ingenieur
wissenschaften
htw saar

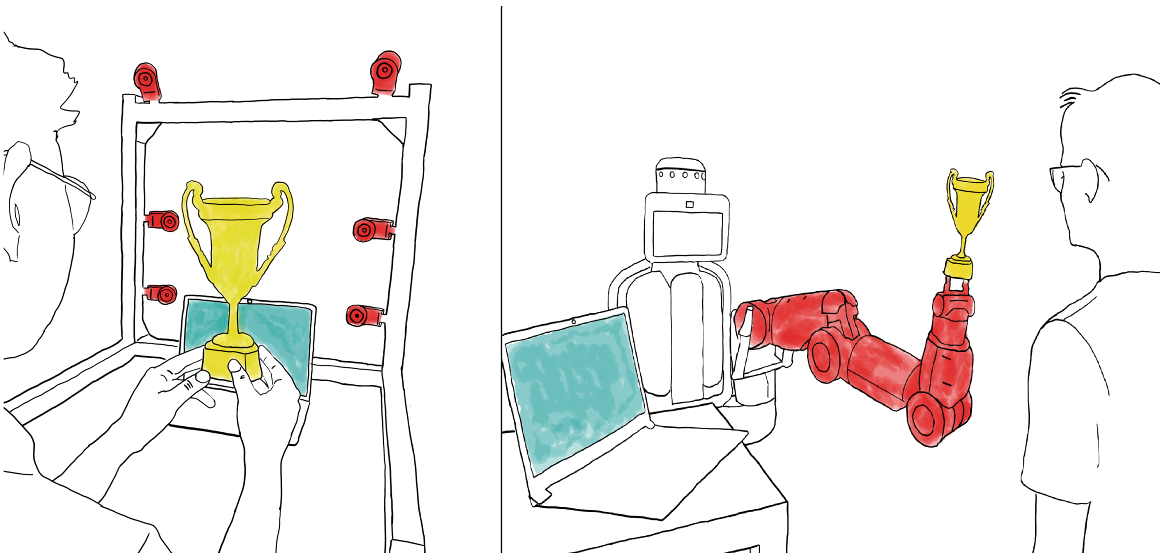
Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

STL SYSTEMTECHNIKLABOR

# Implementing a Humanoid Tele-Robotic Prototype for Investigating Issues in Remote Collaboration

Martin Feick

Technical Report – STL-TR-2017-02 – ISSN 2364-7167



stl.htwsaar.de

Technische Berichte des Systemtechniklabors (STL) der htw saar
Technical Reports of the System Technology Lab (STL) at htw saar
ISSN 2364-7167

Martin Feick: Implementing a Humanoid Tele-Robotic Prototype for Investigating Issues in Remote Collaboration
Technical report id: STL-TR-2017-02

First published: October 2017
Last revision: January 2018
Internal review: Anthony Tang, André Miede

For the most recent version of this report see: https://stl.htwsaar.de/

Title image source: Martin Feick

htw saar – Hochschule für Technik und Wirtschaft des Saarlandes (University of Applied Sciences)
Fakultät für Ingenieurwissenschaften (School of Engineering)
STL – Systemtechniklabor (System Technology Lab)
Prof. Dr.-Ing. André Miede (andre.miede@htwsaar.de)
Goebenstraße 40
66117 Saarbrücken, Germany
https://stl.htwsaar.de

# IMPLEMENTING A HUMANOID TELE-ROBOTIC PROTOTYPE FOR INVESTIGATING ISSUES IN REMOTE COLLABORATION

MARTIN FEICK

Bachelor's Thesis

Applied Computer Science
Faculty of engineering
University of Applied Sciences, Saarland
Hochschule für Technik und Wirtschaft des Saarlandes

Submitted: 28th September 2017

Martin Feick: *Implementing a Humanoid Tele-Robotic Prototype for Investigating Issues in Remote Collaboration* , © April 2017 - September 2017

SUPERVISORS:
Prof. Dr. Anthony Tang
Prof. Dr. André Miede

LOCATION:
University of Calgary, Alberta (Canada)

TIME FRAME:
April 2017 - September 2017

# ABSTRACT

We designed and developed a novel system, ReMa (Remote Manipulator), for supporting remote collaboration on physical tasks through a physical telepresence humanoid robot. The system captures and reproduces object manipulations on a proxy object at a remote location.

The prototype combines latest robotics and motion capture technologies, exploring their capabilities and limitations. We found that directly mapping human and robot action is problematic due to the arrangement and limits of the robot joints.

We applied ReMa to investigate how limited perspective in current video-mediated systems affects remote collaboration. We also explored the impact of a physical proxy manipulated by a remote person. We conducted two user studies and contrasted the results with recent research systems designed for remote collaboration.

Our main findings are: (1) a shared perspective is more effective and preferred compared to the opposing perspective offered by conventional video chat systems, and (2) the physical proxy and video chat complement one another in a combined system: people used the physical proxy to understand objects, and used video chat to perform gestures and confirm remote actions. These research findings validate both the design and implementation of ReMa as an effective research platform.

# PUBLICATIONS

Some materials, ideas and figures from this thesis also appear in the following publications:

Martin Feick, Terrance Mok, Anthony Mok, Lora Oehlberg, and Ehud Sharlin. (2018). Perspective on and Re-Orientation of Physical Proxies in Object-Focused Remote Collaboration. In *CHI 2018: Proceedings of the 2018 SIGCHI Conference on Human Factors in Computing Systems*. ACM. Montréal, Canada

Martin Feick, Lora Oehlberg, Anthony Tang, André Miede, and Ehud Sharlin. (2018). The Way You Move: The Effect of a Robot Surrogate Movement in Remote Collaboration. *HRI '18: Proceedings of the Companion of the ACM/IEEE International Conference on Human-Robot Interaction*. Chicago, USA

*The separation of talent and skill is one of the greatest misunderstood concepts for people who are trying to excel, who have dreams, who want to do things. Talent you have naturally. Skill is only developed by hours and hours and hours of beating on your craft.*

**Will Smith**

# ACKNOWLEDGMENTS

This thesis was written at the University of Calgary in Canada, which would not have been possible without the support of many people. Therefore, it is my pleasure to thank Professor Anthony Tang and Professor André Miede who made this incredible experience possible. I am grateful for their continuing support during my thesis. I also owe my deepest gratitude to Professor Lora Oehlberg and Professor Ehud Sharlin for their intellectual support and their time.

It was an honor for me to be a part of the interaction Lab (iLab) with all its amazing people. Working with all you made it a unique experience, which I will never forget. I would like to thank Sowmya Somanath for helping me get to know the University of Calgary. A special thanks to Kevin Ta and Terrance Mok for their great ideas and their help with both studies, and all iLabbers who voluntarily participated in my pilot studies.

I wish to thank Thomas Quitter and Betina Lang for helping me with the preparation for my stay in Canada.

Another big thanks goes to Dean Hergenhein and Chewie for being such great roommates, giving me an introduction to the Canadian lifestyle and becoming good friends. Furthermore, I would like to thank Karthik Mahadevan, Jessi Stark and Markus Tessmann for the awesome time we had.

At my home institute, I would very much like to thank Nico Kleer and Marek Kohn for sharing the same enthusiasm on solving and discussing problems. We always had a lot of fun, even during the exam preparations.

Finally, I would like to thank my whole family. Without their support this stay would not have been possible.

# CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

## LISTINGS

# 1 INTRODUCTION

We are living in a global connected world, where people from different continents work together to solve problems. Often, these problems involve physical objects where collaborators touch, work and talk about these objects in relation to their environment. Physical objects play an important role in the world. People's experiences are based on repeated interaction with various objects, in response to performance success and errors [31]. Leveraging their past experience, people are capable of manipulating even unknown objects without having knowledge about their properties and behaviors [52]. However, when people collaborate on physical tasks involving objects it affects their communication. For instance, Kruger et. al. [33] define three distinct roles for how people use object orientation in collaboration: to understand, to coordinate and to communicate actions with one another.

*Introducing a common issue in remote collaboration*

Moreover, when collaboration becomes remote, i.e. one person is physically separated from the other, these conversations are more challenging because spatial references and gestures that are clear when co-present are difficult to interpret. Human-Computer Interaction (HCI) researchers have focused on these challenges and recent research in that field has demonstrated that video can indeed help remote collaboration [10, 14, 15, 19, 34]; however, effectively positioning the video and controlling the object present new challenges for collaborators in trying to establish a joint, effective understanding of what is happening [25, 38, 50].

Other researchers try to address these issues with virtual and augmented reality [2, 5], but it is difficult to create unique detailed models from objects, because they are often too complex [15, 42]. Once we use generalized models we lose the unique properties and behaviours of objects. We believe that physicality of objects is important [36], and since robots are now more present in everyday life [32] we wanted to explore how we can use them to support remote collaboration on physical object-related tasks. Moreover, humanoid robot research such as Boston Dynamics[1] with their Atlas robot, continuously try to adapt human behaviours and capabilities to robots. This could lead the way to the vision of using robots as surrogates for humans. To investigate the opportunities of this vision, we focused on a smaller-scale problem that we can explore with current technologies.

1 www.bostondynamics.com

## 1.1 MOTIVATION & RESEARCH QUESTION

In this thesis, we are specifically interested in how robots can support object-focused remote collaboration. Particularly, we are interested in real world scenarios where both collaborators have a copy or a proxy of the object in question. This type of situation may occur, for instance, in a remote assistance situation (e.g. [45], where one collaborator is asking for help in repairing a physical object), or in a remote critique scenario (e.g. [42], where collaborators are discussing and iterating on the design of a physical object) or in very simple scenarios where collaborators may seek to understand how an object works [7, 25]. Object-focused collaboration, particularly when both collaborators have independent objects, presents challenges for conventional remote collaboration technologies such as video chats. First, objects can be viewed from many different perspectives by either collaborator, at any time. Second, the objects themselves are tangible, exist in the physical world, and have details that are difficult to capture or convey via a virtual representation as mentioned before [15, 42]. In the face of these challenges, conventional two-way video chats demand considerable "meta" discussion by collaborators to establish joint understanding or common ground [8] during collaborative activity (e.g. how to orient, view or manipulate the object). Our interest is in understanding how varying collaborators' perspectives on an object (e.g. shared vs. opposing) helps or hinders collaboration. Further, if we can automatically reorient a physical object or a proxy with the help of a robot, how does this help collaborative activity?

*Our research question in Human-Computer Interaction*

Additionally, we wanted to explore the capabilities and limitations of a current state-of-the-art humanoid robot that supports the above elucidated issues in remote collaboration on physical object-related tasks. Previous technical projects with telepresence robots directly map human actions such as hand movements or gesture to a robotic end-effector [37, 40, 41], but there are still many issues due to the different kinematic and speed capabilities. Rakita et. al. [51] showed in their work that relaxing the direct mapping between a human hand and robot end-effector can lead to better results. Our goal is to support remote collaboration on physical object-related tasks and therefore we wanted to investigate alternative ways to use telepresence humanoid robots.

This thesis covers both, a technical and theoretical research question, as we just discussed. It provides an alternative approach for manipulating a remote workspace i.e. manipulating an object via a telepresence robot, and answers the question *how does a robot object manipulator impact remote collaboration on object-related physical tasks?*

## 1.2 THESIS OBJECTIVES

To determine how we can support remote collaboration on physical object-related tasks, we first examine the existing literature on collocated collaboration as well as literature on remote collaboration on physical tasks with a focus on objects. From this, we designed and implemented the novel system ReMa which allowed us to explore how perspective and orientation is used in remote collaboration on physical object-related tasks. Furthermore, we explored the impact of a physical proxy manipulated by a remote person and how people make use of the system.

*Explaining our workflow*

During the implementation, we were faced with current challenges in robotics that we needed to explore. To understand the current limitations of a Baxter robot, we performed a technical analysis of the robot. Following the results of this analysis we developed the ReMa system. We conducted two studies helping us understand how perspective and orientation is used in remote collaboration with 3D physical objects. With the analyzed and evaluated results, we were able to suggest design implications for future systems. Furthermore, we discuss limitations of both our system and our two studies, and how future work can address open questions. Moreover, we provide a technical Robot Operation System (ROS) package of our system making it accessible for future researcher.

In this thesis we make five contributions:

*Contributions of this thesis*

1. A standard ROS package for the Baxter community using an alternative Inverse Kinematics algorithm which works with a physical Baxter robot, as well as in the simulation.
2. A technical evaluation of the research robot Baxter and Inverse Kinematics in relation to (3) and (4).
3. A summary of past work on the role of perspective and orientation for remote object-focused collaboration
4. The design and implementation of a novel system (ReMa) that physically communicates perspective and orientation to a remote site
5. Two user studies that explore how perspective and orientation is used in remote collaboration with 3D physical objects

We distinguish between technical and HCI contributions and, because this thesis covers both, it is structured as shown in the next section.

## 1.3 OVERVIEW

This work is divided into ten chapters which are structured as followed:

Chapter 2 provides the requisite technical background specific to the implementation and technical evaluation of the system. It introduces the mathematical concept of quaternions, and provides a more detailed view into the ROS, as well as basics about our core problem, Inverse Kinematics. Furthermore, it introduces basic knowledge of the Baxter research robot and the OptiTrack system.

Chapter 3 directly addresses contribution (3) to provide a detailed view into the related work on supporting remote collaboration on physical object-related tasks. It points out why we decided to design our system for investigating the difficulties in remote object-focused collaboration.

Chapter 4 directly addresses contribution (4): Based on the previous chapter, we explain the design of the novel system ReMa.

Chapter 5 focuses on the detailed setup of our system. This chapter provides information about system components, and the specific configurations we chose for our final implementation.

Chapter 6 addresses contribution (1) and (2). We show our implementation of the ROS package. Moreover, we discuss the Inverse Kinematics experiment and its results.

Chapter 7 addresses contribution (2) by providing an evaluation of the Baxter robot in relation to our project. Furthermore, it also highlights general issues and limitations of the whole system we designed and developed.

Chapter 8 addresses contribution (5). We conducted two user studies to understand the impact of perspective. We discuss the design of each study, providing background information of participants as well as data analysis/collection. Finally, we also elucidate the findings from our two user studies.

Chapter 9 also addresses contribution (5) and contrasts our findings with recent research findings. Moreover, we also discuss the limitations of our system as well as how future work can address the limitations and open questions. Finally, we provide design implications for future systems.

Chapter 10 summarizes the thesis.

# 2 FOUNDATIONS

In this chapter we provide the mathematical and technical background for our project. We start with introducing four-dimensional numbers called quaternions needed to describe the orientation of objects. Next, we give a brief introduction to our core problem, Inverse Kinematics (IK). Finally, we provide foundations about the Robot Operation System (ROS), the Baxter robot and the tracking system OptiTrack we used, in relation to our project.

## 2.1 MATHEMATICAL FUNDAMENTALS

In this section we briefly introduce four-dimensional numbers, *Quaternions* $\mathbb{H}$, and we show their advantages for our project. Quaternions are a new type of numbers beyond the real numbers $\mathbb{R}$ and even beyond complex numbers $\mathbb{C}$. Nowadays, quaternions are used to describe orientation (e.g. of air planes, space shuttles, smartphones, etc.). In our project, quaternions describe an orientation of a *rigid body* or robot frames (see Section 2.2.4), in relation to the original coordinate system. First, we give a brief throwback about how we can *travel* to a specific point in a coordinate system in different dimensions. If we have a one dimensional coordinate system, we can move a point to the right with positive numbers, and to the left with negative numbers $\in \mathbb{R}$, depending on the axis representation. Numbers in $\mathbb{R}$ are one-dimensional numbers that allow use of very basic movements along a fixed axis.

*Introduction to quaternions*

Let us suppose we want to move a point in two dimensions. We can achieve this with two dimensional-numbers called complex numbers $\mathbb{C}$. With complex numbers we are able to *travel* to a location in our coordinate system by adding numbers, and furthermore we are can rotate in two dimensions by multiplying these numbers. Complex numbers have the form $a + bi$, where $a, b \in \mathbb{R}$ and $i$ is the imaginary part. The multiplication of a complex number by $i$ results in a 90 degree counterclockwise rotation of a point around the origin of the coordinate system. However, to use complex numbers for computations we have to define a rule for $i$:

$$i * i = i^2 = -1$$

Certainly, it is also possible to rotate around an axis with other angles by modifying the multiplication factor. For example, a 45 degree counterclockwise rotation can be achieved by multiplying $p$ with $(1 + i)$.

$$p = (3+2i) * (1+i) = (3+2i) + (3i + 2i^2) = 1 + 5$$

However, to determine the rotation of a point in a three dimensional space we do need four dimensional numbers, called quaternions $\mathbb{H}$. The form of a quaternion is $a + bi + cj + dk$ with $a,b,c,d \in \mathbb{R}$ and $i, j, k$ are the quaternions units [23, 43]. Also quaternions have a rule which looks similar to the rule for complex numbers.

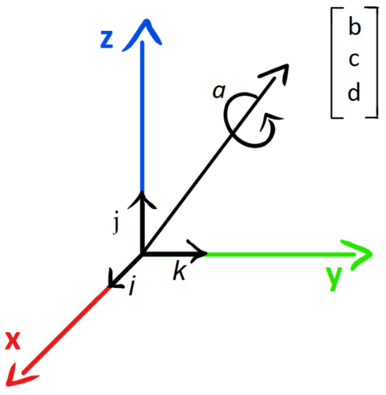


Figure 1: Quaternions

$$i^2 = j^2 = k^2 = i * j * k = -1$$

Quaternions contain real and complex numbers. However, due to the higher abstraction we loose a property of the real and complex numbers. Quaternions are non-commutative, hence we need a definition for the multiplication (see Table A.1). Following, we can rotate a point in three dimensions by multiplying quaternions. Figure 1 provides a visualization of a quaternion, which we explain on a higher abstraction level. To determine the orientation of a *rigid body*, quaternions use a vector $\vec{v}$ $(v1,v2,v3)$ and an angle $\theta$ for rotating around this vector $\vec{v}$. Vector $\vec{v}$ passes through our *rigid body* which can be placed anywhere in a coordinate system. We can rotate the pre-positioned vector $\vec{v}$ around its own axis with an angle $\theta$. Summarized, vector $\vec{v}$ determines the rotation axis that simultaneously represents the position of the *rigid body*, and $\theta$ gives the angle around the rotation axis [23, 43]. In Figure 1 we see the real numbers $a$,$b$,$c$ and $d$ which we define as:

*How do quaterions work?*

$$\begin{aligned} a &= \cos(\theta/2) \\ b &= v1*\sin(\theta/2) \\ c &= v2*\sin(\theta/2) \\ d &= v3*\sin(\theta/2) \end{aligned}$$

Summarized, quaternion multiplications describe rotations in three dimensions. We have to redefine the multiplication for quaternions (see Section A.1), and it has to be a unit quaternion with $norm = 1$.

$$U_q = \frac{q}{||q||} \quad \forall q \neq 0$$

When we use unit quaternions the inverse quaternion is $q^{-1} = a - bi - cj - dk$ which is efficient for calculations, and it does not change the size of the vector (rigid body). *Note:* The multiplication of two unit quaternions results in a unit quaternion. [23]

Before we finally show the advantages of quaternions, following is an example with a 90 degree rotation around the x-axis, and following that another 90 degree around the y-axis. An example with the commonly known Euler angles can be found in Section A.1 .

$$a_x = 0.707; b_x = 0.707; c_x = 0.000; d_x = 0.000$$
$$a_y = 0.707; b_y = 0.000; c_y = 0.707; d_y = 0.000$$
$$q = (a_x + b_x i + c_x j + d_x k) * (a_y + b_y i + c_y j + d_y k)$$

Following the above introduced rules for quaternions we get a solution in the form $a + bi + cj + dk$. The result is $q = 0.5 + 0.5i + 0.5j + 0.5k$ which describes the orientation in the coordinate system. Now, we can rotate an arbitrary vector $\vec{q}$ by multiplying $p * q * q^{-1}$ to determine the position of the rotate vector $\vec{q}$.

*Advantages for our project*

Finally, we want to discuss the advantages to Euler's representation related to our project. Quaternions are very powerful, and we only provided a brief overview related to our project. Quaternions do not suffer from gimbal lock (see Section A.1). Additionally, we can use interpolation between the quaternions, and linear matrix interpolation between two orientations is possible. Moreover, the computational complexity is significantly lower. We have three matrix multiplications when using Euler angles; each has a complexity of $\mathcal{O}(n^3)$. However, with quaternions we only need a quarter of the computational complexity. They also work independently of the coordinate systems representation (left-handed or right-handed), due to the uniqueness of quaternions, and we can easily calculate the inverse. Finally, with unit quaternions we can generate a rotation matrix in constant time $\mathcal{O}(1)$ (see Section A.1). These advantages led the decision to use quaternions for developing our prototype (see Chapter 6).

## 2.2 ROS – ROBOT OPERATION SYSTEM

*Properties of the robot operation system*

ROS – Robot Operation System [48] is a common meta-operating open source framework supporting the development of applications for robots. It is a software providing various libraries and tools. Regarding their package structure, existing packages can be easily reused in other projects. They are fully independent, and therefore applicable in other projects. Thus, ROS has strict standards which must be considered when developing applications with it. ROS provides both a C++ and Python interface. The corresponding libraries are called roscpp and rospy. In the following sections we elucidate the important elements of ROS to provide a basic understanding of how it works. However, it is only a brief introduction that is essential to understand the next chapters; for further details see documentation[1].

