The Development and Evaluation of an Augmented Reality Assisted Multimodal Interaction System for a Robotic Arm

Roland Aigner

DIPLOMARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Digitale Medien

in Hagenberg

im September 2010

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 27. September 2010

Roland Aigner

Contents

Kurzfassung

Industrieroboter werden in der Massenproduktion dafür verwendet, wiederkehrende Aufgaben über lange Zeiträume auszuführen. In der Fertigung geringer Stückzahlen sind gängige Programmiermethoden für solche Systeme hingegen oft ungeeignet, da in diesen Fällen hohe Flexibilität gefordert ist und Prozesse häufig neu erstellt werden müssen. Besonders für Anfänger ist die Bedienung kompliziert, zeitaufwändig und fehleranfällig. Fehlerhafte Roboterprozesse können Werkstücke oder Roboter beschädigen und damit hohe Kosten verursachen, daher müssen die Anwender erst geschult oder Fachkräfte hinzugezogen werden. Aus diesem Grund werden Industrieroboter in klein- und mittelständischen Unternehmen kaum eingesetzt. Die Ursache für dieses Hindernis ist hauptsächlich die unintuitive Benutzerschnittstelle solcher Systeme.

In dieser Diplomarbeit wird anhand eines Roboterarms eine neuartige, multimodale Interaktionstechnik für die Steuerung und Programmierung von Industrierobotern vorgestellt. Als Eingabemedien dienen Six-Degree-of-Freedom Tracking, Spracheingabe und Touchscreen-Interaktion. Zusätzlich werden Augmented-Reality-Techniken eingesetzt, um den Benutzer bei der Bedienung visuell zu unterstützen. Das vorgestellte System ermöglicht natürliche und intuitive Interaktion, die von Anfängern schnell erlernt werden kann und bietet nützliche Zusatzinformationen, die in einer leicht verständlichen Form dargestellt werden.

Des Weiteren wurde im Rahmen dieser Arbeit eine Studie durchgeführt, um die Anwenderfreundlichkeit des Interaktionsmodells zu prüfen. Die Ergebnisse zeigen, dass die geplanten Designziele erreicht wurden, da die Bedienung im Vergleich zu herkömmlichen Methoden speziell für Anfänger intuitiver und schneller erlernbar ist.

Abstract

In mass production, robotic systems are programmed to perform repetitive tasks for a long period of time. In contrast, small lot size production requires high flexibility and frequent reprogramming. However, status-quo methods for programming industrial robots are complicated, hard to learn, and error-prone for novice users. Since incorrect programs can cause severe damage and huge costs, personnel has to be trained, or dedicated experts have to be hired. As a result, robots are not common in small and medium sized enterprises. This circumstance is mainly due to unintuitive human-robot interfaces.

Using the example of a robotic arm, this thesis introduces a novel multimodal interaction approach for operating and programming industrial robots, by combining six degree of freedom tracking with human speech input and touchscreen interaction. For additional operator support, augmented reality is proposed for visualization. The resulting system enables for natural and intuitive interaction with the robotic arm, while providing descriptive visual information, and can be rapidly adopted by novice users.

This thesis also presents the results of a user study, which was conducted to prove the viability of the proposed interaction technique. The data shows that all major design goals were achieved, since it performed superior in most aspects, compared to a rather common robot controlling system.

Chapter 1

Introduction

In contrast to robots as they appear in science fiction literature and movies, most of current robots lack autonomy and need to be advised in a very explicit manner. They are primarily used in industry, for tasks that are repetitive, difficult, unsafe, or unpleasant for humans. On the other hand there are autonomous robots, using *Artificial Intelligence* (AI) technology, intended to carry out non-repetitive, flexible tasks. Still, they usually need very detailed task and environmental knowledge to do so. Some robots can be considered research platforms, such as the robot dog *AIBO*[1], or the humanoid robots *ASIMO*[2] and *Nao*[3], others are targeted on more everyday and practical issues, like the robotic vacuum cleaner *Roomba*[4]. Many researchers see huge potential for robots used as healthcare assistants.

While numerous research projects in the area of *Human-Robot Interaction* (*HRI*) are trying to enable humans to communicate with autonomous systems in a natural manner, HRI techniques for industrial applications seem to be somewhat conservative. They rely on the very same metaphors in each iteration, just replacing input devices, e.g. using joystick-based devices like 3D mice to manipulate and control objects in a way it was done before using buttons-based interfaces. The fact that users (meaning the programmers of those systems) need to have considerable experience and spatial sense remains, since the underlying interaction method stays unaffected. In contrast to this trend, the *European Robotics Technology Platform* (*EUROP*) [1] quotes:

> Human-robot interaction is the ability of a robot and a human to mutually communicate, which may include physical interaction. This involves communication using a common context, possibly embracing a common cognitive view. The interaction may

[1]see http://support.sony-europe.com/aibo
[2]see http://world.honda.com/asimo
[3]see http://www.aldebaran-robotics.com
[4]see http://www.irobot.com

> be multi-modal using sounds, gestures, and physical interaction. They may involve or result in modifications of the environment. In the short term humans will interact with the robot using defined interfaces the human has to learn. After a series of step changes humans will naturally interact with the robot.

Current HRI methods in industry seem to be somewhat out-of-date compared to state of the art interaction techniques of the *Human-Computer Interaction* (HCI) community, like the ones mentioned in the quote above. Although several different methods for programming industrial robots exist, all of them are tedious and time consuming and call for much technical expertise.

Many robot units need to be installed and programmed for very specific applications in mass production scenarios and hence are meant to repeat the very same task for a long time, once this is accomplished. So status quo methods surely are legitimate in many scenarios, but also more natural and intuitive interaction techniques could be applied in numerous cases. Examples are small lot size productions, requiring flexibility and short setup times for frequent reprogramming, maybe even lacking demands of high precision. Extensive startup costs are the major drawbacks for so called *Small and Medium Sized Enterprises* (SMEs), where robots could act as work assistants in many cases. Additionally, specialists are needed to operate the robots because current interfaces are unintuitive and error-prone to novice users by trend. This is a serious issue, since errors in operation may result in severe damage of the robot, the tool, or the entire workcell[5], or may even threaten humans.

1.1 Programming Industrial Robots

1.1.1 Status Quo

The *American Occupational Safety & Health Administration* classifies paradigms of programming industrial robots in the *OSHA Technical Manual* [18]:

Lead-through programming: The process is either taught using a device known as *Teach Pendant* (also *Teach Panel*), as shown in Figure 1.1, or by an external computer, which is connected to the robot controller. The robot is guided through a series of actions (see Figure 1.2 (a)), which are recorded along the way.

Walk-through programming: The operator is in physical contact with the robot and guides its arm to the desired positions (see Figure

[5]The robot (including its controller and all peripherals), workbench and environment are together referred to as *workcell*.

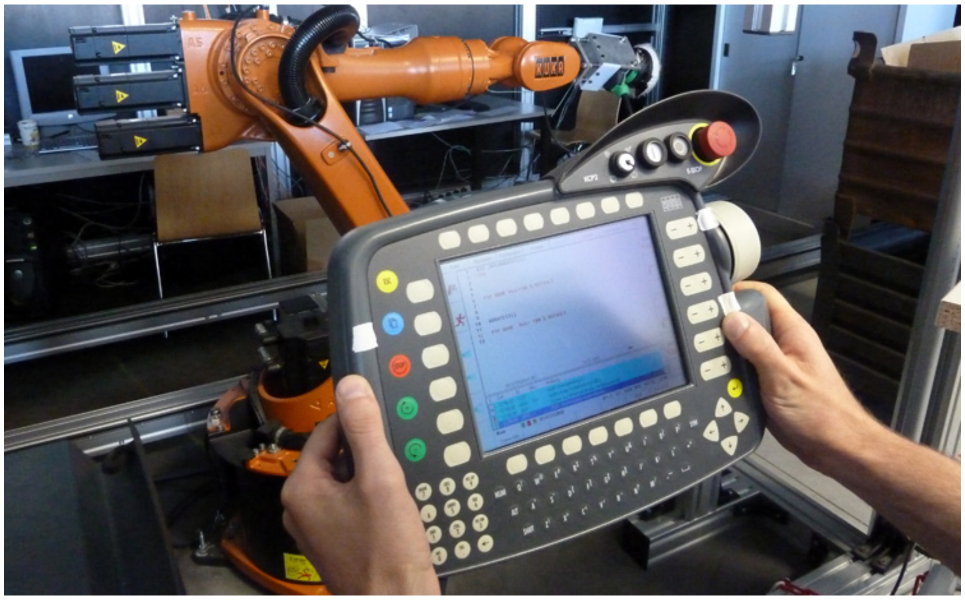
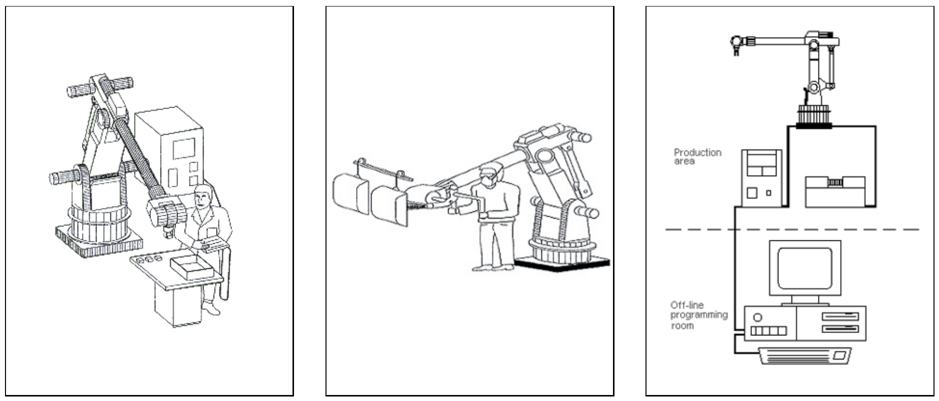


Figure 1.1: Teach pendants are used for programming and operating the robot within the workcell.

1.2 (b)). According to Schraft and Meyer [26], this method is not common in industrial robotics so far, even if there are some products in this area.

Offline programming: The process is programmed completely without a physical robot. Instead, complex computer programs are utilized, which model the entire workcell in *Virtual Reality* (VR), thus programming can take place in a completely virtual manner (see Figure 1.2 (c)). This enables for tests and simulations done entirely in VR, so no robot is needed at all.

Both lead-through and walk-through programming—in which operators act directly on the physical robot—are often also called *teaching* or *online programming*. Schraft and Meyer [26] noted that today's teach pendants are very efficient but complex devices and training takes too long for some applications and situations. Especially this would be the case for trajectory oriented tasks, which happen to be the most time consuming. In those cases, mainly if they involve contact (like welding or polishing processes), offline programming is more sufficient. However, this technique also has obvious limitations, since concise CAD data of workcell and work pieces have to be present. These are often not available in small lot size productions, so offline programming is not a good choice if high flexibility is required.



(a) (b) (c)

Figure 1.2: Today's established robot programming paradigms: *lead-through programming* (a), *walk-through programming* (b) and *offline programming* (c). (Images taken from the *OSHA Technical Manual* [18])

Furthermore, other paradigms exist that tend to translate the duty of instructing robots from *robot experts* to *task experts*. *Visual Programming*[6] uses graphically based interfaces to replace textual commands. In *Programming by Demonstration* (PbD) methods the human demonstrates a task which is observed using different sensors. The demonstration is then interpreted by the system to generate robot programs and actions for execution or it is even used to train an AI system for autonomous operation. PbD is a matter of *High-Level Programming* (also known as *Task-Level Programming*), where humans are intended to just describe *how* to accomplish a task, in a rather abstract manner. Since there are many ways to do so this is a highly complex topic and related research is still in its very beginning.

1.1.2 Specifying a Grasping Sequence

To program a robotic process, basically two entities have to be specified: (i) all positions (or, more precise, the *poses*, thus positions and orientations), which act as waypoints of the procedure and (ii) the procedure itself. For example, a *pick and place* process as illustrated in Figure 1.3, for moving an object from a position A to a position B, might require the following five poses:

- P_1: general safe pose
- P_2: approach to P_3

[6]E.g. *Microsoft Visual Programming Language*, which is a component of *Microsoft Robotics Developer Studio*, or *NXT-G*, the programming language of the *LEGO Mindstorms NXT* robotics kit

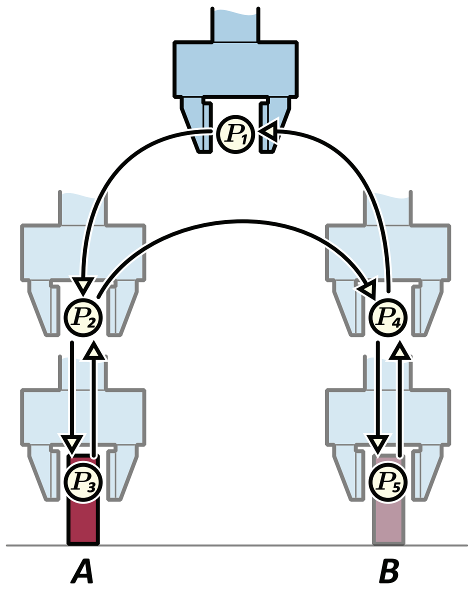


Figure 1.3: This simple *pick and place* process, moving an object from an initial position A to a destination position B, requires five intermediate tool poses (keyframes).

- P_3: object at pose A
- P_4: approach to P_5
- P_5: object at pose B

These poses are also called *keyframes*. Depending on the programming paradigm, there are different ways to define them:

Commands: The user guides the tool to the desired position by either using a *Graphical User Interface* (GUI), or by typing the exact coordinate values into a keyboard. This way maximum precision can be achieved, yet the process is very tedious.

Teach pendant: Using the buttons of the teach pendant (or its 3D mouse if present), the user moves the gripper to the desired position. There usually are several motion modes available, as well as different motion speeds. Using this method in an efficient manner requires much experience.

Lead-by-nose: On some models the user may switch the robot arm limp, so he is able to guide it by hand, while the joint positions can be read back by the system. This way the tool can easily be moved to the designated position. This method is only suitable for low precision demands.

After the positions have been defined, the instruction sequence might be specified as follows:

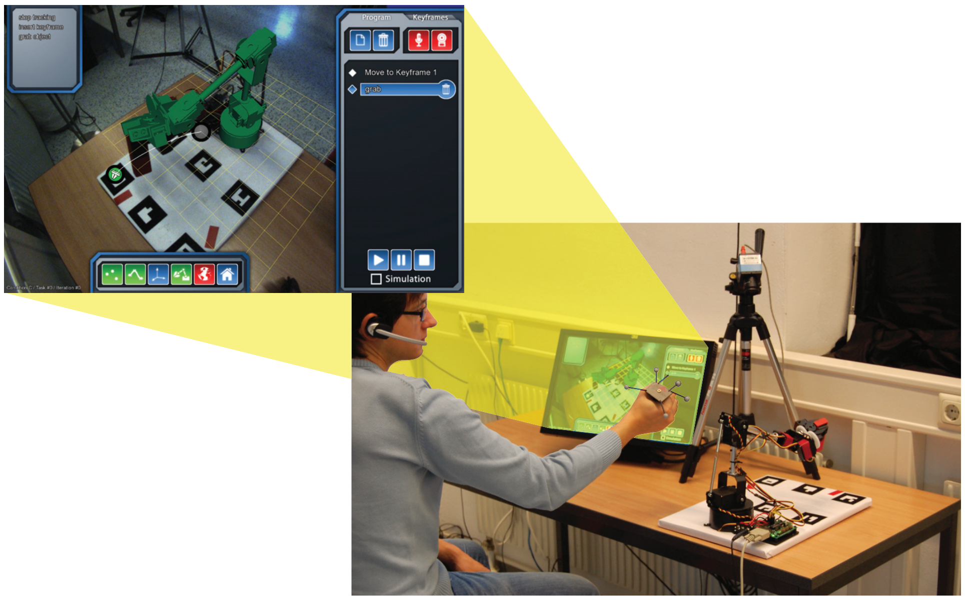


Figure 1.4: *MIMIK* combines 6 DOF tracking, speech input and multitouch interaction with an AR-based visualization.

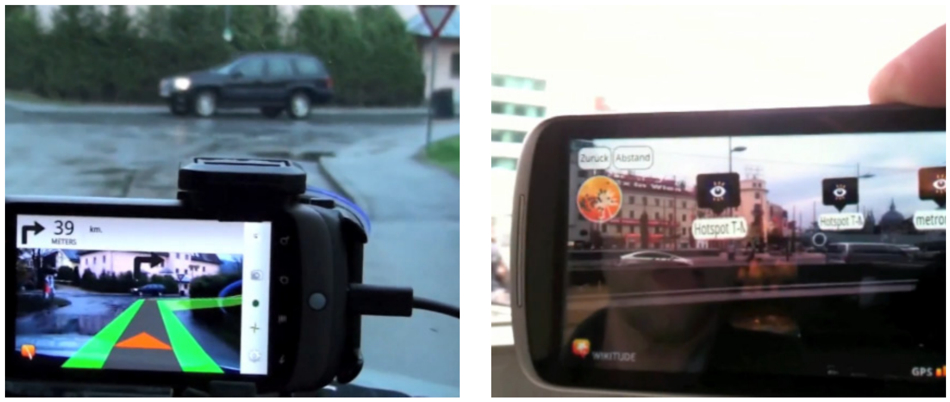
```
Move to P1
Move to P2
Move to P3
Close gripper tool
Move to P2
Move to P4
Move to P5
Open gripper tool
Move to P4
Move to P1 and finish
```

These procedures are usually programmed in specialized scripting languages on the teach pendant or on a computer.

1.2 A New Approach

As suggested in the previous sections, given status quo programming methods lack intuitiveness and efficiency required for SMEs. The programming time only pays off for mass production and is thus unreasonable for manufacturing low quantities. So, alternative interaction methods may be recommendable. *EUROP* [1] quotes:

> Currently, configuration is carried out for a specific task or system at setup or between different tasks by online or offline programming. In the future the process of configuration will be simplified



(a) (b)

Figure 1.5: Two mobile AR applications for smartphones: the navigation system *Wikitude Drive* (a), and the *Wikitude World Browser*, displaying geo-referenced content (b). (Images taken from http://www.wikitude.org.)

through improved user interfaces using more human-compatible modalities.

The intent of this work was to implement and evaluate a HRI interface for controlling and programming a robotic arm that is radically different to conventional ones and thus overcomes current issues. To achieve this, a *multimodal interaction* (MMI) system is proposed, combining six degree of freedom (6 DOF) tracking, speech interaction and touchscreen input, while applying *Augmented Reality* (AR) for visualization (see Figure 1.4).

1.2.1 Augmented Reality

In the consumer field, AR gained more and more popularity in the past few years, e.g. a currently popular application is marketing. This seems to be an effect of the increasing spread of cameras in modern laptops and smart phones. Also, due to the advance in display quality and the increasing computing capabilities of portable devices, some new opportunities show up, like its application for navigation systems, or for augmenting geo-referenced points of interests (see Figure 1.5). This increasing presence of AR seems to lead to its immersion in other fields of potential application. Some communities and researchers seem to be attracted by it for a short time, one of them being the robotics community.

While VR is already common for rapid and intuitive programming as well as for training, AR gained increasing importance in robotics research in the last few years. AR is successively employed in some works, to emphasize environment properties, or to supply additional information to the robot

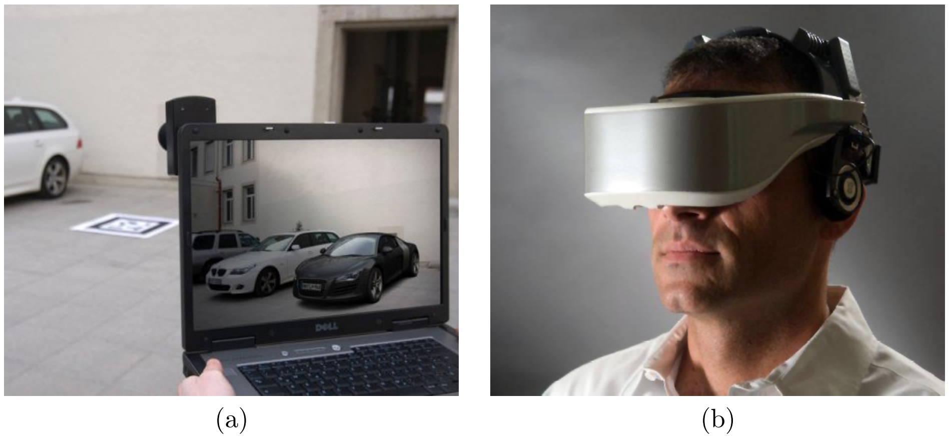


Figure 1.6: VST AR displays superimposed images on a screen (a). (Image courtesy of RTT AG.) The *zSight* is one of *Sensics'* professional HMDs with high-resolution displays (b). (Image taken from http://sensics.com.)

operator. Applications range from anthropomorphic and mobile robots, home assistance and teleoperation systems to industrial robotics.

There are several approaches of implementing AR, each of which involving pros and cons. *Screen-based AR* often comes with the problem of poor spatial coherence. Fixed screen-camera-setups tend to be irritating since they do not adhere to the users view. In contrast to *Video-See-Through* (VST) displays (see Figure 1.6 (a)), the best option imaginable would be *Optical-See-Through* (OST) *Head Mounted Displays* (HMDs) with accurate calibration and tracking. However, to get the image data to the HMDs, the computation unit has to be carried by the user or unpleasant wirings have to be present, so the mobility is limited. Also, especially for the coarse demands of industrial environments, hardware has to be robust and resistant, which is mostly not the case. Additionally, most current HMDs are bulky and therefore uncomfortable. They often also suffer from limited visual field, poor resolution and low color depth (especially OST HMDs). Exceptions are the products of *Sensics* like the *zSight*[7], as shown in Figure 1.6 (b). It is a binocular VST HMD, using OLED microdisplays with a resolution of 1280×1024 pixels per eye. It provides a field of view up to 70° and weighs 450 grams. The major drawback is the price of currently US$24,000, including tracker, stereo audio, microphone and cables. In the future there may be affordable high quality HMDs with wireless data transfer, but at this point they are not an option. *Projector-based AR* can be very advantageous for avoiding problems with spatial coherence, since it enables for augmenting additional data directly onto the affected regions and objects. Projective displays also feature the

[7]see http://sensics.com/products/zSight.php

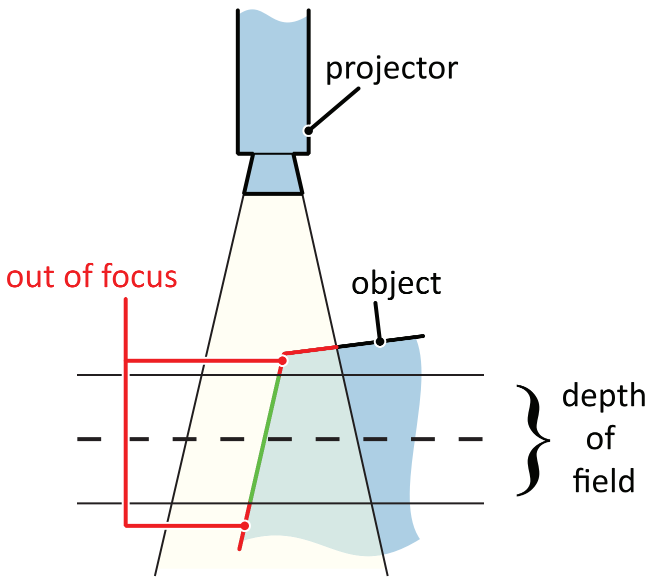


Figure 1.7: Projector-based AR suffers from the low depth of field of current lens projectors. If the distance to the projection surface is varying largely, some regions may be out of focus and become blurry.

benefit of presenting information to several users at a time. This can be very profitable in industrial applications, though it might turn into a drawback when data has to be private. However, there are obvious limitations since data can clearly not be projected into free space, but only onto surfaces. So it is not possible to display off-surface data, such as path trajectories and process waypoints using projectors. Also, there are problems with occlusion, since the robotic arm, as well as the user, may easily mask the projections. Another problem is that most lens projectors have a rather small depth of field. Thus, the projection depth is limited to a small area near the projector's focal plane, while outside this area the projection becomes blurry (see Figure 1.7). Due to that, there are several limitations, especially in acute angles to the projection surface, or if objects with extensive variations in surface have to be highlighted. In contrast to lens projectors, laser projectors do not suffer from this problem. Another advantage of laser projectors is that they are lacking a heavy optical lens system; hence they are much more lightweight and can easier be utilized for mobile solutions. However, Schwerdtfeger et al. [27] mounted a laser projector onto a helmet, and switched to a tripod-mounted system later, since they found that the resulting system was too heavy to be used as a head-mounted device. So head mounted laser projectors also do not seem to be an option at this point.

1.2.2 Multimodality

As defined by Salber et al. [25], in HCI, a *modality* is an interaction method that can be used by an agent to reach a specific goal, the agent being a human or even the system itself. More strictly, it can be described as either a channel enabling an artificial system to receive input from the human, or contrary, a channel enabling a human to perceive output from the system. Thus, examples for *input modalities* are mice, keyboards, touchscreens or touchfoils, or input methods based on computer vision techniques, such as laser pointer tracking or high degree of freedom tracking. Examples for *output modalities* are computer screens, projected displays, force feedback devices and speakers. Summarizing, a modality is *a path of communication between human and artificial system, in either way.*

In contrast to *unimodality*, the term *multimodality* denotes the combination of several modalities, enabling the user to interact with the system through several different interaction channels, either sequentially or concurrently.

