic devices or custom-designed prototypes are available for implementing user interfaces for robots (cf. Figure B.30).

Using Unity3D provides interaction designers with low-cost tracking systems. 6-DoF



tracking of typical HMDs, but also skeleton tracking, as provided by the Leap Motion Controller, are effective mechanism for implementing intuitive MR-RUIs. Exampels for the use of tracking devices were implemented and partially tested on an actual robot:

1. HMD 6-DoF tracking during operator’s walking is applied to movements of an actual mobile robot (cf. Figure 7.5).
2. Altering the operator’s viewing direction is applied to turning the head of a robot (cf. Figure 7.1(d)).
3. In the operator’s viewing direction a ray is cast for extracting target positions for the actual robot (cf. Figure 9.2).
4. Tracked HTC Vive Controllers or hands, tracked by the LMC, are utilized to move the end-effector of a manipulator in realtime with the help of an IK solver running in Unity3D (cf. Figure 7.1(b)).
5. Tracked HTC Vive Controllers are utilized to grab links of a robot in order to physically change the posture of the virtual robot (cf. Figure 7.2).
6. HTC Vive Pucks (Trackers) are used for tracking intermediate positions – the elbow and the shoulder – of a human arm in order to achieve human-like trajectories (cf. Figure B.31(a)).
7. Altering the body posture of a humanoid robot by tracking head pose and poses of both hands (cf. Figure 7.3)
8. Hands, tracked by the LMC, are utilized to define poses for the end-effector of a manipulator via gestures (cf. Figure 7.1(a) and Figure B.31(b-d)).
9. Hand skeleton tracking of the LMC is utilized to measure the distance between thumb and index finger. The normalized value is applied to the closing state of an actual robotic gripper (cf. Figure B.2(c)).

Despite the different kinds of hardware utilizations, the implementations differ especially in the following main aspects:

- “*frequency*”: continuous or single input
- “*directness*”: input defined by pose of a tracked entity or by a gesture
- “*equippedness*”: like empty hand or holding a device

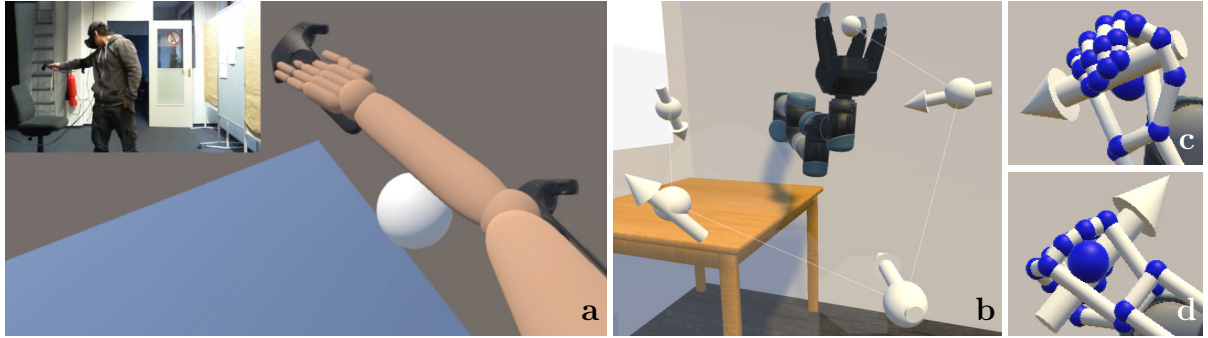
### B.5.1 HMD-Tracking

Tracking systems, which are part of common HMDs like the HTC Vive or the Rift CV are well prepared to explore VEs in a very natural way by moving through them like in the real world. An increasing amount of devices are nowadays provided with “*Inside-Out Tracking*”, which do not require external tracking devices, but instead everything is integrated in the HMD and the controllers. Some of them, like the Microsoft HoloLens are also tetherless. Wearing an HMD for exploring VE is very natural for the user, requires no additional implementation in Unity3D and causes only very low mental workload. In robotics applications this is very useful for different purposes.

As described in Section 7.2.7 walking through the tracking space produces suitable information for moving a mobile robot. In combination with a 360° camera rotations of the operator’s head adjusts the actual view while translations cause a movement of the robot’s base. Control signals for the robot are continuously send via the ROSbridge interfaces implementations in ROS and Unity3D. Depending on the actual goals, variable speeds and directions at the operator side are transduced during use of the system into ROS messages with a certain update rate. The robot executes the incoming commands directly by adjusting appropriate turn rates of the wheels. The feedback loop is closed by the operator who is watching to a 360 video stream and links the walking actions with the video images.

As shown in Figure 7.1(d) tracked controllers of the actual HMD might then be implemented to represent the robotic hands. In combination with an inverse kinematics solver very versatile systems for teleoperation with a high level of telepresence arise. The additional effort for implementing the processing of tracking data is very low, when a one-to-one mapping is desired. Either, the IK are solved at the Unity3D-side and then jointstates are send to the actual robot for changing its configuration as ROS messages via the ROSbridge, or the 6 DoF transforms of the tracked human are send directly.

Additional trackers like the HTC Vive Pucks help to handle the issue of probable DoF mismatch between a human and a robot by adding spatial information from the human’s shoulder and wrist joints. IK solvers like Bio IK support multiple targets. Thus, intermediate positions like an elbow are easy to integrate in to the posture of the robotic arm. Figure B.31(a) shows an implementation with two Vive Controllers for tracking one arm very naturally. No IK algorithm is needed to generate realistic motion of the forearm relative to the upper arm. Despite this would be a limitation for robot’s kinematic capabilities, it is very interesting for generating human-like behavior. Solutions using Bio IK simply need additional target scripts at the desired links of the robot. An additional reference to the tracked entity causes the IK algorithm to take this intermediate pose into account.



**Figure B.31:** Different applications of tracking devices are shown. Realistic arm tracking is achieved by using multiple trackers (a). A trajectory for a robotic manipulator is specified by hand gestures, recognized with an LMC (b). Directions are specified by the “*holding-and-arrow metaphor*”.

## B.5.2 Hand Recognition

One issue of using game controllers or similar handheld devices for tracking hand poses is to lose the utility of the hands as tools. Only empty hands can grasp or manipulate objects. Thus, empty-hands solutions are preferable over the reliable tracking and comfortable implementation of controller use.

One of these empty-hands solutions is shown in Figure B.31(b-d). A system for trajectory generation is implemented using an HMD with front-mounted LMC (cf. Figure B.30(c)). *Tap gestures* create waypoints in the VE, representing the environment of the robot. Waypoints are defined as vectors with the three components and are represented as spheres. A *first gesture* works like holding an arrow, as shown in Figure B.31(c-d), and defines the closing direction of the robot. This two-stage approach works for defining waypoints for manipulator trajectory generation. After extracting 6-DoF poses from user input, these are packed into a ROS message and transferred to a corresponding ROS node, which forwards the given information to generate a trajectory that can be executed on the actual robot.

Another benefit of empty-handed solutions is to use grasping gestures for commanding or training a system. Figure B.2 illustrates the teleoperation of a gripper by a pinch gesture. The gripper closing state is normalized to values in the range of (0 1) and the pinch gestures as well. Additional exponential smoothing and dead-band filtering causes very robust input means for actuating the gripper.

Figure 7.1(b) presents a teleoperation system, which works in a similar way, but with full hand open/close gestures. The operator grasps virtual objects with the controlled virtual gripper and performs the task of putting them into a box. The idea was to command the robot implicitly by scheduling tasks similar to the human actions.

Figure 7.1(a) implements a natural grasp trainer with a learning-by-demonstration-based MR-RUI. The operator performs grasping gestures from a desired direction at a certain region of the object for instructing the robot on different objects.

### B.5.3 HoloLens and the Mixed Reality Toolkit (MRTK)

The first version of the Microsoft HoloLens offers many great features like inside-out tracking, stereoscopic-rendering on a see-through display, basic hand tracking, voice command recognition, and many more. The possibility to operate within the actual robot's environment in a tetherless setup and to enrich the real environment with stereoscopically rendered virtual objects leads to a new potential in MR-RUI design and implementation. Access to many features, based on the integrated components is provided by the “*Mixed Reality Toolkit*” (MRTK), previously known as the “*Windows Holographic Toolkit*”.

Section 9 compares to different methods for specification of grasp poses. Both are based on raycasting. The head-only mode uses the forward direction of the headset for sending the ray out. The physics engine extracts the desired positions for the robot by colliding with objects of the VE, which correspond to recognized objects in the real world.

The second method generates the ray in another way by using the head position as the origin. The desired direction of the ray is computed by extracting a direction vector from head position to the approximated fingertip position of the operator's index finger. Both methods can be as precise as the modeling precision of the virtual world. Once the grasp poses are calculated from user input, the information is wrapped into a ROS message and send to the corresponding ROS node, which takes over and calculates a grasping trajectory for the robot to grasp the selected object in the desired way.

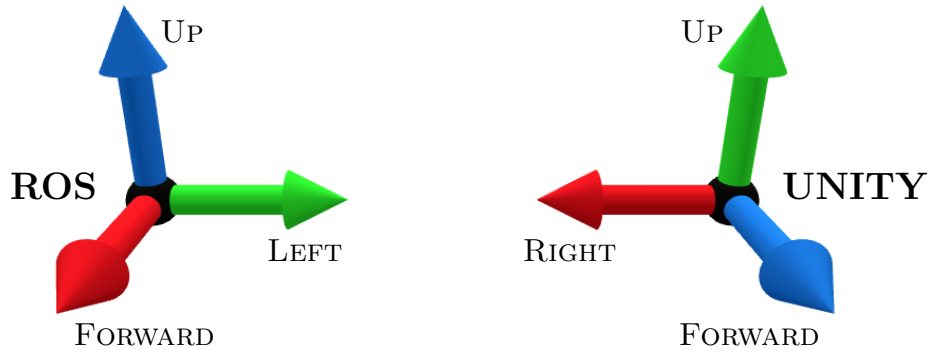
## B.6 Other Tools

Several Unity3D scripts were necessary to implement different scenarios. Tools like coordinate conversion between ROS and Unity3D was essential for interpreting information of the current robot state and for assigning spatial targets to the robot (cf. Section B.6.1). Especially when mobile headsets with inside out tracking like the Microsoft HoloLens are utilized to merge real and virtual worlds by superimposing the actual robot, effective registration methods are needed (cf. Section B.6.2). For improving results based on a simple marker, an adaptive multi-marker tracking were implemented (cf. Section B.6.2). A state machine keeps track of the current state of the MR-RUI, helping the system only to show necessary information, which reduces the need of the user to change operation mode, since this is done implicitly by switching the current state (cf. Section B.6.3). Additionally, it was used to inform the user about the current state.