[1] http://wiki.ros.org/

There are different ROS distributions, for our approach we used ROS *Indigo* that works with Ubuntu version 12.04.03, due to the support for our Baxter robot following Rethink Robotics recommendation.

*Simulation tools Gazebo and Rviz*

We also want to briefly introduce two of the most common simulation environments for robots called Gazebo[2] and Rviz[3]. Simulation environments allow us to the test (e.g. new algorithms, configuration settings, sensors etc.) in a safe environment, very similar to the real world. We can experiment with our robot without taking any risks. In addition to that, in most cases robots are expensive, and we did not want to damage the robot (e.g. servos or sensors) or the environment (e.g. furniture).

Gazebo and Rviz also provide various tools which support people during the development, making it easier to understand the sometimes very complex processes (see Section 2.2.4) or helping with debugging. Furthermore, we were able to develop on our robot without being physically close to it. All in all, our recommendation is to use a simulation environment to improve to productivity, and moreover to have a safe experiment environment.

### 2.2.1 *Filesystem*

In this section we describe the ROS filesystem consisting of the below listed elements:

- Packages: ROS is organized in packages. They are the most atomic, lightweight and independent items in ROS that can be easily reused. This follows the concept of *build your work on the top of others*. A package usually includes ROS nodes, ROS-independent library, a data set, configuration files and third-party piece of software[4]. Essentially, a package is a folder with all the above mentioned content.

- Metapackages: Representing a group of related packages.

- Package Manifests: They are package.xml metadata files which contain important information about a specific package such as dependencies. These manifests must follow a predefined structure.

- Message types (.msg): Message descriptions are used to send information to communicate within ROS (Nodes). They are written in a specific message description language. Further details are available in Chapter 6.

2 http://gazebosim.org/
3 http://wiki.ros.org/rviz
4 http://wiki.ros.org/Packages

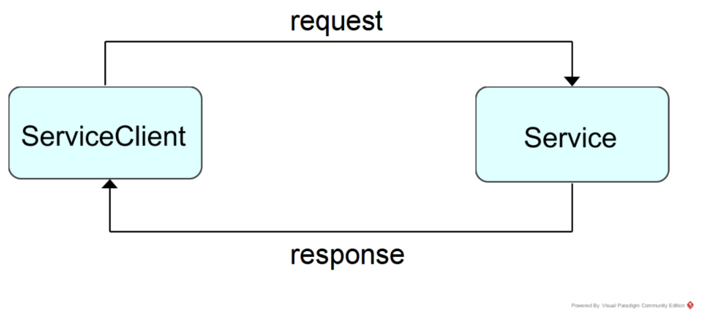


Figure 2: ROS Service

- Service types (.srv): Services coordinate the *request/response* communication via messages within ROS (Nodes). They are written in a service description language similar to messages. There are two different ways to communicate information in ROS, either over services or topics using the publisher-subscriber pattern (see Section 2.2.2). Generally, services are used when a remote procedure call (RPC) is required, which is not possible with topics.

*Catkin for building workspaces*

Above, we introduced the most granular element in ROS, called packages. To build a package we decided to use the package builder Caktin. It provides a very effective way to structure workspaces and packages. Catkin allows users to build multiple, but independent packages together all at once[5]. After initializing the Catkin workspace, users are able to modify, create, build and install certain Catkin packages. When building the workspace, Catkin automatically compiles source code in different programming languages and checks linked dependencies and libraries. To achieve this, every binary Catkin package has environment setup files (e.g. we used the bash shell and therefore our file was called *setup.bash*). Building a Catkin workspace can be achieved through sourcing the setup file with the following command:

```
# source /opt/ros/indigo/setup.bash
```

Subsequently, build the workspace of packages using:

```
# catkin_make
```

*Important to consider with workspaces*

Modifications of the code, dependencies and so on always result in a rebuild of the workspace. Otherwise, the next ROS launch will not consider the changes. With the command *rospack list*, users are able to check which packages are currently installed.

A ROS package is a folder that contains the below listed content files and subfolders. Following a first introduction in this section,

5 http://wiki.ros.org/catkin/workspaces

here is a more detailed look at the structure we must consider for developing code for ROS in Chapter 6:

*ROS package structure*

- include/package_name: header files (e.g. C++ headers)
- msg/: simple folder containing all messages
- src/package_name/: this folder contains the source files (Python, C++)
- srv/: contains all services
- scripts/: executable scripts
- CMakeLists.txt: used to build the software package
- package.xml: defines the properties of packages (name, version, license)

### 2.2.2 *Computation Graph*

*What is a computation graph?*

In this section we describe the general ROS workflow. Therefore, we examine the ROS concept. For additional information look at the website[6]. Essentially, ROS uses the following components: ROS Nodes, ROS Master, Parameter Server, Messages (see Section 2.2.1), Services (see Section 2.2.1), Topics and Bags. All of them provide different data during a computation process. ROS wiki describes their *Computation Graph* as a peer-to-peer network which processes all data from the above mentioned components together[6]. It is a distributed system where each component provides specific information for the calculation process.

- *ROS Nodes:* A Node is a process within ROS that performs computations such as path planning. Nodes are registered at the ROS Master to provide a specific service or functionality in the system (see Figure 3). Usually, there are many different nodes which are reachable via an Uniform Resource Identifier (URI). For instance, nodes can be programmed in different programming language using the ROS libraries roscpp or rospy.
- *ROS Master:* The ROS Master provides a registration and naming services (see Figure 3). Otherwise, a ROS Node is not able to find other ROS Nodes. That means they cannot exchange data such as messages. The ROS Master enables the communication channels within ROS, as well as to the network (see Section 5.2). It also provides the Parameter Server.

6 http://wiki.ros.org/ROS/Concepts

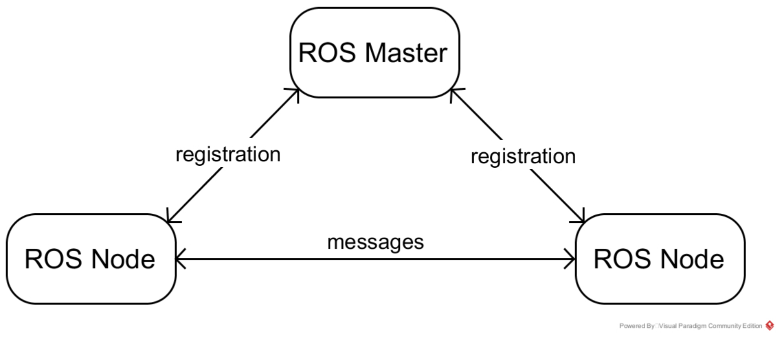


Figure 3: ROS Master communication

- *Parameter Server:* The Parameter Server is running on the ROS Master. It stores important information with a key in a *dictionary* providing it in the ROS network. Stored data are accessible through the above mentioned key. Section 2.2.3 provides a ROS command that requests information from the parameter server.
- *Topics:* Topics are used to exchange messages between different ROS Nodes. Topics, or sometimes also called buses, work with the publisher/subscriber design pattern. The topic name identifies the message that has been published from a specific node. All subscribers (nodes) can use the published message for their own computations. A subscriber registers itself with a topic. A publisher sends messages to the topic which can then be seen by the subscribers. Figure 4 shows the workflow of a single message published by a node through a specific topic. A topic can also have various publishers and subscribers (not illustrated in Figure 4). Topics have strict ROS message types and nodes can only exchange messages when they match the message type.
- *Bags:* The Bag file format is used to record and playback serialized message data such as sensor data. *Note:* We neither describe nor configure them in our approach.

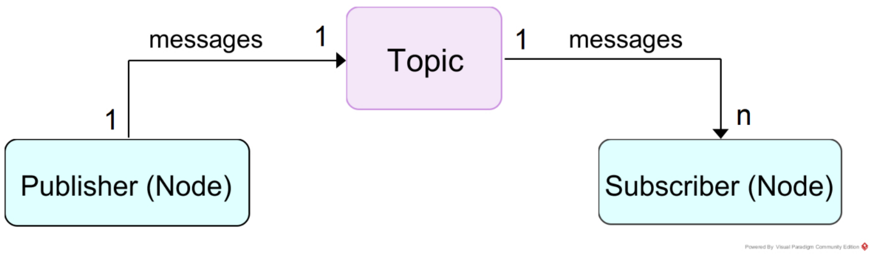


Figure 4: Publisher - Subscriber design pattern

Listing 1: Baxter URDF Model

```
 <link name="right_gripper">
 <gravity>0</gravity>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <box size="0.01 0.01 0.01"/>
      </geometry>
      <material name="black">
        <color rgba="0 0 0 1"/>
      </material>
    </visual>
    <inertial>
      <origin rpy="0 0 0" xyz="0.000000 0.000000 0.000000"/>
      <mass value="0.0001"/>
      <inertia ixx="1e-08" ixy="0" ixz="0" iyy="1e-08" iyz="0"
          izz="1e-08"/>
    </inertial>
</link>
```

### 2.2.3 *Robot Model*

*The robot model is used to...*

In this section we introduce the robot model provided through the Unified Robot Description Format (URDF). We will use the URDF information especially in Chapter 6. The robot model is stored in a XML formatted file, baxter_urdf.xml. It describes the robot with all its properties such as information about kinematics, dynamics and sensors. The UDRF is dynamically generated on every robot start, and eventually gets loaded to the parameter server providing information in ROS. Generally, the URDF has a tree structure that can be used to visualize the sometimes confusing XML file. Another tool we used in our approach is the xml macro language *Xacro*[7]. It supports writing and modifying of the URDF to produce a more elegant, readable and clear XML. Usually, a robot model xml file contains all information about a robot, thus it difficult to read due to its size. Listing 1 provides a code snippet of the baxter_urdf.xml. This is an example of the *right_gripper* that shows the provided data in the baxter_urdf.xml such as default position and orientation.

The RSDK on Baxter provides a package called baxter_description containing the URDF and other accompanying meshes. The currently stored URDF on the parameter server is exportable through a ROS command, to ensure the latest version of the URDF is loaded.

```
$ rosparam get -p /robot_description | tail -n +2 >
                    baxter_urdf.xml
```

7 http://wiki.ros.org/xacro

In general, with the *rosparam* command it is possible to export stored information from the parameter server verifying the integrity of the loaded data (see Section 2.2.2).

### 2.2.4 *Tf - Transform*

*How can we localize the joint positions?*

Before we begin to describe the complex transform library (tf), we explain how the robot works in further detail. The position of each robot element/joint is always given through a coordinate system. That means we have various coordinate systems in our robot (see Figure 5). Basically, we have an original coordinate system called */base*, and the position of the */right_gripper* coordinate system is relative to the original */base* system. This applies for each coordinate system in our robot. Figure 5 (d) shows all the coordinate systems of Baxter's kinematic chain elements (see Section 2.4) provided in the URDF. The pink arrows, and yellow lines symbolize the relation to the */base* frame. Conclusively, we are able to determine the joint positions */right_gripper* relative to our robot */base*. Tf represents the relation between two frames with a 6DoF pose, consisting of a translation and a rotation. Rviz provides tools/programs to request information about the latest positions of the robot frames. Tf frames in ROS are right-handed Cartesian coordinate systems with $x$ forward (red), $y$ left (green) and $z$ (blue) up (see Figure 5 (a)).

*What is transform - tf?*

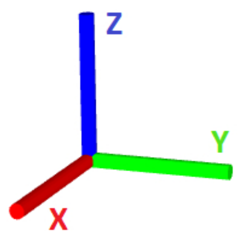
Tf is a library that runs in ROS, tracking all different coordinate frames. It is a standardized tree structured protocol for distributing information about multiple coordinate frames. For instance it is possible to ask the tf library: *What is the position of the robot's gripper relative to its torso?* This information is essential when working with a robot. It is also possible to exchange data between more than one robot, which was not necessary for our project.

There are two different kinds of tf nodes called broadcaster/publisher and listener, written in Python or C++. Following we give a brief description about them:

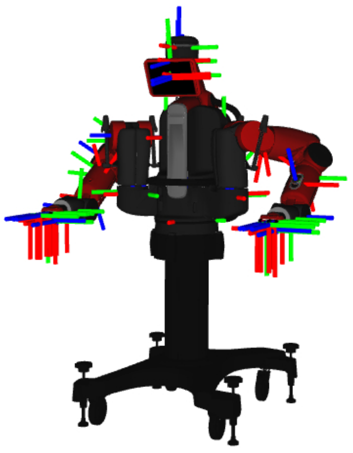
1. *Listener:* These nodes are listening for coordinate frames that get published in the system.
2. *Broadcaster/Publisher:* These nodes inform the distributed system about coordinate frames related their position. Each element of the robot usually has its own broadcast node.

*Mechanisms to overcome issues in the distributed system ROS*

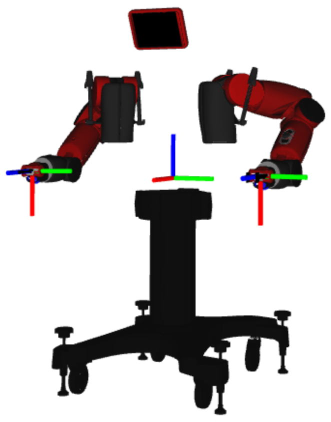
Tf does not use a central storage like the parameter server (see Section 2.2.2). Following that, a node does not have any information about the history of the system when it starts. The advantage of tf is, that we are able to get information directly, and furthermore we know where it came from. In addition to that, without using tf we have two significant problems. First, in distributed systems it is a



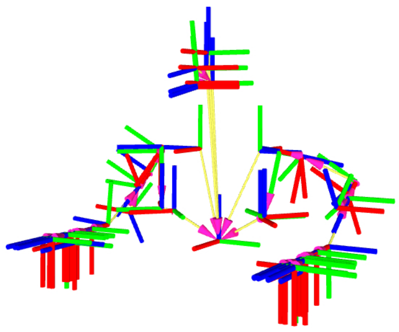
(a) Tf-coordinate system



(b) All Tf's in Baxter



(c) /base related to /grippers



(d) Tf tree on Baxter

Figure 5: Robot Transforms visualized in Rviz

common problem that not all data are available at a certain moment. The last update can be either five milliseconds or five seconds old. The system automatically keeps track of all the updates. Moreover, it provides a timestamp and a frame id. Hence, it is always clear where our data came from and how old they are. Conclusively, we are able to ensure that we use the right data (e.g. the latest updates of our frames or if we want to compare data from a specific point in the past (timestamp)). If we two specific frames from our robot, tf uses a deep search in the tree to find the requested data (see Figure 6).

*Tf - capabilities in ROS*

It is possible to visualize the tf tree getting a better understanding of it. Figure 6 shows Baxter's tf tree at a certain moment with five different frames named */world*, */base*, */torso*, */collision_head_link_1* and */collision_head_link2*. The figure provides information about the relation between */world* and */base*. The information broadcaster, average rate, most recent transform and buffer length got displayed right after the launch. Here, we can see that each frame is related to the */base*

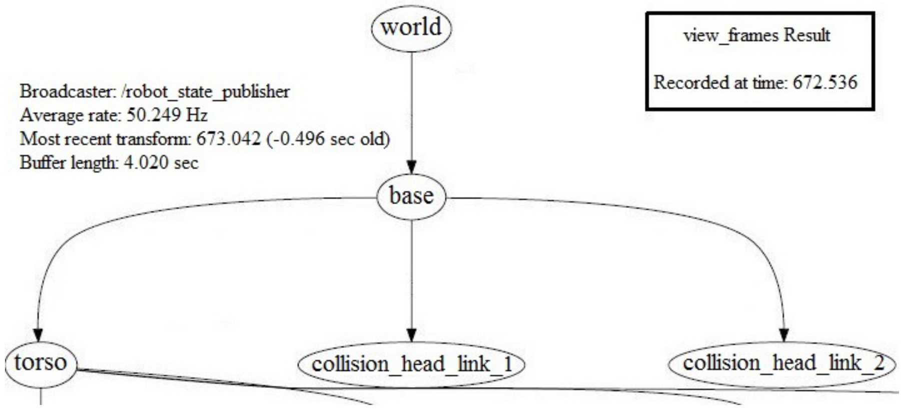


Figure 6: Baxter Frame tree example using Rviz

or */world* frame. */Base* and */world* contain the same data, and therefore we will always use the name */base* frame for referring to the original coordinate system. We even have the ability to display the relation between two frames. Tf provides the timestamp, our Cartesian coordinates $x,y,z$ and the Quaternion/Euler angles for describing the rotation in the spatial environment. The translations of the two coordinate systems are described through a *tf::Vector3*, and the orientation through a *tf::Quaternion* (see Section 2.1). The position *tf::Pose* of a single coordinate system is always related to the robot's */base* system. *tf::Pose* consists of a translation *tf::Vector3* and a rotation *tf::Quaternion*.

For instance, Listing 2 shows the relative position between the two frames */right_ gripper* and */base*. It describes a 90 degree rotation around the y-axis (green). The second entry is recorded at the same moment as Figure 5 (c), where the right gripper is 90 degrees rotated around the y-axis. We can see that the quaternion and the euler angles describing the rotation around the y-axis relative to our */base* system. However, there still is the problem that we need the data and the transform at the same time in a distributed system. Therefore, tf implements a *tf::WaitForTransform* mechanism which blocks all required

*Tf - mechanisms*

Listing 2: Relation between two frames

```
At time 760.064
- Translation: [0.459, -0.553, 0.032]
- Rotation: in Quaternion [0.401, 0.916, 0.006, 0.002]
            in RPY (radian) [3.129, -0.000, 2.316]
            in RPY (degree) [179.267, -0.010, 132.687]
At time 763.104
- Translation: [0.459, -0.553, 0.032]
- Rotation: in Quaternion [0.000, 0.703, 0.000, 0.703]
            in RPY (radian) [0.000, 1.559, 0.000]
            in RPY (degree) [0.000, 89.346, 0.000]
```

resources until either a predefined timeout is reached or the transform is available. Following that, it can delay the whole system, and hence there is a non-blocking solution for it called *tf::MessageFilter*. For instance, if the system has to *wait* for one second to process the required data, it delays the entire process for one second. Time critical applications should consider this, when choosing one of these functions.

Currently, most robots use the second generation of tf, called tf2. It provides more functionality, but it basically follows the concept of tf.

## 2.3 INVERSE KINEMATICS

*What is Inverse kinematics?*

Inverse kinematics (IK) is a well-known problem in robotics, but also in other areas such as computer games and animations. Particularly, in robotics it is a mathematical process which determine the joint parameters for each end-effector of a robot [46]. It is a non-linear equation to map joint parameters to a robot configuration [17]. By robot configuration we mean its kinematic chain (see Section 2.4). In Figure 7 we see an example with 6 degrees of freedom (DoF). Each joint can rotate around an angle $\theta$ to reach the requested goal position $\Gamma_6$. In Section 2.2.4 we saw different coordinate frames for each kinematic chain element related to the original */base* system $\Gamma_0$. In Figure 7 we also see these frames $\Gamma$, where the kinematic chain represents a possible trajectory from the original frame $\Gamma_0$ to the goal frame $\Gamma_6$.

*Different groups of algorithms*

Basically, we provide a desired end-effector positions for the robot, and the IK algorithm calculates the joint angles to reach the requested end-effector pose. After the IK solver finds a solution, the robot can move its arm chain with a specific trajectory to the goal position like in Figure 7. They are three major groups of algorithms to solve the IK problem distributed in analytical, geometrical and numerical approaches. It depends on the scenario and the robot which algorithm performs best. Generally, analytical and numerical algorithms are the most common methods to solve the IK problem with robots. The counterpart of the inverse kinematics is the forward kinematics. Here, the algorithm calculates the corresponding Cartesian coordinates for given robot joint angles.