Salber et al. [25] also characterize aspects of MMI techniques to split them into four groups. Given an initial state s and a target state s' this is a simplified summary of what they call *CARE properties*:

Equivalent: Modalities of a set M are *equivalent* if any *one* of the modalities can be used to reach s' from s.

Redundant: Modalities of a set M are *redundant* if they are *equivalent* and they are *all used within a certain time frame.*

Assigned: A modality m is *assigned* if *no other modality can be used* to reach s' from s.

Complementary: Modalities of a set M are *complementary* if *all of them must be used within a certain time frame* to reach s' from s.

Since several input channels can and will be used at the same time, it is possible for a multimodal input system to support *deictic expressions*, which are characterized by cross-modality references and thus are examples of *complementary modalities.* If it should do so, it is necessary to extract relevant information from each of the involved input channels and interpret the resulting combination of them—this process is called *Signal Fusion.* An example would be to combine the spoken command “move there” with pointing to a position P with the finger. In this case audio and positional (or directional) inputs are the channels to be combined, and cross-modality reference is given by the use of the word “there”, since it refers to another input channel. The resulting high-level command would be “move to position P”.

Bolt’s pioneering work “Put that there” [3] already showed in 1980 that the combination of multiple modalities can lead to a more natural and intuitive user interface—in his case these were voice and 3D magnetic tracking. The combination of pointing and using pronouns enables for deictic input, and thus requires input fusion. Since then considerable effort has been put into multimodal interaction to achieve more human-like communication with machines.

Although both Bolt's work and AR exist for quite a while, there are few examples of multimodal interfaces for AR so far, and there also has been little evaluation of them in general, as stated by Lee and Billinghurst [15].

Some assume that the most intuitive method to instruct a robot would be the way one would instruct a human worker: by simply telling and showing *how* a task has to be accomplished. Since for humans, the most natural means to interact with each other is using speech and gestures, combining them is often considered a valuable approach. A vast field of HRI research deals with exploration of interaction between humans and humanoid robots like Honda's ASIMO or Hiroshi Ishiguro's *Geminoid*[8]. There, researchers try to model human behavior, and to maximize flexibility in communication and social interaction between human and robot, including voice, gestures and mimics. Their HRI methods target multimodal signal fusion to provide a means to interact with machines as naturally, as one would interact with humans. Since this is a highly complicated and extensive topic, the research is still in its infantry. Although flexibility is desirable for operating industrial robots as well, the needs for commanding assistance systems for areas like health care are still very different, since the operational environment is much more random. In this work, HRI is broken down to its very essentials and "natural" interaction in this context does *not* mean to interact with the system on a high level, mimicking human-human communication, but to direct and operate the system in a way that is as intuitive and self-evident as moving a cursor on screen, using a mouse.

[8]see http://www.irc.atr.jp/Geminoid

Chapter 2

Related Work

Milgram et al. [17] state that HRI can benefit from AR in numerous ways. E.g. they mention that visualization limitations can be overcome, and the display can be enhanced with additional, helpful data. Furthermore, the user is able to run and test the program offline, and the process can be planned interactively, enabling for optimizing trajectories, e.g. to avoid potential collisions with obstacles. They also postulate that human-robot collaboration could be significantly improved by clearly separating attributes humans are good at, from those, robots are good at.

In industrial robotics, AR was previously applied for tasks including painting [23] or arc welding [7]. The robot manufacturer *KUKA*[1] investigated the use of AR for robot operator training within the German collaborative research project *MORPHA*[2] and presented the results in the ISMAR 2006 workshop "Industrial Augmented Reality" [2]. In the *KUKA AR Viewer*, information like joint coordinate axes, trajectories and I/O states were embed-

[1]see http://www.kuka.com

[2]see http://www.morpha.de

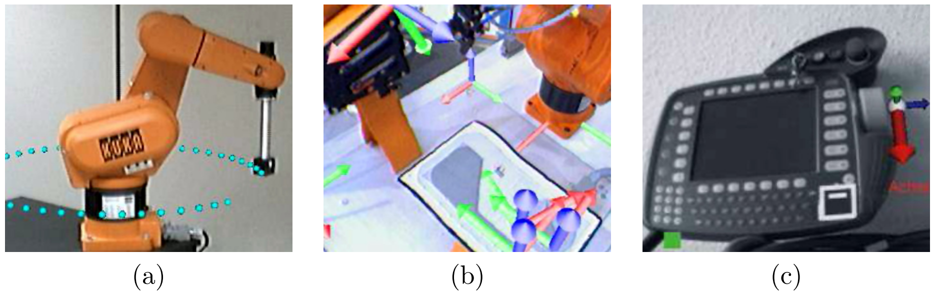
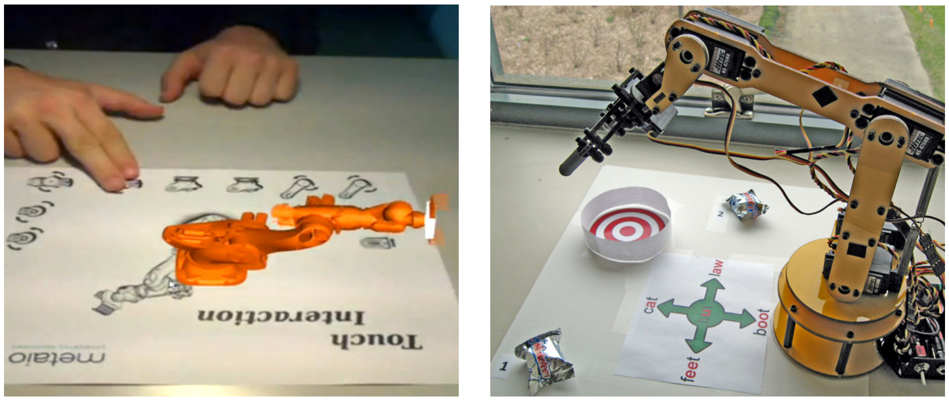


Figure 2.1: The *KUKA AR Viewer* displayed trajectories (a), reference coordinate systems (b), and auxiliary information for handling of the 3D mouse (c).



(a) (b)

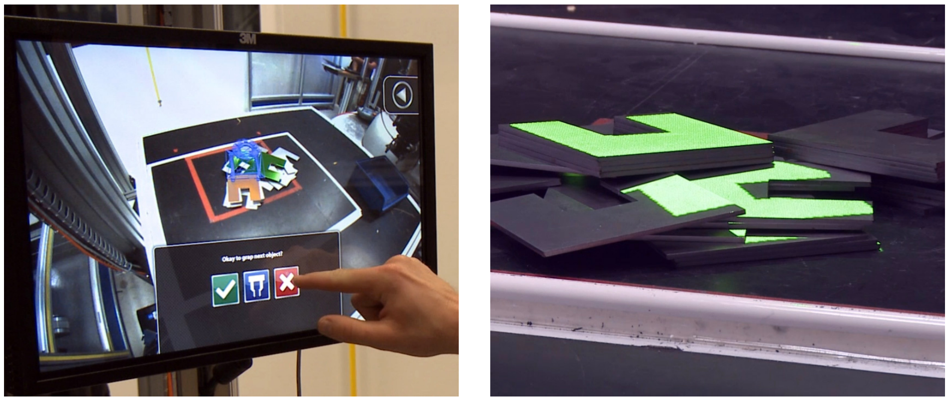
Figure 2.2: An AR-based robot controlling approach by *metaio* allowed for manipulation of joint positions, by occluding marker symbols (a). The *VoiceBot* was controlled by non-verbal voice, using pitch and vowels (b).

ded into a screen-based AR setup to teach students, how to operate robots with a teach pendant, and to make the handling of a 3D mouse more transparent (see Figure 2.1). As a result of their study, the authors state that AR is especially useful for training and to gain awareness of the different reference coordinate systems involved. In contrast, we state that AR and novel interaction techniques should not be utilized only to teach outdated techniques. Instead they should replace them wherever possible.

The German AR technology company *metaio*[3] showed a virtual robot, displayed in screen-based AR[4], whose joints could be rotated individually by occluding AR markers, e.g. with the hand, as shown in Figure 2.2 (a). Although this enabled for controlling a robot without any additional hardware (only requiring the camera used for tracking), the benefit of such a system is questionable. Major problems are expected, since random occlusion of markers as well as faulty marker detection will trigger actions unintentionally. Latter is expected to be an ongoing issue for setups with bad lighting conditions. Furthermore, the same controlling method can simply be accomplished with physical buttons (and in fact it is), more robustly and without any limitations whatsoever.

Kobayashi et al. [13] took advantage of a VST HMD to display robot data in AR, providing helpful information about the robot's knowledge, and presenting it in a human understandable way. Although their work is related to humanoid robots, it is a nice example of applying AR to present abstract information, such as robot strategies, while gaining spatial coherence.

[3]see http://www.metaio.com
[4]see http://www.youtube.com/watch?v=uVeqfpLIdx8&feature=youtu.be&a



(a) (b)

Figure 2.3: *AHUMARI* used both screen-based and projector-based AR to render additional off-surface information (a), while providing in-situ augmentation, directly on the physical objects (b).

The predecessor to this thesis, *AHUMARI*[5] (see Figure 2.3), combined screen-based with projector-based AR to exploit the advantages of both. The goal of *AHUMARI* was to create an autonomic, self-learning system, for picking objects out of a filled box. The user was able to teach grasp actions by leading the robot arm manually to an object and subsequently grab and remove it. A geometric reasoning algorithm then searched for similar areas in the triangulated surface, which was created from a preceding 3D scan. A statically scene-camera setup was used to display an animated vacuum gripper tool, approaching the object next to be picked. Additionally, a steerable projector highlighted the detected areas by different color codes, providing valuable information about detection states and potential problems. *AHUMARI* also successfully demonstrated the benefits of augmenting internal process knowledge and strategies onto real-world objects. However, we found that occlusions are a serious issue, especially in industrial environments where operating robots easily mask the projections, thus finding an adequate position for the projector was hard.

Zaeh and Vogl [29] applied laser projectors for AR, to overcome the depth issue of lens projectors. They used an optical tracked 6 DOF wand to draw trajectories for processing work piece surfaces in scenarios like grinding, polishing and cutting (see Figure 2.4). The authors also used a tablet PC to provide additional information on a VST display, using *ARToolkit*. While their 6 DOF wand is related to our input method, their system is restricted to surface-based tasks and thus does not provide actual 6 DOF input.

[5]see http://mi-lab.org

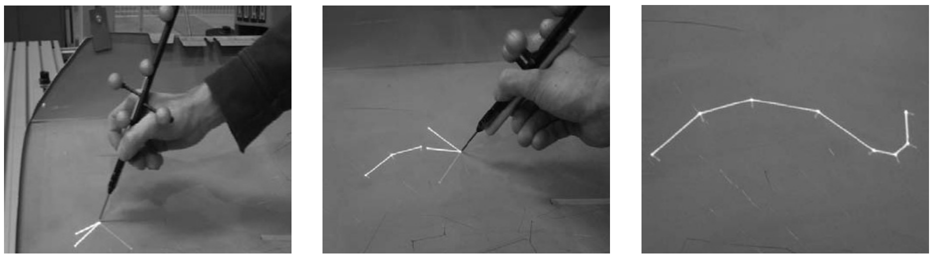


Figure 2.4: The optically tracked stylus was used to draw laser-projected trajectories onto a surface.

House et al. implemented the *VoiceBot* [11], a robotic arm controlled hands-free, only by continuous non-verbal voice (see Figure 2.2 (b)). They evaluated two different kinematic models (forward and inverse kinematics) for controlling the arm using pitch and different vowels. While this interaction technique is highly valuable for individuals with disabilities, it is considered inapplicable to improve controlling of industrial robots since it basically replaces cursor buttons by sounds.

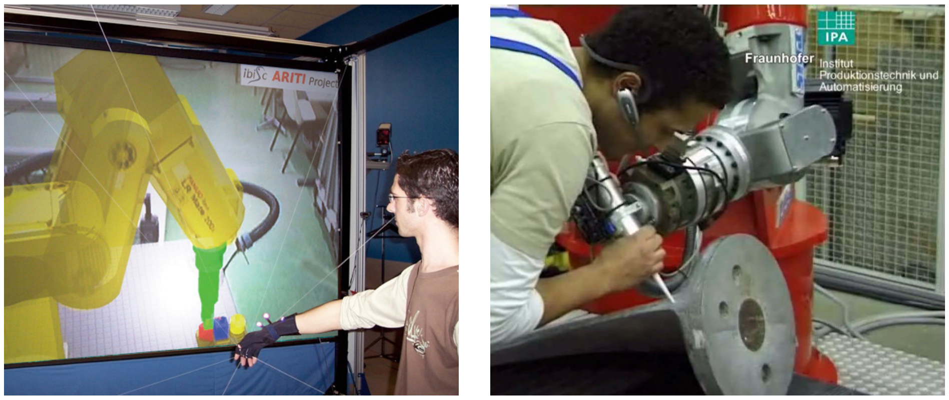
A *WiiMote* hack named *WiiBot*[6] showed an interesting approach for controlling robotic arms. Two engineers applied the motion sensing controller of the video game console *Nintendo Wii* to control a *KUKA* industrial robot. A closer look at the annexed video exposes that they only used gesture recognition to trigger a predefined motion sequence over and over again, so there was no real motion translation from the user's to the robot's arm. Also, the lag of the system was rather high.

Ong et al. [19] used HMDs and *ARToolkit* to control a robotic arm and took safety issues into account. They used marker paddles to quickly define collision volumes, thus obstacles like the workcell or the operator could roughly be provided to the system. Given this information, the robot could quickly be operated in previously unknown environments, avoiding collision volumes.

Boudoin et al. [5] controlled a robotic arm by *equivalent*[7] MMI techniques, that is, different input devices simply replaced each other (see Figure 2.5 (a)). E.g. to simulate a mouse click, a data glove could be used to carry out a close-hand gesture. So basically this just enabled the operator to choose his favorite device for input, the underlying input metaphors—like mouse clicks—persisted. This implies that for every modality there had to be found a correspondent metaphor for each action. We state that it is not always possible to find an adequate and intuitive correspondent; furthermore it is not a reasonable interaction system. Modalities are to be applied only

[6]see http://www.technovelgy.com/ct/Science-Fiction-News.asp?NewsNum=927

[7]according to the CARE-properties, as defined by Salber et al. [25]



(a) (b)

Figure 2.5: The data glove was one of several *equivalent* modalities, used to control the virtual robot arm (a). Force-torque sensors were used for the teaching of this gluing scenario (b).

for roles they are applicable most, moreover they should be combined in a manner they do not replace but complete each other.

Perzanowski et al. [22] built an input system to control mobile robots in a remote setting, combining speech input and hand gestures with touch interaction using a PDA. In contrast to this thesis, the authors focused on deictic input and signal fusion to mimic human-human communication.

Green et al. [9] implemented a multimodal user interface using speech, gestures and gaze-processing for teleoperation of a mobile robot. They used *ARToolkit* paddles to detect basic gestures to control the direction of the robot. For AR-visualization of the scene, they used a VST HMD, which displayed the scene from the perspective of the user. The authors also evaluated this system [8] (omitting gaze-processing) in terms of human-robot collaboration, comparing ego-centric with exo-centric views. They showed that the former enabled for faster task completion time, while latter provided a higher degree of immersion and precision. As a result, the authors state that displaying robot intentions in AR visualization helps reaching common ground between robots and humans.

Schraft and Meyer [26] built a multimodal input system, combining speech input, force-torque sensors and a PDA showing a GUI. They recorded trajectories for replay, while the operator guided the robotic arm and entered speech commands (see Figure 2.5 (b)). In a post-processing step, properties like velocity, position and orientation were smoothed and adopted, to minimize complexity and to cancel out jittering. In contrast to this thesis, which also enables for remote operation, their approach needed the user to manually guide the physical robotic arm via walk-through programming.

Chapter 3

System Design

3.1 Prototype Setups

Adhering to the findings of Chapter 2, a prototype had to be created, to demonstrate intuitive programming of industrial robots, by *reasonably* combining different interaction channels, particularly meeting the needs in programming robotic grasping processes. The program should be able to record keyframes, using a lead-through teaching model. Thus, keyframe poses are not defined in prior, but along the instructions sequentially. To support the user in robot operation, valuable process and robot state information should be presented in a concrete manner. After programming is done, it should be able to simply play back the recorded process on the real robot, or as a virtual simulation, if desired.

User-input can be broken down to three components: (i) guiding of the robotic arm, (ii) triggering actions, such as grasping, and (iii) post-editing of the generated robotic process. Adequate input channels have to be found for those three components, to create a system for robot operation, which achieves the requirements mentioned above. Two prototypes were created, which have been confronted in a user study, one of them representing common input devices (henceforth called *Teach Pendant*), while the other one exhibits a new interaction approach to be proved (henceforth called *MIMIK*[1]). Table 3.1 summarizes the chosen input modalities. The basic setup, as seen in Figure 3.1 (a), included a small robotic arm and a touchscreen display, to operate GUI elements, and for navigation in the rendered VR scene. For safety reasons, handiness and availability, a *Lynxmotion* AL5C hobbyist robotic arm was used (see Figure 3.1 (b)), instead of a real full-size industrial robot. The touchscreen was a 19-inch multitouch LCD panel (*Windows 7 capable*) from *3M*, with a resolution of 1440×900 pixels, detecting up to ten touches simultaneously. In both setups, the GUI was used to perform post-processing

[1]Multimodal Interaction Methodologies for Intuitive operation of Kinematic chains

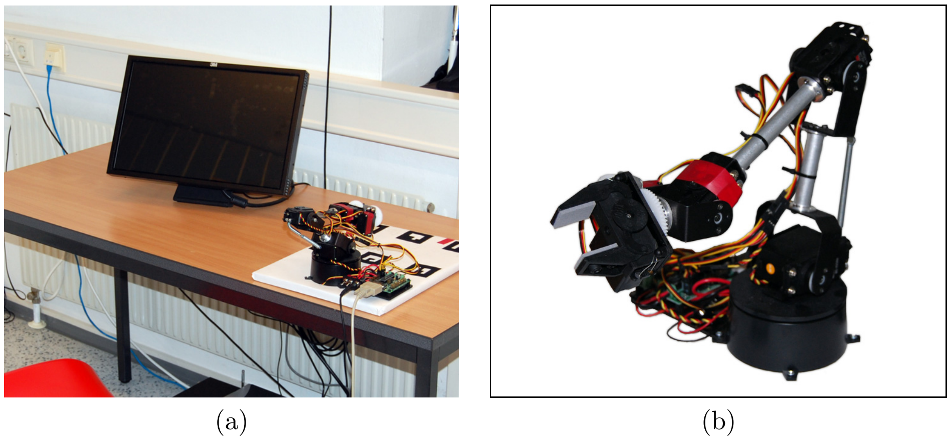


Figure 3.1: The basic setup consisted of a small hobbyist robot and a multitouch capable touchscreen from *3M* (a). The *Lynxmotion AL5C* robotic arm, adjusted with *Heavy-Duty Wrist Rotate* extension, was used for this work (b).

Table 3.1: Chosen input modalities for the two prototypes to be compared. Note that the *Teach Pendant* setup does not actually use a teach pendant, but uses the same

	arm guiding	*action triggering*	*post-editing*
Teach Pendant	3D mouse	touchscreen	touchscreen
MIMIK	3D stick	speech input	speech input + touchscreen

steps, such as deleting or inserting new commands, as well as to start, pause and stop playback or simulation.

3.1.1 *Teach Pendant* Prototype

This prototype was created to simulate status-quo interfaces for evaluation purposes. In this setup, a *3Dconnexion SpaceNavigator* (seen in Figure 3.2) was used to emulate 3D mice, as integrated in common teach pendants. A 3D mouse is an input device similar to a joystick, though it provides fully 6 DOF. 3D mice also are referred to as *3D motion controllers* or *3D navigation devices* and are used in some VR applications as well, e.g. for camera naviga-

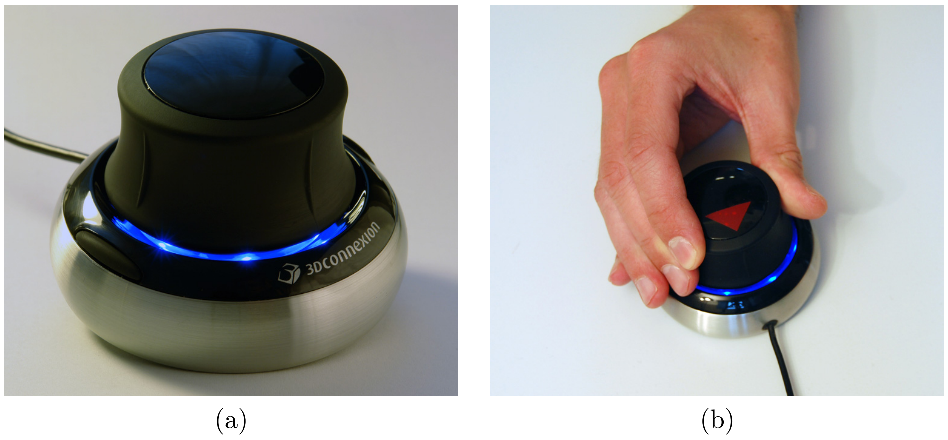


Figure 3.2: A *3Dconnexion SpaceNavigator* 3D mouse was used for *Teach Pendant* (a). (Image taken from http://de.wikipedia.org.) It may be translated and rotated freely and thus provides 6 DOF (b).

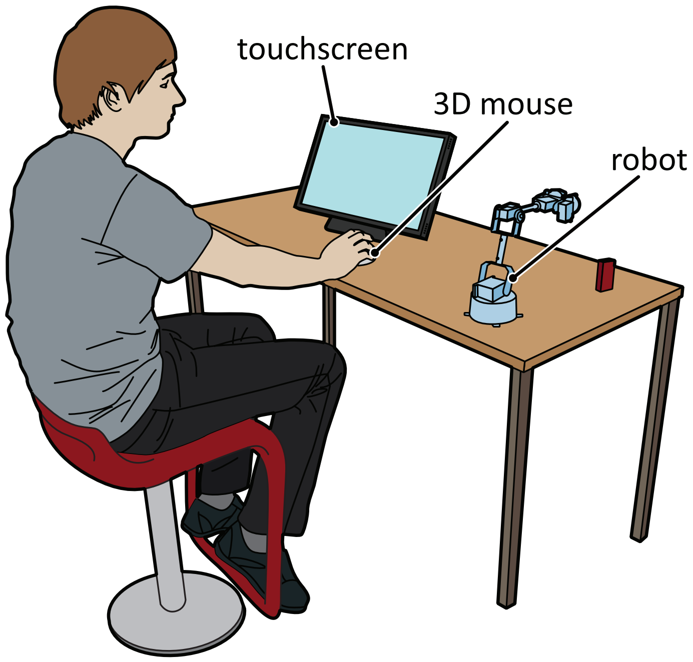


Figure 3.3: The setup, intended to mimic common teach pendants, uses a 3D mouse to manipulate the gripper position, and a touchscreen to provide buttons via a GUI.

tion in *digital content creation* (DCC) tools. In this scenario, the 3D mouse allows for manipulating the pose of the gripper. The touchscreen imitates interaction with physical buttons, by providing a GUI to trigger actions, such as grasping, releasing and setting keyframes. Figure 3.3 illustrates this setup. In terms of visualization, a VR scene is displayed on the screen, showing a model of the robot, keyframes and the tool trajectory.

In coherence with common teaching controllers, *Teach Pendant* offers different motion modes[2]. *Base Motion Mode* and *Tool Motion Mode* specify, whose reference coordinate system the tool is moved in respect to, when the 3D mouse is operated. For definition and further explanation of reference coordinate systems see Section 4.1.2.

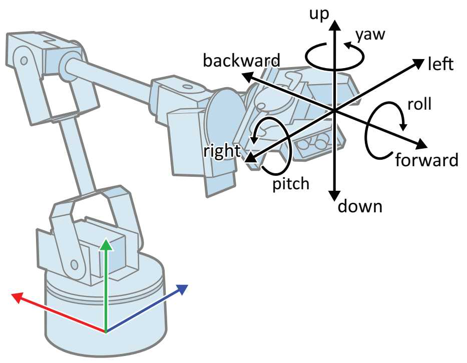
Base Motion Mode: The tool is moved and rotated along base coordinate system axes, no matter how the tool is currently orientated (see Figure 3.4 (a)). E.g. if the user executes an upwards movement by the 3D mouse, the tool is moved along the $\boldsymbol{y}$ axis of the base coordinate system, which would be *global up*. This mode is particularly helpful when the tool is currently rotated, since predicting the effect of 3D mouse motions in these cases is very difficult, especially for beginners.

Tool Motion Mode: The tool is moved and rotated along tool coordinate system axes (see Figure 3.4 (b)). E.g. if the user executes an upwards movement, the tool might not move upwards in a global manner, but along its local up-axis which e.g. may be *global right*, depending on the tool rotation. This mode is advantageous especially if the user tries to move the gripper from an approach pose to the object pose (like P_2 and P_3 in Figure 1.3, respectively), which is often a simple forwards movement with respect to the tool. Instead of having to perform the 3D mouse translation into the exact skew direction in a global manner, the user can activate this mode and move along the tool's local $\boldsymbol{z}$ axis by simply thrusting the 3D mouse forwards and backwards, without risking collisions between gripper and object.