### B.6.1 Coordinate Conversion

Coordinate conversion are of importance if locations and orientations are communicated between Unity3D and ROS or descriptions files are exchanged between the two systems.

Importing robot descriptions or environment specifications in Unity3D necessarily cause such a conversion but also the usage of input methods at the Unity3D-side for setting targets for a robotic manipulator.



**Figure B.32:** Coordinate frame conventions of ROS and Unity3D. Colors mark axes: x-axis – red, y-axis – green, z-axis – blue. The coordinate frames in ROS are left-handed, while Unity3D coordinate frames are right-handed.

The Cartesian coordinates in Unity3D and ROS follow different conventions (cf. Figure B.32). While Unity3D defines transformations according to the left hand rule, ROS implements right handedness. Conversion between the two systems works according to equations (B.2) and (B.3). Definitions and conventions of units in ROS are available on the web<sup>8</sup>. Both systems implement a vector with three components for defining a position in Cartesian space. The assignment of the properties “*x*, *y* and *z*” to the meaningful axes like “*forward and up direction*” differ. Forward direction in Unity3D is represented by the “*z-component*” while ROS uses “*x*”. The up direction in Unity3D is “*y*” and in ROS it is “*z*”. Due to handedness, Unity3D utilizes “*x*” for right direction and ROS “*y*” for *left*.

$$Vec3_{unity} = new Vec3(-Vec3_{ros}.y, Vec3_{ros}.z, Vec3_{ros}.x) \quad (B.2)$$

$$Vec3_{ros} = new Vec3(Vec3_{unity}.z, -Vec3_{unity}.x, Vec3_{unity}.y) \quad (B.3)$$

## B.6.2 Registration

In robotics coordinate registration is defined as the process to obtain an accurate conversion between different coordinate frames. In case of a mobile robot it is of special interest, where it is located within the world frame and in case of a mobile headset it is necessary to know in which pose it is located in the world frame during use. Regarding the ROS-side, mobile robots typically apply SLAM algorithms in order to localize themselves on world-referenced map. Stationary robotic arms can be either mounted very precisely at a

<sup>8</sup><http://www.ros.org/reps/rep-0103.html>

known position, or even better, are automatically registered using markers at reference positions.

In the special case of using a mobile HMD with inside-out tracking, like the Microsoft HoloLens, the device uses SLAM to localize itself on a self-generated 3D map. Bringing the world of the mobile HMD and the robotic world together requires an additional registration process (cf. Figure 9.3).

In the use case described in Section 9, the “*vuforia engine*” was utilized for this additional registration process. By modeling the marker of the real world at the correct pose into the virtual world in Unity3D, the marker position is utilizable as a world anchor and the virtual model is moved automatically to the correct pose by detecting the marker in the real world. For improving the precision and reliability of the detection success multi-marker detection was implemented in Unity3D.

### *Multi-Marker Detection and Fusion*

The basic idea is to use multiple known positions and to calculate a mean transform in a common reference frame. Only detected markers used for calculation of the corrective transform, which moves the virtual world to the correct pose. The corrective transform was calculated by two separate steps. One is to compute the mean position and the other is to generate a mean rotation.

Regarding the mean position, all marker position vectors are transformed to the reference frame of the virtual world anchor and then averaged by summing up component-wise and then dividing by the number of successfully detected markers.

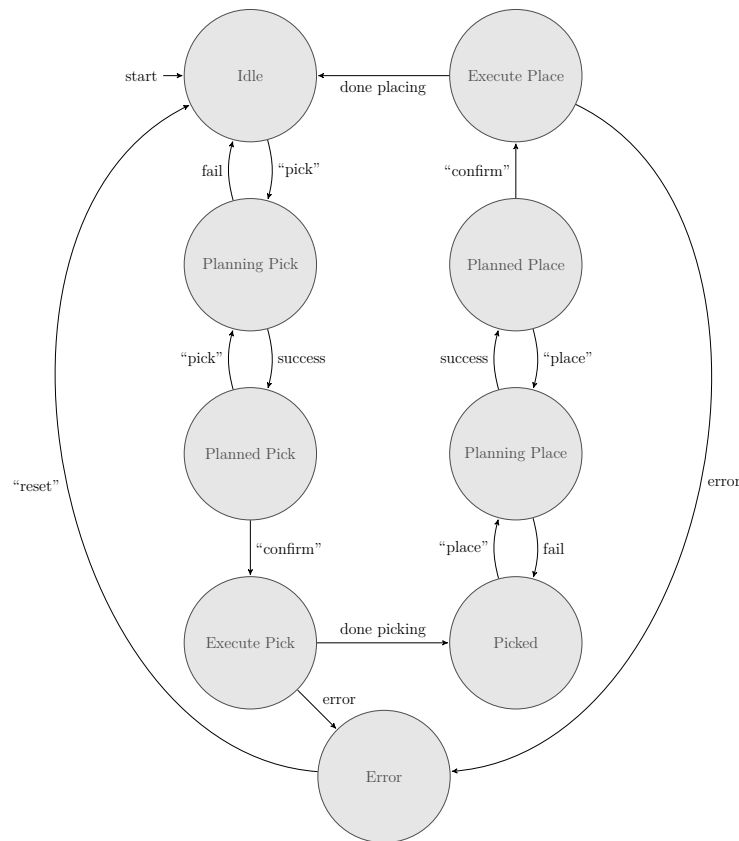
The average rotation is computed by keeping track of the forward and up directions of the recognized markers. These are `Vector3` as well as in the case of positions. Thus, averaging works in the same way. Finally, the correcting rotation is generated calling the Unity3D method `Quaternion.LookRotation(forward,up)` with the averaged and normalized forward and up directions.

### **B.6.3 System State Machine**

The state machine of the system is implemented as suggested by Nystrom [147]. The author explains a practical implementation of states for Unity3D applications in the web.<sup>9</sup> The utility of state modeling in Unity3D for MR-RUIs is very obvious. Different states require different means of interaction and also different information sources are of interest depending on the actual state. Thus, gameobjects have to be enabled or disabled and the set of available input possibilities will change dynamically. Additionally commands to the ROS subsystem are sent if necessary. UI state transitions are caused by either the

---

<sup>9</sup><http://www.gameprogrammingpatterns.com/state.html>



**Figure B.33:** Simplified state machine for a pick-and-place user interface.

user’s input or by the robot’s internal state, which is communicated as an incoming ROS message. Figure B.33 presents the state machine of the pick-and-place MR-RUI described in Section 9.

## State

```

1 namespace RosTools{
  namespace StateMachine{

    using UnityEngine;

6   public abstract class State : MonoBehaviour {
      public State[] nextStates;
      public abstract void SetNext(bool val);
      public abstract State HandleInput(StateController sc);
      public abstract void UpdateState(StateController sc);
11  }
  }
}

```

**Figure B.34:** Abstract class State.

Specific states should implement the abstract class `State` (cf. Figure B.34). In the Unity3D editor reachable states have to be specified in the `nextStates` property. `SetNext(true)`

is called to manually trigger a probable state transition from outside of the state instance. `HandleInput()` is used to process internal triggers which might result in an automatic state transition. `UpdateState()` is the manual pendant of each gameobject's update method. For making sure that update is called by the state controller only on the active state after inputs are handled, this behavior differs from typical gameobjects.

### State Controller

The `StateController` is a gameobject and holds an instance of the current and the previous state. Both states are accessible via getters but only the previous state can be set directly. On `Update()` the two methods `HandleInput()` and `UpdateState()` are called at the current state instance. The control to move on to the next state lies in the state itself, which processes probable triggers to move on to the next state or just executes its task as before and stays active. Since during robot actuation a malfunction could happen or at any time the operator wishes to go back to the initial state, the state controller keeps a reference to the `ErrorState`, which offers a reset method. The state controller has a reference to the current state but also remembers the previous state. This is useful in situations where the operator has the option to improve the results during operation by issuing a command again when a valid has already been given in advance. In Figure B.33 “*Planning Pick*” is reachable by calling “*pick*” from “*Idle*” state or “*Planned Pick*” state. The difference is that when issuing from “*Planned Pick*” state the command fails, there is still the previously planned pick available and the next state will be “*Planned Pick*” again.



## LIST OF FIGURES

1.1	Robotic system with MR user interface. . . . .	5
2.1	Coherence map of the thesis. . . . .	10
3.1	Robot family tree. . . . .	16
3.2	A collection of old visions of robots. . . . .	21
3.3	The first robots. . . . .	23
3.4	A collection of interactive robots. . . . .	24
3.5	A collection of robots suitable for manipulation tasks. . . . .	26
3.6	A possible classification of robots. . . . .	29
3.7	Open loop control. . . . .	30
3.8	Feedback control loop. . . . .	30
3.9	Feedforward control. . . . .	31
3.10	Adaptive control loop. . . . .	31
3.11	Hierarchical paradigm for robot control. . . . .	32
3.12	Reactive paradigm for robot control. . . . .	32
3.13	Hybrid paradigm for robot control. . . . .	32
3.14	A collection of robot control hardware. . . . .	34
4.1	Simplified representation of the reality-virtuality (RV) continuum. . . . .	38
4.2	Technologically mediated experience. . . . .	43
4.3	AIP cube. . . . .	45

4.4	6-DOF-Input-Taxonomy. . . . .	45
4.5	Parameter-Manipulation-Type-Continuum. . . . .	46
4.6	Action-Type-Continuum. . . . .	46
4.7	Degrees-of-Freedom-Continuum. . . . .	46
4.8	IVE-Interaction-Taxonomy. . . . .	47
5.1	Number of publications about mixed reality and robots. . . . .	51
5.2	Remote robot operation. . . . .	52
5.3	Co-located robot operation. . . . .	53
5.4	Virtual reality robot operation. . . . .	53
5.5	Mixed reality robot operation. . . . .	54
5.6	Overall structure of the IMPAct framework. . . . .	58
5.7	Data flow in a “ <i>transmitted experience</i> ”. . . . .	66
5.8	Data flow in a “ <i>simulated experience</i> ”. . . . .	66
5.9	Data flow during “ <i>robot supervision</i> ”. . . . .	66
6.1	Soft-/Hardware setup of the mobile gesture-based remote for chainlike robots. . . . .	87
6.2	Hand interaction methods. . . . .	89
6.3	Comparing continuous hand-gesture-based robot control methods. . . . .	91
6.4	Experiment results of continuous hand-gesture-based robot control methods. . . . .	92
6.5	Ambiguity of reach to grasp hand movements in a tabletop scenario. . . . .	95
6.6	[Hardware setup for analyzing reach to grasp movements. . . . .	96
6.7	Comparison of open-loop and closed-loop ballistic movements. . . . .	97
6.8	Correlation of velocities and target sizes in reach to grasp movements. . . . .	98
7.1	Collection of use cases for combining ROS and Unity3D. . . . .	104
7.2	Virtual physical interaction with a modular robot. . . . .	106
7.3	Transfer of human body and hand tracking to virtual robotic models. . . . .	107
7.4	A bodytracking-based flight interface in VR. . . . .	108
7.5	Immersive teleoperation of a mobile robot equipped with a 360° camera. . . . .	109
8.1	Recorded trajectory during the visual virtual fixture experiment. . . . .	112
8.2	Example of a visual virtual fixture. . . . .	114