In Chapter 6 we use a numerical solver and we will also briefly introduce an analytical approach to solve an inverse kinematics problem. Analytical approaches, however, are closed-form equations which get coordinate inputs and outputs the joint parameters. Numerical approaches use interpolation to find approximate solutions (advantage of quaternions, see Section 2.1). Following the IK problem formalism for our robot with 7DoF. We know the desired end-effector pose $e$, also called the goal position.

$$e = [e1, e2, ..., e7]$$

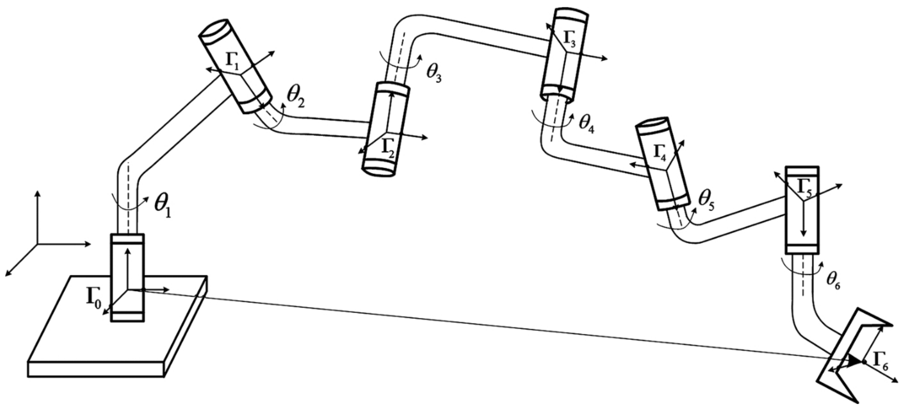


Figure 7: Inverse kinematics problem of a 6DoF robot[8]

The depiction of end-effector position is related to the */base* frame. Generally, it is a Cartesian coordinate system representation with standard $x$, $y$, $z$ and a quaternion $q$ (see Section 2.1). Summarized, we have a translation and an orientation, and we need the joint angles for moving the kinematic chain to the goal position in relation to its original */base* system. We are looking for seven joint values $\theta$, because our robot arm has 7DoF, shown below.

$$\theta = [\theta_1, \theta_2, ..., \theta_7]$$

*Complexity of IK*

IK requests are extremely complex. First, there are several possible trajectories to get from the current end-effector position to the goal position (see Figure 7). Furthermore, our robot has 7DoF, and thus the algorithm has to consider seven different joints. Table 2.4 shows the different capabilities and limitations of the Baxter robot joints. For example, some are twist joints, others are bend joints, and all of them provide different angles they can rotate. The IK algorithm must consider all joint limits, and possible joint collisions of the kinematic chain, when calculating the joint angles. In contrast, Forward Kinematics calculates $e = [e1, e2, ..., e7]$ for given $\theta_1$, $\theta_2$, ..., $\theta_7$.

$$e = f(\theta)$$

In Inverse Kinematics it is vice versa. We know the goal position in our Cartesian coordinate representation and we want to calculate $\theta$.

$$\theta = f^{-1}(e)$$

Following the mathematical definition we have the current joint angles and we want to calculate the change in joint angles needed to reach the requested end-effector position. In this thesis we do not want to go deeper into the complex calculation of a 7DoF inverse kinematics. They are different algorithms which can solve the above

8 http://mechanismsrobotics.asmedigitalcollection.asme.org

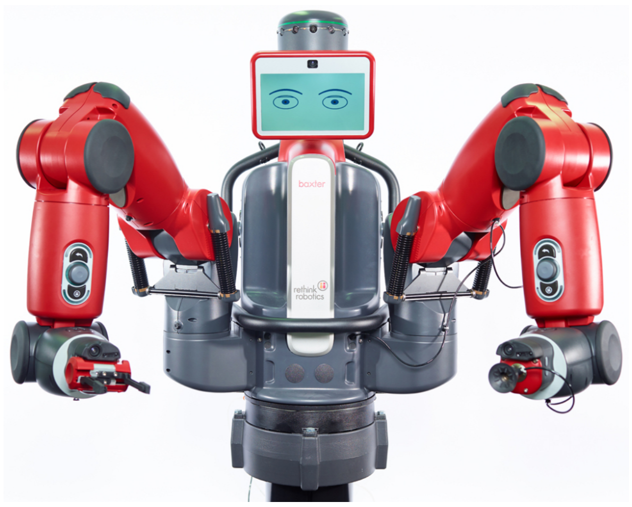


Figure 8: Baxter robot[10]

shown equation such as the popular Ocoros KDL [53]. Common numerical approaches like KDL uses Jacobian methods, pseudoinverse or Newton's method for their calculations. However, it is not necessary to understand how the IK algorithm works for the rest of the thesis. More information about IK, analytical and numerical approaches, as well as examples can be found at [3].

## 2.4 BAXTER RESEARCH ROBOT

In this section we give a short introduction to the Baxter robot we used for our prototype. Baxter is a very common research robot developed by Rethink Robotics[9]. Besides, its presences in research, Baxter is also used in industry, mainly for packaging and handling. Below, we describe the important components of Baxter with particular focus on our project.

Baxter is primarily developed as a *Cobot* (collaborative robot), that means it can work collocated with humans. Therefore, Rethink Robotics implemented safety mechanisms to ensure Baxter cannot hurt humans (e.g. when it works close to people it automatically adjusts the speed of its movements[11]).

*Basic knowledge about Baxter*

There are many different possibilities through which Baxter can grasp objects. From the reader's perspective, on its left arm the standard gripper is mounted (see Figure 8). The right arm shows another kind of gripper called vacuum. There are various grippers which can be mounted on its arm, allowing it to grasp different objects. This results in a higher capability, making the robot more efficient in various applicable scenarios. As with most of today's robots, Baxter has many

9 http://www.rethinkrobotics.com/
10 http://www.pullmanacademic.com.au
11 http://www.rethinkrobotics.com/safety-compliance/

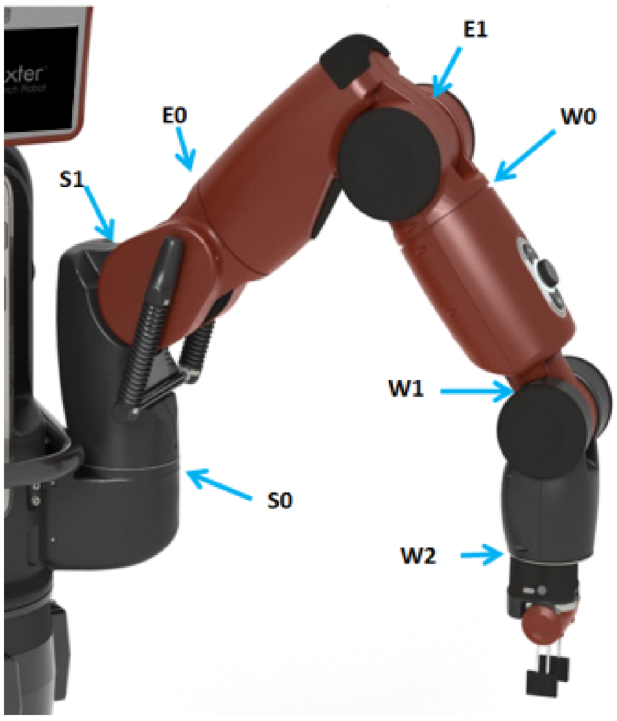


Figure 9: Baxter robot arm[12]

different sensors such as sonar, cameras etc. For information about its construction see the Rethink Robotics website[12]. The sensors we used for our project are described in Chapter 6.

For understanding the future chapters we have to define the term *robot kinematic chain*. Baxter has 7DoF, it basically means that each robotic arm has 7 joints. Each joint has a specific degree of freedom that it can move (see Table 2.4). Figure 9 shows a Baxter Robot arm, where joints are marked with an arrow, and its joint name. They are distributed in three sections:

*...robot kinematic chain?*

1. S = Shoulder (2 joints - S0,S1)
2. E = Elbow (2 joints - E0,E1)
3. W = Wrist (3 joints - W0,W1,W2)

We distinguish between bend and twist joints. S1, E1 and W1 are bend joints, and the remaining are twist joints. Table 2.4 shows the range of motion of each joint in degrees and radians.

A robot kinematic chain describes all joints/elements between the base of a robot and its end-effector (in our case Baxter's gripper). Generally, a robot end-effector is a name for the device at the end of the robot arm e.g gripper or vacuum (see Figure 8). In the following chapters we will only use the term robot kinematic chain. Furthermore, each chain element has its own coordinate system (see Section 2.2.4) due to the tf library and is described in the URDF.

12 http://mfg.rethinkrobotics.com/wiki

| JOINT | RANGE (DEGREES) | RANGE (RADIANS) |
|---|---|---|
| S0 (Twist) | 194.9 degrees | 3.4033 |
| S1 (Bend) | 183 degrees | 3.194 |
| E0 (Twist) | 349.979 degrees | 6.1083 |
| E1 (Bend) | 153 degrees | 2.67 |
| W0 (Twist) | 350.5 degrees | 6.117 |
| W1 (Bend) | 210 degrees | 3.6647 |
| W2 (Twist) | 350.5 degrees | 6.117 |

Table 1: Range of motion - Baxter joints

## 2.5 OPTITRACK

*Requirements for choosing our motion capture system*

In this section we introduce our tracking system OptiTrack [47], which is currently one of most common motion tracking/capturing systems worldwide. It is used by many companys such as NASA, Google, BOEING and UBISOFT [47]. The requirements for choosing a system were 6DoF tracking, high precision and low latency. Furthermore it should be able to recognize fast movements, especially in case of rotations. All these features are given with OptiTrack according to their website [47]. Therefore, we decided to use an approved system like it for our prototype.

OptiTrack works with infra-red light cameras (see Figure 10 (1)), capturing reflective light from objects. At this point the retro-reflective markers are essential (2). They are attached to objects or items we want to track. Through the captured reflections, cameras send position updates to the software, which determines the exact position of these markers, and consequently of the object.

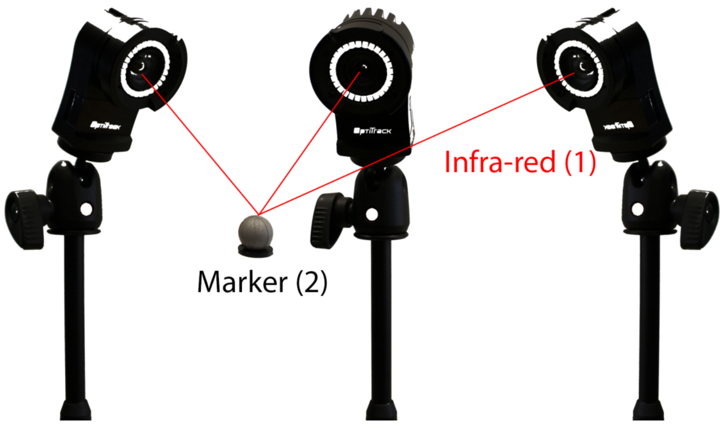


Figure 10: OptiTrack cameras and retro-reflective marker

OptiTrack provides a software, Motive, for performing these calculations. Following, we give an brief overview about Motive and its streaming plugin for Unity3D. In Chapter 5 we will see more of its functionality, especially during the calibration and wanding process (see Section 5.1). Motive offers a preview of each camera connected to the host computer (see Figure 13). It is possible to see reflections and interference with objects and/or other cameras in the spatial environment. It is very helpful when setting up such a tracking environment. It helped us, as we sought the best camera positions and settings for our project. We used the *rigid body* version of Motive, but OptiTrack also offers a *body* version for tracking human bodies.

# 3 RELATED WORK

We want to support remote collaboration on physical object-related tasks, thus we take a look into human-human object-focused collaboration. Therefore, we use the related work chapter to discuss fundamentals in that research area and how it impacts our work in remote collaboration. Particularly, we will discuss three related areas that our work is build on. First, we outline recent work on object-focused remote collaboration, where collaborators discuss and analyze physical objects in remote contexts. Next, we discuss the role of gestures, orientation and perspective of objects in collocated collaboration. Finally, we briefly discuss past research on telepresence robots.

## 3.1 CHALLENGES OF OBJECT-FOCUSED COLLABORATION

*Introducing the related work*

Back in history people always had to find a way to communicate with each other to reach their goals, especially when people work collocated in groups or pairs. It is the most natural process of humans e.g. to administrate, distribute and organize essential resources. Apart from the important life safekeeping tasks, humans collaborate in every day's life and it seems very trivial. Previous studies have shown that human-human collaboration is complex process that depends on various circumstances. For instance, we know from groupware and Computer Supported Cooperative Work (CSCW) studies that human-human collaboration is influenced by the environment, objects, and other people [1]. To design technologies that support and consider these different variables, it is necessary to understand the behavior of collaborators. Hence, we can design novel technologies to support collaboration. Below, we examine into recent work in object-focused collaboration.

In object-focused collaboration, a physical object is the center of collaborative discussion and activity. For instance, Licoppe et al. [38] explore how video is used to support object-focused collaboration in everyday video chat conversations. Beyond the issue that artefacts need to be placed in view of the camera, Licoppe et al. [38] show that how the objects are revealed and manipulated together with ongoing discussion plays an important role in conveying attitudes and interest between video chat participants. For instance, the way that a label on a box of biscuits is revealed to the camera (and a remote partner), signals and emphasizes what is important to each about the object. Similarly, a viewing participant may cock his/her head or appear to move closer to "get a better look", even though this has no meaningful practical effect; rather, the purpose is to engender feelings of interest in the shared experience.

This careful, thoughtful use of object configuration (and camera orientation) may not always happen. Mok & Oehlberg [42] explored a remote critique scenario, where participants were to explain how a complex object (prosthetic hand) worked to an audience via video chat. Their findings reveal that participants frequently forgot to show the audience aspects of the object, or even to ensure that the audience could see the object. This was partly due to the complex interplay between epistemic action (actions used to discover information about the object) and pragmatic action (actions used to explain how the object worked) [30]. It may be that the complexities of perspective and the pragmatic situation (rather than personal, in Licoppe et al. [38]) lends itself to more focus on the object rather than a remote party. Similarly, Jones et al. [25] describe the challenges of positioning a mobile camera view to effectively capture aspects of objects and scenes and convey them to a remote party as "camera work".

Our interest is in building from this work to understand whether the conventional face-to-face, opposing view perspective of video chat systems is a source of some of these problems, and whether an actuated physical proxy can help address these challenges.

## 3.2 GESTURES, PERSPECTIVE AND ORIENTATION IN REMOTE COLLABORATION

Considerable prior work has explored how gestures support collaboration, both from observational studies of collocated collaboration, as well as next-generation systems support. We briefly outline this work, and then illustrate how our work parallels this approach by focusing on how perspective and orientation can be supported.

*Gestures in remote collaboration*

*Gesture:* Tang's seminal studies of collocated interaction on tables underscored the importance of gestures in collaborative work [56]. He discusses how collaborators use hand gestures to communicate significant information such as enacting ideas or pointing to objects. To support gestures, researchers have explored marking up a remote video (e.g. [10, 14, 15, 19, 34]) as a proxy, included simple representations such as telepointers [18], and explored video overlays of bodies and arms [29, 54, 55, 57, 58] to convey additional subtleties of hand-based gestures [28]. Evaluations of these systems not only reinforce the importance of gesture, but also reveal the subtle ways in which gesture enables and engenders collaborative work. In the same way that this seminal work on gesture motivated subsequent system work, related studies of orientation and perspective motivate the present work.

*Orientation:* Kruger et al. [33] revealed the important role of object orientation by studying how people collaborate on a puzzle task. Based on their observational study, they articulate three distinct roles of orientation in collaborative work: comprehension, where the purpose of orienting an object is to personally understand/explore the object; coordination, where the object is reoriented to coordinate access and to define personal/shared working areas, and communication, where the object is re-oriented to explain something to another person. It stands to reason that these functional roles of orientation can play an even more important role in object-focused remote collaboration, particularly with three-dimensional objects (rather than a flat artefacts). Our work in designing ReMa focuses on this aspect of object-focused collaboration, where we explore how explicitly reorienting a remote object helps and hinders remote collaborative work.

*The three distinct roles of object orientation*

*Perspective:* In remote collaboration, collaborators usually have different perspectives of the workspace (due to camera placement, though cf. [10, 24]). This is even more problematic in object-focused collaboration. One cannot, for instance, rotate a piece of paper for a remote collaborator if s/he is not looking at the paper at all. This is a smaller variation of the problem described by Luff et al. [39], where one collaborator's understanding of the space, and how one orients and creates gestures is more difficult to (and sometimes inappropriately) perceive at a remote location. Jones et al. [25] discuss particularly how this happens during mobile video chat, where handheld perspectives of the scene are challenging to produce and capture properly for the remote collaborator. Fussell [13] explores variations on camera angles of a remote workspace for physical tasks, finding that *scene-focused* perspectives outperform *head-mounted* camera angles. Tang et al. [55] shows that task demands may be more easily addressed with some perspectives than others. For instance, shared perspectives are useful for reading text, whereas asymmetric perspectives are desirable (e.g. to create shared vs personal workspaces). Our work begins from the standpoint that different perspectives may be useful, particularly given that in collocated collaboration, people physically occupy different locations in space. Consequently, there is reason to believe that people are accustomed to alternate views of a physical object (e.g., in a face to face situation).

*Perspective issues in remote collaboration*

## 3.3 TELEPRESENCE ROBOTS

Our study builds on a long history of using robots to support telepresence [16]. One line of robotic telepresence research has supported remote camera control, either through a robotic arm [59] or through a mobile telepresence robot (e.g. [35, 44, 50]). Our research instead uses telemanipulation, where a robot manipulates objects in a remote environment [22, 36, 61]. In our case, a human collaborator is located

*Using robots to support remote collaboration* in that remote environment. We provide detail about our remote manipulation system next.

# 4 SYSTEM DESIGN

In Chapter 2 we introduced the two main components we used for developing our prototype system, the Baxter robot and the OptiTrack system. Following the related work in Chapter 3, we wanted a further understanding of how perspective and orientation is used in object-focused remote collaboration. Hence, we designed ReMa - Remote Manipulator.

Our eventual goal of ReMa is to allow two collaborators to explore physical objects, where each collaborator's interactions are reflected

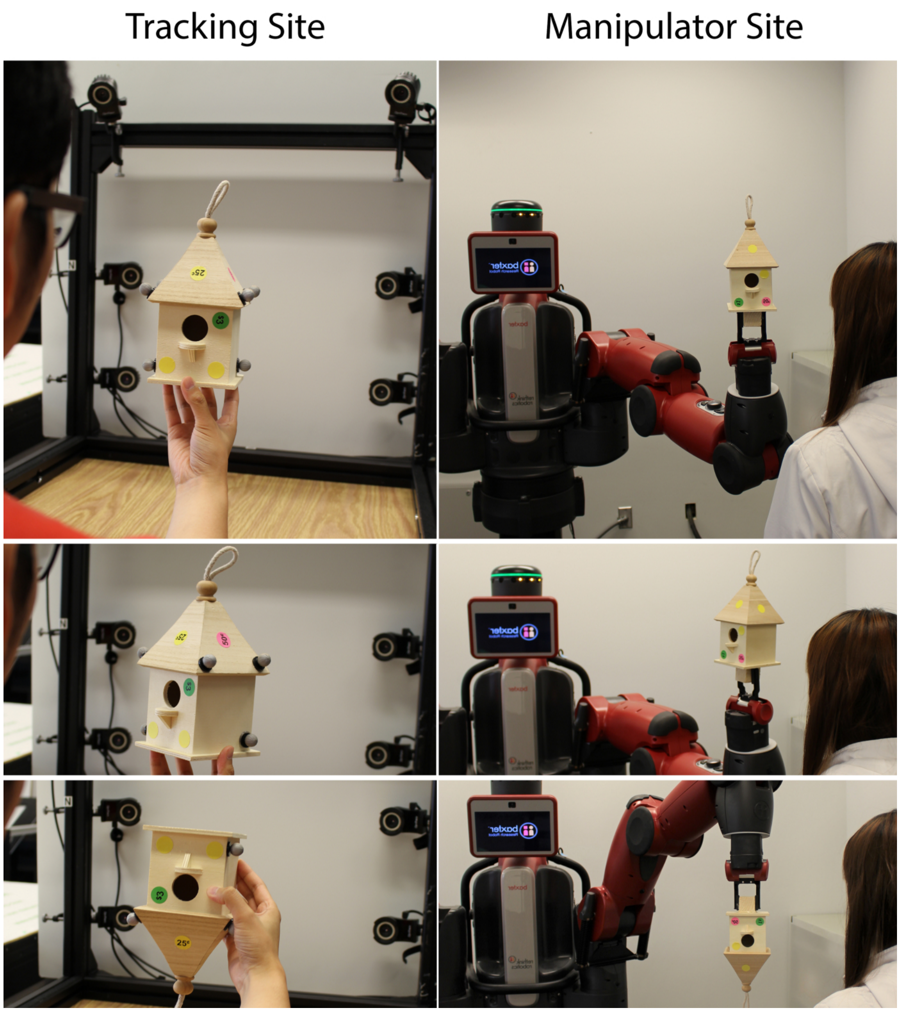


Figure 11: The ReMa system includes a Tracking Site (TS, top-left) and a Manipulator Site (MS, top-right) with bird house object. As the birdhouse is rotated at the TS, the proxy birdhouse at MS is also rotated reproducing a Shared perspective

at the remote site. However, enabling this sort of bidirectional manipulation comes with a well-known set of problems, particularly when collaborators are manipulating the object in opposition to one another (e.g. [6]). Thus, in this first iteration, our focus was specifically on one-way communication of the orientation of the object, abstracted from manipulations of the object in space. This focusing step allowed us to concentrate on the effectiveness and importance of orientation and perspective (i.e. via our study) without having to concern ourselves with resolving the challenges of movement tracking and bidirectional communication.

*What is ReMa ?*

As illustrated in Figure 11, ReMa comprises two separate sites: a Tracking Site (TS) and a Manipulator Site (MS). As we see later in this thesis, Figure 25 shows that there is an option for an additional video medium which was generally disabled in the ReMa-only system.

*ReMa inculdes two sites*

*Tracking Site:* At the Tracking Site, a person's manipulations on an object are captured – both the object's position in space, as well as its orientation.

*Manipulator Site:* Manipulations from the Tracking Site are transmitted to the Manipulator Site and get displayed on a similar proxy object. We can change what is rendered (e.g. in this iteration, orientation only), or how the captured information is interpreted and executed (e.g. *Opposing* or *Shared* perspective). This basically means that we provide two different perspective modes how Baxter will reproduce movements. Below we describe the two different perspective modes in ReMa (see Figure 12).

*Pausing mechanism*

*Pausing:* We also designed a pause mechanism, which allows either collaborator to pause the Manipulator Site in the current orientation. This temporarily disables the Manipulator Site from continuing to mimic the object manipulations from the Tracking Site. We built this into ReMa to allow participants to look at their proxy object independently without having the direct connection between one another.