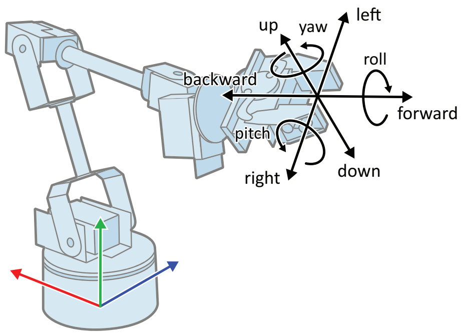
In addition to those two modes, *Constrained Motion Mode* and *Free Motion Mode* specify the presence of motion constraints:

Constrained Motion Mode: In this mode the tool can only be rotated or moved along one single axis. Only the movement *or* rotation with the highest magnitude is performed, all others are eliminated. So if the user wants to perform a backwards movement, e.g. for reaching an approach pose after positioning an object, accidentally sideward movements or rotations can be prevented this way, which in fact are very difficult to

[2]These motion modes are not necessarily present in all available industrial robot controlling systems, also their names and exact implementations might differ. Since usually there are more reference coordinate systems present, professional systems will provide even more modes.

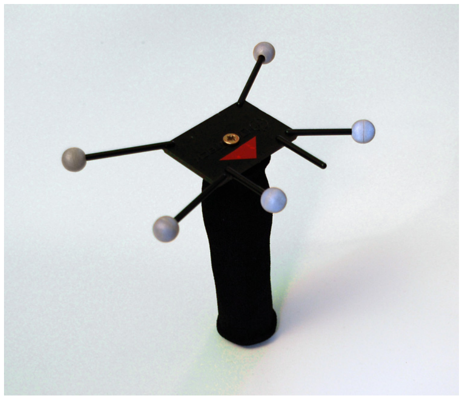


(a)

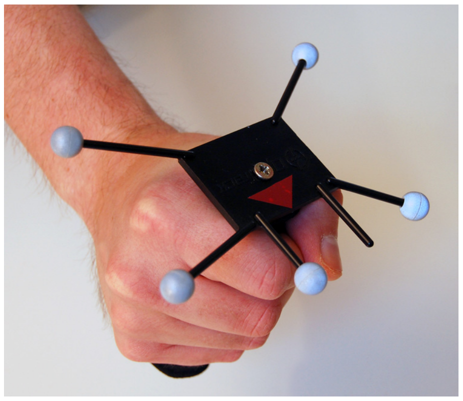


(b)

Figure 3.4: *Base Motion Mode* moves and rotates the gripper with respect to base coordinate system axes (a), while *Tool Motion Mode* uses the axes of the local tool coordinate (b).



(a)



(b)

Figure 3.5: The rigid body, used for *MIMIK* consists of five retroreflective markers, which are attached to a grab handle (a). The enables to utilize the user's arm as an input device for manipulating the pose of the gripper tool(b).

avoid using a 3D mouse. Note that it is not possible to perform skew movements in this mode.

Free Motion Mode: In contrast, this mode allows all movements and rotations to be performed simultaneously, so also skew movements are possible. Furthermore, this mode is time saving for traversing large distances.

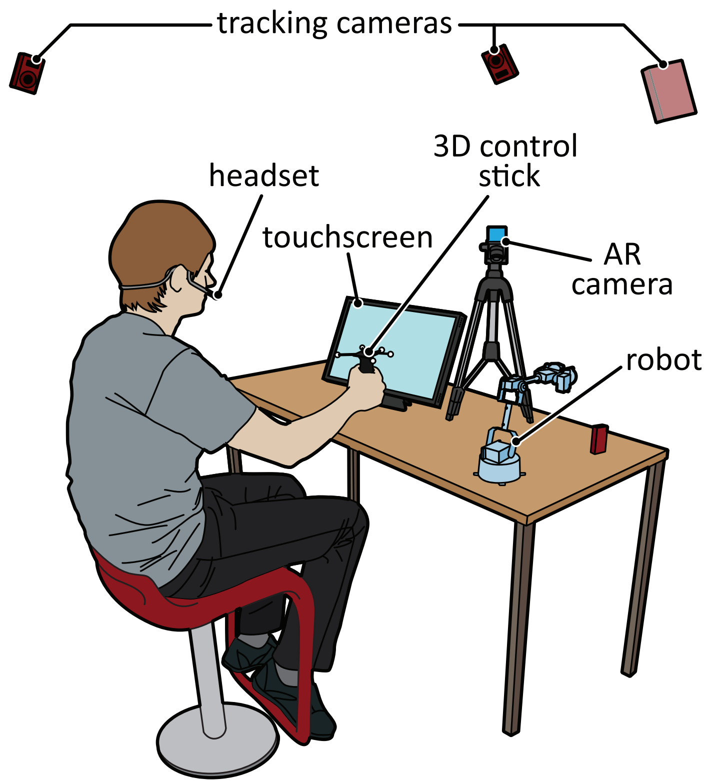


Figure 3.6: The proposed setup consists of the robot, a camera for AR, a multitouch capable touchscreen, and the tracking cameras, used for 6 DOF tracking of the hand-held stick. The operator is wearing a headset to enter speech commands.

3.1.2 *MIMIK* Prototype

The interaction design of *MIMIK* was intended to meet the requirements of being intuitive and natural in the first place, while high precision demands were not considered. It should reduce the mental load during operation, shorten training periods and allow for rapid process teaching. The chosen input modalities were 6 DOF tracking, speech input, touch and multitouch interaction. An optically tracked hand-held rigid body, henceforth called *3D stick*, enables for tracking the pose of the operator's hand (see Figure 3.5). The robotic arm mimics the user's arm movements in an interactive rate. Wearing a headset, the user may trigger actions via speech commands during operation. Additionally, speech commands can also be used during post-editing. Figure 3.6 illustrates this setup. For visualization, the user can switch between AR and VR. In AR-mode, keyframe poses can be displayed, maintaining spatial coherence and providing visual reference to the actual objects. E.g. trajectories can easily be checked, and programming errors can be recognized at a glance. Since a statically screen-based AR setup was chosen for reasons already mentioned in Section 1.2.1, a pure VR scene view providing control about the virtual camera complements the lack of mobility, so the user is able to view the scene from an arbitrarily perspective. To navigate in this virtual scene, touch interaction is considered valuable.

- *Dragging with one finger* rotates the camera around the scene center.
- *Double finger press and drag* pans the camera left, right, up or down.
- *Two finger pinch* moves the camera forward or backward.

In contrast to the system developed by Zaeh and Vogl [29], where the tool of a virtual robot followed an optically tracked stylus on a VST display, *MIMIK* enables to control the actual physical robot. Whenever the robot freezes due to inaccessible target poses, the virtual model tries to follow this pose, while violated joint constraints are rendered, to indicate the kinematic limitations.

Another important feature of this implementation is that only *relative* movements of the operator's arm are translated to the robotic arm. If the user enables the robot, it does not move until the operator starts moving. If *absolute* poses would be adopted, the robot tool would jump to its target pose immediately, as soon as it is activated. Since the user's hand is often at some random pose when the robot is activated, this happens in a hitch of the robotic arm, and is therefore undesirable. Instead, the pose of the user's hand at activation time should be handled as a *reference pose* for tracking, so all successive movements of the robotic arm are also done with respect to the initial gripper pose. Since the robot returns to its default pose as soon as tracking is disabled, this pose was chosen as a reference. This correlation will be illustrated in more detail in Sections 4.3 and 4.4 when all necessary symbols are defined. Another possibility would be to not reset the robot joint configuration on tracking deactivation, so the user would be able to pause tracking, find a new reference pose for the 3D stick and resume tracking again, to continue the gripper movement at this point. This way it would not be that crucial to start with a convenient 3D stick pose, yielding enough space for hand movement. This option was disclaimed, since it was not considered crucial, and this way no additional command for resetting the robot joint configuration was required.

3.2 Interaction Design

3.2.1 Speech Recognition

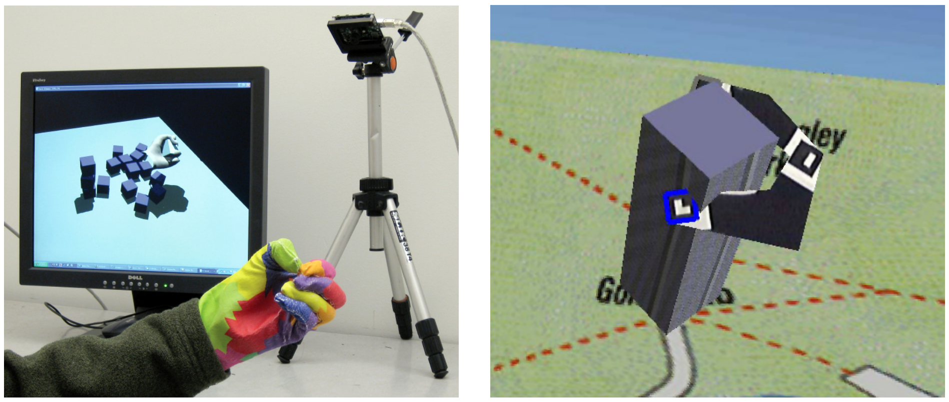
Speech Recognition (SR) and *Text to Speech* (TTS) are often considered some of the most promising interfaces for HCI, maybe because it is also the most natural way for humans to communicate with each other. There are two categories of SR: *Dictation* and *Command and Control* (C&C) mode. While dictation tries to process arbitrary natural human speech, which is hard to accomplish, C&C can rely on a small, task specific command set that is defined in advance (or may grow dynamically). Using natural speech interfaces, it also is a very hard task to construe information, even if the spoken text was delivered accurately by the speech recognizer. As the name suggests, dictation mode is mainly useful for dictating arbitrary text, as for setting up text documents.

Human speech is poorly applicable to specify poses or trajectories in 3D space, since it is hard to verbally describe positions, not to mention orientations. However, it seems to be highly beneficial for performing repetitive actions. Since the command set required for controlling an industrial robot is comparatively small, it is not necessary to provide a complex interface like natural speech. Aside the fact that it would have been way too complicated for this work, Lee and Billinghurst [15] found that humans don't even want to talk to an artificial system in the conversational way they communicate with other humans. They also stated that not having a fixed command set made some users initially frustrated because they had no idea how to instruct the system properly. Hence, C&C speech input is sufficient for triggering commands like "insert keyframe", "grab" and "release" in a very natural way, without requiring carrying an extra physical input device. Only providing few commands however, it would also be possible to use buttons, attached to a hand-held device, but as soon as the command list grows, limits are reached quickly. Additionally, speech input can be expanded to a very powerful and versatile interface, e.g. enabling to assemble several commands into one single macro, supplied with an arbitrarily, user defined identifier like "move current object to bin". Macros could also contain other macros, so a powerful command set could be built quickly by the users themselves. For clarifications, those macros could easily be integrated in the HUD and even be edited like usual program sequences, after all they are nothing else than program fragments. Although considered, the addition of macros was not implemented for *MIMIK*, due to time constraints.

3.2.2 Six Degree of Freedom Tracking

In 1995, Milgram et al. [17] suggest that a feasible option for specifying three-dimensional locations would be, to type the Cartesian coordinates x, y and z into a keyboard. Given the technical opportunities of that time this may have been an adequate choice, but now it seems to be one of the worst options imaginable, especially for teaching robotic processes. However, given a known environment and precise CAD data, specifying the exact positions works well for offline programming.

With *MIMIK*, the user should be able to steer the robotic arm without



(a) (b)

Figure 3.7: A glove with a custom color pattern was used to track the hand pose from a single position (a). *ARToolkit* markers, attached to hand and fingers enabled for grabbing and manipulating virtual 3D content (b).

having knowledge about particular joint or reference coordinate systems. Due to its advantages in navigation in 3D space, 6 DOF tracking was considered a good choice for guiding the robot tool from a remote position in a very intuitive way. Also, robotic and human arms are not too different in terms of kinematic appearance, so it seemed to be a good choice to simply utilize the users arm as an “input device”. The robotic arm should adhere to and imitate the motions of the operator at an interactive rate. So, for the manipulation tool pose, there is no abstraction necessary, since the tool acts as a representation of the user’s hand. In a 2D space this can be compared to the mouse cursor which is a representation of the physical mouse or even of the user’s hand. Ignoring the orientation in 3D space, the pose gained from 6 DOF tracking could be seen as a three-dimensional mouse cursor. Note that the joint positions of the robotic arm do not necessarily match those of the human arm. In most cases this would not even be possible, since the proportions of the robotic arm segments differ from the human ones.

The hand tracking technique proposed by Wang and Popović [28] could be used to get hand position and orientation for positioning the gripper. They proposed vision-based tracking of the user’s hand, wearing a cloth glove, colored with a custom pattern, to gain information about hand position and finger pose (see Figure 3.7 (a)). A single camera retrieved 3D position of the hand, as well as finger poses, so users may manipulate objects in a virtual 3D environment with their fingers. For *MIMIK*, the distance between fingers and thumb could be translated to the gripper opening. To avoid damage on both workpiece and tool, Force-Sensing Resistors (FSR) may be assembled onto the gripper. Since human ability to manipulate objects relies heavily on the contact information gathered, it is not expected to act naturally to grab objects lacking haptic feedback though. One way to overcome this issue is demonstrated by a project named “Dexterous Haptic Interface for Interaction with Remote/Virtual Environments”, where haptic feedback is given by a vibrotactile glove with miniature voice coils embedded, to produce vibrations on the fingertips[3]. Combined with FSR, an idea of the actual force can be given by altering the vibration intensity accordingly. Extending these works, gesture recognition could be used to trigger grab and release commands as shown by Buchmann et al. [6]. They introduced an interaction method enabling for fingertip-based grabbing and manipulation of virtual objects in an urban planning scenario (see Figure 3.7 (b)). Using *ARToolkit* markers attached to a glove, they track position and posture of the user's hand, and perform gesture recognition based on that data. Additionally, they provide haptic feedback with buzzers, which are mounted to the fingertips.

However, speech interaction provides more flexibility since spoken commands like “grab” and “release” can also be used in different applications. E.g. if one replaces the claw gripper by a vacuum gripper, or even by a welding head, the analogy of the human hand is no longer feasible.

A more common way to gain the pose of an object in 3D space was used for *MIMIK*. In rigid body tracking, the environment is scanned for a known constellation of markers, e.g. by installing cameras and processing the captured image data[4]. With knowledge about the configuration of the markers, the pose of the rigid body with respect to the camera can be calculated. Since the object is able to move along three perpendicular axes and may rotate around them as well, it provides six degrees of freedom: the Cartesian coordinates x, y and z, and the *Tait-Bryan angles* ψ, θ and ϕ.

6 DOF tracking is sometimes lumped together with gesture recognition, meaning gestures like pointing or waving are recognized and interpreted. This is not the case in *MIMIK*. However, a gesture-based interaction technique falls into one of two categories: (i) Variations of point-and-click paradigm and (ii) application-specific pose and motion gestures. According to this separation, *MIMIK* is of the prior category since 6 DOF tracking corresponds to *point* and speech commands correspond to *click*.

3.2.3 Touch Interaction

Touch interaction is not considered valuable for controlling 3D objects[5] since three dimensions have to be mapped to a two dimensional surface somehow.

[3]This project is currently active at the *Robotics Institute* (http://www.ri.cmu.edu), which is part of the *School of Computer Science* at the University of Carnegie Mellon.

[4]Other methods would be magnetic, mechanical, inertial or acoustical tracking.

[5]This, of course, depends on the action to be performed—there surely are aspects of DCC like modeling, texturing, etc. that work well on flat surfaces. In this case controlling the gripper pose is meant which is difficult to handle in 2D.

Therefore, some abstractions have to be introduced, which first have to be adopted by the user.

As a result, touch interaction was mainly applied for operating the GUI, e.g. for playback, modifying display settings and post-processing purposes. So, the user is able to delete keyframes and instructions, and to insert movements towards already defined poses, using the touchscreen GUI.

Additionally, touch interaction was considered sufficient for 3D navigation in the VR scene. Reisman et al. [24] proposed multitouch navigation in three-dimensional space, where three fingers were used for rotation about a custom axis. Although these motions were also possible using a single hand, the authors claimed that bimanual interaction was more comfortable. In contrast, in our approach only one hand is required for each of the navigation actions, so the touch display can be replaced by a hand-held device if necessary.

3.2.4 Multimodality

Multimodal interaction is sometimes assumed to necessarily involve deictic commands. Note that *MIMIK* does not, since no cross modality references are present. As a result, there also is no need for signal fusion. According to the already mentioned CARE-properties, defined by Salber et al. [25], all modalities in this work are *assigned*. E.g. 6 DOF tracking is used to guide the robotic arm, while speech commands in fact simply act as voice-buttons, triggering actions that are completely independent of the state of other input channels.

3.2.5 Screen-Based Augmented Reality

For operation itself, as well as for inspection purposes during simulation, AR seems to be the best choice for visualization by nature, since it enables for merging virtual contents with the real world. Even though the visualization could benefit significantly by advanced techniques like stylized *Focus and Context* (F+C) rendering, such as shown by Kalkofen et al. [12], simple flat shaded geometry is displayed in this work, since priorities for the user study are set on evaluating the input methods.

Chapter 4

Implementation

4.1 General

4.1.1 Notation Conventions

Matrix Notation

In this thesis, matrices are denoted by upper case characters. Vectors are written in bold lower case characters. Components of vectors and matrices are written in lower case characters. As an example, $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ are the axes of the basis of a transformation matrix M, while $\boldsymbol{t}$ is a vector, representing its translational part. $x_{\boldsymbol{x}}$, $y_{\boldsymbol{x}}$ and $z_{\boldsymbol{x}}$ are the components of the axis $\boldsymbol{x}$ and m_{32} is the element on row 3 and column 2 of the matrix M.

In contrast to the widely-adopted postmultiplication notation of *OpenGL*, in *OpenSceneGraph* `Matrix` class operators use a premultiplication style (see [16], section 2.3.3.). Transforming a point $\boldsymbol{p}$ by a matrix M, resulting in a point $\boldsymbol{p}'$ is written as

$$\boldsymbol{p}' = \boldsymbol{p} \cdot M \tag{4.1}$$

instead of

$$\boldsymbol{p}' = M \cdot \boldsymbol{p}. \tag{4.2}$$

This also means that vectors are notated as row vectors instead of column vectors, thus

$$\boldsymbol{v}' = \begin{pmatrix} x & y & z \end{pmatrix}. \tag{4.3}$$

Consequently, transforming a matrix M to M' by a translational matrix T and a subsequent rotational matrix R is written as

$$M' = R \cdot T \cdot M \tag{4.4}$$

instead of

$$M' = M \cdot T \cdot R. \tag{4.5}$$

This notation is adopted in this thesis to stay consistent with the program code.

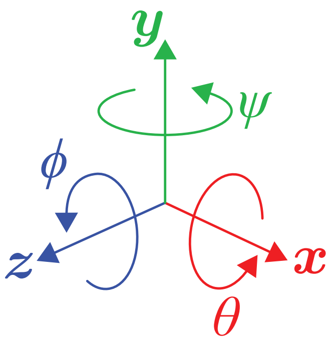


Figure 4.1: The *Tait-Bryan angles* ψ, θ and ϕ denote rotations around the basis axes $\boldsymbol{y}$, $\boldsymbol{x}$ and $\boldsymbol{z}$, respectively.

Tait-Bryan Angles

While different conventions about ψ (yaw), θ (pitch) and ϕ (roll) exist, depending on the coordinate system in charge, in this work they denote rotations around the axes $\boldsymbol{y}$, $\boldsymbol{x}$ and $\boldsymbol{z}$, respectively (see Figure 4.1).

Rotation Conventions

Since different orders of applying rotations result in different transformation matrices, this order has to be noted. This is done by specifying rotation conventions, such as XYZ or ZXY. E.g., a rotation convention of XZY denotes rotations around $\boldsymbol{x}$, $\boldsymbol{z}$ and $\boldsymbol{y}$ in this order, so the resulting rotation matrix R would be

$$R = R_Y(\psi) \cdot R_Z(\phi) \cdot R_X(\theta), \tag{4.6}$$

where $R_Y(\psi)$, $R_Z(\phi)$ and $R_X(\theta)$ are matrices, performing rotations of ψ, ϕ and θ around $\boldsymbol{y}$, $\boldsymbol{z}$ and $\boldsymbol{x}$, respectively.

4.1.2 Kinematic Chains

A robotic arm, also called *manipulator*, is described by a *Kinematic Chain*, which is a set of arm segments connected to each other by joints. The end of the kinematic chain is called the *end effector* or *Tool Center Point* (TCP). Although all joints in the case of this work are revolute joints, there are other types as well, like prismatic joints, cylindrical joints and spherical joints. The state of a single joint is called the *joint position*, even if it describes a rotational state. The combination of all joint positions of a manipulator, hence the overall description of its adjustment, is called the *joint configuration*.

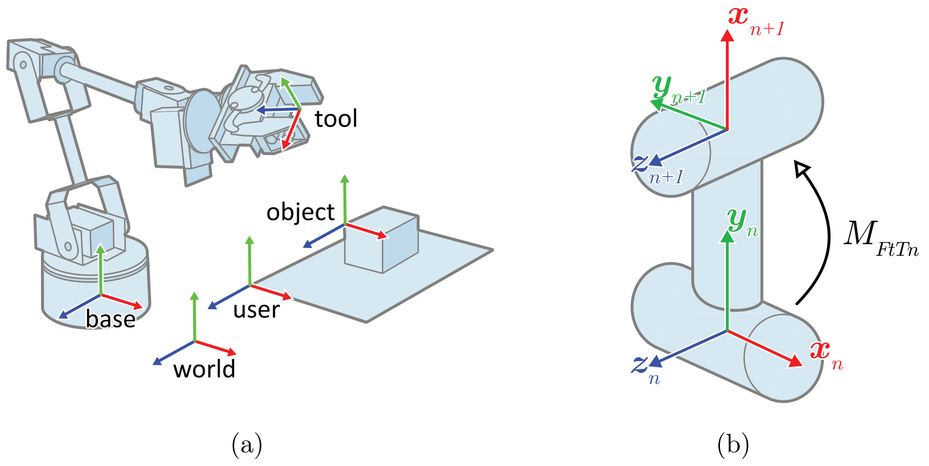


Figure 4.2: For operator convenience, there usually are several reference frames (a). However, in this work base frame and tool frame were adequate. The *Frame-to-Tip* transformation M_{FtTn} represents the translation and rotation of the tip of a segment S_n, with respect to its joint frame F_n (b).

Usually, there are several reference coordinate systems (called *frames*) involved. Figure 4.2 (a) shows some examples:

World frame: global reference frame.

Base frame or robot frame: reference frame of the robot. May be specified relatively to the world frame.

Tool frame or TCP frame: reference frame of the end effector. May be specified relatively to the base frame or the world frame.

User frame: reference frame of workbench. May be specified relatively to the world frame.

Object frame: reference frame of workpiece. May be specified relatively to the user frame or the world frame.

In some instances, the user frame is equated to the *base frame*—number and naming conventions of reference frames are not specified exactly, and may differ according to manufacturer and application. To simplify matters, in this work only two of them are used: the robot frame, and the end effector frame, henceforth called *base frame*, with the basis

$$B_0 = \begin{pmatrix} \boldsymbol{x}_0^T & \boldsymbol{y}_0^T & \boldsymbol{z}_0^T \end{pmatrix}, \tag{4.7}$$

and *TCP frame*, with the basis

$$B_{TCP} = \begin{pmatrix} \boldsymbol{x}_{TCP}^T & \boldsymbol{y}_{TCP}^T & \boldsymbol{z}_{TCP}^T \end{pmatrix}, \tag{4.8}$$

respectively. The TCP frame is specified with respect to the base frame, all other frames (world, tool, etc.) are dispensable in this work and thus equated with the base frame.

The pose of the tip of each segment with respect to its joint frame is called the *Frame-to-Tip* (FtT) transformation. In the segment depicted in Figure 4.2 (b) the Frame-to-Tip transformation M_{FtTn}, transforming from frame F_n to F_{n+1} would be a translation along $\boldsymbol{y}$, followed by a rotation of $90°$ around $\boldsymbol{z}$.

4.1.3 Libraries and Tools

Mayor libraries used for this work are *OpenSceneGraph*[1] (version 2.8.0) for rendering, *osgART*[2] (version 2.0 RC3, wrapping *ARToolkit*[3] version 2.72.1) for AR tracking, *Orocos KDL*[4] for robot kinematic computations, *NaturalPoint OptiTrack TrackingTools*[5] (version 2.1.0) for optical rigid body tracking and *Microsoft Speech API*[6] for speech recognition. The code was entirely written in *C++* using *Microsoft Visual Studio 2008* (version 3.5, SP1) and the program was executed in *Windows 7* (version 6.1).

4.2 Speech Recognition

For speech recognition (SR) *Windows Speech API* (SAPI) 5.4 was used. Instead of the shared speech recognizer[7], in this work an *InProc* recognizer is created, so the application has exclusive control over the associated SR engine. Context-free grammar (CFG) may either be defined in code or via XML-files—in this work the latter was used to create C&C grammar. The XML-files may first be compiled into binary CFG-format using the command line grammar compiler *gc.exe*.