---

8.3	Different levels of difficulty during the visual virtual fixtures user study. . .	114
8.4	Results of qualitative questionnaires. . . . .	116
8.5	Summary of the AttrakDiff questionnaire during virtual fixture experiment. . .	117
8.6	Times to finish the courses. . . . .	118
8.7	Mean distances to obstacles. . . . .	119
8.8	Collisions of the gripper with the obstacles. . . . .	120
8.9	Translational head movements of the operator. . . . .	121
8.10	Rotational head movements of the operator. . . . .	121
9.1	Illustration of the co-located MR human-robot interaction. . . . .	126
9.2	Concept of a pick-and-place implementation with a see-through display. . .	128
9.3	Registration approach with 2D markers. . . . .	130
9.4	Participant during the MR pick-and-place experiment. . . . .	133
9.5	The arrangement of cylinders and target points during the experiment. . .	134
9.6	NASA-TLX results from MR pick-and-place experiment. . . . .	136
9.7	Resulting selection times from the pick-and-place experiment. . . . .	137
9.8	Procedures of the pick-and-place implementation. . . . .	138
B.1	General system architecture for combining ROS and Unity3D. . . . .	159
B.2	Applying hand-tracking to end-effector control. . . . .	161
B.3	Example of using a generic architecture for ROS and Unity3D. . . . .	162
B.4	Original ROS Message Header. . . . .	163
B.5	Abstract Class <code>RosMessage</code> in Unity3D. . . . .	164
B.6	ROS Advertise Message. . . . .	164
B.7	ROS Subscribe Message. . . . .	165
B.8	ROS Publish Message. . . . .	165
B.9	Header Implementation. . . . .	166
B.10	<code>MessageData</code> Marker Interface. . . . .	166
B.11	Example of a <code>MessageData</code> Class: <code>Jointstate</code> . . . . .	167
B.12	Message passing and roundtrip latency between ROS and Unity3D. . . .	168
B.13	A serialized publish message. . . . .	168
B.14	Message factory. . . . .	169

B.15 Deserializing a JSON string with the <code>RosBidgeClient</code> . . . . .	169
B.16 Serializing a ROS message to a JSON string with the <code>RosBridgeClient</code> . . . . .	170
B.17 ROSbridge Manager Asset Options. . . . .	171
B.18 Methods of the <code>RosBridgeClient</code> . . . . .	172
B.19 Connecting to the ROSbridge from Unity3D via websockets. . . . .	173
B.20 Information flow directions during teleoperation. . . . .	173
B.21 Concurrent communication for Windows and Windows UWP. . . . .	174
B.22 Imported robot models in Unity3D. . . . .	176
B.23 A simple rotational joint script with position control. . . . .	178
B.24 Maximum reachability of robotic manipulator. . . . .	180
B.25 IK analysis and visualization. . . . .	181
B.26 Distance function for KD-tree. . . . .	182
B.27 KD-tree construction. . . . .	182
B.28 Radial search in a KD-tree. . . . .	182
B.29 Drawing the KD-tree with a Unity3D particle system. . . . .	183
B.30 Collection of utilizable input devices at the Unity-side. . . . .	184
B.31 Application of tracking devices. . . . .	187
B.32 Coordinate frame conventions of ROS and Unity3D. . . . .	189
B.33 Simplified statemachine for a pick-and-place user interface. . . . .	191
B.34 Abstract class <code>State</code> . . . . .	191

## LIST OF TABLES

4.1	Hand motion taxonomy. . . . .	43
5.1	Results of the card sorting. . . . .	59
5.2	IMPAct template. . . . .	61
5.3	Interaction parameter. . . . .	62
5.4	Paradigm. . . . .	62
5.5	Image/display/vision. . . . .	63
5.6	Input. . . . .	63
5.7	Output. . . . .	64
5.8	Embodiment. . . . .	64
5.9	Immersion/user perception. . . . .	65
5.10	Modalities. . . . .	65
5.11	Space and time. . . . .	66
5.12	Autonomy. . . . .	67
5.13	Behavior. . . . .	68
5.14	Robot appearance. . . . .	68
5.15	Application. . . . .	68
5.16	Comparison of Virtual Fixtures Implementations . . . . .	81
8.1	Total count of collisions. . . . .	120
9.1	Translational and rotational error of the registration. . . . .	129

## LIST OF TABLES

---

9.2	Summary of time, accuracy, usability and task-load. . . . .	135
A.1	Factors of the IMPAct framework. . . . .	149
A.2	Empty IMPAct template. . . . .	155

## LIST OF DEFINITIONS

3.1	(Robot - Oxford Dictionary) . . . . .	17
3.2	(Robot - Kent Schlüssel, 1983) . . . . .	17
3.3	(Robot - McKerrow, 1986) . . . . .	17
3.4	(Autonomy - Zeltzer, 1992) . . . . .	18
3.5	(Automation - Parasuraman et. al, 2000) . . . . .	18
3.6	(Intelligent Robot - Murphy & Arkin, 2000) . . . . .	18
3.7	(Capability - Gunderson & Gunderson, 2004) . . . . .	18
3.8	(Intelligence - Gunderson & Gunderson, 2004) . . . . .	18
3.9	(Autonomy - Gunderson & Gunderson, 2004) . . . . .	19
3.10	(Robot - Bekey, 2005) . . . . .	19
3.11	(Autonomy – Beer et. al, 2014) . . . . .	19
3.12	(Robotic System - Dennis Krupke, 2019) . . . . .	35





## ACKNOWLEDGMENTS

At first, I want to thank my supervisors Prof. Dr. Frank Steinicke and Prof. Dr. Jianwei Zhang, for guiding me through the process of promotion. At the same time, I had all the freedom to realize my ideas. I enjoyed the time spent with Frank in the lab, as much as the time spent together on conferences or group events. I am happy that Jianwei enabled having the IROS conference in Hamburg. So I had the chance to be part of the host team of one of the largest robotics conferences in the world.

I especially want to thank Prof. Dr. Gerd Bruder and Dr. Paul Lubos for introducing me very kindly to the science of human-computer interaction. I learned a lot and I am pleased about this experience.

Furthermore, I want to thank all the other colleagues and co-authors, who listened to my current issues and provided me with useful advice or confronted me with different opinions.

Most importantly, I want to thank my whole family for supporting me in all of my decisions. I want to thank my parents Ina and Rudi, for being patient during my studies and letting me do it my way.

Mainly, I want to thank my wife Anika and my son Niklas for supporting me with focus and for reminding me of the most important things. They (almost ;-)) never complained when I came later or had to work at home on an important topic and either asked or answered the right questions. Everything went as it had to go.



## BIBLIOGRAPHY

- [1] ABB. Yumi, 2019. URL <https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi>. (accessed 2019-10-26). 3
- [2] E. Ackerman. Boston Dynamics Is Getting Ready to Produce Lots of SpotMinis. (accessed 2019-07-27). 3
- [3] G. Adamides, G. Christou, C. Katsanos, M. Xenos, and T. Hadzilacos. Usability Guidelines for the Design of Robot Teleoperation: A Taxonomy. *IEEE Trans. Human-Mach. Syst.*, 45(2):256–262, April 2015. ISSN 2168-2291. doi: 10.1109/THMS.2014.2371048. 56
- [4] Aldebaran Robotics. NAO Robot, 2018. URL <https://www.softbankrobotics.com/emea/themes/custom/softbank/images/full-nao.png>. (accessed 2019-09-24). 24
- [5] Aldebaran Robotics. NAO Specification, 2018. URL [https://en.wikipedia.org/wiki/Nao\\_\(robot\)](https://en.wikipedia.org/wiki/Nao_(robot)). (accessed 2019-09-24). 25
- [6] J. E. Allen, C. Guinn, and E. Horvitz. Mixed-Initiative Interaction. *IEEE Intelligent Systems and their Applications*, 14(5):14–23, 10 1999. doi: 10.1109/5254.796083. 3, 151
- [7] L. Almeida, P. Menezes, and J. Dias. Improving Robot Teleoperation Experience via Immersive Interfaces. In *2017 4th Experiment@International Conference (exp.at'17)*, pages 87–92, June 2017. doi: 10.1109/EXPAT.2017.7984414. 74, 99
- [8] R. S. Andersen, O. Madsen, J.-E. Moeslund, and H. B. Amor. Projecting Robot Intentions into Human Environments. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 294–301, Aug 2016. doi: 10.1109/ROMAN.2016.7745145. 73
- [9] F. Argelaguet and C. Andujar. A Survey of 3D Object Selection Techniques for Virtual Environments. *Computers & Graphics*, 37(3):121–136, 2013. 4
- [10] O. Ariza, P. Lubos, F. Steinicke, and G. Bruder. Ring-Shaped Haptic Device with Vibrotactile Feedback Patterns to Support Natural Spatial Interaction. In *Proceedings of the ICAT-EGVE (International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments) 2015*, pages 175–181, 2015. 184