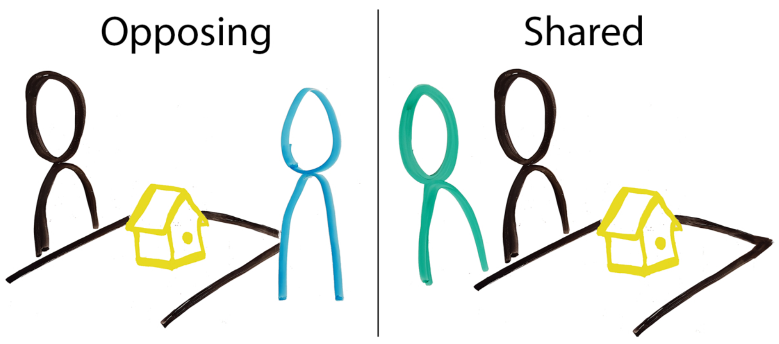


Figure 12: Perspectives: Opposing vs. Shared

*Opposing perspective:* Here, people do not share the same perspective to the object (see Figure 12 (left)). This, for instance, embodies a common situation where collaborators sit at opposite sides of a table, or in video face chats such as Skype or Google Hangouts. Figure 12 (left) shows two humans who are looking at an object from Opposing perspectives. In this case, ReMa reproduces manipulations mimicking a face-to-face collocated scenario.

*Shared perspective:* In Shared perspective, both persons share the same perspective/view to the object (see Figure 12 (right)). It embodies a common situation where collaborators sit at the same sides of a table. Here, ReMa reproduces movements like in a side-by-side collocated scenario.

# 5 TECHNOLOGIES

In this chapter we show the technologies setup we used to develop our prototype system, described in the previous Chapter 4. It is not a detailed guide for setting up all components in the whole system. Information and more detailed overviews can be found on the corresponding websites[1,2]. In the following sections we present our tracking and robot setup, and we illuminate and discuss different setup possibilities.

## 5.1 TRACKING & CALIBRATION

The following Figure 13 shows the architecture of all tracking system components. In our final setup we used six USB Flex 13 cameras. Furthermore, we also used an *OptiHub* to synchronize the cameras and process the collected data to the host computer. All cameras were connected via USB 2.0 cable to the *OptiHub* which was also directly connected to a computer.

There are many different opportunities to construct and structure an OptiTrack environment. Some examples can be found at `http://Optitrack.com/systems/`. The following suggestions should be considered when setting up an tracking environment:

*Important hints for setting up tracking environments*

1. Avoid the set up close to open windows. Incoming sunlight or other infra-red light sources such as computer mouses interfere with the cameras. This leads to worse tracking results or even could system stop the system from working entirely.
2. Reflective floors and illuminating obstacles should be taped or covered with non-reflective materials e.g. sheets.
3. OptiTrack cameras can also interfere with each other, therefore do not place cameras facing each other.

These recommendations are very important in ensuring the system works with a high precision and properly.

*Deciding on the number of cameras to use*

We built our OptiTrack setup with the support of an self designed aluminum frame (80/20 material) (see Figure 14). This frame provides a 1m x 1m x 1m space to manipulate an object. In a first approach we used eight Flex 13 cameras, but we had issues with camera interference (see suggestion (3) above). We decided to remove two

1 https://www.Optitrack.com
2 http://www.rethinkrobotics.com/

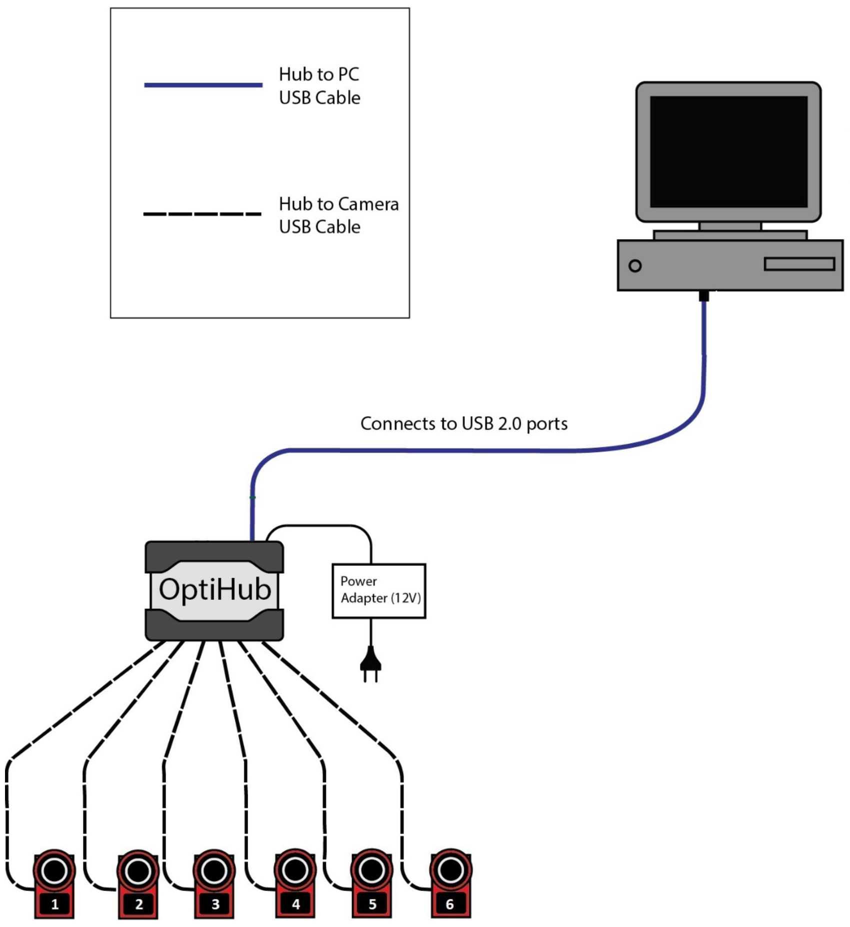


Figure 13: Shows our OptiTrack architecture with all components. Symbols are adapted from the offical OptiTrack website[2]

cameras from our designed frame to get rid of these issues. The result from this was satisfying. We also tested four cameras, that means without both cameras in the middle height of the frame (see Figure 14). It worked fine, but we decided to keep six cameras to support our *main captured area* against errors and obstacles.

After building the frame, setting up all components and the environment, we had to calibrate our tracking volume. Once, the calibration is finished, everything had to stay in the exact same position. Changes within the setup resulted in a mandatory re-calibration of the system.

*...what means camera calibration?*

Following is a short definition that explains the term *camera calibration* in the context of motion capturing with OptiTrack:

"During camera calibration, the system computes position and orientation of each camera and amounts of distortions in captured images. Using calibration data, Motive constructs a 3D capture volume. Specifically, this is done by

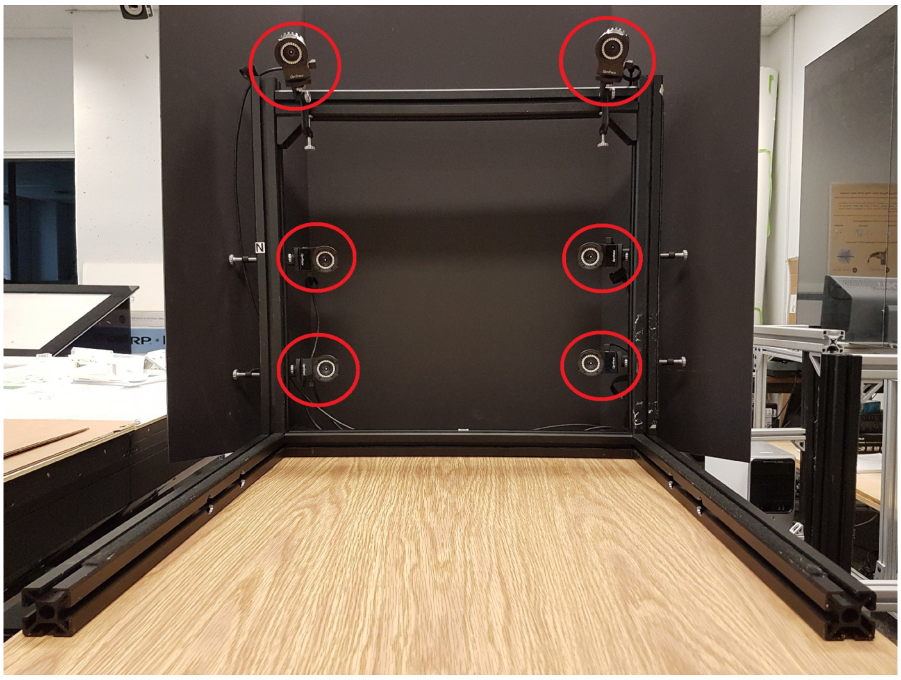


Figure 14: Our final OptiTrack setup with six "red circled" OptiTrack USB Flex 13 cameras attached to an aluminum frame

> observing 2D images from multiple synchronized cameras and associating the position of known calibration markers from each camera through triangulation." [47]

Using the definition above, we collected calibration data from all cameras to calculate and construct our captured volume and synchronize the cameras. This is a mandatory step, especially when 3D information about the object (i.e. markers) is required. For the next calibration step, *masking*, we followed the setup documentation [47]. Masking basically means marking all pixel errors in the camera preview, in order to tell the tracking system to not consider these pixel errors later in the calculation. Usually, these errors are a result of at least one of the above mentioned issues (e.g. interference with another tracking camera, due to them facing each other).

*Wanding process*

Wanding is a part of the calibration and we want to provide a more detailed look into it. As mentioned above, we must collect calibration data. Therefore, a *calibration wand* is required (see Figure 15). These tools are used to collect wanding samples from each camera while waving it within our future captured volume (i.e. in front of the tracking cameras). Once enough samples were collected we initiated the calculation process to render the point cloud.

The standard calibration wands called CW-500 and CW-250 are developed to be used in bigger *captured volumes* (see Figure 15). Higher precision in smaller spaces is only attainable with smaller wanding

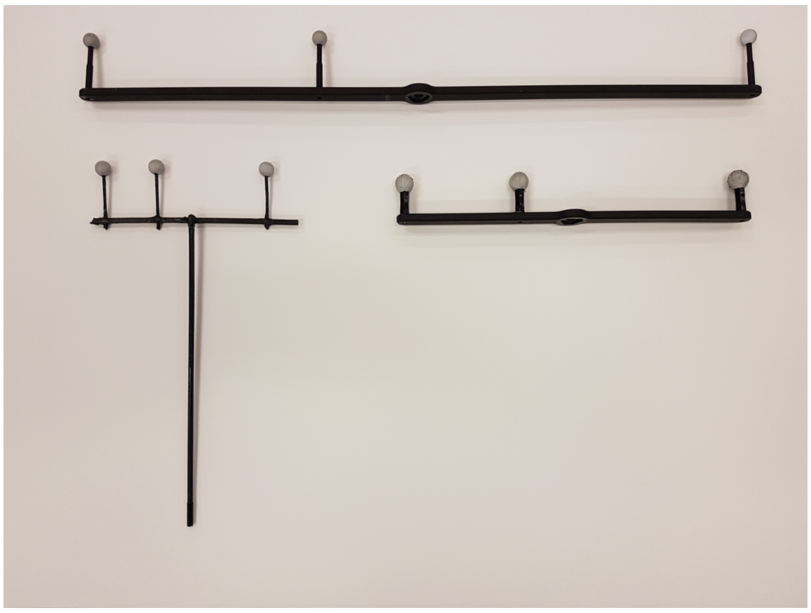


Figure 15: Compare CW-500 (top) and CW-250 (right middle) with our calibration wand (left middle/bottom)

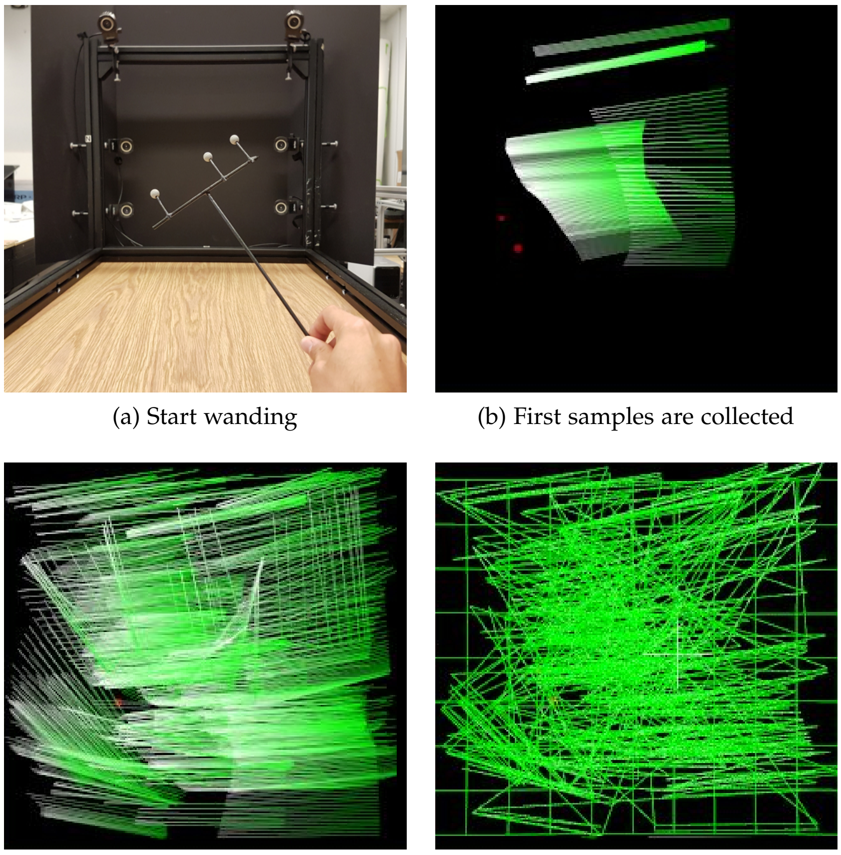
tools, due to the amount of samples which can be collected in a smaller space. Hence, we built our own wanding tool (see Figure 15) for getting the accuracy we needed. We used wood, wood glue and three retro-reflective markers. It is very important to use the right proportions that the system then can recognize the wanding tool. Between the retro-reflective markers we chose 40 millimeters and 120 millimeters distance which conforms to the supported standard ratio.

*Building our own calibration wand*

What we did not consider in our first prototype, was the property of wood that it reflects infra-red light. As a consequence we could not use the first prototype. For solving this problem we painted our calibration wand in a matt black color to avoid reflections during the calibration. After that, we could use it for wanding the tracking volume (see Figure 16 (a)).

Figure 16 shows the different wanding steps in a single camera preview. Figure 16 (b) shows the beginning of the calibration process, due to the amount of samples that have been collected. In contrast, the end result in Figure 16 (c), where a sufficient number of samples are collected. This is a simple visualization making it more clear when the wanding process is finished[3]. After wanding is completed, the next step is to initiate the calculation Figure 16 (d), setting the ground plane and determine the position of the cameras in the Cartesian coordinate system. Here, we basically followed the set up guide at [47].

3 Generally, 2,000 - 10,000 samples are enough [47]



(a) Start wanding

(b) First samples are collected

(c) Sufficient amount of samples

(d) Calculation process

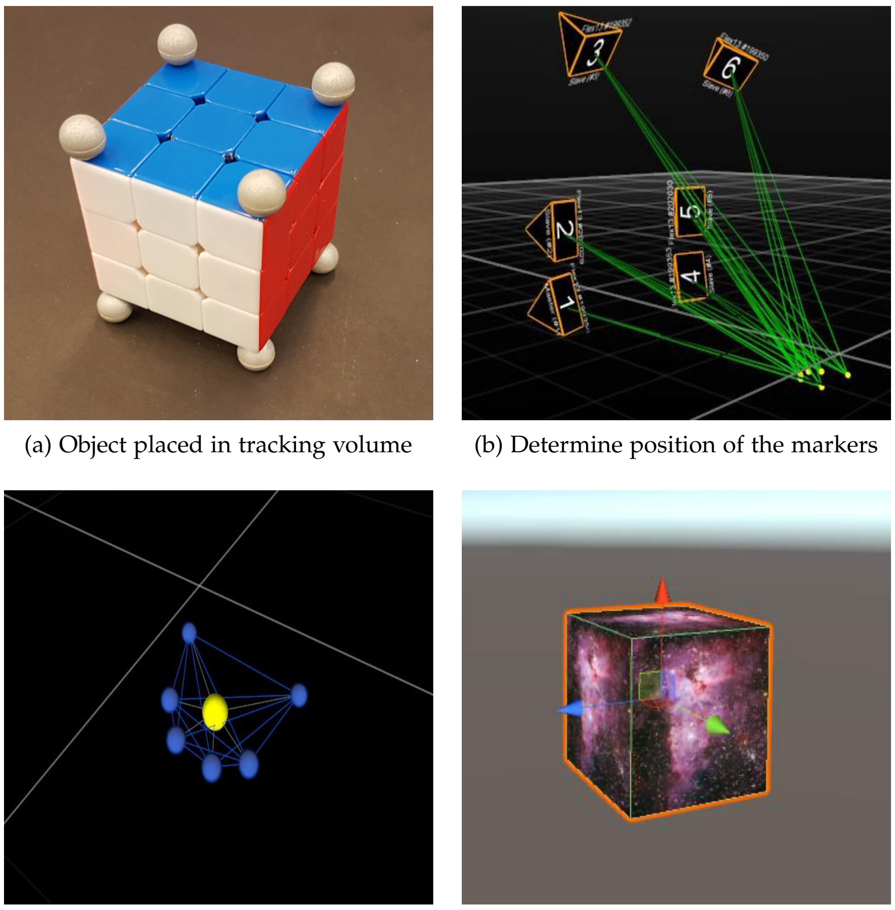
Figure 16: Wanding process camera preview.

As mentioned previously in Chapter 4, we want to track an object in a spatial environment. For realizing that, we had to define a *rigid body* in Motive. At least three visible retro-reflective marker reflections are required to create a rigid body in Motive, allowing it to continuously track the object. That implies that we needed at least three retro-reflective markers attached to our object.

To define a *rigid body*, we placed the prepared object with the attached retro-reflective markers into the tracking volume (see Figure 17 (a)). After that, we assigned each camera the exact position of every single retro-reflective marker (see Figure 17 (b)). Following, the OptiTrack software Motive creates a *rigid body* object (see Figure 17 (c)). Subsequently, the system could track our defined object. It is also possible to track several objects with different properties or marker setups at the same time, due to unique identifiers (UI).

*Streaming-plugin for Unity3D*

For streaming data from the tracking software Motive to Unity3D, a rigid body object using a cube (see Figure 17 (d)) is required. However, we needed data from Motive, in order to animate the rigid body object in Unity3D. OptiTrack provides a *streaming plugin* that allows real-time streaming either with live or recorded data from Motive to



(a) Object placed in tracking volume
(b) Determine position of the markers
(c) Created a rigid body
(d) Animated object in Unity3D

Figure 17: Object tracking

Unity3D. Basically, it uses a client-server approach, where Motive acts as a server and Unity3D as a client. We configured a Motive server as described on their website, and imported the required plugin into our Unity3D project[4]. In the last step, we attached the *OpitrackRigidBody.cs* to our rigid body object in Unity3D which eventually could receive tracking data from Motive (see Section 5.1). When we used more than one object at the same time, we had to use UI's telling both systems which object corresponds to which data stream as mentioned above. The position and orientation of the object is represented through the pivot of the constructed rigid body (seeFigure 17 (c)); marked in yellow.

Information about the tracking settings we chose can be found in Section A.2.

4 http://v110.wiki.Optitrack.com

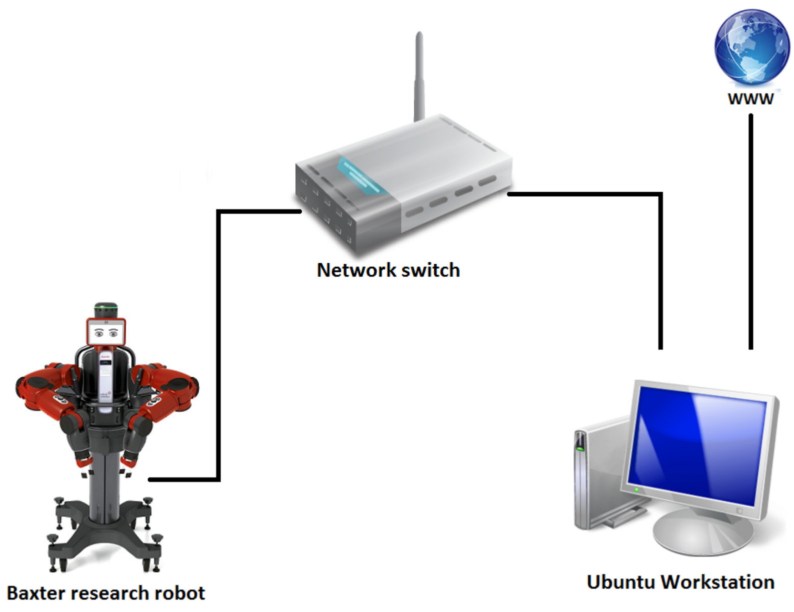


Figure 18: Baxter setup with all relevant components[5]

## 5.2 BAXTER SETUP

Here, we give a brief introduction about our components in our Baxter robot setup. We used a Baxter research robot (model-BR-01) updated to the latest RSDK version 1.2.0, and a standard host computer with an Ubuntu operation system (OS) for our workstation. We also installed the robot operation system (ROS - version Indigo) on our Ubuntu machine. The PC was connected to the internet, and furthermore to our local network (see Figure 18).