There are several possibilities to retrieve information about recognition events (such as *sound start*, *sound end*, *recognition*, *hypothesis* and *interference*). One could implement `ISpNotifySink` for free-threaded notification, or `ISpNotifyCallback` for notification within the thread that initialized the

[1]open source graphics rendering library, based on *OpenGL*, licensed under LGPL, see http://www.openscenegraph.org

[2]open source wrapper of *ARToolkit* for *OpenSceneGraph*, licensed under GNU GPL, see http://www.osgart.org

[3]widely used open source optical tracking library, licensed under GNU GPL, see http://www.hitl.washington.edu/artoolkit

[4]open source library for computation of kinematic chains, licensed under LGPL, see http://www.orocos.org/kdl

[5]proprietary rigid body tracking SDK, see http://www.naturalpoint.com/optitrack/products/tracking-tools/

[6]shipped as a part of the *Windows SDK* (version 5.4)

[7]The shared recognizer comes with its own GUI and the associated SR engine runs in its own process. The shared SR engine can be accessed by all applications, which is convenient especially for desktop applications.

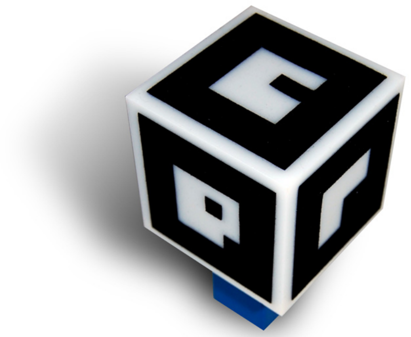


Figure 4.3: Primary, a *ARToolkit* marker cube was used for 6 DOF tracking.

event source, i.e. the recognition context, or the application could just poll for recognition events. Latter was used in this work, for simplicity. When an event is triggered, SAPI provides extensive data along the recognition result, one of them is the SR engine confidence parameter, which is a floating point value in the range of $[0, 1]$ and can be used to filter out low-confidence recognitions. For the English SR engine a threshold of 0.85 was found to be adequate.

Although the speech engine got trained by the speaker and the sound was rather clear (there was hardly any noise and the volume was adjusted properly, so there was no clipping), there were serious recognition problems, seemingly randomly. After switching from the German to the English SR engine, improvements were noticeable, but still there were a lot of false detections, even with high confidence values of about 0.9. Extending the phrase length of some short commands (from "grab" and "release" to "grab object" and "release object", respectively) helped more or less, but still did not lead to a robust and reliable speech recognition, which would be suitable for the user study. Maybe finding individual thresholds for each command, by investigating a longer period of usage, could still lead to improvements. Due to lack of time, no more effort was spent on improving speech recognition performance.

4.3 Six Degree of Freedom Tracking

The first attempt to gain 6 DOF tracking information was done using *ARToolkit*. Since the template markers have to face to the capturing camera to be recognized, a cube as pictured in Figure 4.3 was built, which had different markers on each side so at least one would always be visible sufficiently. Unfortunately, there were several problems due to poor tracking robustness that could not be solved, although highly light absorbing velour foil was used for the markers instead of paper printouts. Clearly, *ARToolkit* relies on good lighting conditions which cannot always be kept. Low contrast easily causes the marker detection to fail (see Figure 4.4). Also, adjusting the aperture of the camera lens accordingly, could not completely avoid this issue. However, the major problems were motion blur and occlusions. Capturing at a framerate of 30 fps, the motion blur was often too high for the system to detect markers in the image—even for moderate movements. The problem with occlusions was even worse, since objects or shadows that overlap the marker in the camera image intercept the contour of the marker, and this causes the edge detection algorithm not to find the marker anymore. Since the whole work rests on a robust tracking system, *ARToolkit* was replaced by *OptiTrack* which performed excellently.

NaturalPoint OptiTrack products are multi-camera motion capturing and tracking systems, using passive retroreflective spherical markers, and proprietary software and hardware. Figure 4.5 (a) illustrates a setup with six cameras, aligned in a circle, so occlusions are avoided at any time, and also high precision is achieved. *OptiTrack FLEX:V100R2* cameras (see Figure 4.5 (b)) exhibit FOVs of 46° with a focal length of 4.5 mm. They provide an operation range of 15 cm to 7 m and a minimum shutter time of 20 us. The LED rings around the lenses emit infrared (IR) light and accordingly, grayscale images are captured through 800nm pass lens filters. Due to the high resolution and framerate of 640×480 pixels (VGA) at 100 fps, and the high depth of field, the tracking is highly robust, and there are no such problems like motion blur.

For *MIMIK*, a setup with three cameras was used, which was easily calibrated within a few minutes, via the included tool. To get image data for debugging purposes, the operation mode of the cameras can be switched to MJPEG, but for tracking they deliver already preprocessed data. For this purpose, thresholding, marker detection and marker processing is done onboard. For high precision demands, there is an operation mode, which calculates marker centroids with subpixel accuracy using grayscale pixel values instead of binary values. Camera parameters like *IR exposure*, *IR intensity* and *threshold* can be set remotely by the program using API calls.

TrackingTools (TT) is an API for 6 DOF rigid body tracking and is built upon the *OptiTrack API*, which is a rather low-level interface for 2D marker tracking in raw camera frames. A rigid body (also called *Trackable*) in TT consists of a special arrangement of several *OptiTrack* markers, as in the 3D stick used for *MIMIK* (see Figure 3.5 (a)). The markers attached to the trackable base are distributed in an asymmetrical manner, so the orientation

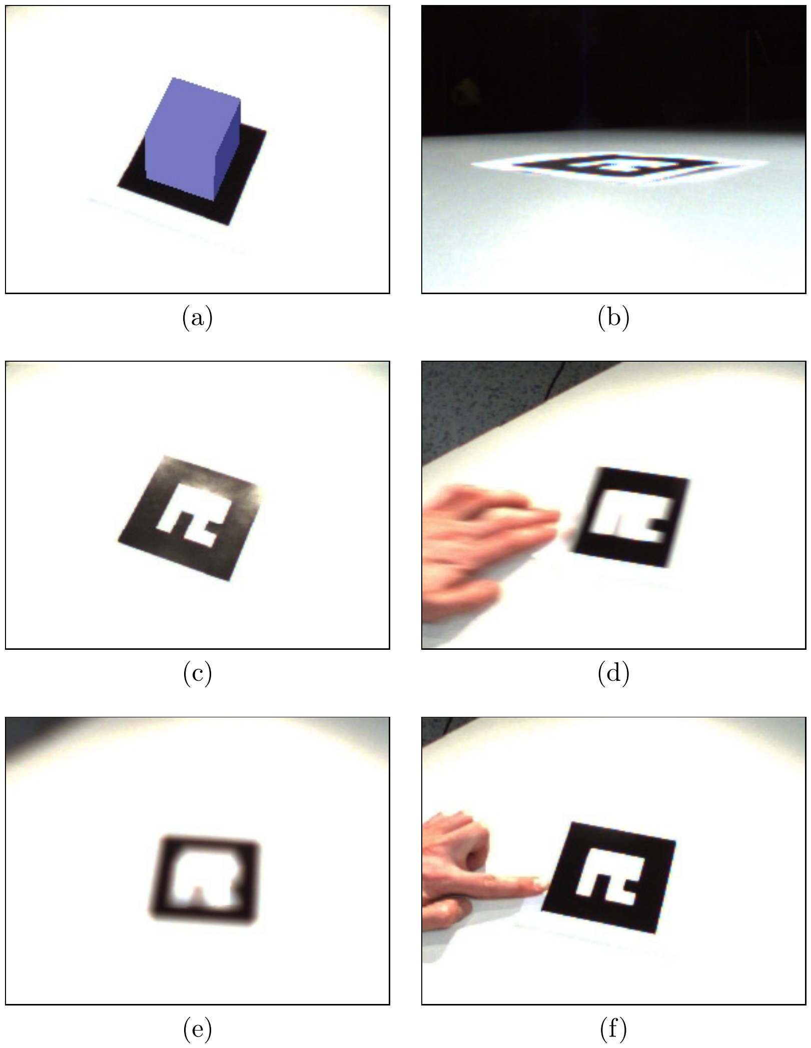
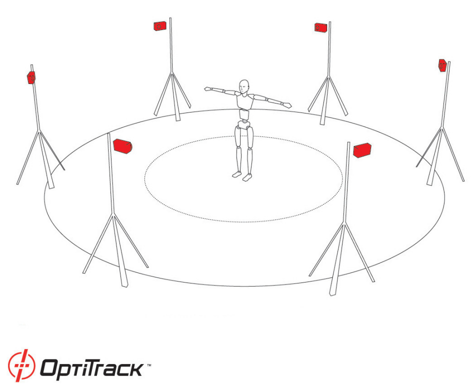


Figure 4.4: This example application renders a cube on top of a detected marker (a). *ARTookit* fails detecting the marker in the camera image in several cases, e.g. if the marker is highly distorted (b), if there are highlights (c), if there is motion blur (d), if it is out of focus (e) or if its contour is intercepted by other objects (f).

is more distinct and the coordinate system can hardly flip over per accident[8]. Figure 4.6 (b) shows the trackable as rendered in a TT VR scene. Note that its pivot point, and thus the transformation of the rigid body maintained from the TT API, is *not* located at the centroid of its marker cloud. It is set to $\begin{pmatrix} 0 & -0.06 & 0 \end{pmatrix}$, where values are specified in meters and were determined

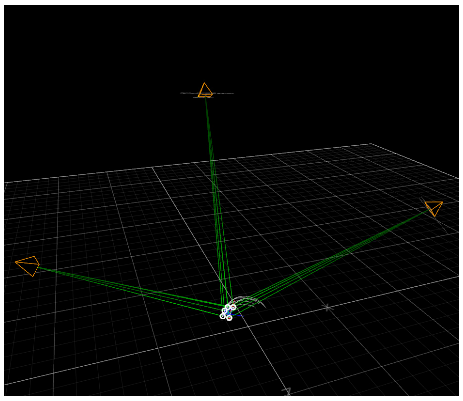
[8]This is in fact a problem *ARToolkit* suffers from, due to the square shape of the markers.



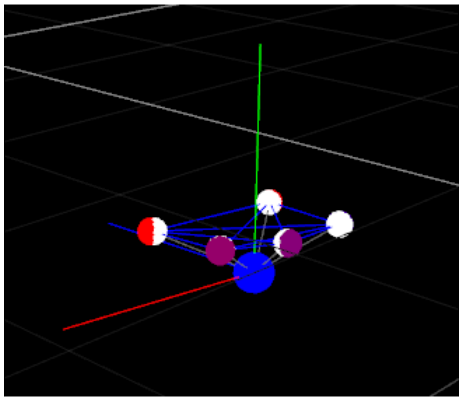
(a)

(b)

Figure 4.5: *OptiTrack* uses multiple IR light emitting cameras (b), which are distributed in the setup, thus occlusions are avoided (b). (Images taken from http://www.naturalpoint.com/optitrack.)



(a)



(b)

Figure 4.6: The setup used for *MIMIK* consisted of three tracking cameras mounted on the ceiling, arranged left, right and in front of the user (a). *TrackingTools* detects the pose of the rigid body by looking for its known constellation of markers, tolerating some adjustable deflection (b). The red spheres represent the actual calculated marker positions, while the white ones display the rigid body, matched into this marker cloud.

empirically, so a rotation of the wrist will result in a rotation around $\boldsymbol{z}_T$ (see Figure 4.7) and can easily be mapped to a rotation of the robot gripper tool.

If the rigid body is temporarily lost by the tracking system, its position and orientation gets extrapolated, based on the motion history. Figure

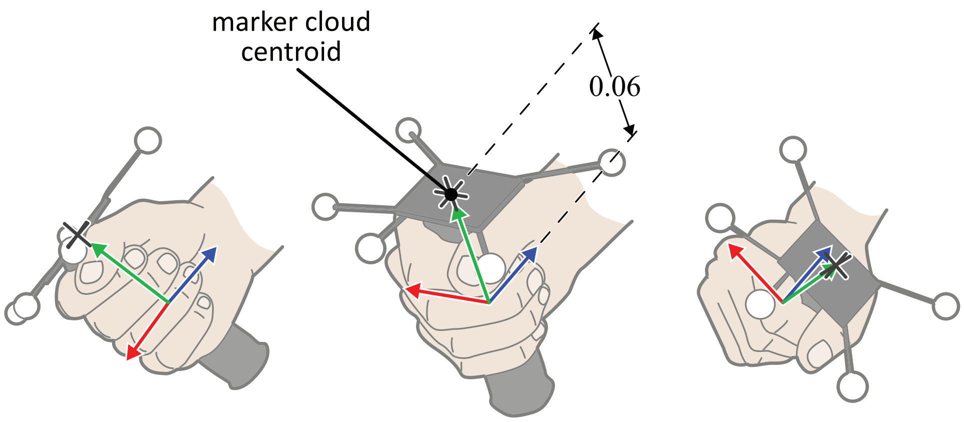
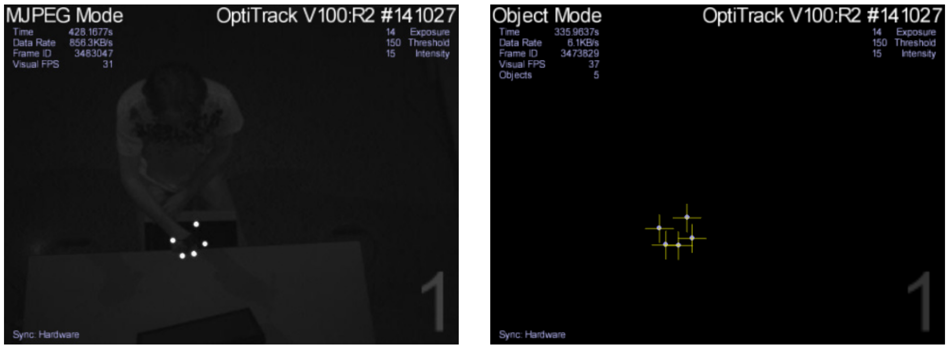


Figure 4.7: The centroid of the rigid body is set below the centroid of the marker cloud, so a wrist rotation performs a rigid body rotation around its z axis.



```
MJPEG Mode                         OptiTrack V100:R2 #141027
 Time         428.1677s                          14   Exposure
 Data Rate    856.3KB/s                          150  Threshold
 Frame ID       3483047                          15   Intensity
 Visual FPS          31
 Sync: Hardware
```

```
Object Mode                        OptiTrack V100:R2 #141027
 Time         335.9637s                          14   Exposure
 Data Rate      6.1KB/s                          150  Threshold
 Frame ID       3473829                          15   Intensity
 Visual FPS          37
 Objects              5
 Sync: Hardware
```

(a) (b)

Figure 4.8: *OptiTrack* cameras can be operated in IR mode (a). The pre-processed, thresholded camera image already contains marker positions with subpixel accuracy (b).

4.8 (b) shows pre-processed data of one of the three cameras, Figure 4.6 (a) depicts the resulting pose of the trackable in a virtual 3D scene. The TT SDK contains a tool to quickly set up such a trackable, including specifying pivot point and several useful convenience settings like *Max Marker Deflection*, *Min Marker Count*, *Min Hit Count*, translational and rotational smoothing settings, and dynamic or static constraints for translation and rotation. Camera calibration and trackables, as well as the whole TT project, containing camera settings, can be saved into files. Those can be loaded by the applications using TT API calls.

At first glance, dynamic rotation constraints seem to be useful, e.g. to limit the rotation around $\boldsymbol{z}_T$ to the range of $[-90°, 90°]$. However, this introduces problems, since all hand movements are evaluated relatively to the pose at tracking activation time. E.g. if the trackable is rotated with $\phi_T = 30°$ at tracking activation, it would have to be rotated to $\phi_T = 120°$ to perform a tool-rotation to $\phi_{TCP} = 90°$. This would be prevented by the TT API, since no trackable would be detected, due to violation of rotation constraints. So constraints are not useful in this case.

The current six DOF, x, y, z, ψ, θ and ϕ, can be achieved each frame after calling an update function of the TT API. For calculating the tracking transformation matrix M_T from these values, care has to be taken due to two reasons. Firstly, TT uses a left-handed coordinate system, so z and ϕ have to be flipped. Secondly, TT uses different conventions for the *naming of pitch and roll*. They represent rotations around the $\boldsymbol{z}$ and $\boldsymbol{x}$ axes, respectively, instead of the other way around, like in most applications. Since TT uses XZY rotation convention, the pose of the trackable, with respect to the TT origin, is calculated as

$$M_T = R_Y(\psi) \cdot R_Z(-\phi) \cdot R_X(\theta) \cdot T, \tag{4.9}$$

where $R_Y(\psi)$, $R_Z(-\phi)$ and $R_X(\theta)$ are matrices, performing rotations of ψ, $-\phi$ and θ around $\boldsymbol{y}$, $\boldsymbol{z}$ and $\boldsymbol{x}$, respectively, and T is a matrix translating about the vector $\begin{pmatrix} x & y & -z \end{pmatrix}$.

As mentioned in Section 3.1.2, tracking transformations are applied relatively, thus with respect to M_{T0}, which is the trackable transformation at tracking activation time. The user may chose an arbitrary starting pose for his hand and the robotic arm adapts only relative movements, with respect to this reference pose (see Figure 4.9). Given the tracking matrix M_T, the relative transformation of the trackable with respect to M_{T0} is calculated as

$$M_H = M_T \cdot M_{T0}^{-1}. \tag{4.10}$$

Due to different physical dimensions of the human and robot arm, M_T was applied in a scaled coordinate system. Thus, the mapping of human arm motion to robot arm motion feels more natural. E.g., note the pan to the left, drafted in Figure 4.10, which is basically a shoulder rotation of 30°. It would result in a much more extensive motion of the robotic arm when transferred unscaled, due to its smaller dimension. Given the dimensions of the *Lynxmotion AL5C*, and assuming an adult operator, a scaling factor of 0.6 was found to be sufficient. So the scaled relative transformation is calculated as

$$M'_H = S^{-1} \cdot M_T \cdot M_{T0}^{-1} \cdot S \tag{4.11}$$

which represents the transformation matrix of the TCP destination, with respect to its reference pose M_{def}. This reference pose was chosen to be the

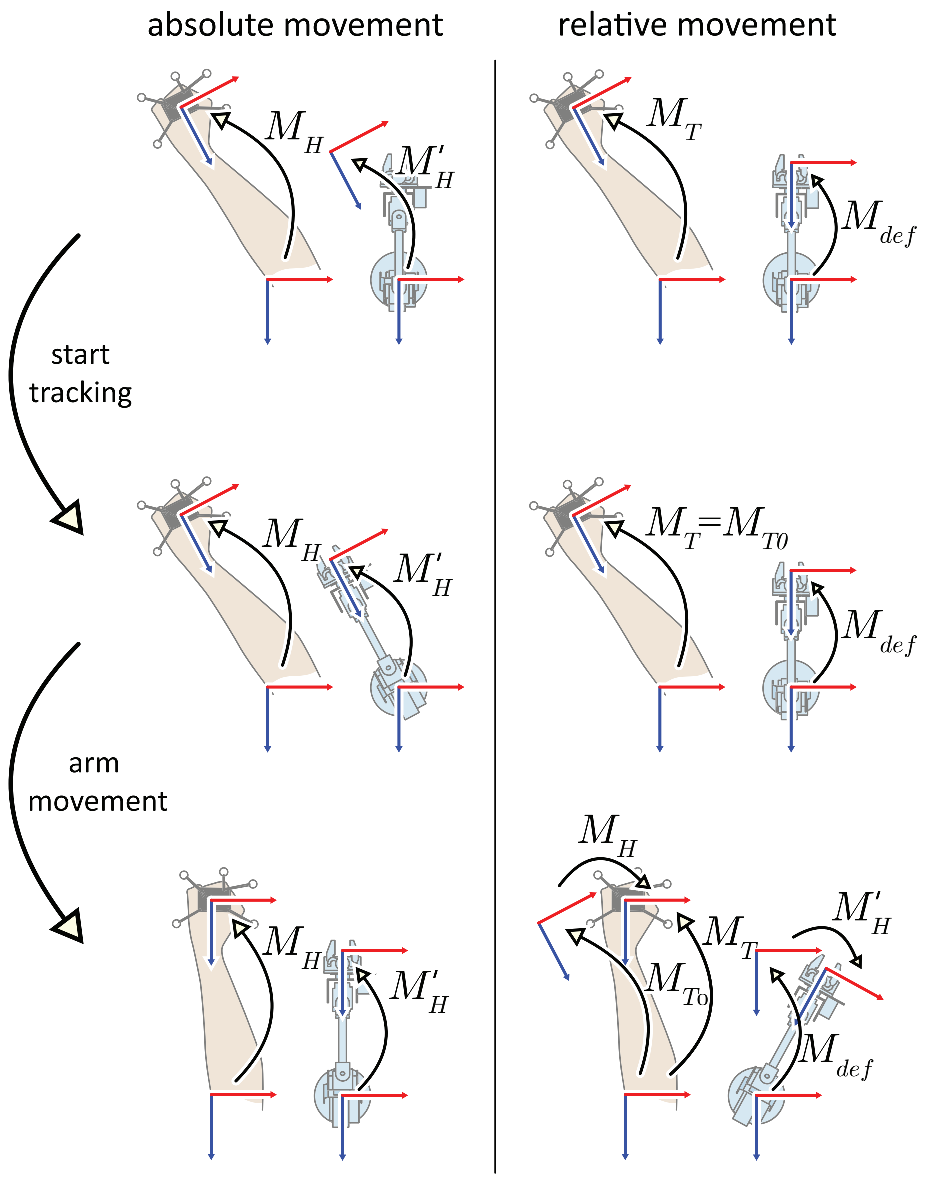
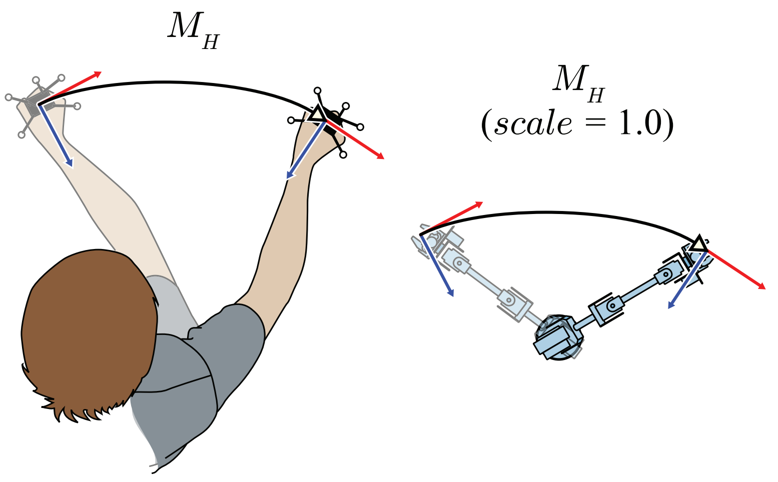
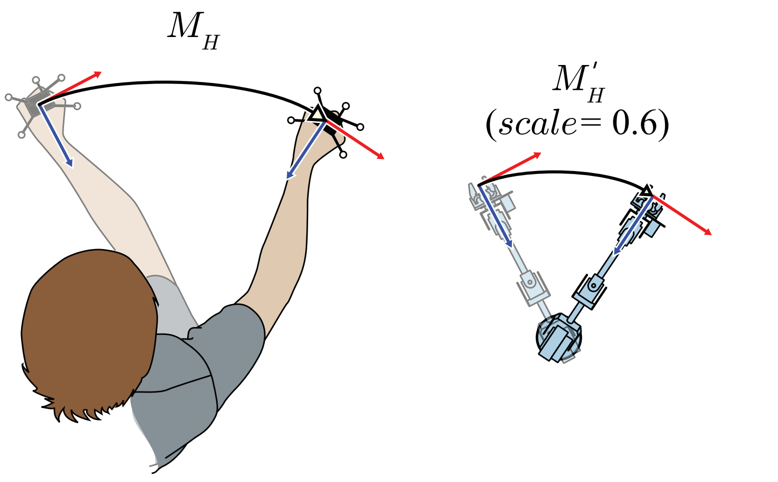


Figure 4.9: If absolute movements would be used, the robotic arm would instantaneously jump to its target position as soon as tracking is enabled (left column). To apply relative movements, a reference pose M_{T0} is introduced, which is set, when the user starts tracking. All subsequent movements are with respect to this pose. Accordingly, the tool reference pose is its default pose M_{def}, and movements are applied with respect to it (right column). Note that the user may choose an arbitrary starting pose in this case, since the arm movement is no longer dependent on the global coordinate system.



(a)



(b)

Figure 4.10: To compensate for the different arm lengths of user and robot, the transformation adopted by the robot is scaled first. Without scaling, the robotic arm would perform a more extensive movement, compared to the user (a). By using an appropriate scaling factor, this can be prevented (b).

robot default pose and will be defined in Section 4.4. M_H' is used later on for inverse kinematics calculations.

4.4 Kinematics

The *Lynxmotion AL5C* incorporates six servo motors with ranges of about $\pm 90^\circ$, whereas five of them are used for joint rotations, and one for opening and closing the gripper. The motors are actuated by the *Lynxmotion SSC-32* servo controller card, which enables for bidirectional communication with a PC via RS-232 (COMM) serial port, so the servos can be controlled remotely by sending ASCII string commands. In those one has to specify servo channel number, servo target position (the range is mapped to values between 500 and 2500) and servo travel time. An example for a command, moving servo 5 to position 1500 within 1000 ms would look like this:

```
\#5P1500T1000
```

Since there are kinematic constraints, due to the physical properties of the robotic arm, not all joints are able to perform rotations of the full servo ranges. E.g., in the assembly at hand, the base joint is able to rotate in the full range of about $[-95°, 86°]$, while the shoulder joint is physically constrained to $[-46°, 90°]$ by the nature of the robotic arm[9]. For opening and closing the gripper, the appropriate servo positions were hardcoded, since all objects used for the study are of the same thickness. However, there are force-sensing resistors (FSRs) available for the *AL5C*, which could be assembled to the gripper claws to read back the amount of pressure, and to stop closing the gripper if some threshold is exceeded.