- [11] O. Ariza, J. P. Freiwald, N. Laage, M. Feist, M. Salloum, G. Bruder, and F. Steinicke. Inducing Body-Transfer Illusions in VR by Providing Brief Phases of Visual-Tactile Stimulation. In *Proceedings of the ACM Symposium on Spatial User Interaction (SUI)*, pages 61–68, Vogt-Kölln-Str. 30, 2016. Hamburg Universität. 184
- [12] I. Asimov. *I, robot*. Doubleday, Garden City, N.Y, 1950. ISBN 978-0-385-42304-5. 22
- [13] I. Asimov. Three Laws of Robotics, 2018. URL [https://en.wikipedia.org/wiki/Three\\_Laws\\_of\\_Robotics](https://en.wikipedia.org/wiki/Three_Laws_of_Robotics). (accessed 2019-09-24). 22
- [14] K. J. Åström and B. Wittenmark. *Adaptive Control*. Dover Books on Electrical Engineering. Dover Publications, 2008. ISBN 9780486462783. 30
- [15] J. A. Atherton and M. A. Goodrich. Supporting Remote Manipulation with an Ecological Augmented Virtuality Interface. *Artificial intelligence and simulation of behaviour (AISB) on New Frontiers in Human-Robot Interaction, Edinburgh, UK*, 2009. 75
- [16] L. Bainbridge. Ironies of Automation. In *Analysis, Design and Evaluation of Man-Machine Systems*, pages 129–135. Elsevier, 1983. 3, 56
- [17] M. Barnett and M. Dixon. *Gods and myths of ancient Greece*. Smithmark Publishers, 1996. 16
- [18] C. Bartneck and M. Okada. Robotic User Interfaces. In *Proceedings of the Human and Computer Conference*, pages 130–140. Citeseer, 2001. 34, 55, 152, 153
- [19] N. Beattie, B. Horan, and S. McKenzie. Taking the LEAP with the Oculus HMD and CAD - Plucking at Thin Air? *Procedia Technol.*, 20:149–154, 2015. ISSN 2212-0173. doi: <http://dx.doi.org/10.1016/j.protcy.2015.07.025>. Proceedings of The 1st International Design Technology Conference, DESTTECH2015, Geelong. 160
- [20] J. M. Beer, A. D. Fisk, and W. A. Rogers. Toward a Framework for Levels of Robot Autonomy in Human-Robot Interaction. *J. Hum.-Robot Interact.*, 3(2):74–99, July 2014. ISSN 2163-0364. doi: 10.5898/JHRI.3.2.Beer. 3, 19, 56, 112
- [21] A. K. Bejczy, W. S. Kim, and S. C. Venema. The Phantom Robot: Predictive Displays for Teleoperation with Time Delay. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 546–551 vol.1, May 1990. doi: 10.1109/ROBOT.1990.126037. 71, 104
- [22] G. A. Bekey. *Autonomous Robots: From Biological Inspiration to Implementation and Control*. MIT press, 2005. 19
- [23] D. J. Benjamin, J. O. Berger, M. Johannesson, B. A. Nosek, E.-J. Wagenmakers, R. Berk, K. A. Bollen, B. Brembs, L. Brown, C. Camerer, et al. Redefine statistical significance. *Nature Human Behaviour*, 2(1):6, 2018. 116
- [24] M. Bischoff. ros-sharp, 2017. URL <https://github.com/siemens/ros-sharp>. (accessed 2019-10-19). 113
- [25] R. A. Bolt. “put-that-there”: Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '80*, pages 262–270, New York, NY, USA, 1980. ACM. ISBN 0-89791-021-4. doi: 10.1145/800250.807503. 127

- [26] D. Bowman, E. Kruijff, J. J. LaViola Jr., and I. P. Poupyrev. *3D User Interfaces: Theory and Practice*. Addison-Wesley, 2004. 4, 39, 41, 55, 151, 154
- [27] J. Brooke. SUS - A Quick and Dirty Usability Scale. *Usability evaluation in industry*, 189(194):4–7, 1996. 116, 134
- [28] G. C. Burdea. Invited Review: The Synergy between Virtual Reality and Robotics. *IEEE Trans. Robot. Autom.*, 15(3):400–410, June 1999. ISSN 1042-296X. doi: 10.1109/70.768174. 50, 70, 111, 125
- [29] C. Calefato, R. Montanari, and F. Tesauri. The Adaptive Automation Design. In *Human Computer Interaction: New Developments*. InTech, 2008. doi: 10.5772/5878. 3, 56, 151, 154
- [30] L. Cancedda, A. Cannavò, G. Garofalo, F. Lamberti, P. Montuschi, and G. Paravati. Mixed Reality-Based User Interaction Feedback for a Hand-Controlled Interface Targeted to Robot Teleoperation. In *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*, pages 447–463. Springer, 2017. doi: 10.1007/978-3-319-60928-7\_38. 79
- [31] Y. Chae, S. Jo, and J. Jeong. Brain-Actuated Humanoid Robot Navigation Control Using Asynchronous Brain-Computer Interface. In *2011 5th International IEEE/EMBS Conference on Neural Engineering*, pages 519–524, April 2011. doi: 10.1109/NER.2011.5910600. 86
- [32] B. Chandrasekaran and J. M. Conrad. Human-robot collaboration: A survey. In *SoutheastCon 2015*, pages 1–8, April 2015. doi: 10.1109/SECON.2015.7132964. 4
- [33] E. Clarkson and R. C. Arkin. Applying Heuristic Evaluation to Human-Robot Interaction Systems. In *Flairs Conference*, pages 44–49, 2007. 154
- [34] R. Codd-Downey, P. M. Forooshani, A. Speers, H. Wang, and M. Jenkin. From ROS to Unity: Leveraging Robot and Virtual Environment Middleware for Immersive Teleoperation. In *Information and Automation (ICIA)*, pages 932–936, July 2014. doi: 10.1109/ICInfA.2014.6932785. 99, 158
- [35] T. H. J. Collett and B. MacDonald. An Augmented Reality Debugging System for Mobile Robot Software Engineers. *Journal of Software Engineering for Robotics*, 2009. 73
- [36] Wikimedia Commons. Kawasaki Robot Control Panel, 1987. URL [https://commons.wikimedia.org/wiki/File:KAWASAKI\\_ROBOT\\_1987\\_CONTROL\\_PANEL\\_\(14844214081\).jpg](https://commons.wikimedia.org/wiki/File:KAWASAKI_ROBOT_1987_CONTROL_PANEL_(14844214081).jpg). (accessed 2019-09-24). 34
- [37] Wikimedia Commons. Fanuc Robot Control Unit, 2019. URL <https://en.oxforddictionaries.com/definition/robot>. (accessed 2019-09-24). 34
- [38] G. Cox. Yan Shi the Artificer, 2018. URL <https://blog.salvius.org/2014/01/a-history-of-robotics-yan-shi-artificer.html>. (accessed 2019-09-24). 20
- [39] J. J. Craig. *Introduction to Robotics: Mechanics and Control*, volume 3. Pearson/Prentice Hall Upper Saddle River, NJ, USA:, 2005. 152, 153, 154
- [40] J. W. Crandall and M. A. Goodrich. Experiments in Adjustable Autonomy. In *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)*, volume 3, pages 1624–1629 vol.3, 2001. doi: 10.1109/ICSMC.2001.973517. 3, 56

- [41] J. W. Crandall, M. A. Goodrich, D. R. Olsen, and C. W. Nielsen. Validating Human-Robot Interaction Schemes in Multitasking Environments. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 35(4):438–449, July 2005. ISSN 1083-4427. doi: 10.1109/TSMCA.2005.850587. 52, 53, 54
- [42] L. Da Vinci. Mechanical Knight, 2018. URL <http://static.t13.cl/images/sizes/1200x675/1518959407-100061683leonardo-robot3.jpg>. (accessed 2019-09-24). 21
- [43] F. Daiber, D. Valkov, F. Steinicke, K. H. Hinrichs, and A. Krüger. Towards Object Prediction Based on Hand Postures for Reach to Grasp Interaction. In *Proceedings of ACM CHI Workshop on The 3rd Dimension of CHI: Touching and Designing 3D User Interfaces (3DCHI)*, pages 99–106, 2012. 94, 96
- [44] H. Das, T. B. Sheridan, and J.-E. Slotine. Kinematic Control and Visual Display of Redundant Teleoperators. In *Conference Proceedings., IEEE International Conference on Systems, Man and Cybernetics*, pages 1072–1077 vol.3, Nov 1989. doi: 10.1109/ICSMC.1989.71462. 70
- [45] P. G. De Barros and R. W. Lindeman. A Survey of User Interfaces for Robot Teleoperation, 2009. 50, 56, 113, 153
- [46] S. W. A. Dekker and D. D. Woods. MABA-MABA or Abracadabra? Progress on Human-Automation Coordination. *Cognition, Technology & Work*, 4(4):240–244, Nov 2002. ISSN 1435-5558. doi: 10.1007/s101110200022. 3, 56
- [47] DFKI. Mixed Reality Systems for Crosssite Production in Industrie 4.0. [https://www.dfki.de/web/news/dfki-cebit-2017/mixed-reality/index\\_html/](https://www.dfki.de/web/news/dfki-cebit-2017/mixed-reality/index_html/). Accessed: 2018-02-26. 127
- [48] J. Di Stefano III. *Schaum's Outline of Feedback and Control Systems, 2nd Edition (Schaum's Outlines)*. McGraw-Hill Education, 2013. 29, 30
- [49] Oxford Dictionary. Robot, 2019. URL <https://en.oxforddictionaries.com/definition/robot>. (accessed 2019-09-24). 17
- [50] M. A. Diftler, J. S. Mehling, M. E. Abdallah, N. A. Radford, L. B. Bridgwater, A. M. Sanders, R. S. Askew, D. M. Linn, J. D. Yamokoski, F. A. Permenter, et al. Robonaut 2-the first humanoid robot in space. In *2011 IEEE international conference on robotics and automation*, pages 2178–2183. IEEE, 2011. 24
- [51] J. L. Drury, J. Scholtz, and H. A. Yanco. Awareness in Human-Robot Interactions. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)*, volume 1, pages 912–918 vol.1, Oct 2003. doi: 10.1109/ICSMC.2003.1243931. 3
- [52] J. S. Dyrstad and J. R. Mathiassen. Grasping Virtual Fish: A Step Towards Robotic Deep Learning From Demonstration in Virtual Reality. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1181–1187, Dec 2017. doi: 10.1109/ROBIO.2017.8324578. 79, 113
- [53] J. S. Dyrstad, E. Ruud Øye, A. Stahl, and J. Reidar Mathiassen. Teaching a Robot to Grasp Real Fish by Imitation Learning from a Human Supervisor in Virtual Reality. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7185–7192, Oct 2018. doi: 10.1109/IROS.2018.8593954. 79