We also used a Netgear[6] network switch between the robot and our local machine. The components within our system were connected via Ethernet Cat 5 cables. *Note:* It is very important to consider that the RSDK version on your robot must match the RSDK version of the workstation.

Baxter is reachable in the network with either its hostname (commonly its serial number) or its Internet Protocol (IP) address. Therefore, we had to configure the communication from our workstation to Baxter and vice versa. The ROS Master (see Section 2.2.2) running on Baxter provides the interface for the communication over IP with Baxter. It has a specific Uniform Resource Identifier (URI) consisting of the above mentioned serial number and a port number.

*Communication within the system*

Our workstation must be able to resolve the ROS Master URI to communicate with it. Additionally, the ROS Master must be able resolve the IP address of our Ubuntu workstation. Basically, it is a stan-

5 http://www.iconarchive.com
6 https://www.netgear.com/

dard network configuration considering the above mentioned properties when setting up the communication in the network. The name resolving works with a *.bash rc* file executed on the host computer.

The communication between our workstation and the Baxter (ROS Master) can be verified through sending a ping to our ROS Master (Baxter), but vice versa it is necessary to test it via SSH following the Rethink Robotics *Hello Baxter* tutorial *Step 2: Verify ROS Connectivity*[7]. After verifying the communication between the components in our system we were able to launch ROS on our workstation. Finally, we enabled the robot to open the communication channels to ROS. This can be easily done by using the provided Python script from Rethink robotics called *enable_robot.py*.

7 http://sdk.rethinkrobotics.com/wiki/Hello_Baxter

# 6 IMPLEMENTATION

In this chapter we provide a detailed look into the implementation of our system. First, we give an overview of the system architecture. Subsequently, we elucidate and discuss every component of our system. We start with the server program and its functionality, as well as the network communication. After that, we show our tracking program and discuss challenges we encountered while implementing it. Then, we explain the core of our system, inverse kinematics, and provide information about the package we created. Furthermore, we will show alternative solutions and contrast them with our approach. Following that, we explain the synchronization mechanism we used. Finally, we show the system modifications we made in order to conduct HCI studies.

## 6.1 ARCHITECTURE

Figure 19 shows a high level architecture of our system. Following the previous Chapter 4, we have a Tracking Site and a Manipulator Site. In the previous Chapter 5, we introduced the main system components, OptiTrack and the Baxter robot, and also briefly explained the software technologies we used. In this section we want to describe the general workflow in our system, in order to provide an understanding of how it works. It is a highly abstracted description, and we will discuss every component shown in Figure 20 later in this chapter.

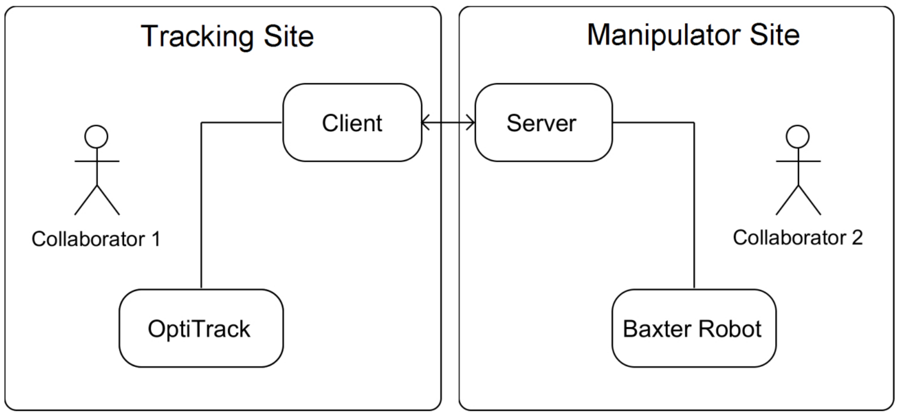


Figure 19: High level system architecture: Shows the Tracking Site with our motion capture rig OptiTrack and the client program (left), and the Manipulator Site with the Baxter robot and the server program (right)

*Basic workflow within the system*

*Tracking Site:* We track object manipulations through humans (see Figure 14). The infra-red cameras send their captured information to Motive which combines it to determine the position of the object in a Cartesian coordinate system in relation to its spatial environment. Motive streams these coordinate updates to Unity3D via its streaming plugin. The client program stores the current position of the object to compare each coordinate update received by Motive with the stored position (see Figure 19). If the system recognizes a difference it continues with processing the update (`Trigger principle`). It encapsulates the updated coordinates in a package and send it to our server at the Manipulator Site. Furthermore, it sets the update coordinates as the current coordinates.

*Manipulator Site:* The server receives the data package. All previous steps worked with a standard Cartesian coordinate representation of the position, but Baxter is unable to execute a movement to a position given through Cartesian coordinates. Baxter's end-effector position is represented by its seven joint angles. Simplified, we have a position and a quaternion, and we are seeking the robot's joint angles for moving its arm chain to the requested goal position. It is a well-known problem in robotics called Inverse kinematics (IK) (see Section 2.3). After an algorithm solves the IK problem, we send joint updates to Baxter, and it adjusts the position of its arm chain.

The following sections provide a detailed look at each of the above mentioned steps and is shown in Figure 14.

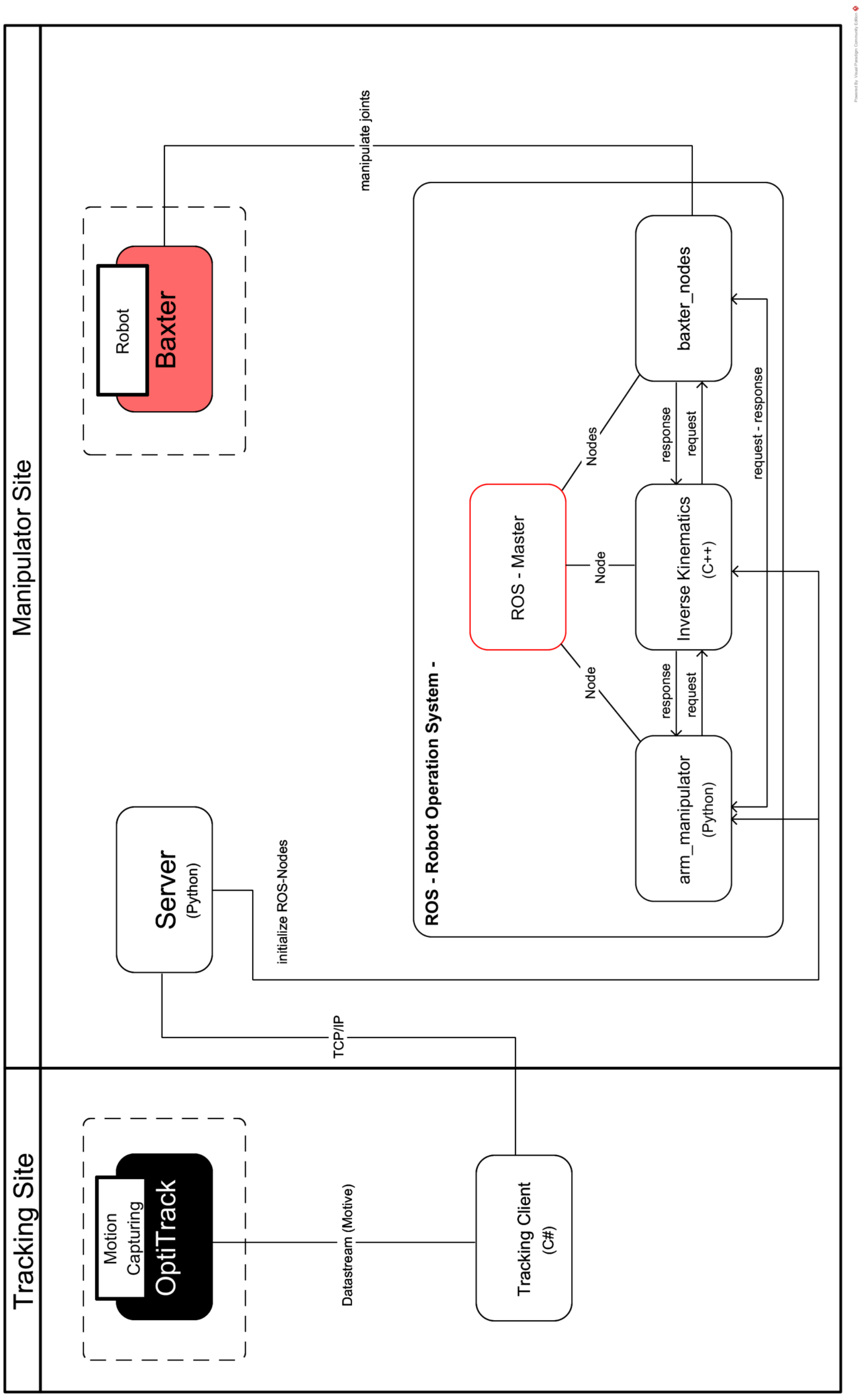


Figure 20: Technical system architecture

## 6.2 SERVER

In this section we explain the communication between the various components in our network. Moreover, we give a further look into our server program consisting of different scripts for network communication, robot initialization and data processing/administration.

As we know, the communication within the internet and in local networks work with different layers (see OSI-Model[1]). To exchange messages from one host to another within a network, we commonly use IP-networks. There are various protocols we can use for the communication (e.g. TCP, UDP, FTP and SMTP). Each of these protocols are used in specific cases. In our project, we exchange bits of data between a server program and a client. Therefore, it is reasonable to choose either TCP or UDP. Both use the transport layer of the OSI model. The decision depends on different circumstances. In general, TCP verifies that a message reached the destination host. Therefore, the destination host sends a confirmation back to the sender. If it does not get a response from the recipient, it can send the message again. In contrast, UDP does not verify that the message reached the goal, and thus it is faster. We decided to use TCP in our system approach due to the verification and the non-significant performance losses compared to UDP.

*Why TCP ?*

We used Python for programming our TCP server for two simple reasons. First, using Python to program a TCP server is very simple and can be done with a few lines of code. Second, we also initialized the ROS nodes in Python (rospy), due to the documentation provided by Rethink Robotics. It is also possible to use C++ (roscpp) to program the TCP server and initialize the robot, but we decided to keep our first approach with Python. We did not see a lack in performance which could justify the effort to change the programming language. We also used sockets for our client-server communication, where a server is listening to its socket for a client request. If the request was successful, server and client finally communicate directly. The connections work over a combination of IP and port number. When choosing an arbitrary connection port for the TCP server, we considered that it must be open in the router configuration, and enabled for TCP communication.

*Robot initialization and server program*

The server imports the ROS interface to enable the communication. It is divided into five python programs which provide different functionality. Above, we explained the network part of the server. We also implemented programs for the robot grippers, head and screen manipulations, and to administrate and process received coordinate updates.

Our server program gets TCP packages over the network, encapsulate the packages and processes this data for further computations.

1 http://docwiki.cisco.com/wiki

Basically, it gets a position consisting of $x$, $y$ and $z$, and a quaternion $q$ (see Section 2.1). The data undergoes consistency checks e.g. logical checks. The system always knows the current position and orientation of the end-effector in Cartesian coordinate representation. Furthermore, it knows the position and orientation of each kinematic chain element relative to the origin, due to the transform library (tf - see Section 2.2.4). The program calculates the corresponding joint angles with an IK algorithm for the requested position (see Section 2.3). The IK algorithm runs as a service in ROS (see Section 2.2.2). For processing the required data, we needed appropriate message types (see Section 2.2.1). Therefore, we used four different messages to communicate with the services, and to get the current joint angles of the robot. It was not necessary to develop custom message types, because ROS and Rethink Robotics already provided them. We used *Pose* and *PoseStamped* from the ROS geometry_msgs package to build the IK message type. For the communication with the IK service, we also used the existing *SolvePositionIK* message type from *baxter_core_msgs*.

*Communication with the IK services*

*Required message types*

In order to compute the IK, we call the responsible service with its name and corresponding message type (see Listing 3 (1) and (2)). After that, we build the message with the latest data from our distributed system (see Listing 3 (3)). For getting the required information from the distributed system we used the *wait_for_service()* function. If the IK algorithm finds a solution, and the result passes another consistency check, it eventually sends the joint angle updates to the robot to adjust its position (see Listing 3 (4)). Otherwise, the error handling routine releases the resource and unblocks the system (see Section 6.6).

To use the grippers, the server program has to initialize and calibrate them. Subsequently, we can easily control them in the program at any time. The *gripper.position()* function receives a value between zero and 100, where zero means that the gripper is closed and 100 means the gripper is entirely open.

*Embody current robot state*

To represent the current state of Baxter, we decided to use head movements and different faces on its screen. Generally, in our approach, Baxter's head follows the end-effector during a movement and focuses on the person in front of Baxter when the movement is finished. To realize that, we mapped the head position to the y value of the robot's end-effector frame. The position of the head is given through the *tf* library. However, Rethink Robotics provides an easy-to-use interface to manipulate the head position of the robot. The head can move 180 degrees to cover the whole workspace in front of Baxter. Regardless of whether the robot uses its right or left arm, the head can always follow the position of the object. Obviously, it can only follow one robotic arm which can be chosen by the user. After a movement is finished, Baxter's head will face the user. We did not

Listing 3: Calling IK service

```
def send_trac_ik_request(limb, pos, orient):
    # (1) Servicename/path
    ns = "ExternalTools/" + limb + "/PositionKinematicsNode/
        TracIKService"
    # (2) Calling service (name, msg type)
    trac_iksvc = rospy.ServiceProxy(ns, SolvePositionIK)
    trac_ikreq = SolvePositionIKRequest()
    # (3) Build message (timestamp, frame, position)
    hdr = Header(stamp=rospy.Time.now(), frame_id='base')
    poses = {
        str(limb): PoseStamped(header=hdr,
                               pose=Pose(position=pos,
                                   orientation=orient))}
    trac_ikreq.pose_stamp.append(poses[limb])

    try:
        rospy.wait_for_service(ns, 5.0) # (4) Block (name,
            timeout)
        resp = trac_iksvc(trac_ikreq) # (4) Actual request
```

implement a dynamic localization to find the position of the person in front of the robot. Theoretically, it is possible to realize it, due to the sonar sensor on the robot's head that can localize humans in front of the robot to ensure safety with the user (see Section 2.4).

*Publishing images to the robot screen*

Furthermore, to emphasize Baxter's interaction, we provide different faces for the different states of the robot shown in Chapter 4. Currently, there is no python interface provided by Rethink Robotics to publish images to the screen. Therefore, we had to use the standard publisher-subscriber pattern from ROS (see Section 2.2.2). Basically, we load an image as an OpenCV[2] 2D array, and convert it into a ROS message which we then publish to the */robot/xdisplay* topic. The screen sensor subscribes the topic, and therefore finally shows the picture. Summarized, we can call our function at any time in the program with a link to a local image to publish it to the robot's screen. Images should have a 1024 x 600 resolution to match the screen's resolution.

Finally, the communication between the server and ROS is only possible because a part of the server also runs as a ROS node (see Figure 20 - arm_manipulator node). Summarized, due to the server's initialization as a node, it is responsible for the communication with the client, processing data and providing an interface to ROS (see Figure 20).

2 http://opencv.org/

## 6.3 TRACKING CLIENT

In this section we describe our client program which is connected to the above elucidated server program. It is also responsible for handling the streaming data from OptiTrack. The tracking client is a C# program in Unity3D. We created an arbitrary model using a cube in Unity3D, and enabled the communication/streaming between OptiTrack (Motive) and Unity3D (see Section 5.1). We implemented a standard C# TCP client following Microsoft's .NET documentation[3].

We encountered two challenges we had with our client program.

*Challenges with the tracking client*

- OptiTrack has a high precision, it even recognizes small orientation and position changes, as a result of a barely jittering human hand. This means we cannot process every rotation or position changes to the server program.
- Unity3D works with a left-handed Cartesian coordinate system, but the end-effector pose of the robot kinematic chain is represented through a right-handed tf.

In the case of the first problem, we decided to provide different modes. Depending on the complexity of the object, users can decide whether they want to recognize rotations, positions, or both. For instance, users can enable a setting to recognize one centimeter position and 45 degree rotation changes, even during run time. It is also possible to keep one orientation and only use position manipulations or vice versa. The client listens to the position/orientation update stream of the object, and triggers the system when the configured alteration is reached. Subsequently, the system sends a TCP package with the updates to the Server.

The issue with different coordinate systems shows another advantage of quaternions. Positions are trivial for translating from a left-handed into a right-handed coordinate system. Suppose we have a $\vec{v}$ in a left-handed system, and we want the corresponding $\vec{v'}$ in a right-handed system. We can easily achieve it by inverting the z value of $\vec{v}$.

$$\vec{v} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \equiv \begin{pmatrix} x \\ y \\ -z \end{pmatrix} = \vec{v'}$$

We can translate rotations between the different systems by using the uniqueness of quaternions. Therefore, we transform the quaternions from Unity3D's left-handed system into a right-handed *tf* form.

3 https://docs.microsoft.com/en-us/dotnet/

Generally, quaternions give an independent representation of rotations. However, later in this chapter we will see that we have to convert a quaternion into a rotation matrix (see Section A.1). Hence, we have to swap axes which basically means negating or reversing the trihedron of the quaternion. Finally, we can express the same orientation in a right-handed coordinate system with $q = (a, -b, -d, -c)$.

*Pausing the system*

We also give users the opportunity to either pause the system at the Manipulator or Tracking Site at any time (see Chapter 4). To achieve this, we implemented a *signal handler* giving the ability to pause the system anytime. The signal handler can process various inputs e.g. keyboard, mouse or a pedal. Reproducing the signal leads to continue the system. It is realized through the logical *XOR* function, which pauses the system if anyone presses the required button. However, it will toggle the current state when both buttons get pressed simultaneously. Another possibility to pause the system is to go outside the marked tracking volume (1m x 1m x 1m space). It also is a function we implemented to ensure the user's safety. If the OptiTrack system can not track the object anymore, it automatically stops sending updates to our server. Otherwise, it could result in uncontrolled movements of the robot.

## 6.4 INVERSE KINEMATICS & COLLISION AVOIDANCE

The last two sections mainly covered the communication between the components over a network, and processing data to ROS. In this section, we elucidate the core of our system called inverse kinematics, and our approach for avoiding collisions.

*Standard Ik solver on Baxter*

Rethink Robotics provides the common Orocos KDL inverse kinematics solver for their Baxter robot [53]. KDL uses the kinematic chain for the calculation which makes it applicable for various robots through a standard URDF model (see Section 2.2.3). The algorithm runs on ROS and provides a service to solve the IK problem. Unfortunately, the solving rate and the speed was inappropriate for our scenario (see Section 6.4). We decided to look for another IK solver, and we found a solver called TRAC-IK [4]. This algorithm also uses the URDF robot kinematic chain representation to solve the IK problem which means it should run on every robot providing an URDF. Furthermore, it offers a very similar interface to KDL. In this section, we introduce the TRAC-IK algorithm on a high abstraction level. We also show and discuss the performance improvement, and how we replaced the standard IK solver with the TRAC-IK solver. Finally, we provide a standardized ROS package for the Baxter robot, whereby future researchers can easily use TRAC-IK on their Baxter robot and in the simulator (e.g. Gazebo[4]) to considerably enhance their robot's performance.

4 http://gazebosim.org

Collision avoidance is one of the main research questions in robotics. It describes the behavior of the robot after it detects a possible collision with its sensors. It is related to the sensor's accuracy and interpretation of the environments. It is still an open research question in highly dynamic environments (as is the case with vehicles). In our approach, we have a static environment consisting of a table, an object and a robot. Baxter also has specific sensors recognizing two different kinds of collision called impact and squish. Impact means the robot hits something (e.g. a table). Squish occurs if the robot exceeds a predefined torque for executing a movement. In our final approach, we did not use any specific algorithms for collision avoidance, except the IK solver which is responsible for avoiding collisions of the joints and Baxter's standard collision triggers (impact and squish). We did not do motion planning, where specific collision libraries such as OMPL, SBPL or CHOMP are required.

*Collision avoidance in our system?*

*TRAC-IK experiment:* Patrick Beeson and Barrett Ames who are researchers from Traclabs, introduced the TRAC-IK open-source library in November, 2015 [4]. It is an alternative algorithm to the KDL solver which works inefficiently when close to joint limits. TRAC-IK can be used with various robots such as NASA Robonaut 2 arm, Atlas 2015 arm or the famous robot PR2 developed by Willow Garage, due to the URDF compatibility. Basically, it uses two IK implementation to handle the existing issue with joint limits.

> "A simple extension to KDL's Newton-based convergence algorithm that detects and mitigates local minima due to joint limits by random jumps. The second is an SQP (Sequential Quadratic Programming) nonlinear optimization approach which uses quasi-Newton methods that better handle joint limits." [4]

Before we decided to replace the IK solver on Baxter, we wanted to ensure that it increased the performance of our robot. Our goal was to maximize the solving rate and minimize the calculation time of IK requests. Furthermore, KDL uses an iterative approach, where the algorithm terminates when it finds a solution. Otherwise, it runs for five milliseconds until it terminates. In the previous Section 2.3 we saw that there often is more than one solution to solve the problem. Choosing the first possible solution sometimes results in elaborate movements.