The kinematic chain of the *AL5C* is shown in Figure 4.11, where measurements were taken manually, using the physical robot. For convenience reasons the frame of the base rotation joint is assumed to be identical with the robot root frame, while the base joint rotation of the real robotic arm happens in fact at $\begin{pmatrix}0 & 0.045 & 0\end{pmatrix}$, where values are specified in meters. In the kinematic model, however, the base rotation is assumed to be applied at $\begin{pmatrix}0 & 0 & 0\end{pmatrix}$, and the FtT transformation of segment S_1 is expanded accordingly to $\begin{pmatrix}0 & 0.067 & 0\end{pmatrix}$. The FtT transformations are just translations, with one exception: The last transformation additionally applies a rotation, so the gripper faces along the $-\boldsymbol{z}_{TCP}$ axis. This convention was introduced for better coherence with the 6 DOF tracking part, where $\boldsymbol{z}_H$ points backwards (see Figure 4.12). Also note that the coordinate system in this work differs from the one usually used in robotics, to be consistent to other libraries (e.g. *OpenSceneGraph*). As common in computer graphics, $\boldsymbol{y}$ represents the up-axis, instead of $\boldsymbol{z}$ which is more common in some other domains.

To retrieve the pose of the TCP from a given joint configuration, an operation called *Forward Kinematics* (FK)[10] has to be applied. The joint

[9] The actual ranges were measured manually.

[10] More precisely, this has to be called *Forward Position Kinematics* (FPK), in contrast to *Forward Velocity Kinematics* and *Forward Force Kinematics* (FFK).

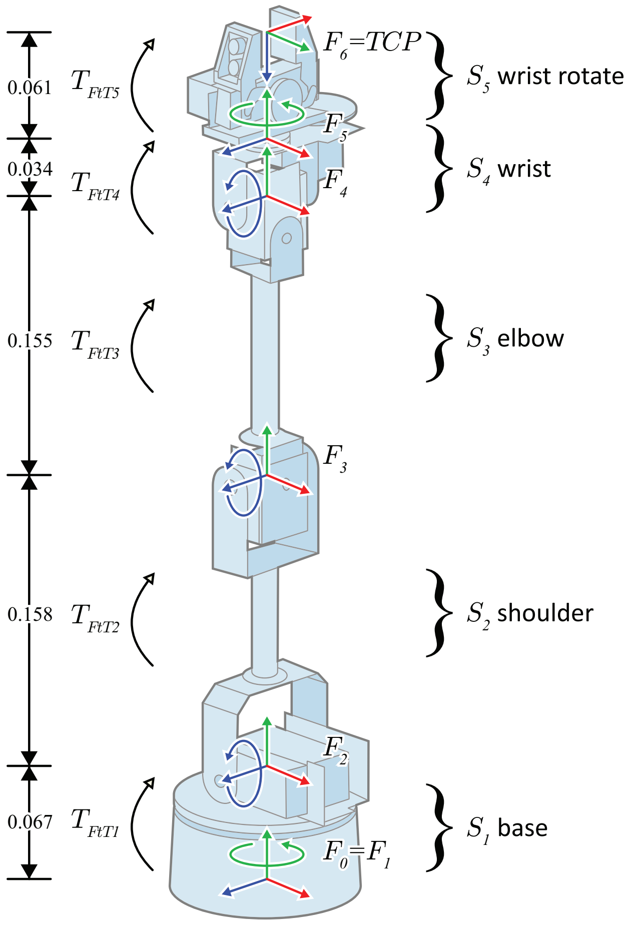


Figure 4.11: The kinematic chain of the *AL5C* is composed of five segments and only provides revolute joints. Note that this pose forms a singularity, since the joints of S_1 and S_5 are aligned. The dimensions are specified in meters.

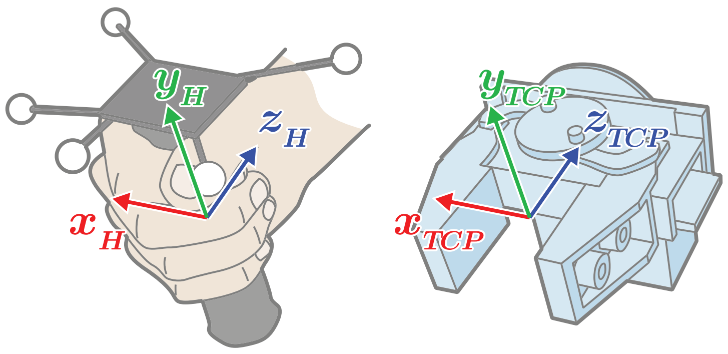


Figure 4.12: The bases of both rigid body and TCP are defined with $\boldsymbol{y}$ being up, and $\boldsymbol{z}$ being backwards.

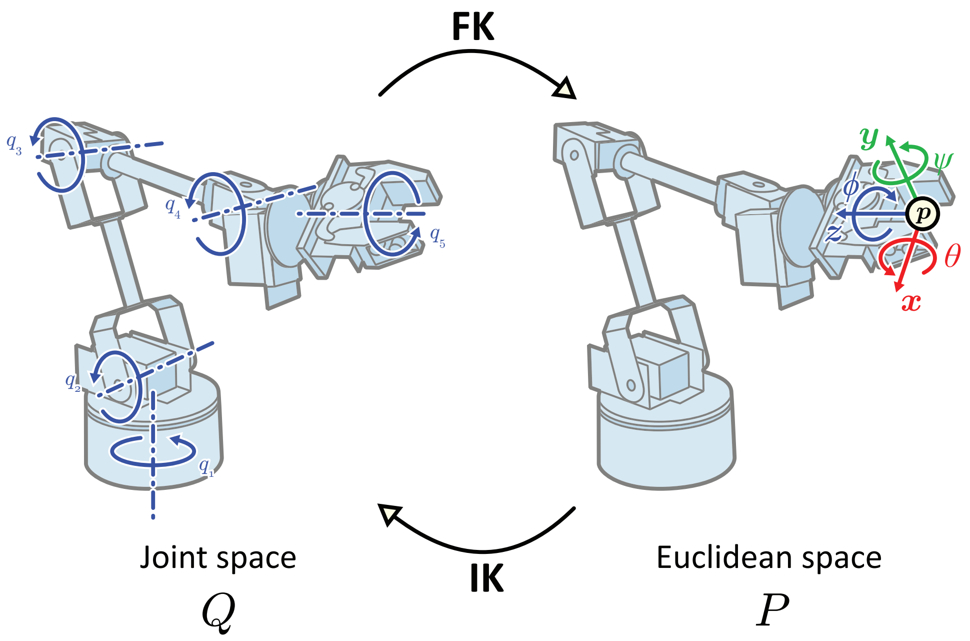


Figure 4.13: IK is the inverse operation of FK. While FK transforms from joint space Q into Euclidean space P, IK transforms from Q into P. Note that Q only offers 5 DOF in this case, since the *AL5C* features five joints.

configuration of a robotic arm with n DOF[11] can be represented as a vector $\boldsymbol{q}$ in an n-dimensional space Q, called the *Joint Space*,

$$\boldsymbol{q} = \begin{pmatrix} q_1 & \dots & q_n \end{pmatrix}, \tag{4.12}$$

whereas each element q_i in this vector represents the position of one joint, respectively. On the other hand, a pose $\boldsymbol{p}$ in 3D *Euclidean Space* P can be described as the combination of its coordinates and rotations. While there are several possibilities to describe positions and rotations in 3D space, the Cartesian coordinates x, y and z as well as the Tait-Bryan angles ψ, θ and ϕ are used here for illustration. Hence, the pose in Euclidean space is a combination of the values x, y, z, ψ, θ, and ϕ, resulting in a vector in 6 dimensional space,

$$\boldsymbol{p} = \begin{pmatrix} x & y & z & \psi & \theta & \phi \end{pmatrix}. \tag{4.13}$$

So, formally speaking, FK can be described as the transformation from joint space Q to Euclidean space P (see Figure 4.13). We can see that this operation is rather simple, since it just requires to start with the robot frame,

[11] In robotics the degrees of freedom simply refers to the number of joints, so depending on the configuration of a kinematic chain, if a robot exhibits a 6 DOF arm, this does not mean that its end effector is able to reach each pose within the volume of reaching space.

and apply both the transformation resulting from the current joint position (only rotations in this case), as well as the FtT transformation of each arm segment successively, until the TCP is achieved. For the *AL5C*, every subsequent frame of B_i of a kinematic chain with n DOF can easily be calculated as

$$F_{i+1} = M_{FtTi} \cdot R(q_i) \cdot F_i, \tag{4.14}$$

where F_1 equals the base frame, $R(q_i)$ represents a rotation of q_i around the according rotation axis of the joint of arm segment S_i, and M_{FtTi} is its FtT transformation. So, the TCP frame can be found as

$$F_{TCP} = F_{n+1}, \tag{4.15}$$

since the end effector is the end of the kinematic chain. Its transformation with respect to the base frame can be represented as a single matrix by multiplying all transformation matrices, which results in

$$M_{TCP} = M_{FtTn} \cdot R(q_n) \cdot M_{FtTn-1} \cdot R(q_{n-1}) \cdots M_{FtT1} \cdot R(q_1). \tag{4.16}$$

The opposite transformation from P to Q is called *Inverse Kinematics* (IK)[12] and has been studied for many decades. In contrast to FK, the IK problem unfortunately has to deal with several difficulties and is computationally expensive. Given a kinematic chain and a pose to be achieved by the end effector, the solution is not always unique. In the example, shown in Figure 4.14 (a), there obviously are two solutions for joint configurations to reach the TCP pose. There also may be poses that cannot be reached at all. The fact that in many (if not most) instances there are joint constraints to be maintained, makes computation even more difficult. There are two approaches to solve IK problems analytically:

Geometric: The solution is calculated by analyzing the kinematic chain geometrically and using trigonometry accordingly. This is very cumbersome, even for very simple manipulators, and an approach has to be set up specifically for the kinematic chain at hand.

Algebraic: According to Kucuk and Bingul [14], the algebraic approach is more beneficial for manipulators with more than two joints, and whose arms can extend into three dimensions. However, not all mathematical solutions may result in the end effector achieving the desired positions[13], so they have to be checked via FK, so invalid ones can be dropped.

[12] Again, this is more precisely called *Inverse Position Kinematics* (IPK), in contrast to *Inverse Velocity Kinematics* (IVK) and *Inverse Force Kinematics* (IVK).

[13] An example for reasons for this is the "±" introduced, when solving an quadratic equation. Mathematically, both results are correct, but physically only one is.

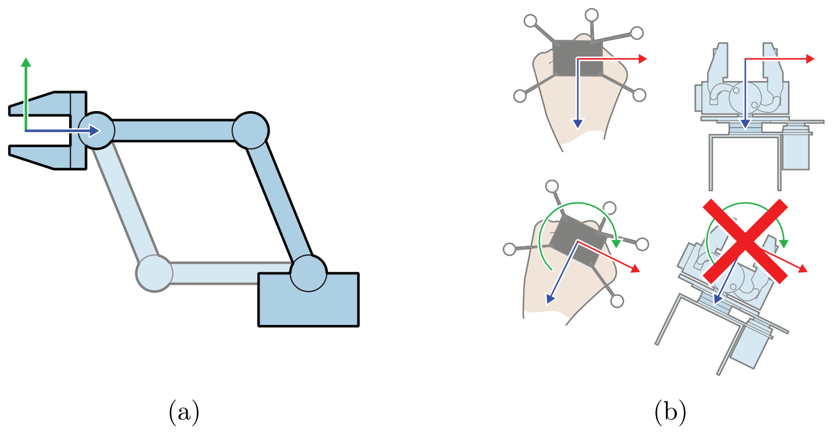


Figure 4.14: IK often yields multiple solutions since a TCP pose may be achieved by more than one joint configuration (a). Lacking one degree of freedom, the *AL5C* cannot perform rotations around the $\boldsymbol{y}_{TCP}$ axis (b).

Mathematical solutions may not correspond to physical solutions, since there are no joint constraints in the mathematical model. In this case, depending on the requirements of calculation speed and accuracy, a numerical approximation method, which tries to minimize an error function, could be sufficient. For IPK an iterative solver from the *Orocos Kinematics and Dynamics Library* (KDL) was used, which adheres to joint limits. According to the Orocos KDL API reference [20], it implements a solver, which is based on *Newton-Raphson* iterations. There are some issues to consider when dealing with iterative solvers, so it is able to find a solution. First of all, the start configuration has to be chosen carefully. E.g. if there are no rotations applied at all, the solver cannot find a solution, since the robot is in a singular configuration[14] because two rotation axes (in this case $\boldsymbol{y}_1$ and $\boldsymbol{y}_5$) are collinear aligned. Also, the start configuration cannot be chosen randomly. If the IK solution is too far from the current configuration in joint space, the solver might run into a local minimum and get stuck, especially when joint limits are involved. Since *MIMIK* does not head for high precision, ϵ can be raised[15] and the maximum iteration count n_{it} can be dropped to speed up the IK solving process, so an interactive rate can be maintained. In this work $n_{it} = 150$ and $\epsilon = 0$ were used for the IVK solver, while $n_{it} = 300$ and $\epsilon = 10^{-3}$ were used for the IVP solver. For some computational problems within KDL (e.g. due to singularities) that seriously hit performance, it was necessary, to modify KDL and check for infinite (`1.#INF` or `-1.#INF`)

[14]In robotics, singularities are poses that can be reached by an infinitely number of joint configurations. During operation, singularities can lead to unpredictable motion.

[15]in calculus, ϵ stands for the error

and indeterminate `-1.#IND` numbers in intermediate results, and cancel the process if they were detected.

To map the hand motions of the user to the robot gripper, the transformation gained from 6 DOF tracking, as described in Section 4.3, is used to calculate the target pose for the IK solver. To do this, the TCP pose M'_H with respect to the robot reference pose is used, as defined in Equation 4.11. Since the robot default pose M_{def} was chosen to be the reference pose, the target pose M_{dest} can be found as

$$M_{dest} = M'_H \cdot M_{def}, \tag{4.17}$$

where M_{def} is calculated by FK, with Equation 4.16, using a default joint configuration defined as

$$\boldsymbol{q}_{def} = \begin{pmatrix} 0 & -45 & 145 & 0 & 0 \end{pmatrix}, \tag{4.18}$$

where joint positions are specified in degrees.

However, the *AL5C* only provides 5 DOF, so in addition to the limited working envelope[16], there are further restrictions in agility. In contrast, the user may move the 3D stick freely in space. Consider the situation depicted in Figure 4.14 (b), where the trackable gets rotated around the local $\boldsymbol{y}_T$ axis. The arm lacks the ability to rotate the TCP around the $\boldsymbol{y}_{TCP}$ axis accordingly. Thus, some modification has to be applied to the transformation gained from 6 DOF tracking, to enable the solver to find a solution at all.

It is conspicuous that only poses can be reached by the *AL5C*, which "face from" the robot's up-axis, $\boldsymbol{y}_0$. The extensions of their $\boldsymbol{z}_{TCP}$ axes cross the $\boldsymbol{y}_0$ axis, as can be seen in 4.15. So an approach to prepare the input transformation for the IK solver is twisting the frame accordingly. To simplify this, it is assumed that the base frame F_0 equals the global coordinate system; hence it is the identity matrix I.

Since rotating the target frame M_{dest} around $\boldsymbol{y}_{dest}$ would result in a completely different orientation of the target pose, which differs substantially from the orientation of the user's hand pose, the rotation has to be performed around a rotation vector parallel to $\boldsymbol{y}_0$. To accomplish this, the translation of M_{dest} is first eliminated, so its basis is located at the base frame origin. Subsequently, the rotation is applied, and M_{dest} is translated back to its prior position. As can be seen in Figure 4.16, the rotation angle α is found between the projections of $\boldsymbol{z}_{dest}$ and the distance vector

$$\boldsymbol{d} = \boldsymbol{t}_0 - \boldsymbol{t}_{dest} = -\boldsymbol{t}_{dest},$$

[16]In robotics, the *working envelope* or *work area* is the volume that can be achieved by the TCP.

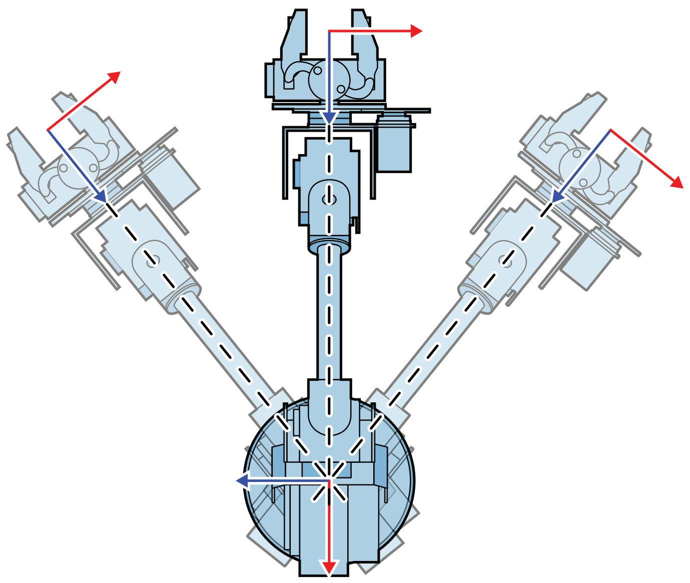


Figure 4.15: Since the *AL5C* lacks a joint for according TCP rotations, IK can only find solutions for TCP poses, whose $\boldsymbol{z}_{dest}$ axis cross the up-axis of the base joint, $\boldsymbol{y}_0$.

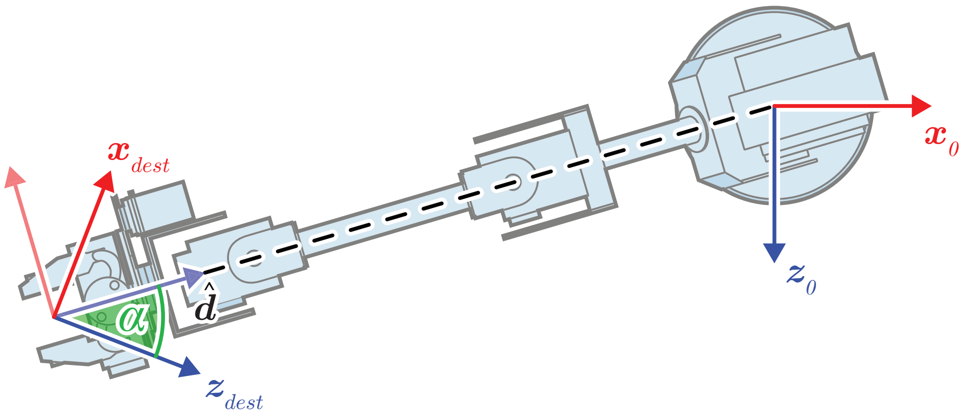


Figure 4.16: To compensate for the lacking revolute joint, the target pose has to be twisted in a way, so the $\boldsymbol{z}_{dest}$ axis point towards the $\boldsymbol{y}_0$ axis. The according angle α can be found between the $\boldsymbol{z}$ axis of the target pose, thus $\boldsymbol{z}_{dest}$, and the distance vector $\boldsymbol{d}$.

onto the XZ-plane, where $\boldsymbol{t}_0$ and $\boldsymbol{t}_{dest}$ are the translational parts of F_0 and M_{dest}, respectively. Since F_0 is assumed to be I, $\boldsymbol{t}_0$ equals $\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$. To omit y-values, they are set to 0, so two auxiliary vectors

$$\boldsymbol{z}'_{dest} = \begin{pmatrix} x_{\boldsymbol{z}_{dest}} & 0 & z_{\boldsymbol{z}_{dest}} \end{pmatrix} \tag{4.19}$$

and

$$\boldsymbol{d}' = \begin{pmatrix} x_{\boldsymbol{d}} & 0 & z_{\boldsymbol{d}} \end{pmatrix} \tag{4.20}$$

are defined, thus $\boldsymbol{z}_{dest}$ and $\boldsymbol{d}$, respectively, projected onto the XZ-plane. Using cross and dot product, sine and cosine of α can be calculated with

$$\boldsymbol{n} = \hat{\boldsymbol{z}}'_{dest} \times \hat{\boldsymbol{d}}' \tag{4.21}$$

as another auxiliary vector, so

$$\cos\alpha = \hat{\boldsymbol{z}}'_{dest} \cdot \hat{\boldsymbol{d}}' \tag{4.22}$$

$$\sin\alpha = \begin{cases} \|\boldsymbol{n}\| & \text{for } y_n \geq 0 \\ -\|\boldsymbol{n}\| & \text{for } y_n < 0 \end{cases}, \tag{4.23}$$

where $\hat{\boldsymbol{z}}'_{dest}$ and $\hat{\boldsymbol{d}}'$ are normalized variants of $\boldsymbol{z}'_{dest}$ and $\boldsymbol{d}'$, respectively and y_n is the y-component of $\boldsymbol{n}$. From these two values, α can easily be calculated as the argument of the complex number

$$r = \cos\alpha + i\sin\alpha, \tag{4.24}$$

$$\alpha = \arg r, \tag{4.25}$$

where i denotes the imaginary parts of complex numbers. Finally, the modified (i.e. twisted) target pose M'_{dest} is calculated by

$$M'_{dest} = M_{dest} \cdot T^{-1}_{dest} \cdot R(\alpha) \cdot T_{dest}, \tag{4.26}$$

where $R(\alpha)$ is a rotation of α around $\boldsymbol{y}$, T_{dest} is a matrix, translating about $\boldsymbol{t}_{dest}$ and T^{-1}_{dest} is its inverse.

4.5 Touch Interaction

As already mentioned, the joint buttons can be used to manipulate joint angles separately. As long as the buttons are pressed, the respective joints are rotated with an angular velocity of 60 degree per second. Given the updated joint configurations, an FK problem is solved using KDL, according to Equation 4.16, to find the pose of the TCP.

To integrate multitouch interaction for camera navigation, the default *MatrixManipulator* of *OpenSceneGraph* was extended. For this purpose, the *Windows API* (version 6.0A) was used which comes with native multitouch capabilities. If multitouch gestures are detected by *Windows 7*, it sends the `WM_GESTURE` message, so the application has to check for it in the window procedure. Using *OpenSceneGraph*, this needs for some unaesthetic hacks, since the windows procedure is not accessible due to windowing system abstraction. Along with the `WM_GESTURE` message, the *Windows API* provides a handle to get a gesture info structure with extensive information like gesture state (begin, inertia, end), center position and distance of pinch or drag. Using this data, the *OpenSceneGraph* class `osgGA::TrackballManipulator` was modified, so drag and pinch gestures performed camera translation and rotation, instead of middle and right mouse buttons.

4.6 3D Mouse

To read motion input data from the *SpaceNavigator* in *Windows*, *3Dconnexion* [4] recommends, to use the RAW Input API, which is a simple and stable means to access any Human Interface Device (HID). By registering for receiving raw input from the specified device type, the application gets data from this device, as soon as it is plugged, with no need for driver installation. If HID data is present, the `WM_INPUT` message is sent and the *Windows API* provides a handle to a raw input structure, which contains information about translation and rotation vectors. The translation vector represents the current motion direction of the 3D mouse, while the norm of this vector denotes its distance from the idle position. The rotation vector represents a rotation axis, while its length denotes the rotation angle. Both translation distance and rotation angle do not seem to come in meaningful units, in this work they have been divided by 400, which is the highest value observed for both, so ranges of about $[-1, 1]$ are processed. Translations and rotations were interpolated over time, whereas values of 0.08 m/s and 15 degree per second were chosen, respectively. As common for HIDs, the coordinate system is right-handed with $\boldsymbol{z}$ pointing down, as seen in Figure 4.17. To maintain consistency with other libraries, the axes were switched accordingly for subsequent calculations.

For motion with respect to the robot frame, the yaw-rotation has to be removed from the 3D mouse input, since the *AL5C* cannot perform this kind of rotation, lacking an according joint. In free motion mode, this issue is met by modifying the rotation matrix R_S (which is built of the rotation information obtained from the 3D mouse) in a way that yaw-rotation is eliminated. On the other hand, in constrained motion mode, implementation of yaw rotation is simply skipped, so only a rotation around one single axis is performed at a time. To build a new rotation matrix from scratch, only containing the single rotation currently to be performed, R_S is decomposed in its Tait-Bryan angles.