- [54] A. El Saddik. Digital Twins: The Convergence of Multimedia Technologies. *IEEE MultiMedia*, 25(2):87–92, Apr 2018. ISSN 1070-986X. doi: 10.1109/MMUL.2018.023121167. 4
- [55] M. R. Elara, C. A. Acosta Calderon, C. Zhou, P. K. Yue, and L. Hu. Using Heuristic Evaluation for Human-Humanoid Robot Interaction in the Soccer Robotics Domain. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids 2007)*, Pittsburgh, USA, Nov, 2007. 154
- [56] M. R. Endsley. Level of Automation Effects on Performance, Situation Awareness and Workload in a Dynamic Control Task. *Ergonomics*, 42(3):462–492, 1999. doi: 10.1080/001401399185595. PMID: 10048306. 3
- [57] H. C. Fang, S. K. Ong, and A. Y. C. Nee. Novel AR-Based Interface for Human-Robot Interaction and Visualization. *Advances in Manufacturing*, 2(4):275–288, Dec 2014. ISSN 2195-3597. doi: 10.1007/s40436-014-0087-9. 77, 125, 136
- [58] S. Feiner, B. Macintyre, and D. Seligmann. Knowledge-Based Augmented Reality. *Commun. ACM*, 36(7):53–62, July 1993. ISSN 0001-0782. doi: 10.1145/159544.159587. 70
- [59] J. A. Frank, M. Moorhead, and M. Kapila. Realizing Mixed-Reality Environments with Tablets for Intuitive Human-Robot Collaboration for Object Manipulation Tasks. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 302–307, Aug 2016. doi: 10.1109/ROMAN.2016.7745146. 78
- [60] J. A. Frank, M. Moorhead, and V. Kapila. Mobile Mixed-Reality Interfaces That Enhance Human-Robot Interaction in Shared Spaces. *Frontiers in Robotics and AI*, 4:20, 2017. ISSN 2296-9144. doi: 10.3389/frobt.2017.00020. 73
- [61] Thomas W. Frey and H. Jean Page. Virtual HUD using an HMD. In Ronald J. Lewandowski, Loran A. Haworth, Henry J. Girolamo, and Clarence E. Rash, editors, *Helmet- and Head-Mounted Displays VI*, volume 4361, pages 251 – 262. International Society for Optics and Photonics, SPIE, 2001. doi: 10.1117/12.438000. 183
- [62] K.-C. Gan and E. R. Hoffmann. Geometrical Conditions for Ballistic and Visually Controlled Movements. *Ergonomics*, 31(5):829–839, 1988. doi: 10.1080/00140138808966724. 96
- [63] A. Gaschler, M. Springer, M. Rickert, and A. Knoll. Intuitive Robot Tasks with Augmented Reality and Virtual Obstacles. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6026–6031, May 2014. doi: 10.1109/ICRA.2014.6907747. 77
- [64] D. Gelernter. *Mirror Worlds: Or the Day Software Puts the Universe in a Shoebox... How It Will Happen and What It Will Mean*. Oxford University Press, 1993. 4
- [65] FRANKA EMIKA GmbH. Franka emika, 2019. URL <https://www.franka.de/>. (accessed 2019-10-26). 3
- [66] J. González-Gómez, H. Zhang, E. Boemo, and J. Zhang. Locomotion Capabilities of a Modular Robot with Eight Pitch-Yaw-Connecting Modules. In *9th International Conference on Climbing and Walking Robots. CLAWAR06*, September 2006. 85, 86, 87
- [67] J. P. Gunderson and L. F. Gunderson. Intelligence = Autonomy = Capability. *Performance Metrics for Intelligent Systems, PERMIS*, 2004. 18, 19, 56, 152

- [68] C. Hand. A Survey of 3D Interaction Techniques. In *Computer graphics forum*, volume 16, pages 269–281. Wiley Online Library, 1997. 4
- [69] S. G. Hart and L. E. Staveland. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. *Advances in psychology*, 52:139–183, 1988. doi: 10.1016/s0166-4115(08)62386-9. 116, 134
- [70] S. Hashimoto, A. Ishida, and M. Inami. TouchMe: An Augmented Reality Based Remote Robot Manipulation. In *The 21st International Conference on Artificial Reality and Telexistence, Proceedings of ICAT2011*, 2011. 125
- [71] M. Hassenzahl, M. Burmester, and F. Koller. AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität. In *Mensch & Computer 2003*, pages 187–196. Springer, 2003. doi: 10.1007/978-3-322-80058-9\_19. 116, 134
- [72] F. A. Haugen. Basic dynamics and control. *Porsgrunn: TeachTech*, 01 2010. 30
- [73] M. Haun. *Handbuch Robotik: Programmieren und Einsatz Intelligenter Roboter*. Springer-Verlag, 2013. 17
- [74] F. Herrero-Carrón, F. B. Rodríguez, and P. Varona. Study and Application of Central Pattern Generator Circuits to the Control of a Modular Robot. Master’s thesis, Escuela Politécnica Superior, Universidad Autónoma de Madrid, 2007. 86, 87
- [75] K. Hertkorn, M. A. Roa, M. Brucker, P. Kremer, and C. Borst. Virtual Reality Support for Teleoperation using Online Grasp Planning. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2074–2074, Nov 2013. doi: 10.1109/IROS.2013.6696637. 160
- [76] Lawrence J. Hettinger and Gary E. Riccio. Visually induced motion sickness in virtual environments. *Presence: Teleoperators and Virtual Environments*, 1(3):306–310, 1992. doi: 10.1162/pres.1992.1.3.306. 157
- [77] S. Hirose. *Biologically Inspired Robots: Snake-Like Locomotors and Manipulators*. Oxford science publications. Oxford University Press, 1993. ISBN 9780198562610. 86, 87
- [78] L. Hohl, R. Tellez, O. Michel, and A. J. Ijspeert. Aibo and webots: Simulation, wireless remote control and controller transfer. *Robotics and Autonomous systems*, 54(6):472–485, 2006. 25
- [79] Honda. H0, 2018. URL [https://i.dailymail.co.uk/i/pix/2014/10/18/1413644947731\\_wps\\_47\\_MUST\\_CREDIT\\_Honda\\_Rex\\_Fea.jpg](https://i.dailymail.co.uk/i/pix/2014/10/18/1413644947731_wps_47_MUST_CREDIT_Honda_Rex_Fea.jpg). (accessed 2019-09-24). 23
- [80] A. J. Ijspeert. Central Pattern Generators for Locomotion Control in Animals and Robots: A Review. *Neural Networks*, 21(4):642–653, 2008. ISSN 0893-6080. doi: 10.1016/j.neunet.2008.03.014. <ce:title>Robotics and Neuroscience</ce:title>. 86
- [81] Dr. Tetsunari Inamura. SIGVerse, 2019. URL <http://www.sigverse.org/wiki/en/index.php?SIGVerse>. (accessed 2019-10-26). 101
- [82] T. Inamura, T. Shibata, H. Sena, T. Hashimoto, N. Kawai, T. Miyashita, Y. Sakurai, Ma. Shimizu, M. Otake, K. Hosoda, et al. Simulator platform that enables social interaction simulation—sigverse: Sociointelligence simulator. In *2010 IEEE/SICE International Symposium on System Integration*, pages 212–217. IEEE, 2010. 101



- [83] International Federation of Robotics (IFR). Executive Summary World Robotics 2018 Industrial Robots, 2018. (accessed 2019-07-27). 3
- [84] N. Ishikawa and K. Suzuki. Development of a human and robot collaborative system for inspecting patrol of nuclear power plants. In *Proceedings 6th IEEE International Workshop on Robot and Human Communication. RO-MAN'97 SENDAI*, pages 118–123, Sep. 1997. doi: 10.1109/ROMAN.1997.646967. 4
- [85] J. A. Jacko. *Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Third Edition*. CRC Press, Inc., Boca Raton, FL, USA, 3rd edition, 2012. ISBN 1439829438, 9781439829431. 154
- [86] I. Jang, J. Carrasco, A. Weightman, and B. Lennox. Intuitive Bare-Hand Teleoperation of a Robotic Manipulator Using Virtual Reality and Leap Motion. In Kaspar Althoefer, Jelizaveta Konstantinova, and Ketao Zhang, editors, *Towards Autonomous Robotic Systems*, pages 283–294, Cham, 2019. Springer International Publishing. ISBN 978-3-030-25332-5. 99
- [87] H. Jin, L. Zhang, S. Rockel, J. Zhang, Y. Hu, and J. Zhang. A Novel Optical Tracking Based Tele-Control System for Tabletop Object Manipulation Tasks. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 636–642, Sept 2015. doi: 10.1109/IROS.2015.7353439. 160
- [88] A. Kamimura, H. Kurokawa, E. Yoshida, K. Tomita, S. Kokaji, and S. Murata. Distributed Adaptive Locomotion by a Modular Robotic System, M-TRAN II. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2370–2377 vol.3, sept.-2 oct. 2004. doi: 10.1109/IROS.2004.1389763. 86, 107
- [89] K. E. Kaplan, K. A. Nichols, and A. M. Okamura. Toward human-robot collaboration in surgery: Performance assessment of human and robotic agents in an inclusion segmentation task. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 723–729, May 2016. doi: 10.1109/ICRA.2016.7487199. 4
- [90] D. Kent, C. Saldanha, and S. Chernova. A Comparison of Remote Robot Teleoperation Interfaces for General Object Manipulation. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, HRI '17*, pages 371–379, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4336-7. doi: 10.1145/2909824.3020249. 113, 131
- [91] B. Keyes, M. Micire, J. L. Drury, and H. A. Yanco. Improving Human-Robot Interaction through Interface Evolution. In *Human-robot interaction*. IntechOpen, 2010. doi: 10.5772/8140. 154
- [92] S. Kim, Y. Kim, J. Ha, and S. Jo. Mapping System with Virtual Reality for Mobile Robot Teleoperation. *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, pages 1541–1541, 2018. 80
- [93] T. Komura and W.-C. Lam. Real-Time Locomotion Control by Sensing Gloves. *Journal of Visualization and Computer Animation*, 17:513–525, 2006. 86
- [94] D. Krupke. Development of Bio-Inspired Locomotion Using Modular Robotic Simulation and Control System. Master's thesis, Universität Hamburg, 2013. 86
- [95] D. Krupke, G. Li, J. Zhang, H. Zhang, and H. P. Hildre. Flexible Modular Robotic Environment for Research and Education. In *26th European Conference on Modelling and Simulation. ECMS2012*, 2012. 87, 88