*Issues with current IK Solver*

Traclabs offers a ROS package to compare their IK solver to KDL[5]. We were able to re-use their code for our Baxter robot. As shown in the previous sections, we only needed to initialize a ROS node enabling the communication to ROS. For calculating the IK, KDL and TRAC-IK, we need the robot kinematic chain (URDF) which is loaded

5 https://bitbucket.org/traclabs/trac_ik

to the parameter server (see Section 2.2.2). In this approach we only worked in the Gazebo simulation to avoid any damage to the robot. We modified the C++ code for our tests, using different configurations to find the optimal settings for our real-time representation of manipulations with Baxter. TRAC-IK provides the different modes: Speed, Distance, Manip1 and Manip2.

*Different TRAC-IK modes*

- *Speed:* algorithm terminates after it finds the first solution
- *Distance:* algorithm runs until timeout, return result minimizes sum of the squared error from seed
- *Manip1:* algorithm runs until timeout, return result maximizes $\sqrt{\det(J * J^T)}$
- *Manip2:* algorithm runs until timeout, return result minimizes $\mathrm{cond}(J) = |J| * |J^{-1}|$

*Settings for our experiment*

Moreover, it allows the use of an error tolerance for the 3D pose, which means that it does not have to find the exact solution. We can also change the timeout value. In the experiment we use the same amount of samples as Traclabs to test the algorithm on Baxter. That means 10,000 randomly generated, but reachable joint configurations for our robot. They are generated, with the C++ *random* function between the possible angles for each joint following the URDF. After that, the program calculates the forward kinematics, and finally uses the result for the IK request in the experiment. We ran 10 tests with different settings, each of them five times in order to get an average result, due to the random generated joint angles. Table 2 shows the significant performance improvement of the IK solver with default settings error = 1e-5, timeout = 1e-3 (same as KDL) and in Speed mode (see above).

We could increase the success rate up to 99,8% by setting timeout = 1e-2. However, our system is time critical, and therefore increasing the timeout value was inconsequential. Due to such a high success rate, we decided to keep the error tolerance = 1e-5 as recommended by Traclabs. Tests in other modes (see above) sometimes showed more *sensible* movements in the Gazebo simulation, but the algorithm would always run until it reached the timeout value.

*IKFast as an alternative*

*Alternative:* As part of the motion planning environment OpenRAVE IKFast is another common IK solver provided by Rosen Diankov [9]. It is an analytical approach offering a closed-form solution for the IK problem. Following the OpenRAVE website, IKFast can find IK solutions in about four microseconds. It independently generates a closed-form C++ code for any robot which provides either a Collada DAE format or OpenRAVE's customer XML format instead of an URDF. Generally, it is possible to convert an URDF into a DAE.

| IK-SOLVER | SUCCESS RATE | AVG TIME |
|---|---|---|
| Orocos KDL | 62.3% | 2.22 ms |
| TRAC-IK | 97.8% | 0.81 ms |

Table 2: Performance improvement with TRAC-IK

We chose TRAC-IK for two reasons: (1) it provides various configurations, and (2) to provide future researchers with insight about its effectiveness for Baxter.

## 6.5 INVERSE KINEMATICS PACKAGE

In the last section, we showed why we decided to use TRAC-IK for our prototype. Here, we want to show how we developed an easy-to-use TRAC-IK package for Baxter. Our ROS package follows the standard requirements described in Section 2.2.1. Here, we explain the development of the ROS package, making the robot more effective in finding IK solutions. We used a Catkin package structure to maintain consistency within the ROS community (see Section 2.2.1). We implemented the code for the ROS node, created the launch file to run our ROS packages, and modified the CMakeLists.txt as well as the package.xml. Baxter's RSDK already provides code for standard communication over messages and services. Re-using their code package was possible with for a few modifications.

Alongside the auto-generated code in the package.xml and the CMakeLists.txt, we had to link the required dependencies and packages for our program. In order to process data in ROS, we used the provided message types *baxter_core_msgs*, *sensor_msgs* and also *gazebo_msgs* due to the Gazebo simulation. We also used the C++ template library Eigen, that is required when using the numerical IK solver TRAC-IK. It is a mathematical library to support linear algebra (e.g. through vectorization), making it significantly faster due to for example, less vector multiplications. Furthermore, we needed the kdl_parser, and the TRAC-IK library to compute the IK. To determine the position of the robot we used *tf* and *tf_conversions* as shown in Section 2.2.4.

*Messages types for the TRAC-IK package*

Initially, we want to explain the launch file allowing us to basically run our ROS package in the distributed system. It is a .XML file and uses ROS commands to provide information for the ROS launch. For instance, we declare default values for variables in our code such as default timeout value or the error value epsilon. Additionally, we also load the URDF file to the parameter server, and furthermore launch the ROS nodes for the right and left robot arm providing the interface for the communication with the system. For implementing the IK

*Requirements for launching a ROS program* service, we used the provided code from Rethink robotics in order to provide the same interface as with the standard IK solver. They provide a C++ class for processing the standard ROS messages, and a callback function for the IK services. We modified their C++ code so that we could use the TRAC-IK algorithm, and distinguish between the different IK solvers. Finally, we implemented the *kinematics.cpp* which is the core of the ROS package we provide.

*Processing data from robot sensors*

We used the two standard message types, geometry_msgs and sensor_msgs. The geometry message contains a position and a quaternion for our coordinate updates. We decided to use the numerical approach to solve the IK problem but we also needed the current joint angles (see Section 2.3). The most common way to read data from sensors is through sensor_msgs. These are also required to send updates to the robot sensors such as the joint servos. In the next step, the system has to calculate the goal position in relation to our */base* coordinate system. Therefore, the tf library provides a *tf::poseStampedMsgToTF* method to convert geometry_msgs into a tf for further processing (see Listing 4 (1)). Subsequently, we can use the *tf_listener.transformPose* to convert the message into a target frame (see Listing 4 (2)). In last step before we actually call the IK solver, we have to convert the target tf into a KDL::Frame, due to the required input for the TRAC-IK algorithm. A KDL::Frame consists of rotation matrix $R$ and a vector $\vec{v}$ to describe the desired end-effector pose (see Listing 4 (3)). Constructing the rotation matrix of a given quaternion can be done in constant time $\mathcal{O}(1)$ (see Section A.1). Finally, we call the IK solver which returns either a negative integer, if it did not find a solution, or a positive value, when it terminated successfully (see Listing 4 (4)). We handle errors in the system with the ROS_ERROR message appearing in the user console.

## 6.6 SYNCHRONIZATION

In the previous Section 2.2, we introduced the distributed system ROS, and the mechanisms to overcome the difficulties of such as system time stamps or frame id. We wanted to create a *real-time* system with various components, and thus we also needed mechanisms to control the message flow. Additionally, we also show the already existing standard synchronization methods of the systems. Figure 21 shows a flow diagram to visualize the communication between the different components in ReMa.

Our Tracking client program captures object manipulations in 30 FPS. However, it depends on the computer's FPS due to the *Update()* function in Unity3D. If there is a significant change of coordinates (see Section 6.3), it triggers the client program to send a TCP package including the coordinate updates to the server (see Figure 21 (2.1) and (3)).

Listing 4: Core program IK solver

```
// (1) Required messages
geometry_msgs::PoseStamped pose_msg_in = pose_stamp;
tf::Stamped<tf::Pose> transform;
tf::Stamped<tf::Pose> transform_base;
tf::poseStampedMsgToTF(pose_msg_in, transform);
...
...
// (2) Convert frame to our base_frame
tf_listener.transformPose(base_frame, transform, transform_base);
...
// (3) Frame consists of a vector and a rotation matrix
KDL::Frame goal_frame;
tf::transformTFToKDL(transform_base, goal_frame);
...
// (4) Call IK solver input & ouput are in KDL format
int valid=ik_solver->CartToJnt(input_position, goal_frame,
    output_angles);
```

*Mechanism to control server requests*

The server implements a *Requesthandler* which handles requests from the Tracking client program. To ensure that it proceeds with the latest data, the *Requesthandler* uses a semaphore to control the message flow. It acquires the requested coordinates, and furthermore blocks the input until the resource gets released (see Figure 21 (3.1)). The server also stores the current position and orientation globally in our program, providing them for computations. In the next step, the server calls the IK service in ROS in order to compute the joint angles for the robot (see Figure 21 (3.2)). Therefore, it needs the current joint angles, the time stamp, the goal position consisting of a position and a quaternion, and finally the related frame. The blocking function *wait_for_service* calls the service with a timeout of five milliseconds, allowing ROS to process the required data (see Figure 21 (3.2.1)). At this point, the system inherently blocks the communication channels to compute the IK. If the algorithm terminates successfully, the system will forward the solution to the actual function to manipulate the robot arm using the above mentioned message type *sensor_msgs* (see Figure 21 (3.2.2.1)). In contrast, if the algorithm does not find a solution for the IK problem the program releases the resource, and opens the communication channels for receiving new coordinate updates (see Figure 21 (3.3)).

*Moving Baxter's arm chain*

However, as we saw in the last two sections, TRAC-IK finds a solution in most cases. To move the end-effector of the robot to the required position, we can use two different methods, either the blocking function *move_to_joint_position* or the non-blocking solution called *set_to_position*, depending on the scenario. The difference between these functions is simple. The function *move_to_joint_position* blocks the system until the arm chain finally reaches the required end-effector position. It is possible to set a threshold and an accuracy for moving the robot arm chain. However, to achieve the goal of a real-time system, we decided to use the non-blocking function *set_to_position* to avoid system delay. The robot will move its arm chain until either it reaches the goal position or another position is requested. It also implies that we release the resource directly after the response of the IK solver. However, in the next section we elucidate our modification for the study. In the study approach, we use the blocking function as shown in Figure 21.

## 6.7 MODIFICATIONS FOR USER STUDIES

In Chapter 4, we designed a system which provides different perspectives, Opposing and Shared, of an object in a remote workspace. In the next Chapter 7 we will see the limitations of the system, particularly the robot itself.

*Reasons for modifying our prototype*

Since participants would not know the limitations of the robot, and how to ensure their safety, we modified our system to run HCI studies. We did not want to give participants the ability to reach their remote collaborator with the robot arm, because they could not see each other. Therefore, Baxter was not able to enter the remote participant's workspace, because of a safety mechanism we implemented. Furthermore, we introduced Baxter as a Cobot which basically means it is developed for working in collocated tasks with humans. Therefore, for instance, Rethink Robotics's robot automatically stops when it hits something e.g. a human arm to ensure safety.

*Simplified the implementation to overcome robot limitations*

In the first iteration, we were interested in perspective and orientation manipulations and therefore we only passed orientation updates to the server (see Chapter 4). Figure 24 shows the different areas where Baxter can provide certain orientations. Because of the joint arrangement, some gripper position and orientation combinations are impossible for Baxter to perform. Rakita et. al. [51] encountered similar issues when mapping a human hand to a robotic end-effector, which was ineffective due to humans's different kinematic capabilities. We used a dictionary to overcome this limitation, where we simplified the implementation in two ways. First, we allow for any pitch, yaw and roll in a Cartesian coordinate system to be rendered at 0° , 90°, 180° and 270°. Our implementation relies on a dictionary of end positions, where for some orientations of the object, the object needs to be positionally shifted slightly on x, y and/or z axes. While the data on how the robot should move is sent over a local network with very low latency, the Baxter requires 2.5 seconds average (minimum: 2.0 seconds, maximum: 3.5 seconds) to physically respond and reach the correct orientation at the MS. Second, our study implementation does not allow the Baxter robot to stop mid-way during a reorient-

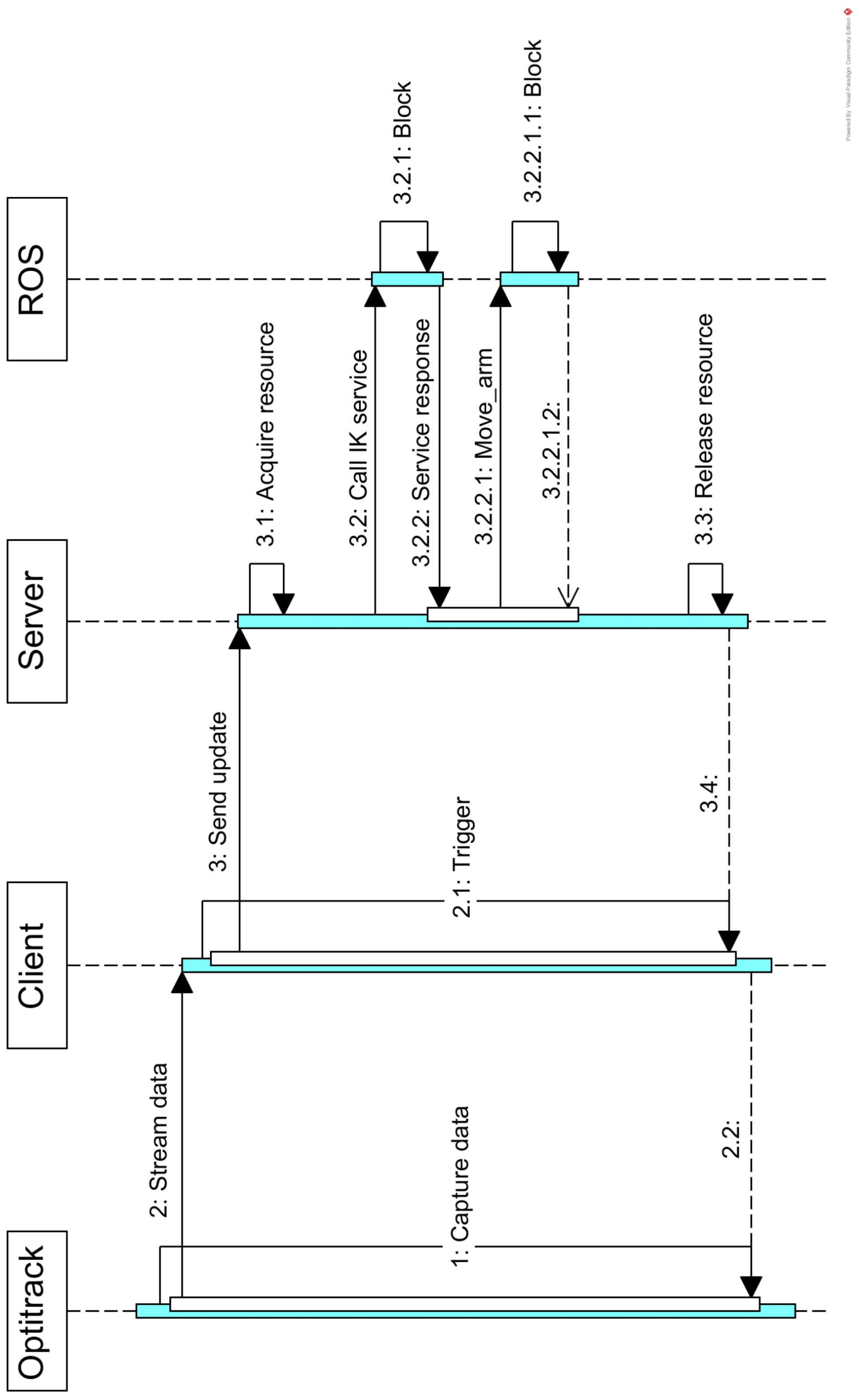


Figure 21: Communication flow between the different system components

ing act, and go to a new orientation. As a result, if the TS rotates to a position and decides to rotate the object again while ReMa is still performing the first orientation, the ReMa finishes the first reorienting act before beginning the second reorienting act, compounding interaction latency.

We also changed the TRAC-IK mode to *Distance* which performed best for our scenario. Although the algorithm always runs until the timeout value is reached, the computation time was sufficient. Apart from the above mentioned modifications, we used the developed system described in the sections before.

Particularly, in our first study we provided two different perspective modes to use our system. In Chapter 4, we designed a system which provides an Opposing and a Shared perspective. Next, we want to give a technical background of the different perspective modes.

*Providing the different perspective modes*

We did not modify the program itself, the different perspectives could be easily achieved through changing the camera positions in the OptiTrack software. In Section 5.1, we showed the OptiTrack software Motive, and explained the setup of our motion capture rig. To provide the different perspective modes Opposing and Shared, we inverted the OptiTrack virtual camera setup to mirror the tracked orientations. Thus, the coordinates are sent in an inverted state, and reproduce the requested orientation on the object accordingly.

In the previous Section 6.2 we explained our realization for head movements and different faces to embody the current state of Baxter. For our study, we did not want to distract people with *extraordinary*

*Head movements and display changes*

faces. We decided to use a green face if Baxter finished its movements or it is aligned with the remote object. Otherwise, its screen will turn yellow to signalize something will happen or is in progress. Furthermore, our robot moves its head towards the object to reinforce the next movement. Finally, its head moves back to the user's position when it is finished.

## 6.8 SUMMARY

Summarized, we implemented ReMa, a system which tracks object manipulations through humans and reproduces these manipulations at a remote location with the help of a telepresence robot. To realize this, we developed a server and a client program running on two different machines. All components work independently and can be easily exchanged. For instance, we could apply the server/client program to a different robot and/or tracking system. The client program at the Tracking Site receives position updates from six OptiTrack cameras via Motive's streaming-plugin, and processes data packages to the remote server at the Manipulator Site. The server program initializes the robot, handles requests and processes data to the IK solver for moving the robot end-effector to the requested goal position.

As discussed previously, we did not use the standard Orocos KDL solver for our robot. To significantly improve the robot performance, we used the TRAC-IK algorithm and developed a package which allows the Baxter robot to use the algorithm. Our ROS package is standardized, described in Section 2.2.1, and therefore can be used for future research with Baxter robots. Finally, we also showed its advantages compared to the standard IK solver and briefly introduced IKFast as an alternative.

We developed two slightly different programs. The original program executes position and orientation updates. Returning to our HCI research question, where we were interested in how perspective and orientation matters/is used in remote collaboration on physical object-related tasks. We modified our program as described in Section 6.7. Essentialy, we only passed orientation updates (quaternions) to the IK solver and used pre-tested positions in order to ensure Baxter could provide the requested position-orientation combination.

# 7 SYSTEM EVALUATION

In this chapter we elucidate and discuss the technical limitations of our system, as well as limitations of the robot and the tracking system. We start with the system's precision, then discuss its usability. Finally, we show the overall limitations and challenges of the system. This chapter provides an evaluation of the system we originally created and also the system with the study modifications. However, most difficulties and limitations are independent of the system version.

## 7.1 ACCURACY & USABILITY

*Accuracy of the system components*

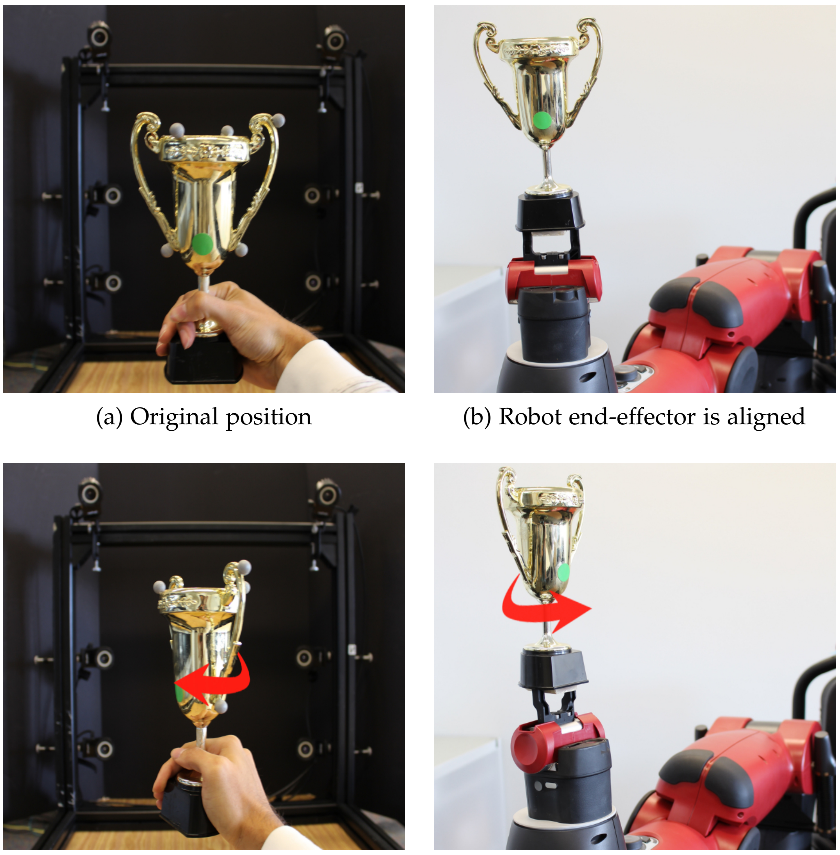
In this section we talk about the preciseness of the system, as well as the usability. OptiTrack's tracking of the retro-reflective markers through infra-red light is very precise. Following the calibration we only have an average error of 0.82 mm which is negligible for our project. Also, Baxter's accuracy plays almost no role, with an average error of +/- 5 mm following Rethink robotics hardware specifications [1]. In the implementation chapter we chose a maximum error of 1*e-5 for IK pose requests (see Section 6.4), which also results in a significant low error. Summarized, we do not have issues with accuracy in our system, due to the sophisticated components we used. Furthermore, we kept the delay in the whole system as low as possible. We usually measured an average delay of under half a second till the robot started the movement, even if we consider the network latency (e.g. through the internet). That means the robot reacts immediately when changing the position of the object in front of the tracking system. However, the Baxter requires 1.2 seconds average (minimum: 0.8 seconds, maximum: 3.5 seconds) to physically reach the correct orientation at the MS. The system we used for our user study has an average delay of 2.5 seconds (minimum: 2.0 seconds, maximum: 3.5 seconds), due to the *Distance* IK mode and the blocking function for executing requests, this ensured a 100% success rate.