To get rid of the yaw-rotation in R_S, XZY rotation convention is assumed so R_S can be described as

$$R_S = R_Y(\phi) \cdot R_Z(\theta) \cdot R_X(\psi), \tag{4.27}$$

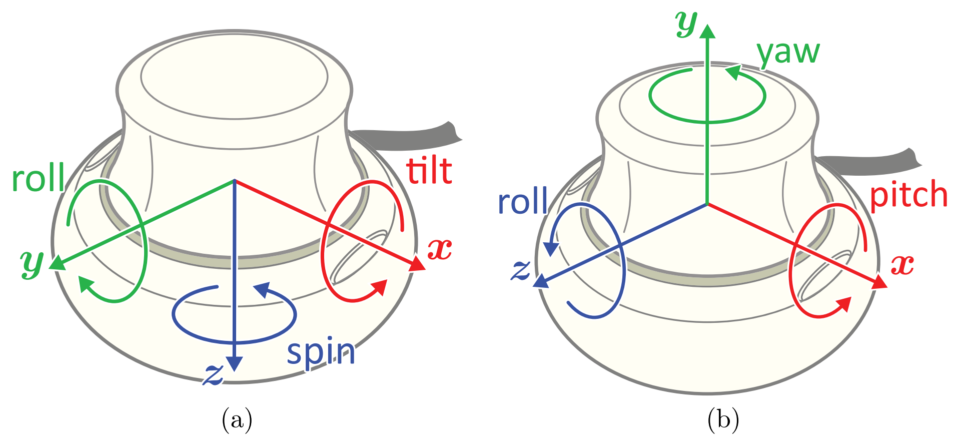


Figure 4.17: The coordinate system, as well as the naming conventions for the *SpaceNavigator* (a) differ from those, used in this work, so they were adapted accordingly (b).

and yaw-rotation can be eliminated by multiplying with its inverse

$$\begin{aligned} R'_S &= \underbrace{R_Y^{-1}(\phi) \cdot R_Y(\phi)}_{I} \cdot R_Z(\theta) \cdot R_X(\psi) \\ &= R_Z(\theta) \cdot R_X(\psi), \end{aligned} \tag{4.28}$$

with R'_S being R_S without yaw-rotation. Therefore, we can say that

$$R'_S = R_Y^{-1}(\phi) \cdot R_S. \tag{4.29}$$

To achieve the rotation angles ψ, θ and ϕ, it is helpful to take a look at the rotation matrix R_S. From Equation 4.27 we get

$$\begin{aligned} R_S &= \begin{pmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{pmatrix} \cdot \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix} \\ &= \begin{pmatrix} \cos\psi\cdot\cos\phi & -\cos\psi\cdot\sin\phi\cdot\cos\theta - \sin\psi\cdot\sin\theta & -\cos\psi\cdot\sin\phi\cdot\sin\theta + \sin\psi\cdot\cos\theta \\ \sin\phi & \cos\phi\cdot\cos\theta & \cos\phi\cdot\sin\theta \\ -\sin\psi\cdot\cos\phi & \sin\psi\cdot\sin\phi\cdot\cos\theta - \cos\psi\cdot\sin\theta & \sin\psi\cdot\sin\phi\cdot\sin\theta + \cos\psi\cdot\cos\phi \end{pmatrix}. \end{aligned} \tag{4.30}$$

Note that the element r_{21} is the sine of ϕ, so the roll rotation can be calculated by taking the arc sine of this value. Retrieving θ and ψ is more difficult. Since

$$r_{23} = \cos\phi \cdot \sin\theta \quad \text{and} \tag{4.31}$$

$$r_{22} = \cos\phi \cdot \cos\theta, \tag{4.32}$$

a division of $\frac{r_{23}}{r_{22}}$ removes $\cos\phi$ and results in the tangent of θ:

$$\frac{r_{23}}{r_{22}} = \frac{\cos\phi \cdot \sin\theta}{\cos\phi \cdot \cos\theta} = \frac{\sin\theta}{\cos\theta} = \tan\theta. \tag{4.33}$$

θ can be calculated as the argument of r_{23} and r_{22}, and similarly, ϕ is achieved by the argument of $-r_{31}$ and r_{11}. Summarizing, the rotation values can be computed with

$$\psi = \arg(-r_{31} + ir_{11}), \tag{4.34}$$
$$\theta = \arg(r_{23} + ir_{22}) \quad \text{and} \tag{4.35}$$
$$\phi = \arcsin r_{21}, \tag{4.36}$$

where i denotes the imaginary parts of complex numbers. Given these angles, the rotation matrices can be built accordingly for constrained motion mode, and yaw rotation can be eliminated from R_S for free motion mode, using Equation 4.29.

If two or more displacement values are of similar size in constrained motion mode, e.g. translation values x and y are both about 0.5 and a little alternating, this would result in a permanent oscillation between translation along $\boldsymbol{x}$ and $\boldsymbol{y}$, depending on which value is currently greater, which can change very rapidly. To avoid this, a hysteresis value h was introduced. So, in the prior example, the current translation is performed along $\boldsymbol{x}$, the motion is not switched to $\boldsymbol{y}$-translation until $y > x+h$. Once this happened, it is not switched back to $\boldsymbol{x}$-translation until $x > y + h$. This hysteresis mechanism involves all motions, translations as well as rotation values. Also, a threshold t was implemented, so if the 3D mouse was previously idle, motion does not start until a value exceeds t. For hysteresis and threshold, adjustments of $h = 0.15$ and $t = 0.3$ were used. Rotation values, which are represented in radian units, are scaled by 4.0 so they are transformed into ranges, which are comparable to the translation values.

4.7 Results

4.7.1 *Teach Pendant* Interface in Detail

Figure 4.18 shows a screenshot of the display of *Teach Pendant* (see Table A.2 in Section A.2 for descriptions). The user may display or hide robot, TCP trajectory, and keyframe positions and basis axes by pressing the corresponding buttons in the *display window*. By default, robot, keyframes and trajectory are displayed, while keyframe basis axes are hidden.

In addition to the possibility of manipulating the TCP frame using the 3D mouse, there are joint buttons in the *teach pendant window*, which provide control over each one of the arm joints independently. This is inspired

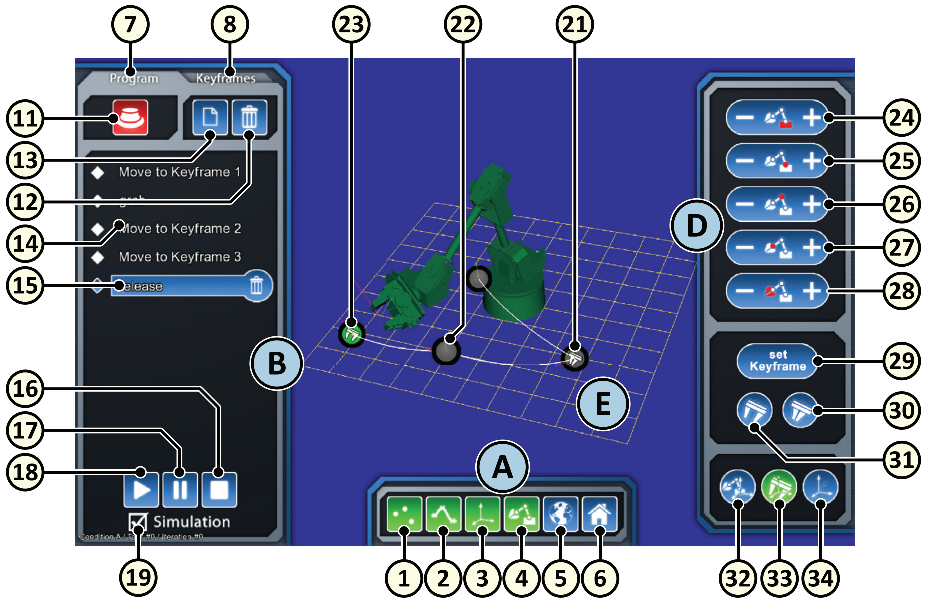


Figure 4.18: The display of *Teach Pendant* consists of three windows for GUI elements and a VR scene view. For an explanation of according elements see Table A.2 in Section A.2.

by common teach pendants. Although it is difficult to direct the TCP to a desired pose using these buttons, they may be helpful for performing base or wrist rotations. To activate 3D mouse and joint buttons, the *3D mouse control on/off switch* has to be pressed. Subsequently, the user may guide the TCP to a desired pose and press the *insert keyframe*, *insert grab command* or *insert release command* buttons accordingly. Whenever a new keyframe or a grab or release command is inserted, movements or gripper instructions are registered in the *instruction sequence* of the *program window*. Additionally, new keyframes are charted in the *keyframe list*, which can be found found in the *keyframe tab*. Switching between different motion modes is also performed using GUI buttons.

When the user is finished and the 3D mouse is disabled again, the program can be edited, using the touchscreen GUI. To do this, keyframes can either be selected in the 3D scene or the *keyframe list*, to insert new movements or gripper commands at any point of the instruction sequence. Whenever a keyframe is deleted, all instructions associated with it are deleted too. In contrast, if instructions are deleted, the associated keyframes stay available, even if they are not referenced by an instruction anymore, so they can be reused later on. In both cases the displayed trajectory is updated accordingly.

The visualization provides VR rendering exclusively, whereas only single

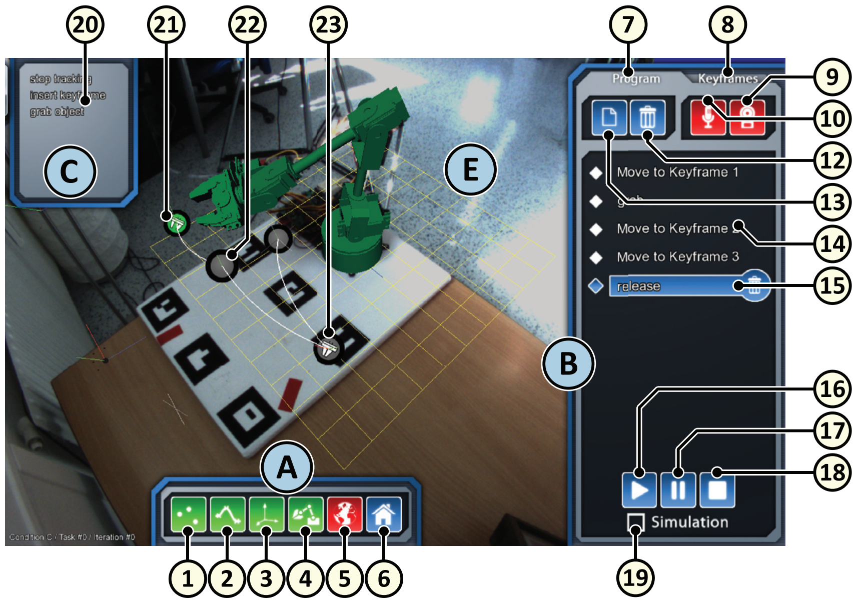


Figure 4.19: The display of *MIMIK* consists of three windows for GUI elements and a scene view, which is switchable between AR and VR. For an explanation of according elements see Table A.2 in Section A.2.

touch interaction is provided. The scene may be rotated using single touch press and drag. If the user tries to perform movements, which are not possible due to kinematic constraints of the robotic arm, according joint constraints are displayed.

4.7.2 *MIMIK* Interface in Detail

The display of *MIMIK* is very similar to the one of *Teach Pendant*, as can be seen in Figure 4.19 (see Table A.2 in Section A.2 for descriptions). However, the 3D scene area shows an AR depiction of the robot. Since the AR setup is stationary, it is possible to switch to VR and to navigate through the scene, using touch and multitouch gestures. Pressing the *home* button returns to the default scene view, in case the user gets lost.

To program a grasping process with *MIMIK*, it is necessary to first activate speech recognition by pressing the *Microphone on/off switch*. As soon as it is enabled, a context dependent list of available speech commands is displayed in the *command window*. Subsequently, 6 DOF tracking can be enabled, either using the GUI button, or the according speech command. As soon as 6 DOF tracking is enabled, the user may guide the TCP with his hand, and keyframes can be inserted by the speech-command "insert keyframe". If the user's hand achieves poses, which are not adoptable by

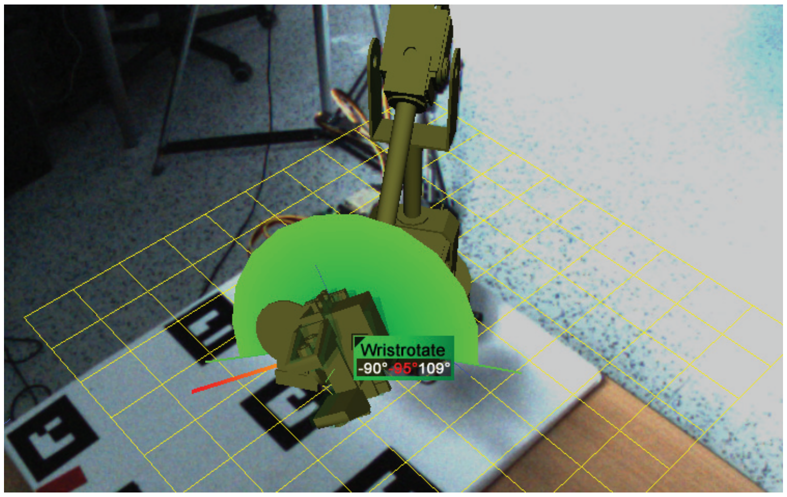


Figure 4.20: Whenever joint ranges would have to be exceeded, to achieve the target pose, the constraining joints are highlighted, and angular ranges, as well as the inexecutable rotation are displayed.

the robotic arm due to kinematic constraints, the physical robot stops. In the display, the robot geometry is then rendered semi-transparent and an additional, fully opaque virtual robot model tries to follow the target pose, violating joint constraints, so kinematic problems are visualized (see Figure 4.20). Joints that are rotated outside of their physical boundaries are then highlighted, and angle information (minimum, maximum, invalid target angle) is displayed in place. To find this invalid joint configuration, another IK solver implementation was used, which does not consider joint constraints. The joint angles resulting from this unconstrained solver are then compared to the actual boundaries, and joints violating them are marked accordingly.

Since IK yields multiple solutions and only one of them can be displayed at a time, the visualization just shows *one example* of joint violations. The solver can hardly be taught to compute the solution, which is most similar to the one, displayed in the the previous render pass. As a result of this, the representation of the unconstrained robot sometimes jitters and flips over, switching between different IK solutions.

Grab and release actions are triggered by speech commands too. If the previously set keyframe is too far from the current position in Euclidean space (a threshold of $0.025\,\mathrm{m}$ was used), a keyframe is inserted automatically[17].

As soon as the user is finished and tracking is disabled again, the program can be edited just like in *Teach Pendant*, but using the combination of the touchscreen and speech interaction. For playback the user can choose to either run a VR/AR-simulation by activating the simulation check or to run the program on the physical robot as well.

[17]As a distance measure only the Euclidean distance was used, rotations should also be considered but were ignored so far.

Chapter 5

User Study

To validate the major interaction design, a user study was conducted for gaining quantitative and qualitative measurements of robot operation and programming, given the two prototypes. The study was composed of a preceding questionnaire to collect demographic data, followed by the experiment and a concluding questionnaire, gathering subjective data about different aspects of each of the setups. Latter was composed of questions about aspects to be rated on a Likert scale from 1 (very bad) to 5 (very good). Additionally, the participants had the opportunity to give additional feedback about their impressions, either textually or verbally.

The research questions to be answered by this study were if, and by what degree teaching of simple processes can be sped up by the combination of 6 DOF tracking and speech input, and if it would be preferred by operators, compared to status quo methods, such as using a teach pendant.

5.1 Participants

Twelve voluntary participants were recruited from the local university, containing students, secretaries and professors. Three of them were female, nine were male, the ages ranged from 22 to 44 years ($M = 26.67$, $SD = 5.76$). They spent an average of 8.67 hours per day on a computer ($SD = 2.15$). Five were highly skilled in programming or scripting, four had no experience whatsoever, while none of them was familiar with robot programming. Eight participants have already used a *WiiMote*, six were also experienced in motion capturing and three have operated a 3D mouse before. All of them were familiar with touchscreens, seven of them knew multitouch gestures. Ten already were aware of the concept of keyframes prior to the study (mainly from animation and video editing). All of them have been in contact with VR or computer/video games before, and four already were familiar with AR.

5.2 Apparatus

There were two setups to be compared, called *Teach Pendant* and *MIMIK* in the following. In both of them, the participants were sitting in front of a table, with the touchscreen device and the robotic arm standing on it. Although the same names are used here for convenience reasons, the setups, as described in Section 3.1, have slightly been modified:

Teach Pendant combined 3D mouse control for manipulating the TCP with GUI buttons on the touchscreen for grasping, releasing, setting keyframes, and switching motion modes (see Figure 3.3). This setup was meant to represent current teaching methods, using teach pendants. Even if physical buttons are attached to those, the concept is the same, and thus the designs are considered to be comparable. To keep the interface simple, multitouch navigation and joint buttons (as described in Section 4.5) were removed for the study. FK has not been considered advantageous in the given tasks, and only could have led to confusion. Also, the scene view showed the robot in a well-arranged pose in VR, so no navigation was necessary.

MIMIK combined 6 DOF tracking for TCP manipulation with speech interaction for simple commands, such as starting and stopping tracking, releasing, grasping and setting keyframes (see Figure 3.6). The touchscreen GUI was used to activate and deactivate speech recognition, and later on for starting replay. The scene view was set to AR mode by default and was not explained to the participants to be switchable to VR.

5.3 Procedure

Two tasks had to be performed by the participants for each setup, respectively. They concern about simple grasping processes, in which an object has to be moved from an initial position A to a target position B. The assignment was, to program fully functional robot programs, which can be replayed later on, so several keyframes had to be set on proper poses. After an explanation of the particular setup, the participants were allowed to familiarize themselves with the system for up to two minutes, before they started with the first task. To keep the trials short, post-editing was not allowed and also not explained to the participants. The tasks were designed to involve very simple processes, regarding the program cycle, so they could be accomplished without the need for post-editing.

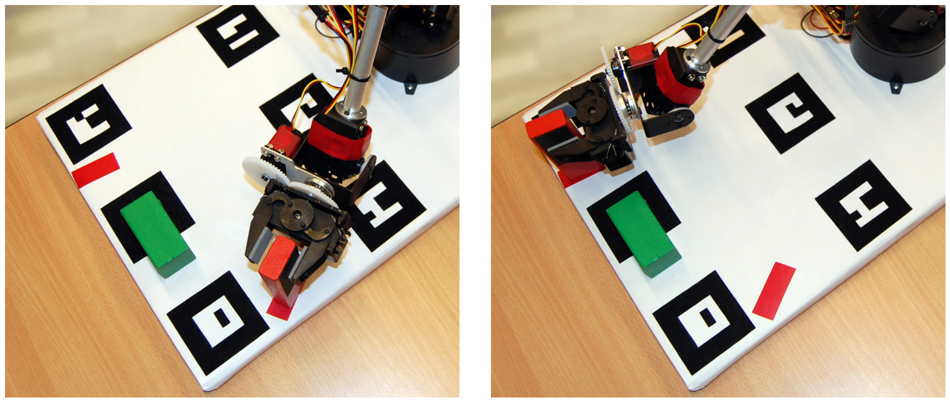


Figure 5.1: *Task 1* required the participants to program a robotic process, which picks an object from an initial position (a), moves it over an obstacle, and places it on a target position (b). The only measurement taken was task completion time; positioning accuracy was not a decisive factor.

5.3.1 Task 1

Task 1 is illustrated in Figure 5.1. An object had to be moved from A to B, avoiding an obstacle. So a possible solution is illustrated in Figure 5.3 (a), requiring six keyframes ($P_1 \cdots P_6$) to be specified, while P_0 already was present.

- P_0: general safe pose
- P_1: approach to P_2
- P_2: object at pose A
- P_3: object lifted above A
- P_4: object lifted above B
- P_5: object at pose B
- P_6: departure from P_5

Simple enough, the according instruction sequence would be as follows:

```
Move to P1
Move to P2
Close gripper tool
Move to P3
Move to P4
Move to P5
Open gripper tool
Move to P6
Move to P0 and finish
```

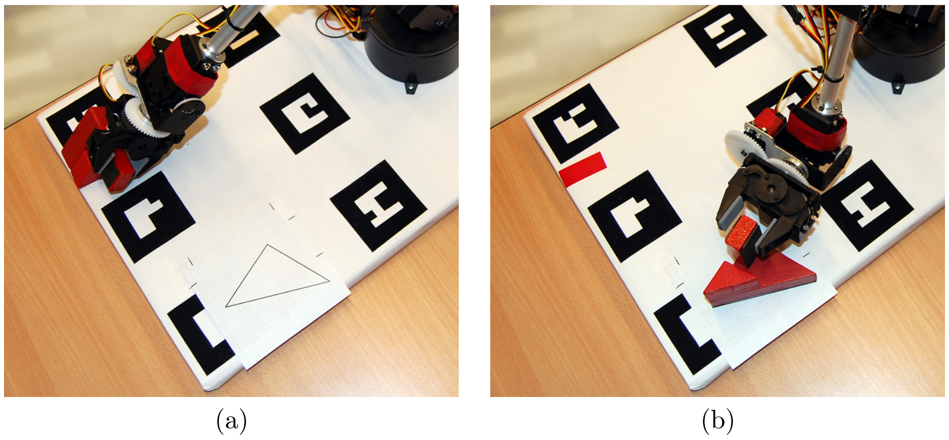


Figure 5.2: *Task 2* required the participants to program a robotic process, which picks an object from an initial position (a), and places it on a target position (b). To pick the object, the TCP had to be rotated by $\theta = 45°$ and $\phi = 90°$. Measurements regarding task completion time, as well as positioning accuracy were taken.

5.3.2 Task 2

Figure 5.2 shows *Task 2* which did not include an obstacle. While *Task 1* was rather easy, *Task 2* was much more difficult, since it involved wrist rotations, and hence is a much more common scenario in practice. An object had to be grasped at position A, which required the wrist rotate joint to be adjusted accordingly. Subsequently, the object had to be rotated and placed at B, which was labeled with the shape of the object's ground area. An example solution is given in Figure 5.3 (b), again requiring six new keyframes along with P_0. Rotations are denoted in XZY rotation convention according to TCP frame axes as defined in Section 4.1.2.

- P_0: general safe pose
- P_1: approach to P_2 with $\theta = 45°, \phi = 90°$
- P_2: object at pose A with $\theta = 45°, \phi = 90°$
- P_3: object lifted above A with $\theta = 45°, \phi = 90°$
- P_4: object lifted above B with $\theta = 0°, \phi = 0°$
- P_5: object at pose B with $\theta = 0°, \phi = 0°$
- P_6: departure from P_5

The according instruction sequence is identical to the one shown for *Task 1*. *Task 2* was also designed to address a common comprehension problem we call the *wrist rotation problem* henceforth. Since the wrist can only be rotated within a range of approximately $[-90°, 90°]$ about the $\boldsymbol{z}_{TCP}$-axis, it

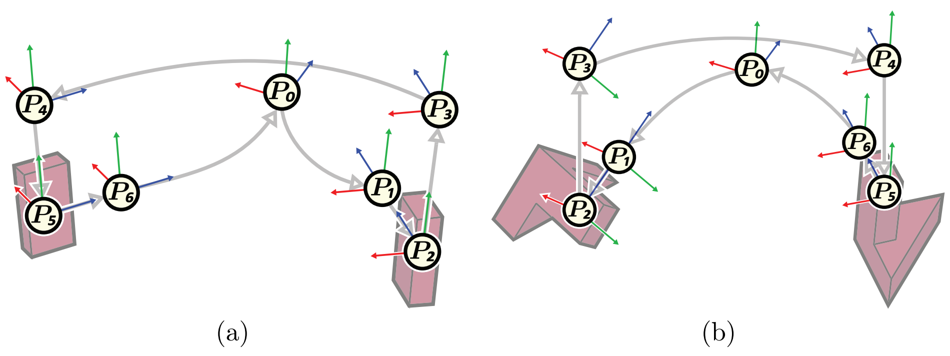


Figure 5.3: Solutions for *Task 1* (a) and *Task 2* (b) may require seven keyframes in each case, including the general save pose P_0. Note that in *Task 2* the rotation of the object takes place between P_3 and P_4, to avoid the ground collision problem.

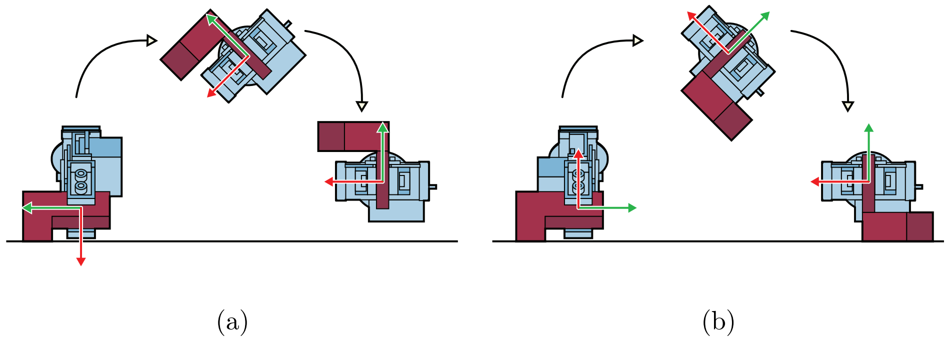


Figure 5.4: The wrist rotation problem refers to the issue, occurring when the gripper is rotated the wrong way by the time the object gets picked, since the wrist rotation joint is limited to about $[-90°, 90°]$. If the object is picked with a TCP rotation of $\phi = -90°$, it is oriented upside-down when the gripper is rotated back to $\phi = 0°$ and thus cannot be placed (a). However, if it is picked with a TCP rotation of $\phi = 90°$, it can be placed correctly.

is important to pay attention to its rotation during picking up the object. If the wrist is rotated into the wrong direction, it is not possible to perform the rotation, necessary to place the workpiece as claimed (illustrated in Figure 5.4). So, for *Task 2*, a trivial strategy had to be worked out by the participant in advance. Note that the rotation of the workpiece does not happen until it reaches P_3, thus it is lifted. If the rotation would be performed simultaneously with the lifting, it may cause one side of the workpiece to collide with (or to

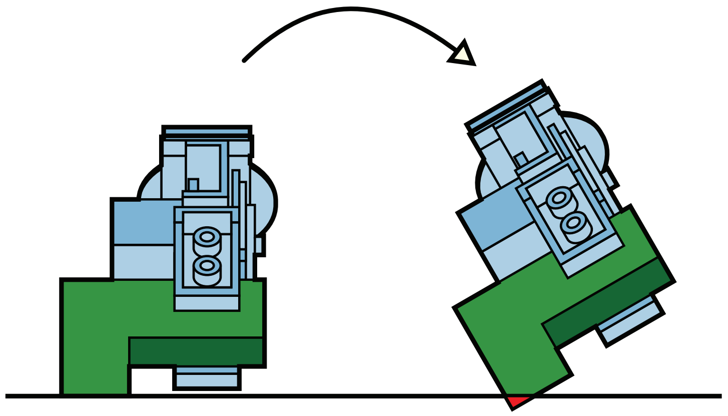


Figure 5.5: The ground collision problem refers to the issue that may occur when the object is rotated while being lifted in *Task 2*. If rotation takes place along the translation between P_2 and P_3, as defined in the example solutions, the outer edge of the object will collide with the ground plane if the translation is not fast enough. Thus, rotation as to be performed *after* the object has been lifted.

be "pressed into") the ground plane, as illustrated in Figure 5.5. This issue will be called *ground collision problem* henceforth.