- [96] D. Krupke, G. Bruder, and F. Steinicke. Analyses of Spatial Ballistic Movements for Prediction of Targets in Reach to Grasp Tasks. In *Eurographics Symposium on Virtual Environments (Poster)*, nov 2015. 10
- [97] D. Krupke, P. Lubos, G. Bruder, J. Zhang, and F. Steinicke. Natural 3D Interaction Techniques for Locomotion with Modular Robots. *Mensch und Computer 2015-Proceedings*, 2015. 10
- [98] D. Krupke, P. Lubos, L. Demski, J. Brinkhoff, G. Weber, F. Willke, and F. Steinicke. Evaluation of Control Methods in a Supernatural Zero-Gravity Flight Simulator. In *Proceedings of the GI-Workshop VR/AR*. GI, 2015. 108
- [99] D. Krupke, L. Einig, E. Langbehn, J. Zhang, and F. Steinicke. Immersive Remote Grasping: Realtime Gripper Control by a Heterogenous Robot Control System. In *VRST '16 Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, pages 337–338, nov 2016. 10
- [100] D. Krupke, N. Hendrich, J. Zhang, and H. Zhang. Gait Optimization Based on Physics Simulation of 3D Robot Models with a Modular Robotic Simulation System. In *ASSISTIVE ROBOTICS: Proceedings of the 18th International Conference on CLAWAR 2015*, pages 612–619. World Scientific, 2016. 107
- [101] D. Krupke, P. Lubos, L. Demski, J. Brinkhoff, G. Weber, F. Willke, and F. Steinicke. Control Methods in a Supernatural Flight Simulator. In *Proceedings of IEEE Virtual Reality (VR) Video Presentation*, 2016. 10, 108
- [102] D. Krupke, F. Steinicke, and J. Zhang. Target Prediction in an Immersive Pick-and-Place Scenario. In *DGR Days electronic proceedings*. DGR (Deutsche Gesellschaft für Robotik), jun 2016. 10
- [103] D. Krupke, S. Starke, L. Einig, F. Steinicke, and J. Zhang. Prototyping of Immersive HRI Scenarios. In *Human-Centric Robotics*, Proceedings of CLAWAR 2017: 20th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, pages 537–544. CLAWAR Association, World Scientific, sep 2017. doi: 10.1142/9789813231047\_0065. 10, 99, 113, 114, 128
- [104] D. Krupke, F. Steinicke, P. Lubos, Y. Jonetzko, M. Görner, and J. Zhang. Comparison of Multimodal Heading and Pointing Gestures for Co-Located Mixed Reality Human-Robot Interaction. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, Oct 2018. doi: 10.1109/IROS.2018.8594043. 10, 60, 62, 77, 103, 113, 114
- [105] D. Krupke, J. Zhang, and F. Steinicke. Virtual Fixtures in VR - Perceptual Overlays for Assisted Teleoperation, Teleprogramming and Learning. In Gerd Bruder, Shunsuke Yoshimoto, and Sue Cobb, editors, *Proceedings of the ICAT-EGVE (International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments) 2018*. The Eurographics Association, The Eurographics Association, nov 2018. 10, 81
- [106] D. Krupke, J. Zhang, and F. Steinicke. IMPAct: A Holistic Framework for Mixed Reality Robotic User Interface Classification and Design. *Multimodal Technologies and Interaction*, 3(2), 2019. ISSN 2414-4088. doi: 10.3390/mti3020025. 10
- [107] C.-P. Kuan and K.-Y. Young. Challenges in VR-Based Robot Teleoperation. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 3, pages 4392–4397, Sept 2003. doi: 10.1109/ROBOT.2003.1242280. 75

- [108] D. Labonte, P. Boissy, and F. Michaud. Comparative Analysis of 3-D Robot Teleoperation Interfaces with Novice Users. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(5):1331–1342, 2010. doi: 10.1109/tsmcb.2009.2038357. 39
- [109] H. Lahamy and D. Litchi. Real-Time Hand Gesture Recognition Using Range Cameras. In *Proceeding of Canadian Geomatics Conference*, pages 1–6, 2010. 95
- [110] J. D. Lee and K. A. See. Trust in Automation: Designing for Appropriate Reliance. *Hum. Factors*, 46:50–80, 2004. doi: 10.1518/hfes.46.1.50.30392. 3
- [111] A. Leeper, K. Hsiao, M. Ciocarlie, L. Takayama, and D. Gossow. Strategies for Human-In-The-Loop Robotic Grasping. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 1–8, March 2012. doi: 10.1145/2157689.2157691. 33, 39, 126
- [112] F. Leutert, C. Herrmann, and K. Schilling. A Spatial Augmented Reality System for Intuitive Display of Robotic Data. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 179–180, March 2013. doi: 10.1109/HRI.2013.6483560. 73, 125
- [113] G. Li. *Hierarchical Control of Limbless Locomotion Using a Bio-Inspired CPG Model*. PhD thesis, 07 2013. 86, 88
- [114] R. W. Lindeman. *Bimanual Interaction, Passive-Haptic Feedback, 3D Widget Representation, and Simulated Surface Constraints for Interaction in Immersive Virtual Environments*. PhD thesis, George Washington University, 1999. AAI9923558. 46, 47, 55, 149, 150, 152
- [115] X. Liu and K. Fujimura. Hand Gesture Recognition Using Depth Data. In *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition*, pages 529–534, 2004. 95
- [116] Y. Liu and Y. Zhang. Toward welding robot with human knowledge: A remotely-controlled approach. *IEEE Transactions on Automation Science and Engineering*, 12(2):769–774, 2014. 26
- [117] M. L. Lupetti, C. Germak, and L. Giuliano. Robots and Cultural Heritage: New Museum Experiences. In *Proceedings of the Conference on Electronic Visualisation and the Arts (EVA)*, pages 322–329, Swindon, UK, 2015. BCS Learning & Development Ltd. doi: 10.14236/ewic/eva2015.36. 125
- [118] J. Mace. ROSbridge suite, 2015. URL [http://wiki.ros.org/rosbridge\\_suite](http://wiki.ros.org/rosbridge_suite). 111
- [119] A. Makhal and A. K. Goins. Reuleaux: Robot Base Placement by Reachability Analysis. *CoRR*, abs/1710.01328, 2017. doi: 10.1109/irc.2018.00028. 137
- [120] I. Malý, D. Sedláček, and P. Leitão. Augmented Reality Experiments with Industrial Robot in Industry 4.0 Environment. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pages 176–181, July 2016. doi: 10.1109/INDIN.2016.7819154. 78
- [121] F. Martín, C. Agüero, J. M. Cañas, G. Abella, R. Benítez, S. Rivero, M. Valenti, and P. Martínez-Martín. Robots in therapy for dementia patients. *Journal of Physical Agents*, 7(1):48–55, 2013. 25
- [122] H. Martins and R. Ventura. Immersive 3-D Teleoperation of a Search and Rescue Robot Using a Head-Mounted Display. In *Emerging Technologies Factory Automation, 2009. ETFA 2009.*, pages 1–8, Sept 2009. doi: 10.1109/ETFA.2009.5347014. 99, 160

- [123] M. Mast, M. Burmester, B. Graf, F. Weisshardt, G. Arbeiter, M. Španěl, Z. Materna, P. Smrž, and G. Kronreif. Design of the Human-Robot Interaction for a Semi-Autonomous Service Robot to Assist Elderly People. In *Ambient Assisted Living*, pages 15–29. Springer, 2015. doi: 10.1007/978-3-319-11866-6\_2. 125
- [124] P. Mckerrow. *Introduction to Robotics*. Addison-Wesley Longman Publishing Co., Inc., 1991. 28, 34, 55, 151, 152, 153, 154
- [125] P. J. McKerrow. Robotics, an Academic Discipline? *Robotics*, 2(3):267–274, 1986. doi: 10.1016/0167-8493(86)90035-5. 17
- [126] M. Meehan, S. Razzaque, M. C. Whitton, and F. P. Brooks. Effect of latency on presence in stressful virtual environments. In *IEEE Virtual Reality, 2003. Proceedings.*, pages 141–148, March 2003. doi: 10.1109/VR.2003.1191132. 157
- [127] Microsoft. Microsoft HoloLens Specification. <https://www.microsoft.com/en-us/hololens/>, 2016. Accessed: 2018-02-26. 129, 133
- [128] P. Milgram and H. Colquhoun. A Taxonomy of Real and Virtual World Display Integration. *Mixed reality: Merging real and virtual worlds*, 1:1–26, 1999. 38, 41, 55, 150, 151
- [129] P. Milgram and F. Kishino. A Taxonomy of Mixed Reality Visual Displays. In *IEICE Transactions on Information and Systems, Special issue on Networked Reality*, 1994. 37, 38, 39, 55, 111, 126, 127, 151, 152
- [130] P. Milgram, S. Zhai, D. Drascic, and J. Grodski. Applications of Augmented Reality for Human-Robot Communication. In *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, volume 3, pages 1467–1472 vol.3, Jul 1993. doi: 10.1109/IROS.1993.583833. 72
- [131] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum. In *Telemanipulator and telepresence technologies*, volume 2351, pages 282–293. International Society for Optics and Photonics, 1995. 38, 39, 55, 150, 151, 152
- [132] C. A. Miller and R. Parasuraman. Beyond Levels of Automation: An Architecture for More Flexible Human-Automation Collaboration. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 47, pages 182–186. SAGE Publications Sage CA: Los Angeles, CA, 2003. 3, 56, 152
- [133] Y. Mizuchi and Te. Inamura. Cloud-based multimodal human-robot interaction simulator utilizing ros and unity frameworks. In *2017 IEEE/SICE International Symposium on System Integration (SII)*, pages 948–955. IEEE, 2017. 99, 101
- [134] R. R. Murphy and R. C. Arkin. *Introduction to AI Robotics*. MIT press, 2000. 18, 31, 55, 152, 153, 154
- [135] NASA. Robonaut, 2018. URL <https://i.pinimg.com/originals/dc/0d/99/dc0d99615218a2b7bc18dce325408d56.jpg>. (accessed 2019-09-24). 24
- [136] NASA. Robonaut Specification, 2018. URL <https://www.nasa.gov/audience/forstudents/5-8/features/nasa-knows/what-is-robot-58.html>. (accessed 2019-09-24). 25

- [137] K. Negishi, Y. Liu, T. Maruyama, Y. Matsumoto, and A. Namiki. Operation Assistance Using Visual Feedback with Considering Human Intention on Master-Slave Systems. In *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2169–2174, Dec 2016. doi: 10.1109/ROBIO.2016.7866651. 78
- [138] New York Times. Early Robotic Surgery, 1985. URL <https://www.nytimes.com/1985/06/25/science/a-robot-arm-assists-in-3-brain-operations.html>. (accessed 2019-09-24). 24
- [139] L. A. Nguyen, M. Bualat, L. J. Edwards, L. Flueckiger, C. Neveu, K. Schwehr, M. D. Wagner, and E., Zbinden. Virtual Reality Interfaces for Visualization and Control of Remote Vehicles. *Autonomous Robots*, 11(1):59–68, Jul 2001. ISSN 1573-7527. doi: 10.1023/A:1011208212722. 72
- [140] K. Nickel and R. Stiefelhagen. Visual Recognition of Pointing Gestures for Human-Robot Interaction. *Image Vision Comput.*, 25(12):1875–1884, dec 2007. ISSN 0262-8856. doi: 10.1016/j.imavis.2005.12.020. 86
- [141] C. W. Nielsen, M. A. Goodrich, and R. W. Ricks. Ecological Interfaces for Improving Mobile Robot Teleoperation. *IEEE Trans. Robot.*, 23(5):927–941, Oct 2007. ISSN 1552-3098. doi: 10.1109/TRO.2007.907479. 125
- [142] S. B. Niku. *Introduction to Robotics: Analysis, Control, Applications.*, 2011. 17, 27, 55, 60, 149, 151, 152, 153, 154
- [143] N. J. Nilsson. Shakey the robot. Technical report, SRI International Menlo Park CA, 1984. 23
- [144] M. Noeske, D. Krupke, N. Hendrich, J. Zhang, and H. Zhang. Interactive Control Parameter Investigation of Modular Robotic Simulation Environment based on Wiimote-HCI's Multi Sensor Fusion. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, pages 478–483, sept. 2012. doi: 10.1109/MFI.2012.6343044. 86, 88
- [145] S. Nof. *Handbook of industrial robotics*. John Wiley, New York, 1999. ISBN 0-471-17783-0. 23
- [146] M. V. Noyes and T. B. Sheridan. A Novel Predictor for Telemanipulation through a Time Delay. *20th Proc. Annu. Conf. Manual Control*, 1984. 71
- [147] R. Nystrom. *Game Programming Patterns*. Genever Benning, 2014. 190
- [148] E. Oyama, N. Tsunemoto, S. Tachi, and Y. Inoue. Experimental Study on Remote Manipulation Using Virtual Reality. *Presence: Teleoperators and Virtual Environments*, 2(2):112–124, 1993. doi: 10.1162/pres.1993.2.2.112. 71
- [149] O. Palinko, F. Rea, G. Sandini, and A. Sciutti. Robot Reading Human Gaze: Why Eye Tracking Is Better Than Head Tracking for Human-Robot Collaboration. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5048–5054, Oct 2016. doi: 10.1109/IROS.2016.7759741. 76
- [150] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. A Model for Types and Levels of Human Interaction with Automation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 30(3):286–297, May 2000. ISSN 1083-4427. doi: 10.1109/3468.844354. 3, 18, 56, 152