*System delay*

In case of usability it depends on what people expect from the robot. Baxter does not execute a movement similar to humans. It has to readjust all joints when it moves its arm chain to another position. It depends on the goal position and orientation of the end-effector, and the current state of the joints. Following our implementation, we use the *Speed* mode in TRAC-IK which sometimes results in elaborate movements compared to other TRAC-IK solutions in different modes.

*Usability of the ReMa system*

1 http://sdk.rethinkrobotics.com/wiki/Hardware_Specifications



(a) Original position

(b) Robot end-effector is aligned

(c) Simple 90 degree clockwise rotation

(d) Robot executes a 270 degree counter-clockwise rotation

Figure 22: Robot trajectory issues (arrows adapted from photoshop brusheezy package[2])

However, even other modes cannot solve the issue of readjusting all joints.

*Issues related to the robot joint limits*

When we introduced Baxter in Section 2.4, we also showed its joint limitations (see Table 2.4). Following the table, we can see that, for example, Baxter's wrist can rotate 350 degrees. That means depending on the current state of the wrist servo, Baxter will not execute a *natural* human movement, due its joint limit. Figure 22 illustrates the issue which appears if we rotate an object. Let us suppose we have a right-handed coordinate system and we rotate an object 90 degree clockwise around the y-axis (horizontal axis - see Figure 22) . In this case, Baxter executes a *simple* 90 degree clockwise rotation through a 270 degree counterclockwise rotation, because of its wrist joint limit. Figure 22 (c) and (d) show that the robot wrist turns in counterclockwise direction instead of mimicking the human's movement in the clockwise direction. It may seem trivial and unimportant, but it definitely confused user in both studies (e.g. *"Can you rotate the object 90*

2 https://www.brusheezy.com/members/aura_id

*degrees to the right"* (Group 5 MS - participant), but Baxter will rotate it 270 degrees to the left). Both result in the same end-effector orientation, but the trajectory from current to goal position is different.

If we consider all joint limitations from (see Table 2.4), it becomes clear that the robot is not constructed for human-like movements. Furthermore, it also highlights the complexity of the request position for the IK algorithm. In our tests we explored lots of unexpected large and extraordinary movements, in particular when we switched from a end-effector, face-down to a face-up orientation. As we showed before it is also a result of the joint limitations of the robot. Here, the robot has to readjust its whole arm chain. Figure 23 shows the different orientation face-down to face-up. In figure Figure 23 (c) the movement is executed by a human through a simple tilt of the wrist. The whole arm of the participants stays in the same position as in Figure 23 (a). However, as visible in Figure 23 (d) the robot has to adjust its entire arm chain resulting in an unexpected bigger movement.

*Elaborate robot movements*

Baxter is mainly developed for packing tasks such as grasping an object and placing it in a box[3]. In these cases its joint limits and capabilities are sufficient.

## 7.2 LIMITATIONS

In this section we talk about the technical limitations of our system, in particular the Baxter robot. We have already seen issues related to the joint limitations of the robot. In this section we dive further into the robots capabilities.

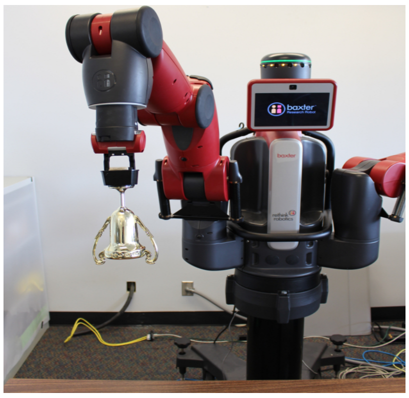
As we briefly mentioned in the Chapter 6 Baxter cannot provide every rotation at a position. In the previous Section 7.1, we saw that its movements are often elaborate, due to the joint limitations. However, many rotations are often physically not reachable for the robot due to its joint arrangement. At the beginning of our research, we expected that it is a problem of the IK algorithm and following the previous Section 6.4 we were able to significantly improve the performance. However, we encountered another issue with the robot itself. For Baxter, some gripper position and orientation combinations are impossible to perform. Figure 24 shows the different areas in green, where Baxter can provide basic rotations face-up (a), face-down (b) and face-left (c), face-right (d).

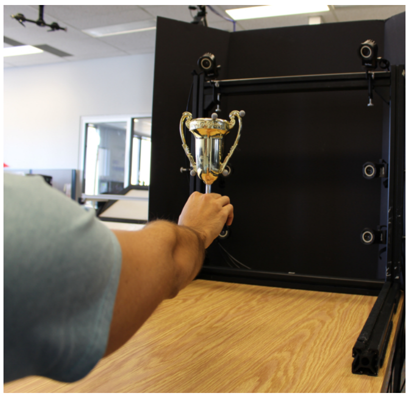
*Arrangement of robot joints led to system limitation*

There are many orientations in between these basic orientations, but for illustration purposes, it is sufficient to provide only the above mentioned orientations. The main reason why Baxter cannot provide all orientations at a position is the arrangement of its joints, and the above elucidated joint servo limits. As we can see in Figure 24 (c) (d), the green marked area shows where Baxter can provide the rotation face-left and face-right. This is only a 2D image and does not con-

3 http://www.rethinkrobotics.com/baxter/

(a) Face-down orientation human arm


(b) Face-down orientation robot arm chain


(c) Face-up orientation human arm

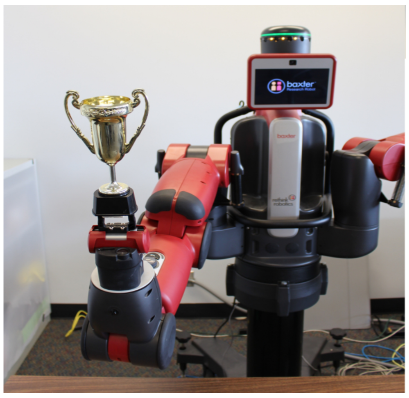
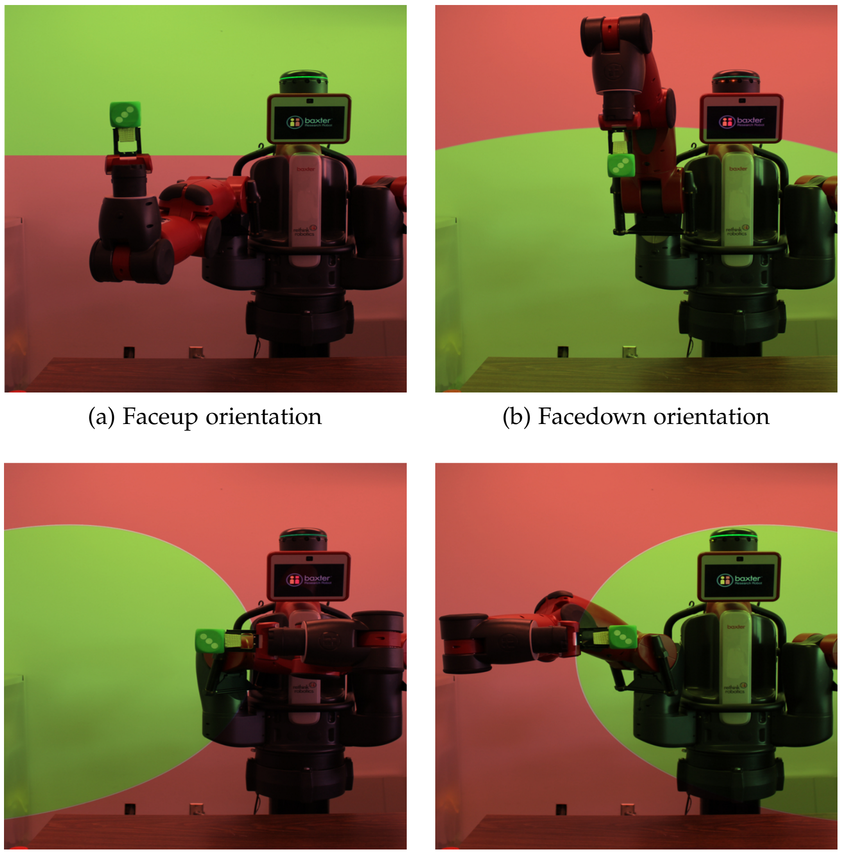

(d) Face-up orientation robot arm chain

Figure 23: Re-adjust robot arm chain

sider depth. However, even from this limited perspective, it is clearly visible that the end-effector position, where Baxter can reach the orientations face-left and face-right are only overlaying in a very small region. Figure 24 (a) (b) also shows the orientation face-up and face-down which illustrates the same issue. Baxter is constructed for executing specific tasks such as packing (see Section 2.4). Hence, the robot capabilities in relation to the tasks it typically performs are appropriate. For a packing tasks such as grasping an object and placing it into a box, Baxter mostly needs the orientation face-down which it provides almost in the entire workspace (see Figure 24 (b)). Considering all possible orientations in 3D space shows how limited our prototype eventually is. Therefore, we decided to use the dictionary implementation ensure that users are not faced with these issues (see Section 6.7).

Another question which arose was *how can we determine if a movement is finished or not*. We send coordinate updates to our server program triggered by position and orientation changes. However, it is a


(a) Faceup orientation (b) Facedown orientation

(c) Faceleft orientation (d) Faceright orientation

Figure 24: Robot end-effector position-orientation combinations: The figures show the different areas where Baxter can provide end-effector position and orientation combinations in 2D space (green)

fixed value and often people were in the middle of a movement when the robot updates its position. Using a time threshold does not solve the problem, because it is still unclear how long a movement will take. We also considered a signal executed by users in order to tell the system when a movement is finished. This seemed very unnatural to us, so we decided against it. Furthermore, depending on the function we use to call the IK service, it either results in staccato or smooth movements. The *move_to_joint_position* service function blocks all communication channels to ROS which means it will execute the movement without interruptions. As shown before each IK call will result a delay of the whole system, but movements will be smoother. Nevertheless, we still have the problem of not knowing when a movement is finished in order to send updates to the server program. Another approach is to use the *set_to_position* position function which does not block the system. Thus, there is no extra delay in the system. However, while executing a movement, the robot gets many position updates resulting in staccato movements. We could not find a solution for entirely eliminating the problem. The *set_to_position* function does not have this delay and therefore we chose this approach. We tried to overcome the staccato movements by using a fast deterministic greedy path smoothing algorithm. However, we still needed the trajectory before we actually could use the algorithm. This took us into recent autonomous vehicles research about real-time motion/-path planning considering dynamic environments which is beyond the scope of our work.

*Limitations of an OptiTrack system*

Another limitation of the system is the tracking system. Using retroreflective markers has a simple disadvantage. If someone covers the markers (e.g. with his/her hands), the tracking does not work anymore. We tried many different marker setups, to ensure that we always have enough visible markers for the cameras to determine the position (at least three). However, we could only predict how participants would hold the object. In our user study we also explored other issues with the tracking. Some participants wore watches, rings and/or glasses. All of these items interfere with the infra-red cameras which sometimes resulted in errors. However, even with the above mentioned problems with the tracking, in most cases the system worked as expected. Using an asymmetric marker setup also made the tracking more resistant and stable against interference, errors and covered markers.

# 8 USER STUDIES

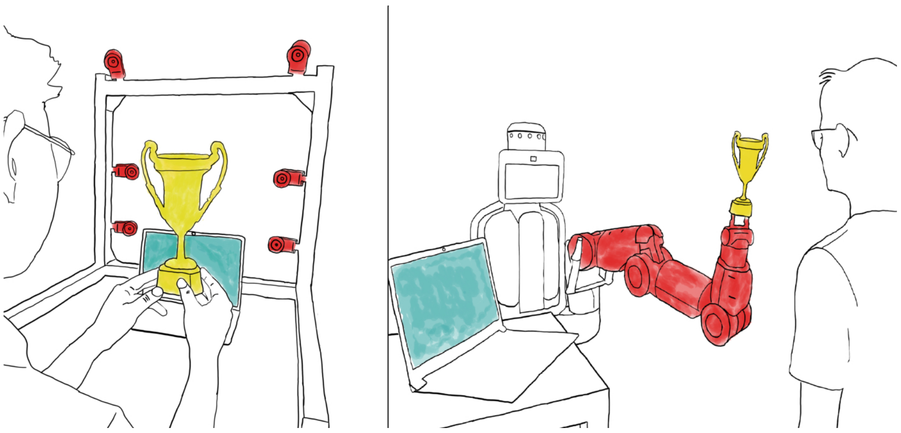


Figure 25: ReMa study setup: The system detects manipulations on an object (Left-yellow) using a set of sensors (Left-red), and then reproduces these with a proxy object (Right-yellow) using a Baxter robot arm (Right-red). ReMa allows shows the Manipulator Site collaborator (Right) the object with the same orientation as at the Tracking Site (Left). Collaborators can optional use video chat (blue) depending on the condition

*ReMa study setup*

Following Chapter 3 and Chapter 4 we designed two studies to evaluate and understand how people would make use of rotation and perspective information in collaborative *matching* tasks. In both studies, both the Tracking Site (TS) and Manipulator Site (MS) participants had their own physical proxy (of the other participant's object). For our studies we decided to move our whole setup to another location where we were able to observe the behavior of the participants. It was also easier to administrate our various components and assisting participants during the study if questions arose. Therefore, both persons were in the same room but they could not see one another (back-to-back). We also switched to a local network solution to avoid any issues due to the internet connection. In the following results and findings sections for both studies, we use a vignette approach where we: first, give a concrete example from our study and subsequently, we explain our conclusion based on these observations.

## 8.1 DESIGN STUDY 1 - THE IMPACT OF PERSPECTIVE

In Study 1, we focused on how different perspectives of a proxy object affects a pair's collaborative interactions and conversation.

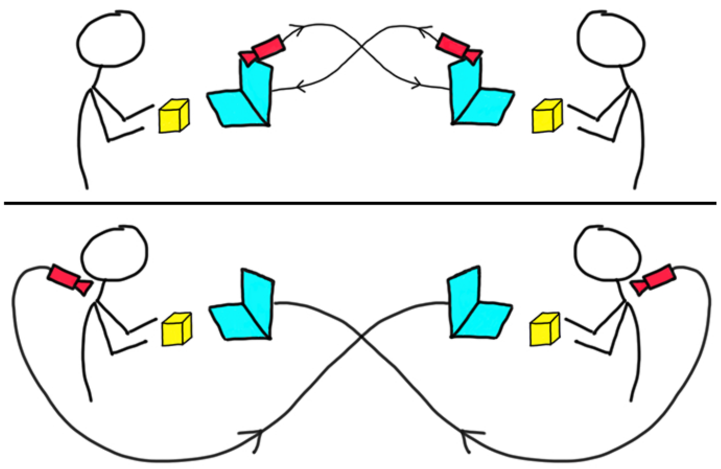


Figure 26: Study 1: VC study setup "Opposing" (top) and "Shared" (bottom)

*Study Variables:* Our central interest was in comparing an Shared perspective, where participants share the same view of an object (see Figure 27 (bottom)), with the Opposing perspective offered by conventional video chat systems (see Figure 27 (top)). We implemented these two perspectives in two technical settings, allowing us to compare videochat (VC) interactions (see Figure 26) with the ReMa system. As illustrated in (see Figure 27), we used a 2x2 within-subjects design (Opposing vs Shared, and VC vs. ReMa), where each pair experienced all four conditions once (each with a different task object/arrangement of stickers).

*Different objects for our user stuy*

*Task design:* We were looking for a task where participants have to explore an object with the use of our system. We chose an alteration of an observation task, in order to have the ability to slightly control the communication flow. Basically, we used a sticker task where participants had to communicate the right position, colour, value, and orientation of the stickers attached to an object given by us (see Figure 28 (2) and (3)). We bought two different objects, a bird house

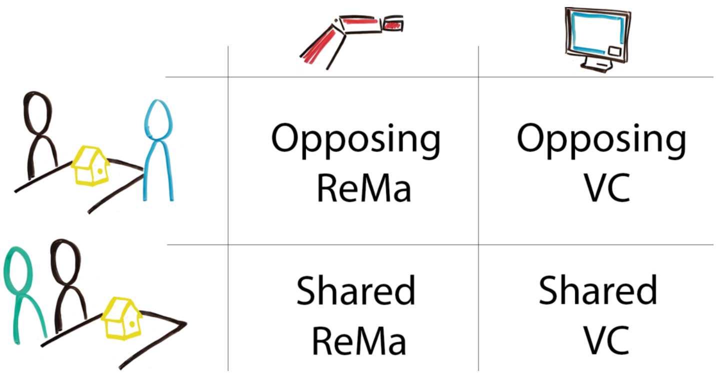


Figure 27: Study 1: Compared the different perspectives Shared vs. Opposing using our ReMa system and a common video chat tool

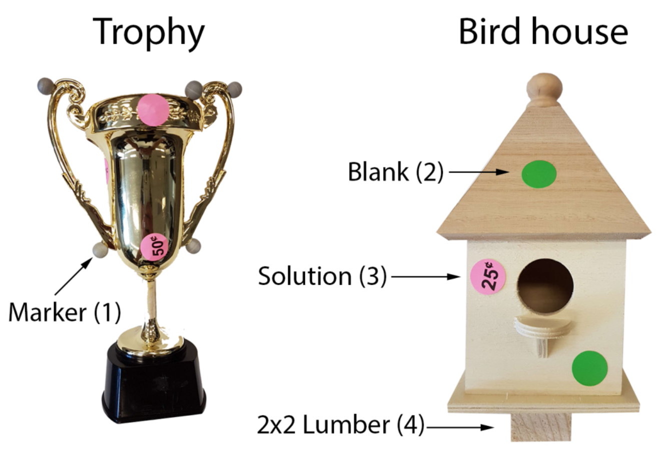


Figure 28: Study 1 objects: Trophy and Bird house

and a trophy. We decided to use these objects due to the different shapes and the various possibilities to attach stickers (e.g. inside the trophy). We prepared the two study objects differently. The object for the Baxter robot needs to be tangible by the robot (i.e. its gripper). We tested many possibilities such as tape, but in most cases the gripper did not have enough grip to hold the object in a fix position during a movement of the robotic arm. Finally, we used two by two lumber which we notched on the sides to improve the grip (see Figure 28 (4)). This approach was sufficient for both objects. For attaching the lumber to the bird house we used wood glue, and for the trophy hot glue. For capturing motions of an 3D object we had to use retro-reflective markers (see Figure 28 (1)). We used hot glue to attach the reflective markers to our study object, and we also prepared backup objects for our study. All in all, we prepared 8 objects for our study. We could easily attach the stickers to the trophies and bird houses. We chose two different sticker setups for trophy and bird house. The place of the stickers where always same during the entire study. We only constrained that we did not run e.g. two bird houses successive to minimize the learning effect.

*Finding from our pilot studies*

Before we ran our actual study, we conducted four pilot studies in our lab. Following that, we changed the position of participants in front of Baxter. Sitting in front of the robot felt intimidating due to its size and appearance. However, as our participants stood in front of Baxter they felt more comfortable and therefore in the actual study one person was sitting in front of the motion capture rig and the second one stood in front of the robot.

*Participants:* We recruited 16 participants (eight pairs; six females; ten males), aged 19-54 with a range of backgrounds including electrical, mechanical and software engineering, computer science, art history and sports science. Each participant was provided with $20 remuneration for their participation. All participants reported experience with video chat software tools like FaceTime or Skype.

*Data collection:* We collected data from six sources: a pre-study questionnaire for demographic information; video of participants as they completed tasks; video feeds of participants during video chat (VC) conditions; ReMa's internal logging (e.g. numbers of rotations; which orientation to which orientation; timing, etc.); field notes and observations; and a post-study interview eliciting participants' experiences with the system. For recording the video material we used two Microsoft full hd cinema lifecams at the Tracking Site for a face and an over the shoulder perspective to capture participants' behaviours and actions. Another cam recorded a side view at the Manipulator Side to provide video material of the participant's interactions with the object and the robot. For the VC variation we used two laptops, a Samsung Ultrabook Serie 9 and a Microsoft Surface Pro 3 to record the laptop facecam. For recording the voice communication and the interviews we used Yeti USB microphones from Blue. We ran 8 groups of participants which resulted in over 40 hours of video material we analyzed.

*Study process:* Here, we describe our study process in the actual study. We introduced our team and explained the consent process for studies at the University of Calgary. After that, we showed participants the study location, the system and explained the study task. To get familiar with the system, participants performed a warm-up task with a third object (network card). Both participants had the chance to explore the system before the actual study. Subsequently, they performed the different trials (see Figure 27). We either started with the ReMa system or the VC condition, due to the counterbalance approach of our study. They performed the two variations Shared and Opposing perspectives with different objects in each system. We did not use the same object in two successive tasks, and we also flipped participants in the Baxter conditions after they finished a trial. At the end of our study we gave both participants a questionnaire and we had a short interview to better understand their experience. Our questionnaire can be found in Section A.3, but essentially we asked them about their background and previous experience with video conferencing tools such as Skype or Google Hangouts. In the second part of the questionnaire we asked them to rate the different variation related to questions given by us (see Section A.3). After that, we used key questions to start a discussion (see Section A.3). Besides, these questions the interview was very open, and we encouraged participants to tell us their opinions and thoughts about the system, as well as suggestions to improve it.