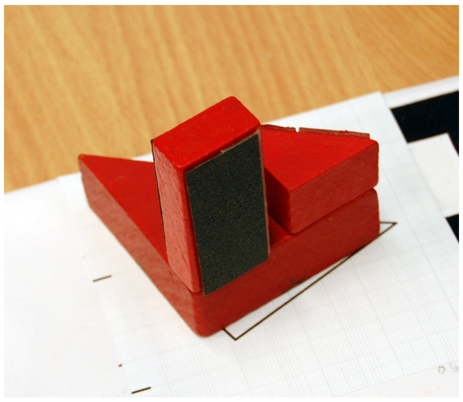
5.4 Experimental Design

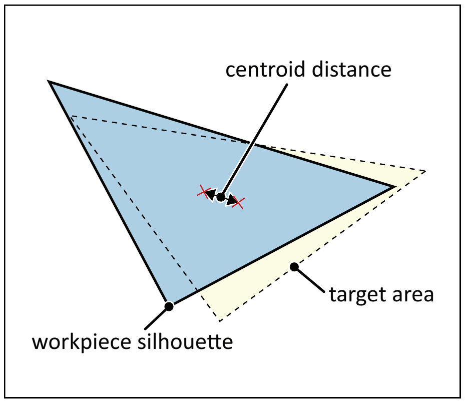
The duration of the experiment was targeted at one hour per participant. The study had a counterbalanced within-subject design, where the participants were split into two groups. The order of *Teach Pendant* and *MIMIK* was varied by *Group*. All of the factors were within-subject, with two levels for *Group* (Group 1 vs. Group 2, six participants per group), two levels for *Setup* (*Teach Pendant* vs. *MIMIK*), two levels for *Task* (*Task 1* vs. *Task 2*), and five levels for *Trial* (1, 2, 3, 4, 5). The total amount of iterations was 6 participants $\times$ 2 groups $\times$ 2 setups $\times$ 2 tasks $\times$ 5 trials $=$ 240 iterations. Measurements taken were:

Task completion time: The time between activating and deactivating 3D mouse (in *Teach Pendant*) or high DOF tracking (in *MIMIK*).

Collisions count: The number of collisions between gripper tool and work piece for both picking up and placing.

Failure count: The number of trial cancellation, which occurred, whenever the workpiece got knocked over during picking up (only possible in *Task 1*), or whenever a prior major collision caused a dislocation in a way, it


(a)



(b)

Figure 5.6: In *Task 2*, the object had to be placed on a triangle-shaped marker (a). The distance of the centroids of target area and placed object served as an indicator for positioning accuracy.

could not be grasped any more. Also, on wrist rotation problems (see Section 5.3.2), the trial was canceled.

Program error count: The number of relevant keyframes that were missing in the taught robotic process.

Positioning accuracy: Distance between the centroids of target area and placed workpiece base area (see Figure 5.6). This measurement was only taken in *Task 2*. Rotational inaccuracies have not been relevant in this case, since they resulted from seized grasping, which was mostly not preventable by the participants. Also, they were not correctable later on, since the *Lynxmotion AL5C* is missing a joint for according rotations. Using a 6 DOF robotic arm, these deflections could be compensated.

In failed trials, there obviously were no data for task completion time and position accuracy, so the trial means, calculated from the remaining participants were inserted in these cases.

Since speech input was not robust enough to be applied during the experiments[1], a Wizard of Oz (WoZ) study was conducted. For this purpose, the actions “start tracking”, “stop tracking”, “insert keyframe”, “grab” and “release” were triggered remotely by the investigator, using a wireless keyboard. This was done without the participant’s knowledge.

[1]Although it worked well in many cases, ill-detected commands may have required to repeat the whole trial, e.g. if a “release” command could have been triggered accidentally, while moving or placing the object. This would have led to serious complications and so it was decided to fake speech input.

Chapter 6

Results and Discussion

6.1 Measurements

Figure 6.1 shows the grand means of task completion time measurements. In *Task 1* they were 51.90 s ($SD = 10.76$) for *Teach Pendant* vs. 36.23 s ($SD = 9.60$) for *MIMIK*. A within subjects analysis of variance (ANOVA) showed that the results were statistically significant with $F_{1,10} = 21.121, p < 0.0001$. *Task 2* took longer since it was more complicated, there the means were 121.13 s ($SD = 26.08$) for *Teach Pendant* vs. 50.42 s ($SD = 13.05$) for *MIMIK*. The results were statistically significant with $F_{1,10} = 216.098, p < 0.0001$. So *MIMIK* enabled for trial durations faster than those of *Teach Pendant* by 30 % (*Task 1*) and 58 % (*Task 2*), respectively. There were major differences by *Participant* (see Figure 6.2). In *Task 1* the participant's means

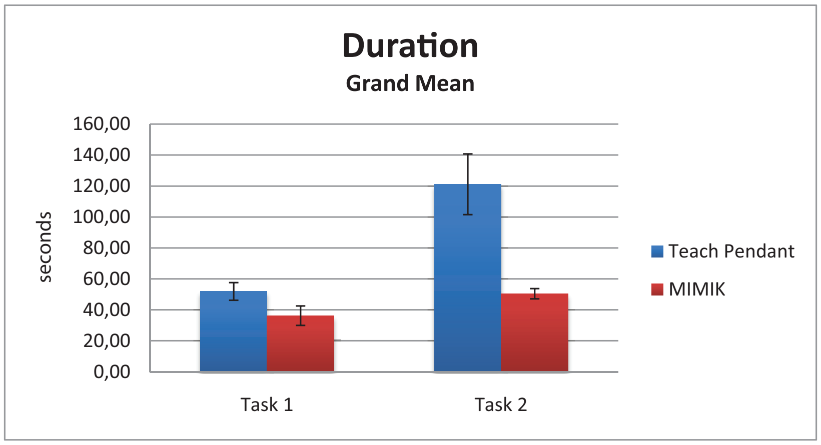
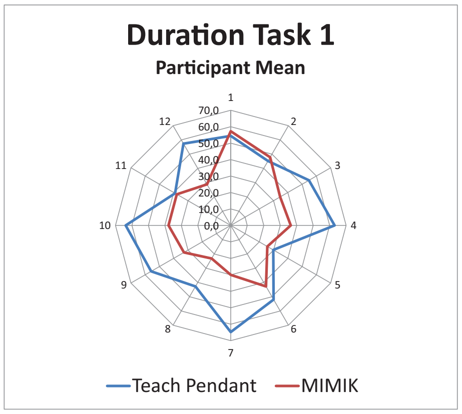
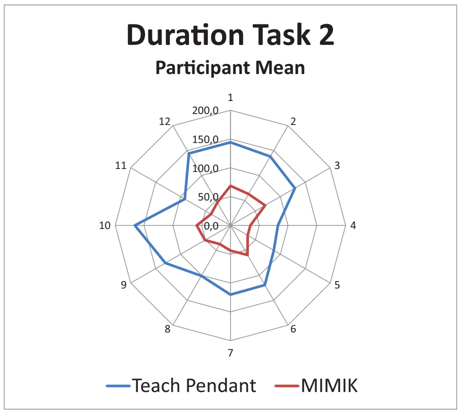


Figure 6.1: The grand means of task completion time show major differences, especially for *Task 2*.



(a)



(b)

Figure 6.2: The participants means for task completion time show that there only were two participants, being faster with *Teach Pendant* in *Task 1* (a), while all of them were faster with *MIMIK* in *Task 2* (b).

of *Trial* range from 29.8 s ($SD = 3.42$) to 64.8 s ($SD = 20.09$) for *Teach Pendant* and from 23.2 s ($SD = 7.46$) to 57.2 s ($SD = 18.02$) for *MIMIK*. In *Task 2*, they range from 82.4 ($SD = 21.87$) to 166.0 ($SD = 12.94$) for *Teach Pendant* and from 34.4 s ($SD = 10.99$) to 69.4 s ($SD = 22.17$) for *MIMIK*. This suggests different participant priorities, which also agrees to the observations. Some went for higher speed, others for higher precision—even in *Task 1*, where precision was not a decisive factor. However, this was the case in both setups, and grand means, as well as shortest completion durations, show clearly that *MIMIK* is superior regarding task completion time. The p-values show that these differences were not significant in both cases. Figure 6.3 shows the shortest task completion durations measured, which for *Task 1* were 25 s for *Teach Pendant* and 17 s for *MIMIK*. For *Task 2* they were 64 s for *Teach Pendant* and 26 s for *MIMIK*. This shows that the fastest trials were noticeable superior in *MIMIK*, especially in the more complicated task. Most participants preferred constrained motion mode, and hence primarily used it during the experiment, which means that movements and rotations had to be accomplished successively, causing higher task completion times. There also were serious differences by *Trial* as can be seen in Table 6.1. Comparing means of trials 1 and 5 shows that during *Task 1*, the participants improved by 26 % for *Teach Pendant* and by 33 % for *MIMIK* ($F_{4,40} = 12.603, p < 0.0001$). During *Task 2*, they improved by 31 % for *Teach Pendant* and by 13 % for *MIMIK* ($F_{4,40} = 10.808, p < 0.0001$). The downside trend in Figure 6.4 (a) shows that there is a noticeable learning effect, which is similar for both setups, with a serious advance of *MIMIK*.

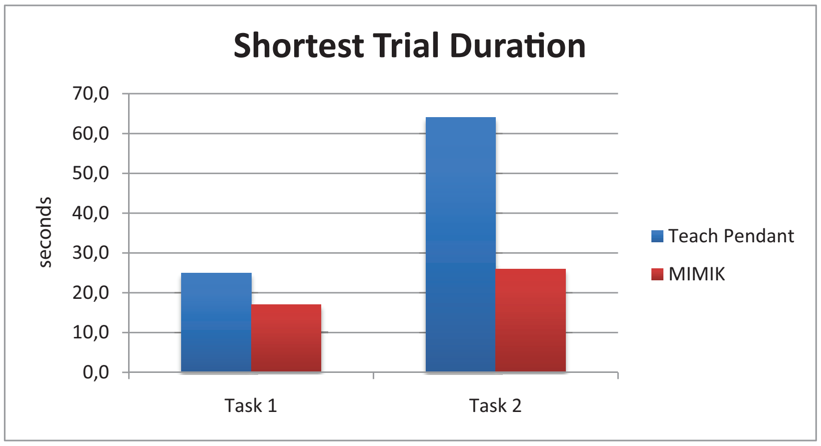
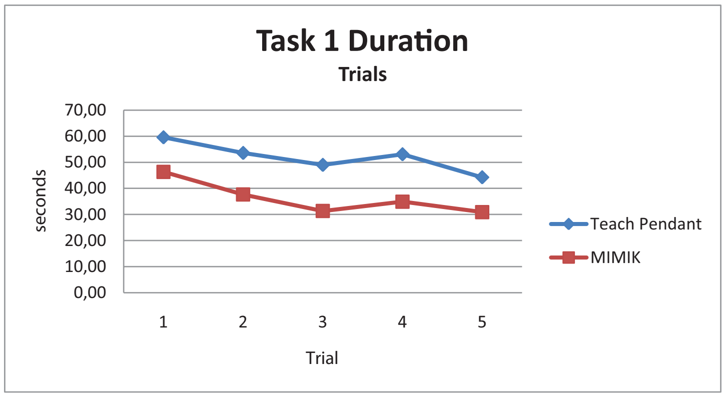
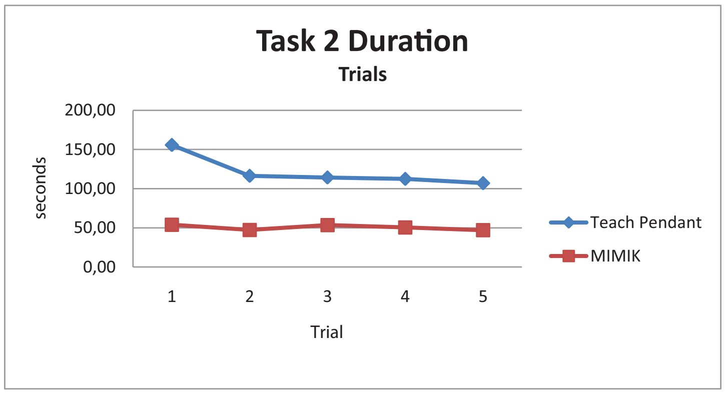


Figure 6.3: The overall shortest task completion times show a major advantage of *MIMIK*.



(a)



(b)

Figure 6.4: The trials means for task completion time of *Task 1* (a) and *Task 2* (b) show a noticeable learning effect in *Task 1*.

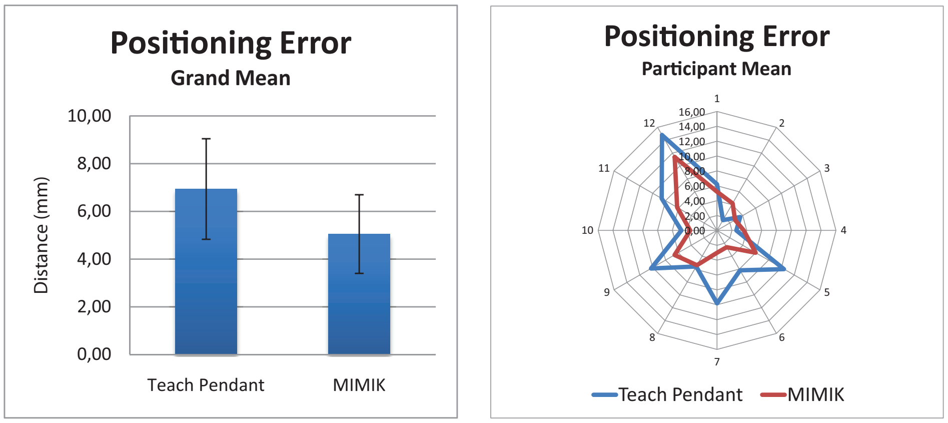
Table 6.1: The improvement in task completion time in *Task 1* was greater for *MIMIK*, while it was greater for *Teach Pendant* in *Task 2*.

	Teach Pendant			*MIMIK*		
	Trial 1	*Trial 5*	*Speed-up*	*Trial 1*	*Trial 5*	*Speed-up*
Task 1	59.6 s	44.3 s	25.73 %	46.3 s	30.9 s	33.27 %
Task 2	155.7 s	107.0 s	31.26 %	53.9 s	46.9 s	12.98 %

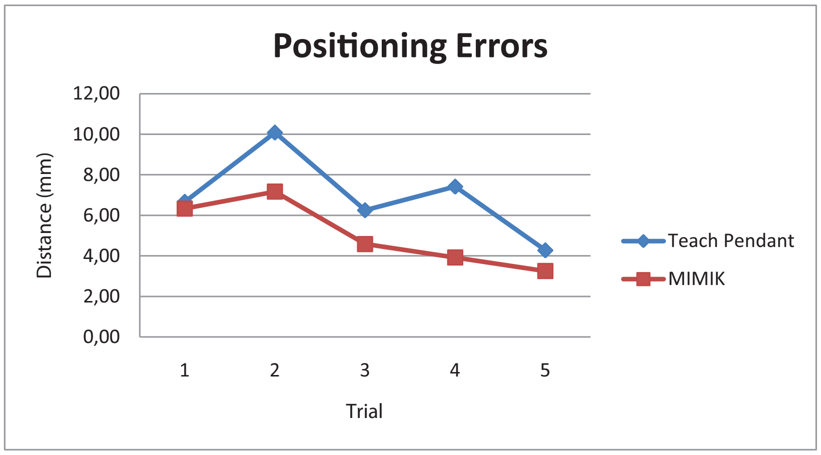
Since the order of both tasks was not counter-balanced due to the higher complexity of *Task 2*, there is not much learning effect in *Task 2*, as seen in Figure 6.4 (a), with one exception: As wrist rotations were introduced, the participants had a hard time to figure out how to grab the object using *Teach Pendant*, which also was observed during the experiments. Once they learned, how the task can be accomplished, they were much faster and showed only slightly improvements thenceforth. *MIMIK* did not show this effect, participants intuitively knew how to reach the object with the gripper: Simply as they would with their own hand. This suggests that the usage of *Teach Pendant* for each new situation has to be "learned" first and thus is less intuitive. In contrast, the participants seemed to already have adopted the usage of *MIMIK* during *Task 1*, and no strategy has to be worked out for new task requirement. This seems to be due to the chosen interaction modality of 6 DOF tracking, since humans perform simple arm for their whole lives and thus know, how to grasp objects without thinking about it.

Figure 6.5 shows the positioning accuracy measurement results of *Task 2*. The grand means were 7.04 mm (SD = 3.832) for *Teach Pendant* and 5.05 mm (SD = 2.436) for *MIMIK*. The results were statistically significant with $F_{1,10} = 6.945, p < 0.05$. Again, there were considerable alterations by *Participant*. Over the five trials the means range from 1.60 mm (SD = 0.894) to 14.80 mm (SD = 4.604) for *Teach Pendant* and from 2.60 mm (SD = 1.817) to 11.40 mm (SD = 4.561) for *MIMIK*. An ANOVA showed that these values were not significant ($F_{4,40} = 1.415, \text{ns}$). It is surprising that the accuracy of *Teach Pendant* performed that bad, after all the TCP may be guided to the exact destination spot with a joystick-like controlling mechanism, enabling for more subtle movements which are not even affected by hand jittering, in contrast to in *MIMIK*. This is not expected on professional robotic systems, and although there sometimes were mechanical problems complicating the task[1], this may be an indication to imperfect im-

[1]Sometimes the base rotation joint jammed, so the arm did not rotate first, then suddenly jumped for some degrees at once.



(a) (b)

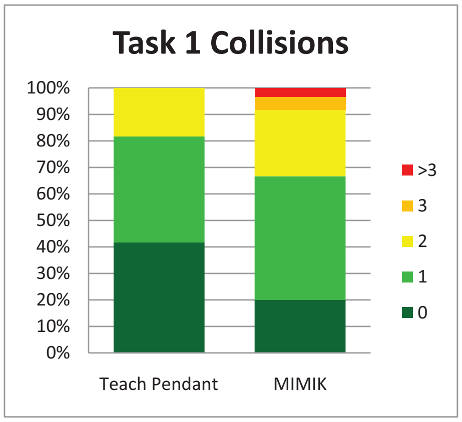


(c)

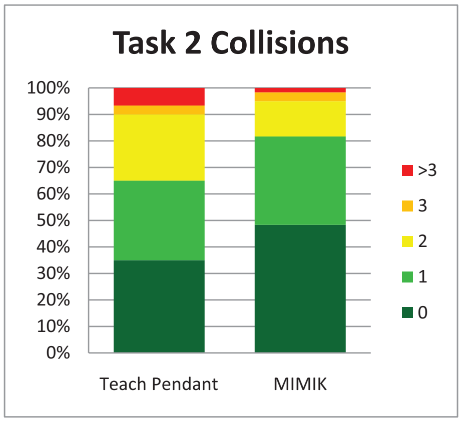
Figure 6.5: The grand mean for positioning error was higher for *Teach Pendant* (a). Only two participants were less inaccurate using *MIMIK* (b). The means over trial show a downward trend, which is steeper for *Teach Pendant* (c).

plementation or unsatisfactory parameterization (too high values may have been chosen for threshold and/or motion speed). However, this also is a result of several participants, simply dropping the workpiece after some time because they found the control to be cumbersome, got frustrated and thus compromised accuracy for speed. Nonetheless, there is a downward trend noticeable by *Trial*, which is steeper for *Teach Pendant*, so it may catch up with *MIMIK*, and even outperform it after some more iterations.

Figures 6.6 (a) and (b) depict the ratio of collisions between gripper tool and workpiece for *Task 1* and *Task 2*, respectively. *Teach Pendant* shows fewer collisions in the simpler *Task 1*. The proportion of participants, col-



(a)



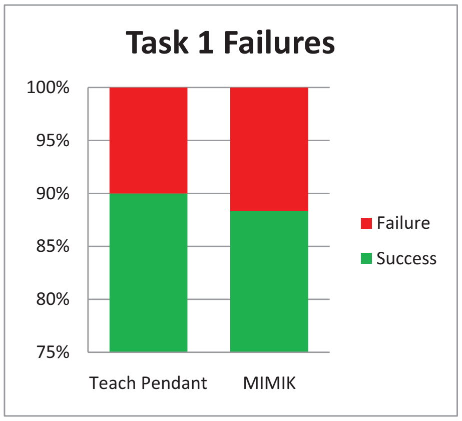
(b)

Figure 6.6: In terms of collisions, *Teach Pendant* performed superior in *Task 1* (a), and worse in *Task 2* (b).

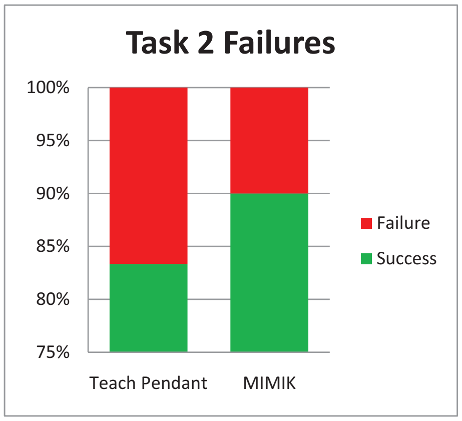
liding less than two times was 82 %, in contrast to *MIMIK* with only 67 %. In *Task 2*, *Teach Pendant* performed worse with 65 %, compared to *MIMIK* with 82 %. Because *Task 1* only needs for some simple translations of the 3D mouse in *Teach Pendant*, it is easy to accomplish, without risking collisions at any point. Most the time the TCP frame axes are aligned with the base frame axes (especially the up axes which seem to be the most crucial ones), so there is no major spatial sense required. As soon as rotations are involved, there are much more collisions due to unexpected TCP movements. In contrast, *MIMIK* is sensitive to every movement of the operator's arm. Combined with the minor latency of the system[2], which also may be irritating, this had to be got used to first. The fact that there were less collisions during the subsequent *Task 2* (which is more complicated after all) might imply that the sensibility and latency of the system has been adopted by the participants by then. Also, it might be an issue that *MIMIK* provides all DOF at once, without supporting constraints. So, even rotations are adapted in *Task 1*, where they are superfluous. As a result, sometimes the gripper was slightly askew, while approaching the object, which then hardly fit into the opened yaws, causing collisions more easily.

Task failure counts are shown in Figures 6.7 (a) and (b). There are only minor differences in *Task 1*, favoring *Teach Pendant* with 90 % against *MIMIK* with 88 %. In *Task 2*, *Teach Pendant* performed worse with 83 %, compared to *MIMIK* with 90 %. Failures in *Task 1* were mainly due to

[2]The latency was due to the robot controller, not the 6 DOF tracking system, so it was present in both setups. Though, whenever arm motions are adopted one-to-one it seems to be more an issue.



(a)

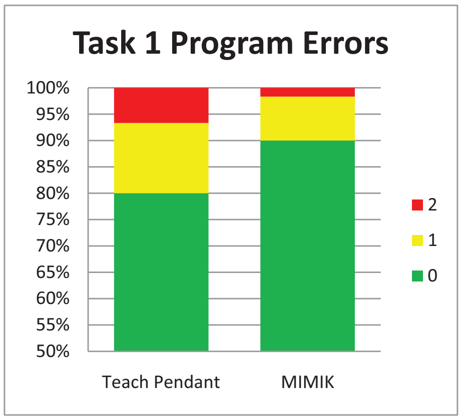


(b)

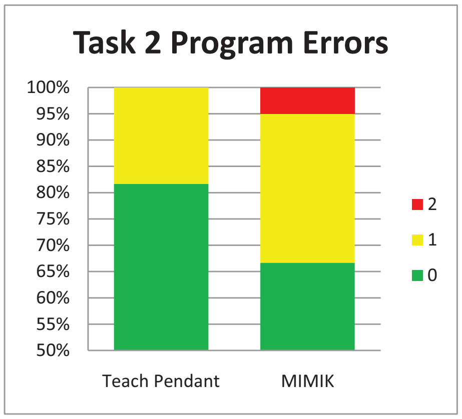
Figure 6.7: In *Task 1* (a), *MIMIK* caused slightly more failures, while in *Task 2* (b) it performed superior.

knocking over of the workpiece. For reasons already discussed above, in *Task 1* collisions were more frequent in *MIMIK*. Since movement speeds were higher by nature until the participants became used to the system sensibility and latency, collisions often resulted in the work piece tumbling down.