- [151] R. Parasuraman, S. Galster, P. Squire, H. Furukawa, and C. Miller. A Flexible Delegation-Type Interface Enhances System Performance in Human Supervision of Multiple Robots: Empirical Studies with RoboFlag. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 35(4):481–493, July 2005. ISSN 1083-4427. doi: 10.1109/TSMCA.2005.850598. 3
- [152] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. Situation Awareness, Mental Workload, and Trust in Automation: Viable, Empirically Supported Cognitive Engineering Constructs. *Journal of Cognitive Engineering and Decision Making*, 2(2):140–160, 2008. doi: 10.1518/155534308X284417. 3, 39
- [153] C.-B. Park and S.-W. Lee. Real-Time 3D Pointing Gesture Recognition for Mobile Robots with Cascade HMM and Particle Filter. *Image Vision Comput.*, 29(1):51–63, January 2011. ISSN 0262-8856. doi: 10.1016/j.imavis.2010.08.006. 86
- [154] PARO Robots U.S. Inc. PARO, 2018. URL <https://robots.ieee.org/robots/paro/Interactive%201/Media%20Player/HD-Q3/paro-int1-1@2x.jpg>. (accessed 2019-09-24). 24
- [155] PARO Robots U.S. Inc. PARO Specification, 2018. URL [https://en.wikipedia.org/wiki/Paro\\_\(robot\)](https://en.wikipedia.org/wiki/Paro_(robot)). (accessed 2019-09-24). 24
- [156] L. Peppoloni, F. Brizzi, C. A. Avizzano, and E. Ruffaldi. Immersive ROS-Integrated Framework for Robot Teleoperation. In *3D User Interfaces (3DUI), 2015 IEEE Symposium on*, pages 177–178, March 2015. doi: 10.1109/3DUI.2015.7131758. 99, 160
- [157] M. I. Posner, M. J. Nissen, and R. M. Klein. Visual Dominance: An Information-Processing Account of its Origins and Significance. *Psychol. Rev.*, 83(2):157, 1976. doi: 10.1037//0033-295x.83.2.157. 136
- [158] PTC Inc. Vuforia. <https://www.vuforia.com/>, 2016. Accessed: 2018-02-26. 129
- [159] Y. Y. Qian and R. J. Teather. The Eyes Don’t Have It: An Empirical Comparison of Head-Based and Eye-Based Selection in Virtual Reality. In *Proceedings of the 5th Symposium on Spatial User Interaction, SUI ’17*, pages 91–98, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5486-8. doi: 10.1145/3131277.3132182. 76, 127
- [160] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: An Open-Source Robot Operating System. *ICRA workshop on open source software*, 3(3.2):5, 2009. 114, 128, 158
- [161] D. Rakita, B. Mutlu, and M. Gleicher. A motion retargeting method for effective mimicry-based teleoperation of robot arms. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, HRI ’17*, pages 361–370, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4336-7. doi: 10.1145/2909824.3020254. 79
- [162] J. Regenbrecht, A. Tavakkoli, and D. Loffredo. A Robust and Intuitive 3D Interface for Teleoperation of Autonomous Robotic Agents through Immersive Virtual Reality Environments. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 199–200, March 2017. doi: 10.1109/3DUI.2017.7893340. 74
- [163] W. Robinett. Synthetic Experience: A Proposed Taxonomy. *Presence: Teleoper. Virtual Environ.*, 1(2):229–247, May 1992. ISSN 1054-7460. doi: 10.1162/pres.1992.1.2.229. 43, 54, 66, 149, 150, 152, 153, 154

- [164] Robotiq. 3-Finger Adaptive Gripper, 2018. URL <https://i0.wp.com/www.thinkbotsolutions.com/wp-content/uploads/2017/10/Robotiq-3-Finger-Gripper.jpg>. (accessed 2019-09-24). 26
- [165] Universal Robots. Universal robots, 2019. URL <https://www.universal-robots.com>. (accessed 2019-10-26). 3
- [166] T. Rodehutsors, M. Schwarz, and S. Behnke. Intuitive Bimanual Telemanipulation under Communication Restrictions by Immersive 3D Visualization and Motion Tracking. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids) 2015*, 2015. doi: 10.1109/humanoids.2015.7363547. 78, 99
- [167] E. Rosen, D. Whitney, E. Phillips, G. Chien, J. Tompkin, G. Konidaris, and S. Tellex. Communicating and Controlling Robot Arm Motion Intent through Mixed-Reality Head-Mounted Displays. *The International Journal of Robotics Research*, 0(0), 2019. doi: 10.1177/0278364919842925. 80
- [168] L. B. Rosenberg. The Use of Virtual Fixtures as Perceptual Overlays to Enhance Operator Performance in Remote Environments. Technical report, Stanford Univ Ca Center for Design Research, 09 1992. 111, 112, 122, 179
- [169] L. B. Rosenberg. The Use of Virtual Fixtures to Enhance Operator Performance in Time Delayed Teleoperation. Technical report, Stanford Univ Ca Center for Design Research, 03 1993. 111, 112, 179
- [170] L. B. Rosenberg. Virtual Fixtures as Tools to Enhance Operator Performance in Telepresence Environments, 1993. 71, 81, 111, 112, 179
- [171] B. Sadrifaridpour, H. Saeidi, J. Burke, K. Madathil, and Y. Wang. Modeling and control of trust in human-robot collaborative manufacturing. In *Robust Intelligence and Trust in Autonomous Systems*, pages 115–141. Springer, 2016. 26
- [172] A. Sandygulova, A. G. Campbell, M. Dragone, and G. M. P. O’Hare. Immersive Human-Robot Interaction. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction, HRI ’12*, pages 227–228, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1063-5. doi: 10.1145/2157689.2157768. 99
- [173] M. Schilling, S. Kopp, S. Wachsmuth, B. Wrede, H. Ritter, T. Brox, B. Nebel, and W. Burgard. Towards a Multidimensional Perspective on Shared Autonomy. In *2016 AAAI Fall Symposium Series*, 2016. 56
- [174] K. Schlüssel. Robotics and Artificial Intelligence across the Atlantic and Pacific. *IEEE Trans. Ind. Electron.*, IE-30(3):244–251, Aug 1983. ISSN 0278-0046. doi: 10.1109/TIE.1983.356735. 17
- [175] J. Scholtz. Theory and Evaluation of Human Robot Interactions. In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, pages 10–ff, Jan 2003. doi: 10.1109/HICSS.2003.1174284. 56
- [176] J. Scholtz, J. Young, J. L. Drury, and H. A. Yanco. Evaluation of Human-Robot Interaction Awareness in Search and Rescue. In *Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on*, volume 3, pages 2327–2332. IEEE, 2004. doi: 10.21236/ada456128. 153
- [177] F. Serra and J.-C. Baillie. Aibo programming using open-r sdk tutorial. *ENSTA. France*, 2003. 25

- [178] Shadow Robot Company. Shadow Motor Hand, 2018. URL <https://www.shadowrobot.com/wp-content/uploads/2012/12/MotorHandBulb.jpg>. (accessed 2019-09-24). 26
- [179] S. Shamsuddin, H. Yussof, L. Ismail, F. A. Hanapiah, S. Mohamed, H. A. Piah, and N. I. Zahari. Initial response of autistic children in human-robot interaction therapy with humanoid robot nao. In *2012 IEEE 8th International Colloquium on Signal Processing and its Applications*, pages 188–193. IEEE, 2012. 25
- [180] T. B. Sheridan. Musings on Telepresence and Virtual Presence. *Presence: Teleoper. Virtual Environ.*, 1(1):120–126, January 1992. ISSN 1054-7460. doi: 10.1162/pres.1992.1.1.120. 38
- [181] T. B. Sheridan and R. Parasuraman. Human-Automation Interaction. *Reviews of Human Factors and Ergonomics*, 1(1):89–129, 2005. doi: 10.1518/155723405783703082. 3
- [182] T. B. Sheridan and W. L. Verplank. Human and Computer Control of Undersea Teleoperators. Technical report, Massachusetts inst of tech Cambridge man-machine systems lab, 1978. 56, 152
- [183] M. Slater and M. V. Sanchez-Vives. Enhancing our Lives with Immersive Virtual Reality. *Frontiers in Robotics and AI*, 3:74, 2016. ISSN 2296-9144. doi: 10.3389/frobt.2016.00074. 4
- [184] M. Slater and S. Wilbur. A Framework for Immersive Virtual Environments (FIVE): Speculations on the Role of Presence in Virtual Environments. *Presence: Teleoperators & Virtual Environments*, 6(6):603–616, 1997. 55, 150, 151, 152, 153, 154
- [185] M. Slater, V. Linakis, M. Usuh, R. Kooper, and G. Street. Immersion, Presence, and Performance in Virtual Environments: An Experiment with Tri-Dimensional Chess. In *ACM virtual reality software and technology (VRST)*, volume 163, page 72. ACM Press New York, NY, 1996. 38
- [186] Sony. AIBO Specification, 2018. URL <https://en.wikipedia.org/wiki/AIBO>. (accessed 2019-09-24). 25
- [187] Sony. AIBO Robot, 2018. URL <https://en.wikipedia.org/wiki/AIBO#/media/File:Aibo-DASA.JPG>. (accessed 2019-09-24). 24
- [188] SRI International. Shakey, 2018. URL <http://archive.computerhistory.org/resources/still-image/Robots/shakey.102635321.lg.jpg>. (accessed 2019-09-24). 23
- [189] S. Starke, N. Hendrich, D. Krupke, and J. Zhang. Evolutionary Multi-Objective Inverse Kinematics on Highly Articulated and Humanoid Robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017)*, sep 2017. 10
- [190] S. Starke, N. Hendrich, and J. Zhang. A Memetic Evolutionary Algorithm for Real-Time Articulated Kinematic Motion. In *IEEE Congress on Evolutionary Computation*, 2017. doi: 10.1109/cec.2017.7969605. 113, 177
- [191] F. Steinicke. *Being Really Virtual - Immersive Natives and the Future of Virtual Reality*. Springer, 2016. ISBN 978-3-319-43076-8. doi: 10.1007/978-3-319-43078-2. 127
- [192] C. R. Stoker, E. Zbinden, T. T. Blackmon, B. Kanefsky, J. Hagen, C. Neveu, D. Rasmussen, K. Schwehr, and M. Sims. Analyzing Pathfinder Data Using Virtual Reality and Superresolved Imaging. *Journal of Geophysical Research: Planets*, 104(E4):8889–8906, 1999. doi: 10.1029/1998JE900019. 73