Concerning robot program errors, Figures 6.8 (a) and (b) show that *Teach Pendant* performs worse in *Task 1*, while it performs better in *Task 2*. In *Task 1*, the proportion of participants, producing less than two errors for *Teach Pendant* was 93 %, in contrast to *MIMIK* with 98 %. In *Task 2*, it were 100 % for *Teach Pendant* vs. 95 % for *MIMIK*. The results for *Task 2* are unexpected, since in the questionnaire the mental load was rated much higher for *Teach Pendant*, and accordingly more programming mistakes were anticipated. The higher error rate was caused by the preferred way, some participants controlled the robotic arm with *MIMIK*: While they lifted the object from P_2, they already rotated the gripper to $\phi = 0$, so it was oriented horizontally. Hence, the resulting robotic processes suffered from the ground collision problem, as described in Section 5.3.2. Although participants might not have been aware of this, it was counted as a program error. In contrast, using *Teach Pendant*, the participants mostly moved in constrained motion mode, and first of all just lifted the object, which often was hard to accomplish for them, due to the rotated TCP frame. Once they have managed this, they quickly set a keyframe, before bothering with subsequent duties, like rotating the gripper tool. Although this result seems to be due to distraction problems caused by the mental load of *Teach Pendant*, it could be a hint for *MIMIK* being more viable for motion *capturing*, thus recording



(a)



(b)

Figure 6.8: The number of program errors for *Task 1* was higher for *Teach Pendant*, while in *Task 2* it was higher for *MIMIK*.

and replaying exact trajectories, as demonstrated by the operator. Enhanced by manual or automated post-processing and optimization steps, this would also be more viable for trajectory-based tasks like welding or polishing.

6.2 Observations

One participant was not able to perform the required rotations using *Teach Pendant* and needed to be assisted. While in this setup most of the participants happened to move the TCP in a way they did not intend (especially with the rotated gripper tool in *Task 2*), one in particular had major problems imagining motions along TCP frame axes.

Although explicitly advised about the wrist rotation constraints, four participants were not aware of the wrist rotation problem (see Section 5.3.2) until it occurred. Once it did, some of them were confused and had to cogitate for a few seconds to avoid it. This observation suggests that the right choices in operation were often made by accident without realization of potential problems. Once the problems occurred, and the operator became aware of it, the challenge of avoiding them caused an impact on the time and focus required to solve the task. In *MIMIK* however, there was no such observation, since the participants knew how the object has to be grasped without thinking about it.

Two participants of the group starting with *MIMIK* got used to speech interaction that much they accidentally tried to give speech commands in *Teach Pendant* as well, even though they were not even wearing a headset

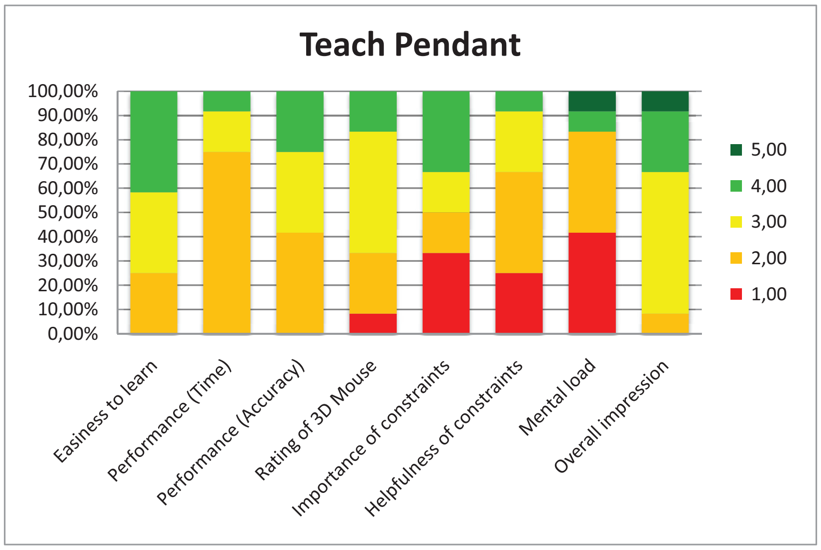


Figure 6.9: The summary of the questionnaire for *Teach Pendant* shows moderate results. Especially, the mental load was rated to be very high. The aspects had to be rated on a Likert scale from 1 (very bad) to 5 (very good).

any more. This seems to confirm that speech input is easily adopted and thus provides a very natural way of triggering commands.

Finally, one participant was not able to figure out how to perform pitch rotations with both *MIMIK* and *Teach Pendant*.

6.3 Questionnaire Results

Figures 6.9 and 6.10 show the analysis of the post-experiment questionnaires. As can be seen, *MIMIK* was rated superior throughout all aspects. It was found to be easier to learn and to provide better performance in task completion time as well as in positioning accuracy. A 3D mouse was considered not to be very sufficient for controlling a robotic arm while 6 DOF tracking was found to be more suitable. Not knowing about the WoZ experiment, the usefulness of speech input was graded very high, while the combination of 6 DOF tracking with speech was perceived as highly valuable. Both importance and helpfulness of constraints visualization on screen was judged moderately. Participants had to concentrate noticeable harder using *Teach Pendant*. In terms of overall impression *MIMIK* clearly was the favorite.

Nine Participants preferred constrained motion mode against free motion mode. Polled about what setup the participants was more attracted to and which one they would use again, only one chose *Teach Pendant*.

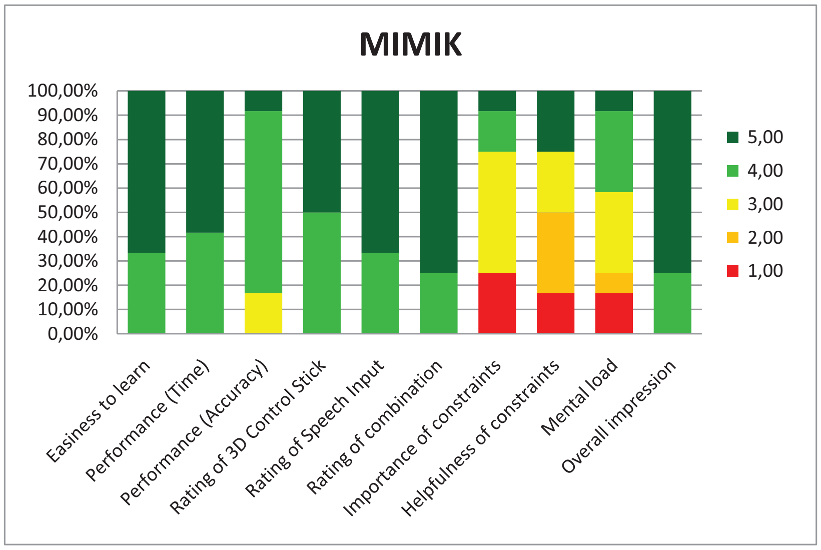


Figure 6.10: The summary of the questionnaire for *MIMIK* show that most aspects are rated positively. The AR-based constraints visualization were not rated that helpful. The aspects had to be rated on a Likert scale from 1 (very bad) to 5 (very good).

Some stated that a 3D mouse would be fine as long as no rotations are involved. From this point on, it would be unintuitive, tedious and difficult to imagine, in what direction a 3D mouse movement will guide the TCP. Further comments include that a *MIMIK* would be "much more intuitive to handle", "less precise, but very simple to learn and handle", "much easier to handle" and "intuitive and quickly to be learned". One participant stated that *MIMIK* would be more valuable for *Task 2*, while *Teach Pendant* would be more feasible for *Task 1*. The benefit of *Teach Pendant* was said to be "task dependent", "may yield accurate results, given some training", and "needs for more training and instruction", while the preceding instructions were "too much information to handle at a time".

One participant noted that the AR visualization on screen would have hardly been noticeable because he was that focused on the physical robot, while another one said it would have been helpful, since the joint constraints were sometimes hard to imagine. Whenever he got stuck he would have consulted the visualization, and gained valuable information from it.

Suggestions for improvements of *MIMIK* were, to provide the possibility of specifying the motion radius for the 3D stick (and thus the scaling according to Equation 4.11) by oneself and to provide acoustic feedback (TTS or simple beeps) to acknowledge commands. Also, it was mentioned that implementing the feature of pause tracking, as discussed in Section 3.1.2 would be helpful. If the 3D stick reference pose (thus the pose at tracking activation time) was not chosen well, it sometimes happened that space for hand motion was short because environmental objects like monitor and table were in the way.

For *Teach Pendant*, speech interaction would have been considered valuable, as well as the opportunity of adjusting motion speed. Some participants stated that subtle movements were difficult to perform which seemed to be due to disadvantageous parameterization.

6.4 Fairness

Without doubt, a professional robot controlling system would yield better results, especially for positioning accuracy. One could state that it is not representative, or not “fair” to compare with an unbalanced system like *Teach Pendant*. Regarding to that, one has to admit that *MIMIK* also suffers from serious issues, and could massively be improved as well. So, not the best implementations possible of either got compared, but two rudimentarily, premature prototypes. In fact, their very basic interaction methods got confronted, thus 3D mouse and buttons/GUI vs. 6 DOF tracking and speech input.

Regarding task completion time, one could also state that *Teach Pendant* suffers from slow maximum motion speed of the TCP, which is true, but in fact this is a result of the basic interaction technique, which after all is joystick-like. Although the motion speed can be (and in this work is) controlled in a floating manner, by utilizing an analogue controlling device, the maximum speed must not be chosen too high. Movements in unintended directions may easily cause damage then, and as observed, this in fact often happens to beginners. So, there always is a threshold, which should not be exceeded. Since unexpected movements do not seem to be that common in *MIMIK*, this threshold may be chosen quite higher there.

In addition to variable motion speed, professional systems also provide several convenience features like different adjustments for 3D mouse motion, depending on the position of the operator. Anyway, more of them would have been too much to learn for the participant for an experiment with a duration of one hour. So the interface was reduced to only provide essential features. Similarly, *MIMIK* could be enhanced by several motion speed granularities, switched by speech input. Also, motion and rotation constraints could dynamically be defined by the user; so many collisions could be avoided, gaining improved results for collision count as well as failure count.

Regarding the choice of participants, the study was unaffected by prejudice. Since there were no robot programmers participating, results for *Teach Pendant* were not affected by prior experience with similar systems.

Chapter 7

Conclusion and Future Work

This thesis presented a novel multimodal interaction method for more natural and intuitive operation and programming of robotic arms. For this purpose, assigned input channels were combined with AR-based visualization. The system was especially designed, to meet requirements of high flexibility and fast operation, compromising accuracy aspects.

The results of a user study, confronting it with a rather conventional approach, show that all major design goals were achieved. Participant feedback seems to approve the viability of the chosen input modality combination of high DOF tracking and speech input. Notably, the high rating of speech input seems to confirm its acceptability as a very natural means to operate artificial systems, once current technical constraints are overcome. Although some findings were unexpected, they could be explained, and allow for further improvements.

First of all, the usage of a professional, high-precision industrial robotic arm with 6 DOF would surely improve the experience.

To allow for higher precision, it would be thinkable to introduce several motion speed modes for respective situations, thus higher speed translation for extensive movements, lower speed translation for subtle movements. An analog button could be incorporated into the 3D stick, to control this ratio in a floating manner, or speech commands could be used to switch between different discreet speed modes. Even defining areas, associated with according motion speeds, would be possible, similar to those Ouramdane et al. [21] used for virtual guides. This way, also the option of constraining motions within certain regions could be provided. These regions could either be prototyped by hand, or automatically be generated if environmental data is at hand. For rapid prototyping of approximate environmental geometry, either hand-held tracked objects, such as in Ong et al. [19], or even video assisted geometry modeling, like in *VideoTrace* by van den Hengel et al. [10], would be an option. Given this data, an intelligent facilitating system could be accomplished, adjusting constraints and motion speed on the fly. Also, constraining

DOF by speech commands, to avoid unintended TCP movements, could easily be implemented, e.g. only allowing for forward and backward translations, once the object approach pose was achieved. For trajectory-based tasks like welding, polishing or painting, a useful constraint that could automatically be kept, would be the distance between tool and object surface. For these tasks there would be no need to set keyframes, but trajectories could be sampled, recorded, and replayed. Also, there is the possibility of post-processing those paths, either manually or automatically, so unintended movements can be removed, and even higher precision can be achieved.

AR is still believed to be advantageous for visualization, although it has not been the scope of the user study, and thus has not been proved. The tasks have not been designed to require the support, AR visualization could provide, since they were comparatively simple.

In addition to industrial robotics, there also may be other potential applications, benefiting from this system, like power gear ratio scenarios, where heavy loads have to be moved, e.g. within storage depots. For microsurgery applications, robots could scale the hand movements of the surgeon down to very subtle ones, while compensating for jittering. Also, for teleoperation of robots in space or undersea, or even of evacuation robots in hazardous environments, this technique could be valuable.

However, there still are several issues to deal with. Firstly, there is the problem of ill-detected speech commands, which might get even worse in noisy industrial environments. Secondly, the tracking system has to be robust, which increases hardware costs. Also, HMDs would be useful, so the augmented information is displayed in place, but as already mentioned in Section 1.2.1, current HMDs suffer from several problems. Without doubt, several years of progress in these fields will yield usable, affordable and robust devices to overcome these issues.

Chapter 8

Acknowledgements

I owe my most sincere gratitude to my supervisor, Prof. Michael Haller, for extensive feedback, and especially for his extraordinary support.

Thanks are due to the volunteers, participating in the user study, taking the time to contribute to my work. I also wish to thank Gerold Meisinger and Christian Herzog for reviewing the thesis; and Stefan Wasserbauer for producing the outstanding presentation video.

Moreover, I wish to express my gratitude to Horst Glasner and Thomas Mamoser from *Promot Automation*, Christian Peer from *KUKA Robotics Austria*, and Martin Ankerl and Gernot Stübl from *PROFACTOR* for their cooperation and valuable input.

Finally, I wish to thank my parents and brother, for their encouragement and support.

Appendix A

Additional Tables

A.1 Joint Constraints of the *Lynxmtion AL3C*

Table A.1 summarizes the angle ranges and according servo positions for each joint, as determined empirically, by matching the AR overlay to the physical robot.

Table A.1: The angle ranges and servo positions were determined empirically.

Joint	*Servo position*		*Angle*	
	min	*max*	*min*	*max*
Base	55	246	−95°	86°
Shoulder	71	194	−46°	90°
Elbow	53	210	−8°	149°
Wrist	51	240	−71°	96°
Wrist rotate	57	240	−90°	109°

A.2 GUI Table

Table A.2 lists all elements of the user interfaces for *Teach Pendant* and *MIMIK*, which are shown in Figures 4.18 and 4.19.

Table A.2: Description of GUI elements of prototypes A and B

#	*Description*
A	Display Window
B	Program Window
C	Command Window
D	Teach Pendant Window
E	3D scene
1	Show/hide keyframes
2	Show/hide trajectory
3	Show/hide keyframe basis axes
4	Show/hide robot geometry
5	Switch scene display mode between AR and VR
6	Home—reset scene camera perspective
7	Program Tab: switch to program tab containing instruction sequence and playback elements
8	Keyframe Tab: switch to keyframe tab containing keyframe list
9	Tracking on/off switch: enables/disables the robot's movement according to high 6 tracking data
10	Microphone on/off switch: enables/disables speech recognition
11	3D mouse control on/off switch
12	Delete: deletes selected object
13	New Program: delete all instructions and keyframes
14	Instruction sequence
15	Currently selected element, containing quick-delete button
16	Start/resume playback of program or simulation
17	Pause playback of program or simulation
18	Stop playback of program or simulation, robot returns to default pose
19	Simulation check: if checked, the physical robot is not moved during playback
20	Command list: a listing of currently feasible speech commands
21	Keyframe tagged with release command
22	Ordinary keyframe
23	Keyframe tagged with grab command
24	Increase/decrease base joint position
25	Increase/decrease shoulder joint position
26	Increase/decrease elbow joint position
27	Increase/decrease wrist joint position
28	Increase/decrease wrist rotate joint position
29	Insert keyframe
30	Insert grab command
31	Insert release command
32	Switch to *Base Motion Mode*
33	Switch to *Tool Motion Mode*
34	Toggle *Constrained* and *Free Motion Modes*

Appendix B

Content of the CD-ROM/DVD

File System: ISO9660 + Joilet **Mode:** Single-Session DVD

B.1 Master Thesis

Pfad: /

da_0810305001.dvi . .	Master thesis (without graphics)
da_0810305001.pdf . .	Master thesis
da_0810305001.ps . . .	Master thesis (PostScript-file)

Pfad: /latex/

hgbthesis.cls	Hagenberg thesis class file
hgb.sty	Hagenberg thesis style file
da_0810305001.tcp . .	TeXnicCenter project file
ressources.bib	BibTeX document
*.sty	Auxiliary style files
*.tex	LaTeX document files

Pfad: /latex/images/

*.eps	Encapsulated PostScript images and figures

B.2 Videos

Pfad: /videos/

MIMIK.mov	Presentation video with background music
MIMIKnomusic.mov . .	Presentation video without background music

B.3 Study Data

Pfad: /study/

RawDataQuestionnaire.csv	Raw data, collected during the user study (comma seperated values, german)
RawDataExperiment.csv	Raw data, collected during the user study (Excel spreadsheet, german)
ReportExperiment.xlsx .	Experiment data and analysis (Excel spreadsheet)
ReportQuestionnaire.xlsx	Questionnaire data and analysis (Excel spreadsheet, participant comments in german)

Bibliography

[1] Bischoff, R. and T. Guhl: *Robotic visions to 2020 and beyond—the strategic research agenda for robotics in europe*, Sept. 2009. http://www.robotics-platform.eu/cms/upload/SRA/2010-06_SRA_A4_low.pdf.

[2] Bischoff, R. and J. Kurth: *KUKA: Concepts, tools and devices for facilitating human-robot interaction with industrial robots through augmented reality.* http://www.ismar06.org/data/2a-KUKA.pdf.

[3] Bolt, R.A.: *"Put-that-there": Voice and gesture at the graphics interface.* In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pp. 262–270, New York, NY, USA, 1980. ACM, ISBN 0-89791-021-4.

[4] Bonk, M.: *Programming for the 3D Mouse.* 3Dconnexion, June 2010. Contained in the 3Dconnexion Software Developer Kit, Rev. 5794 from 06/18/2010.

[5] Boudoin, P., C. Domingues, S. Otmane, N. Ouramdane, and M. Mallem: *Towards multimodal human-robot interaction in large scale virtual environment.* In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pp. 359–366, New York, NY, USA, 2008. ACM, ISBN 978-1-60558-017-3.

[6] Buchmann, V., S. Violich, M. Billinghurst, and A. Cockburn: *FingARtips: gesture based direct manipulation in augmented reality.* In *GRAPHITE '04: Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pp. 212–221, New York, NY, USA, 2004. ACM, ISBN 1-58113-883-0.

[7] Chong, J.W.S., A.Y.C. Nee, K. Youcef-Toumi, and S.K. Ong: *An application of augmented reality (AR) in the teaching of an arc welding robot.* In *Singapore-MIT Alliance Symposium*, 2005.

[8] Green, S.A., J.G. Chase, X. Chen, and M. Billinghurst: *Evaluating the augmented reality human-robot collaboration system.* Int. J. Intell. Syst. Technol. Appl., 8(1/2/3/4):130–143, 2010, ISSN 1740-8865.

[9] Green, S.A., X. Chen, M. Billinghurst, and J.G. Chase: *Collaborating with a mobile robot: an augmented reality multimodal interface.* 17th International Federation of Automatic Control (IFAC-08) World Congress, 6-11 Jul:15595–15600, 2008.

[10] Hengel, A. van den, A. Dick, T. Thormählen, B. Ward, and P.H.S. Torr: *VideoTrace: rapid interactive scene modelling from video.* In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, p. 86, New York, NY, USA, 2007. ACM.

[11] House, B., J. Malkin, and J. Bilmes: *The VoiceBot: a voice controlled robot arm.* In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pp. 183–192, New York, NY, USA, 2009. ACM, ISBN 978-1-60558-246-7.

[12] Kalkofen, D., E. Mendez, and D. Schmalstieg: *Interactive focus and context visualization for augmented reality.* In *ISMAR '07: Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1–10, Washington, DC, USA, 2007. IEEE Computer Society, ISBN 978-1-4244-1749-0.

[13] Kobayashi, K., K. Nishiwaki, S. Uchiyama, H. Yamamoto, S. Kagami, and T. Kanade: *Overlay what humanoid robot perceives and thinks to the real-world by mixed reality system.* In *ISMAR '07: Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1–2, Washington, DC, USA, 2007. IEEE Computer Society, ISBN 978-1-4244-1749-0.

[14] Kucuk, S. and Z. Bingul: *Robot Kinematics: Forward and Inverse Kinematics, Industrial Robotics: Theory, Modelling and Control.* Pro Literatur Verlag, Germany / ARS, Austria, 2006. http://sciyo.com/articles/show/title/robot_kinematics__forward_and_inverse_kinematics, ISBN: 3-86611-285-8.

[15] Lee, M. and M. Billinghurst: *A wizard of Oz study for an ar multimodal interface.* In *ICMI '08: Proceedings of the 10th international conference on Multimodal interfaces*, pp. 249–256, New York, NY, USA, 2008. ACM, ISBN 978-1-60558-198-9.

[16] Martz, P.: *OpenSceneGraph Quick Start Guide: A Quick Introduction to the Cross-Platform Open Source Scene Graph API.* Skew Matrix Software LLC, Louisville, CO 80027 USA, 2007. http://www.lulu.com/product/paperback/openscenegraph-quick-start-guide/1144911.

[17] Milgram, P., S. Zhai, and D. Drascic: *Applications of augmented reality for human-robot communication.* In *IROS93: Int'l Conf. on Intelligent Robots and Systems, Yokohama, Japan*, 1993.

[18] Occupational Safety & Health Administration: *OSHA Technical Manual (OTM), Section IV, Chapter 4. Industrial Robots and Robot System Safety*, Nov. 2008. http://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html.

[19] Ong, S.K., J.W.S. Chong, and A.Y.C. Nee: *Methodologies for immersive robot programming in an augmented reality environment.* In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pp. 237–244, New York, NY, USA, 2006. ACM, ISBN 1-59593-564-9.

[20] The Open Robot Control Software Project: *Orocos Kinematics and Dynamics Library (KDL) API Reference*, June 2010. http://people.mech.kuleuven.be/~orocos/pub/devel/kdl/latest/api/html/.

[21] Ouramdane, N., S. Otmane, F. Davesne, and M. Mallem: *FOLLOW-ME: a new 3d interaction technique based on virtual guides and granularity of interaction.* In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pp. 137–144, New York, NY, USA, 2006. ACM, ISBN 1-59593-324-7.

[22] Perzanowski, D., A.C. Schultz, W. Adams, E. Marsh, and M. Bugajska: *Building a multimodal human-robot interface.* IEEE Intelligent Systems, 16(1):16–21, 2001, ISSN 1541-1672.

[23] Pettersen, T., J. Pretlove, C. Skourup, T. Engedal, and T. Løkstad: *Augmented reality for programming industrial robots.* In *ISMAR '03: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, p. 319, Washington, DC, USA, 2003. IEEE Computer Society, ISBN 0-7695-2006-5.

[24] Reisman, J.L., P.L. Davidson, and J.Y. Han: *A screen-space formulation for 2d and 3d direct manipulation.* In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pp. 69–78, New York, NY, USA, 2009. ACM, ISBN 978-1-60558-745-5.

[25] Salber, D., A. Blandford, J. May, R.M. Young, J. Coutaz, and L. Nigay: *Four easy pieces for assessing the usability of multimodal interaction: the CARE properties.* In *Proceedings of INTERACT'95*, pp. 115–120. ACM Press, 1995.

[26] Schraft, R.D. and C. Meyer: *The need for an intuitive teaching method for small and medium enterprises.* In *ISR 2006 - ROBOTIK 2006: Proceedings of the Joint Conference on Robotics*, vol. 1956. Citeseer, 2006. http://www.smerobot.org/08_scientific_papers/#2006.

[27] Schwerdtfeger, B., D. Pustka, A. Hofhauser, and G. Klinker: *Using laser projectors for augmented reality.* In *VRST '08: Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pp. 134–137, New York, NY, USA, 2008. ACM, ISBN 978-1-59593-951-7.

[28] Wang, R.Y. and J. Popović: *Real-time hand-tracking with a color glove.* In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pp. 1–8, New York, NY, USA, 2009. ACM, ISBN 978-1-60558-726-4.

[29] Zaeh, M. and W. Vogl: *Interactive laser-projection for programming industrial robots.* In *ISMAR '06: Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 125–128, Washington, DC, USA, 2006. IEEE Computer Society, ISBN 1-4244-0650-1.

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —

Breite = 100 mm
Höhe = 50 mm

— Diese Seite nach dem Druck entfernen! —