- [193] D. J. Sturman, D. Zeltzer, and S. Pieper. Hands-On Interaction with Virtual Environments. In *Proceedings of the 2Nd Annual ACM SIGGRAPH Symposium on User Interface Software and Technology*, UIST '89, pages 19–24, New York, NY, USA, 1989. ACM. ISBN 0-89791-335-3. doi: 10.1145/73660.73663. 42, 43, 55, 151
- [194] Y. Takahashi, M. Takeda, and M. Asada. Continuous valued q-learning for vision-guided behavior acquisition. In *Proceedings. 1999 IEEE/SICE/RSJ. International Conference on Multisensor Fusion and Integration for Intelligent Systems. MFI'99 (Cat. No.99TH8480)*, pages 255–260, Aug 1999. doi: 10.1109/MFI.1999.815999. 4
- [195] Y. Tamura, M. Egawa, S. Yano, Y. Kumita, T. Maeda, M. Kato, and H. Asama. Relationship between Sense of Agency and Task Performance in Target Search Task. In *Complex Medical Engineering (CME), 2012 ICME International Conference on*, pages 392–396. IEEE, 2012. doi: 10.1109/iccme.2012.6275710. 38
- [196] T. Tanimoto, K. Shinohara, and H. Yoshinada. Research on Effective Teleoperation of Construction Machinery Fusing Manual and Automatic Operation. *ROBOMECH Journal*, 4(1):14, May 2017. ISSN 2197-4225. doi: 10.1186/s40648-017-0083-5. 38
- [197] R. M. Taylor. Situational Awareness Rating Technique (SART): The Development of a Tool for Aircrew Systems Design. *AGARD, Situational Awareness in Aerospace Operations 17 p(SEE N 90-28972 23-53)*, 1990. 116
- [198] N. Tesla. Radio Controlled Boat, 2018. URL [https://www.thevintagenews.com/wp-content/uploads/2016/09/200px-Tesla\\_boat1.jpg](https://www.thevintagenews.com/wp-content/uploads/2016/09/200px-Tesla_boat1.jpg). (accessed 2019-09-24). 21
- [199] N. Tesla. Radio Controlled Boat Story, 2018. URL <https://www.thevintagenews.com/2016/10/01/1898-nikola-tesla-tricked-entire-crowd-believing-control-boat-shouting-commands-fact-invented-radio-control-piloted-boat/>. (accessed 2019-09-24). 22
- [200] E. Thro. *Robotics: Intelligent Machines for the New Century*. Facts on File science library. Facts on File, 2003. ISBN 9780816047017. 16
- [201] A. Tikanmäki, T. Bedrník, R. Raveendran, and J. Röning. The Remote Operation and Environment Reconstruction of Outdoor Mobile Robots Using Virtual Reality. In *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1526–1531, Aug 2017. doi: 10.1109/ICMA.2017.8016043. 74
- [202] J. Tilley. Automation, Robotics, and the Factory of the Future, 2017. URL <https://www.mckinsey.com/business-functions/operations/our-insights/automation-robotics-and-the-factory-of-the-future>. (accessed 2019-07-27). 3
- [203] F. Tobe. Why Co-Bots Will Be a Huge Innovation and Growth Driver for Robotics Industry, 2015. URL <https://spectrum.ieee.org/automaton/robotics/industrial-robots/collaborative-robots-innovation-growth-driver>. (accessed 2019-07-27). 3
- [204] R. Toris, J. Kammerl, D. V. Lu, J. Lee, O. C. Jenkins, S. Osentoski, M. Wills, and S. Chernova. Robot Web Tools: Efficient Messaging for Cloud Robotics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537, Sept 2015. doi: 10.1109/IROS.2015.7354021. 111, 129, 157, 170


- [205] Unimation. PUMA 500, 2018. URL [https://www.researchgate.net/profile/Fabio\\_Caraffini/publication/259350973/figure/fig3/AS:667834128547855@1536235472492/The-PUMA-560-robotic-arm\\_W640.jpg](https://www.researchgate.net/profile/Fabio_Caraffini/publication/259350973/figure/fig3/AS:667834128547855@1536235472492/The-PUMA-560-robotic-arm_W640.jpg). (accessed 2019-09-24). 23
- [206] Unimation. Unimate, 2018. URL <https://robots.ieee.org/robots/unimate/Photos/SD/unimate-photo1-full.jpg>. (accessed 2019-09-24). 23
- [207] Unity Technologies. Unity 3D, 2015. URL <https://unity3d.com/>. 111
- [208] Universal Robot. UR-5 Specification, 2018. URL [https://en.wikipedia.org/wiki/Universal\\_Robots](https://en.wikipedia.org/wiki/Universal_Robots). (accessed 2019-09-24). 26
- [209] Universal Robots. UR-5, 2018. URL [https://www.robots.com/images/robots/Universal/Universal\\_UR5\\_0002.jpg](https://www.robots.com/images/robots/Universal/Universal_UR5_0002.jpg). (accessed 2019-09-24). 26
- [210] M. Usoh, E. Catena, S. Arman, and M. Slater. Using presence questionnaires in reality. *Presence: Teleoperators & Virtual Environments*, 9(5):497–503, 2000. 116
- [211] C. Vandeveld and J. Saldien. Demonstration of OPSORO – An Open Platform for Social Robots. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 555–556, March 2016. doi: 10.1109/HRI.2016.7451853. 125
- [212] R. Vassallo, A. Rankin, E.-C.-S. Chen, and T.-M. Peters. Hologram Stability Evaluation for Microsoft HoloLens. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 10136, March 2017. doi: 10.1117/12.2255831. 77, 129
- [213] M. M. Veloso, P. E. Rybski, S. Lenser, S. Chernova, and D. Vail. Cmrobotbits: Creating an intelligent aibo robot. *AI magazine*, 27(1):67–67, 2006. 25
- [214] D. Vernon, S. Thill, and T. Ziemke. The Role of Intention in Cognitive Robotics. In *Toward Robotic Socially Believable Behaving Systems-Volume I*, pages 15–27. Springer, 2016. doi: 10.1007/978-3-319-31056-5\_3. 125
- [215] D. Whitney, E. Rosen, D. Ullman, E. Phillips, and S. Tellex. ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, Oct 2018. doi: 10.1109/IROS.2018.8593513. 99, 113
- [216] C. D. Wickens. Multiple resources and performance prediction. *Theoretical issues in ergonomics science*, 3(2):159–177, 2002. 113
- [217] Wikipedia. Golem of Prague, 2018. URL <https://en.wikipedia.org/wiki/Golem>. (accessed 2019-09-24). 21
- [218] Wikipedia. Golem of Prague, 2018. URL <https://vignette.wikia.nocookie.net/simpsons/images/7/73/800px-Golem.png/revision/latest?cb=20170104185721>. (accessed 2019-09-24). 21
- [219] Willow Garage. Personal Robot 2 (PR2), 2018. URL <https://robots.ieee.org/robots/pr2/Interactive%201/Media%20Player/HD-Q3/pr2-int1-1@2x.jpg>. (accessed 2019-09-24). 26
- [220] Willow Garage. PR2 Specification, 2018. URL [https://en.wikipedia.org/wiki/Willow\\_Garage](https://en.wikipedia.org/wiki/Willow_Garage). (accessed 2019-09-24). 26

- [221] B. Wilson, M. Bounds, D. McFadden, J. Regenbrecht, L. Ohenhen, A. Tavakkoli, and D. Loffredo. VETO: An Immersive Virtual Environment for Tele-Operation. *Robotics*, 7(2), 2018. ISSN 2218-6581. doi: 10.3390/robotics7020026. 74
- [222] R. Yamashina, M. Kuroda, and T. Yabuta. Caterpillar Robot Locomotion Based on Q-Learning Using Objective/Subjective Reward. In *2011 IEEE/SICE International Symposium on System Integration (SII)*, pages 1311–1316, Dec 2011. doi: 10.1109/SII.2011.6147638. 87
- [223] H.-D. Yang, A.-Y. Park, and S.-W. Lee. Human-Robot Interaction by Whole Body Gesture Spotting and Recognition. *18th International Conference on Pattern Recognition (ICPR'06)*, 4:774–777, 2006. 86
- [224] D. Zeltzer. Autonomy, Interaction, and Presence. *Presence: Teleoper. Virtual Environ.*, 1(1): 127–132, January 1992. ISSN 1054-7460. doi: 10.1162/pres.1992.1.1.127. 3, 18, 44, 45, 54
- [225] S. Zhai and P. Milgram. Input Techniques for HCI in 3D Environments. In *Conference Companion on Human Factors in Computing Systems, CHI '94*, pages 85–86, New York, NY, USA, 1994. ACM. ISBN 0-89791-651-4. doi: 10.1145/259963.260059. 45, 55, 151, 152, 153
- [226] J. Zhang, E. Langbehn, D. Krupke, N. Katzakis, and F. Steinicke. Detection Thresholds for Rotation and Translation Gains in 360° Video-Based Telepresence System. *IEEE Transactions on Visualization and Computer Graphics (TVCG), Special Issue on IEEE Virtual Reality (VR)*, 24(4): 1671–1680, 2018. 110
- [227] J. Zhang, E. Langbehn, D. Krupke, N. Katzakis, and F. Steinicke. A 360° Video-based Robot Platform for Telepresent Redirected Walking. In *Proceedings of ACM SIGCHI Conference on Human-Robot Interaction (HRI) workshop on Virtual, Augmented and Mixed Reality for Human-Robot Interaction*, International Workshop on Virtual, Augmented and Mixed Reality for Human-Robot Interaction. ACM, ACM, 2018. 10, 110
- [228] K. Čapek. Universal Robots, 2018. URL [https://upload.wikimedia.org/wikipedia/commons/8/87/Capek\\_play.jpg](https://upload.wikimedia.org/wikipedia/commons/8/87/Capek_play.jpg). (accessed 2019-09-24). 21



Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, October 31, 2019

  
\_\_\_\_\_  
(Dennis Krupke)