*Analysis:* We conducted a thematic analysis of our data, identifying recurring themes in participant behaviour as they engaged with the

| CONDITION | AVG TIME | VARIANCE |
|---|---|---|
| Opposing-ReMa | 6m29s | 2m10s |
| Opposing-VC | 5m19s | 1m33s |
| Shared-ReMa | 4m40s | 1m52s |
| Shared-VC | 3m24s | 1m22s |

Table 3: Study 1: Task completion times

system, and correlating these with participant's responses in the data collected from the interviews. In addition, we conducted a modified interaction analysis (Jordan & Henderson [26]), where we identified unusual incidents, and used these as points for further understanding of how participants worked with one another. We also used communication flow diagrams to analyze the differences in communication during the trials.

## 8.2 RESULTS & FINDINGS STUDY 1

All participant pairs completed the tasks in the different trials. Generally, Shared perspective trials were better than Opposing perspective trials in terms of completion time. Average completion times for each condition were as follows Table 3.

Pairs were more efficient using the Shared perspective rather than Opposing, regardless of the tool. While we were generally not interested in comparing task completion times between the VC and ReMa conditions (recall that ReMa introduces substantial latency due to the physicality of the robot), we still observe that one of the ReMa conditions is faster than one of the VC conditions.

*Ratings from our participants*

The utility of the Shared perspective is corroborated by data from the questionnaire. On a 10-point Likert scale response to the question, *"Which of these would you prefer to use next time (1-definitely; 10-definitely not)"*, participants overwhelmingly chose the Shared perspective options (median scores: Shared-ReMa (1.5), Shared-VC (2), Opposing-VC (4.5), Opposing-ReMa (5)). Responses followed a similar pattern for participant's rating on ease of use (median scores: Shared-VC (1), Shared-ReMa (2), Opposing-VC (4), Opposing-ReMa (5)). Based on our analysis of participants' behaviour, we observed two principal challenges participants face in Opposing perspective trials that they did not have in Shared perspective trials: first, the Opposing view conditions meant that a participant could not show his/her partner and see for him/herself what was being discussed, and second, in Opposing-VC conditions, partners had a hard time knowing how to "follow along" because of the perspective problem. With the Shared

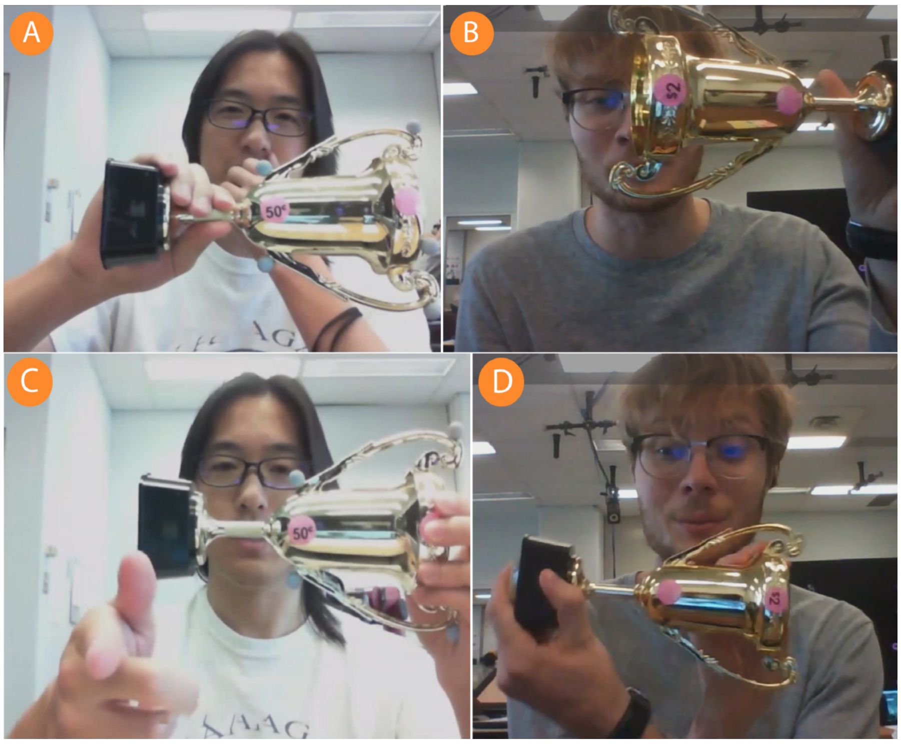


Figure 29: Group 3 Opposing-VC: Frank (A) tries to explain what he sees on one side of the trophy, but Joe rotates his trophy in the wrong direction (B). Frank explains the orientation of his trophy to Joe (C), but Joe is still confused whether he is holding his trophy in the correct orientation (D)

perspective conditions, participants used different strategies made available to them because they knew what the other person saw.

*Overall observations from study 1*

We observed that generally, participants had difficulty organizing and coordinating activity with an Opposing perspective because they had difficulty understanding what the other participant could see. In both VC and ReMa trials, we observed participants turning an object, and pausing the turn to check if the partner could see what was expected. This problem was exacerbated in VC trials, where both participants could turn an object in whatever way they wanted. When they tried to synchronize movement, even a simple misstep was difficult to recover from. This seems to be a symptom of the problem that others have observed [12, 21, 60], where people have difficulty mentally rotating the object and understanding the object from another person's perspective. This problem is well illustrated by the difficulties experienced by G3, where one Frank's re-orienting manipulations on the object are difficult for Joe to copy onto his own object.

***Vignette 1:*** *Group 3, Opposing-VC. Frank orients the object for Joe so that Joe can see the right orientation of the trophy for his sticker (A). Joe tries to align his trophy with Frank's trophy by using the VC preview (B). Joe is uncertain if this is the right orientation of the trophy. Frank and Joe*

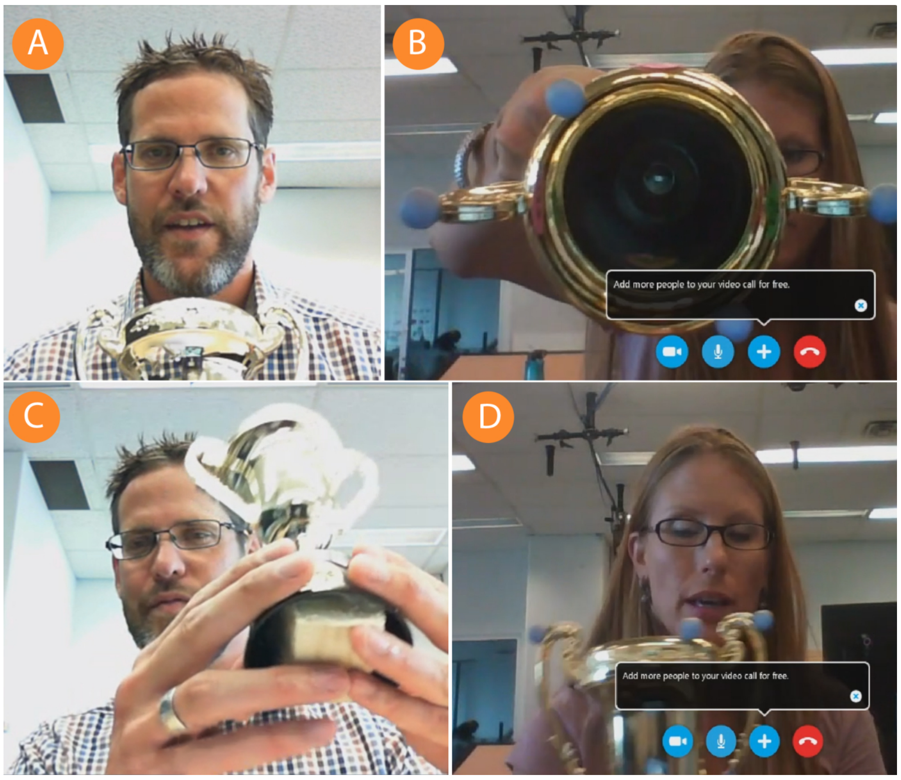

A
B
C
D
Add more people to your video call for free.
Add more people to your video call for free.

Figure 30: Group 1 Opposing-VC. Brenda wants to show Alan a sticker inside the trophy (B), but Alan cannot see the sticker (A). Alan tells Brenda to orient the trophy that both shared the same perspective (C, D)

*try to determine whether they share the same view or not. "The flat part of the trophy is facing you" (C). Ultimately, Joe re-orients his trophy, but is still confused about this orientation: "This feels really weird cause this isn't the side I'm looking at" (D).*

*Issues with mirroring effect*

*Problem of Left-Right:* Vignette 1 shows difficulties in the Opposing-VC particularly with a mirroring effect. Joe uses the video to align his trophy with Frank's trophy. However, he gets confused, as he is observing three different views of the object: Frank's, his own physical object, and the preview in the video chat. Joe is uncertain how or whether indeed he should be matching Frank's view, and in what way: should he rotate left or right, counter-clockwise or clockwise; should he be showing Frank what he is looking at, or should he be doing the same operations so he is looking at the same thing Frank is? Trying to use VC to reach a shared orientation was challenging for most participants, as the perspective draws one's attention away from one's own object. Thus, rotational manipulations on the remote object were difficult to reproduce for most participants.

*Seeing and Showing at the Same Time:* Vignette 2 from G1's Opposing-VC trial (see Figure 30) illustrates how the Opposing perspective results in challenges with both, showing part of the object and describing it. Here, the problem is further exacerbated by the use of video on a flat 2D display.

***Vignette 2:*** *Group 1, Opposing-VC. Brenda shows Alan the inside of the trophy so he can see where to put a sticker (B). Alan is unable to understand from the video which sticker Brenda is referring to, or the orientation of the trophy (A). Brenda tells Alan to reorient the trophy so that they share the same perspective (C,D), but she has a hard time simultaneously showing Alan the inside of the trophy and describing it. She leans forward to look inside her trophy to describe what Alan should be seeing. After struggling to do this, Alan tells Brenda to reorient her trophy to match his perspective on his trophy: "No. Look at... Look inside like I'm looking inside."*

This vignette illustrates that when Brenda is trying to show Alan something in the video, she has difficulty describing it to Alan (which requires her to see it) and showing it to him at the same time. When she points the object towards Alan, she can no longer see it (and is thus relying on memory). On the other hand, when she looks at it to describe to Alan, he can no longer see what she is talking about. As Alan struggles to map his view of Brenda's changing object to his own view of his own object, he realizes that ultimately the video does more harm than good. They later resort to using verbal descriptions of how to rotate the object into the right orientation. Beyond this, we observe that the camera capture itself is problematic: when Brenda tries to show Alan the inside of the trophy, she holds it too close to the camera such that Alan cannot understand the trophy's orientation, which means he cannot extract contextual 3D spatial information from the 2D video.

*Pause Workaround:* Groups experienced similar orientation confusion in the Opposing-ReMa condition; however, five of eight groups developed a clever workaround by re-purposing the "pause" functionality (originally designed to allow participants to examine their objects independently) to create a Shared perspective on the object. In Group 8, TS participant describes this idea:

*Using pause functionality to create a shared perspective*

***Vignette 3:*** *Group 8, Opposing-ReMa. Ava (TS) shows the trophy so that Mia (MS) can see the right side. Ava then pauses the system and orients the trophy for herself that both can look at the same side of the trophy. "So I pause it, then I turn it so I can see the same side," Ava explains. "Ooooh, smart!" replies Mia.*

*"Reset" Strategy Given a Shared Perspective:* The participants generally worked very well in the Shared perspective trials. With Shared-VC, most teams readily identified a "start" position/orientation that they used for the rest of the task. Here, after successfully affixing each sticker, they would revert their own respective objects to the "start" orientation to begin again. In the following vignette, one participant works with the other to establish what the "start" orientation will be.

***Vignette 4:*** *Group 2, Shared-VC. At the beginning of the task, Nancy holds the trophy right-side up, "If you look at the bottom of the trophy, there is a flat side." Ned looks at the wrong side, so he rotates his trophy to find the flat side. To confirm and establish this position, Nancy says to Ned, "Put the flat side in front of you [and] let's call that original position."*

This strategy allowed participants to easily revert to a "known state" if they got into a confusing situation that was difficult to recover from.

*Shared-ReMa Prevents Exploration:* The mental model provided by ReMa in the Shared perspective was straightforward for participants to understand. The automatic reorientation meant that participants did not need to describe re-orientation actions (and potentially have them misinterpreted or reproduced incorrectly), as in the VC conditions. Yet, the tradeoff was that TS participants could not look explore the object, to understand it, or to look ahead at next steps properly without affecting their partner. This problem is illustrated in the following vignette:

*ReMa prevented participants from exploring the object*

***Vignette 5:*** *Group 6, Shared-ReMa. Jon (TS) tells Emi (MS) where to put the sticker, "It's a 25 cent yellow sticker upside-down." Emi begins to attach the sticker, but pauses and asks about a nearby sticker (which could act as a landmark): "Actually, do you see the two. . . ?" Jon twists the bird house to check if there's a sticker left, which startles Emi, who was about to put a sticker down. "What!? Stop, Jon!" Jon sheepishly returns the birdhouse to his original position, "Oops, sorry, I forgot. . . I am controlling the robot arm."*

Thus, TS participants would need to hold the object in place while Manipulator Site participants worked, and could not "look ahead" at other parts of the object without affecting their partner's activities.

*ReMa Conditions - Slow confirmation:* Participants appreciated that ReMa's automatic reorientation meant they shared the same perspective each time the object was reoriented. This reduced the number of interpretation errors between participants: *"I could just assume that we are looking at the same side of the trophy"* (G8-P15). However, when using ReMa, the Tracking Site participants did not know when Manipulator Site had finished re-orienting the object for the other participant, and when/whether the MS participant had completed an action/instruction step on their own object. This is illustrated by Group 5, where the MS participant slows the interaction down by asking several confirmatory questions of his partner to ensure both are looking at the same thing:

*Difficulties as a result of the system delay and no visual feedback from the remote site*

***Vignette 6:*** *Group 5, Opposing-ReMa. Harry (MS) looks at the bird house ReMa has oriented for him. He starts talking about an empty sticker he sees on the left side of the bird house. Ben (TS) is craning his neck to look at the same side of the bird house, because he does not want to move the object, but he cannot see the sticker: "Empty sticker? There is no empty sticker in front..." (Ben). Harry wants to confirm that they share the same perspective:*

*"I just want make sure we are looking on the same side. Is there a little desk in front of the house?"*

This type of confirmatory behaviour was common across all pairs, because neither participant had a strong understanding of what the partner could see. To overcome this problem, pairs would frequently resort to slowing down the interaction, and then ask questions about what the other participant could see, or what they were doing.

*Summary of study 1*

In summary, this study shows that the Shared perspective was far more straightforward for participants to adopt. Pairs developed interaction strategies around this perspective to allow them to complete the task efficiently. In contrast, the Opposing perspective, which is how conventional video chat tools are oriented, caused problems for participants: they had a hard time distinguishing between left-right rotations, and could not see and show aspects of the object at the same time. At the same time, the study raised several questions about the role of ReMa, leading us to design the second study.

## 8.3 DESIGN STUDY 2 - ROLE OF PHYSICAL PROXY

*Focus of the second user study*

Our second study focused on how the presence of or lack of proxy changes collaborative behaviour. Specifically, how is a physical proxy used in the presence of a video channel? We are interested here in situations where both the video chat channel is available alongside a system like ReMa, which can manipulate a physical proxy to mimic actions on another object. *What role does each of these channels play in supporting the collaboration?*

*Study Variables:* As in Figure 31, the second study had three conditions: a video-only condition (VC-Only), a physical proxy-only condition (ReMa-Only), and a combined condition with both video and a physical proxy (VC+ReMa). Based on the findings from our first study, all conditions used a Shared perspective. Participant pairs experienced all three conditions. The video-only and physical proxy-only conditions were presented either first or second (counter-balanced across pairs); the combined condition was always presented last.

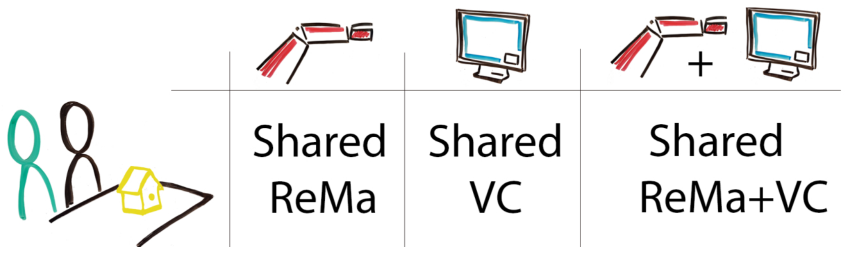


Figure 31: Study 2: Compared Shared video chat, ReMa and a combination of both

*Participants:* We recruited 16 participants (eight pairs; nine females, seven males), aged 18-29 with backgrounds including computer science, actuarial math, animal and medical science, arts, linguistics, and electrical and chemical engineering. No participants from Study 1 were permitted to participate in Study 2. For participating in study each person received $20 CAD remuneration.

*Task Design:* For our second study, we modified the task objects to consider scenarios where object details are more subtle and perhaps more difficult to identify over video alone. In real world scenarios it is not always possible to see orientation, context, complexity, or detail of real-world artefacts (e.g. subtle material cues in design critique [42], larger-scale physical tasks [49]). We were aiming for a study which simulates it. Therefore, we reduced the size of the stickers with the information. On the stickers were different letters we chose by going through the alphabet and eliminated letters that were (a) ambiguous to differentiate between (e.g. W/M) or (b) too symmetrical (e.g. O, X). We aimed for letters that would have a very clear orientation and did not have two "right" answers as to what they could be. Finally, we used the capital letters A - B - G - K - L - R - Y. We reused the plain stickers from the first study, where participants had to write the solution letter with a pen. Both participant's objects have stickers in the same locations. The content of these stickers differ between the participants. Some have a letter written on it, in different orientations. Others are blank, which corresponds to a sticker with a letter on the collaborator's object. Both participants had blank markers on their objects; this meant that both participants needed to exchange knowledge to write letters on their own object's blank markers, such that the objects match at the end of the task. In each study the stickers are at the exact same position on the object. The two bird houses had different sticker setups to minimize the learning effect. We did not flip participants between tasks, in order to get stronger opinions about the different roles. We deactivated the different screen images and the head movements following our observations and interviews from our first study. We decided to not experiment with different faces to avoid distraction. We revised our study tasks to highlight specific issues with physical proxies. In Study 1 we observed that participants could glean considerable information from the video, not only including object orientation, but also specific marker details (e.g. colour and content).

*Task modifications in order to match real world scenarios*

*Data Collection and Analysis:* We followed the same data collection and analysis approach as in Study 1.

*Study process:* We followed the same study process as in Study 1, but with the new study variables (see Figure 31).

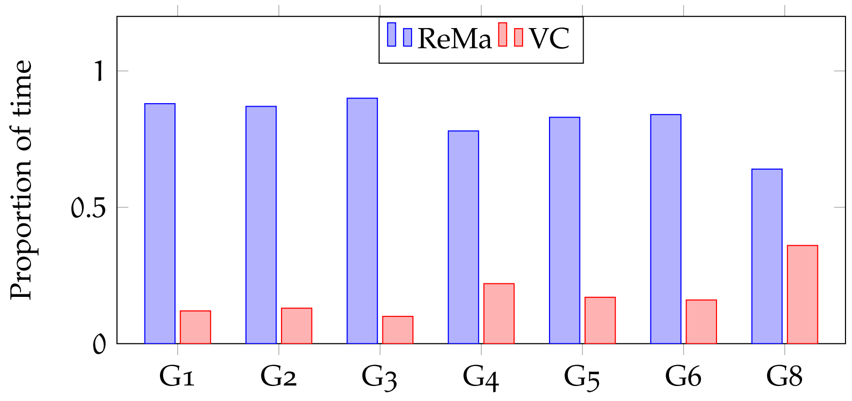


Figure 32: Proportion of time MS participants focused on ReMa (blue) vs. video chat (red) (G7 video data lost)

We also conducted pilot studies before we ran the actual study. We explored having the laptop on the table is insufficient for the combined variation. The laptop position was too far away from the physical object, thus we re-positioned the laptop as close as possible to the robotic arm.

## 8.4 RESULTS & FINDINGS STUDY 2

We focus our analysis on the VC+ReMa trials, where participants had access to both VC and the physical proxy. Based on our analysis, we find that the physical proxy obviated the problems participants have in describing, interpreting and carrying out reorientation acts on the proxy object, while video chat helped participants understand what has happened to the remote proxy object, and gesture. To illustrate this differential use of the video chat and the proxy object, we contrast VC+ReMa against the VC-only and ReMa-only conditions.

*Distinct Roles for VC and ReMa:* From a video analysis of the VC+ReMa trials, we collected data on how MS participants used each channel. Figure 32 illustrates the proportion of time the MS participants focused their visual attention on ReMa compared to the video chat. Between video chat and ReMa, MS participants disproportionately spent their visual attention on the ReMa-manipulated proxy object. Our analysis of the VC+ReMa condition shows that the proxy object was used as a shared workspace (e.g. for the MS to understand in what orientation the object ought to be, or what to do), whereas the video was used for confirming that actions/steps had been taken (e.g. MS ensuring that the TS's object had been rotated to the correct orientation), and for gestures. The following vignette, from Group 2, typifies the VC+ReMa experience of seven of our eight groups:

***Vignette 7:*** *Group 2, VC+ReMa. Susan (TS) rotates her object, telling Larry (MS), "I'm gonna turn it". The robot rotates the object and Susan*

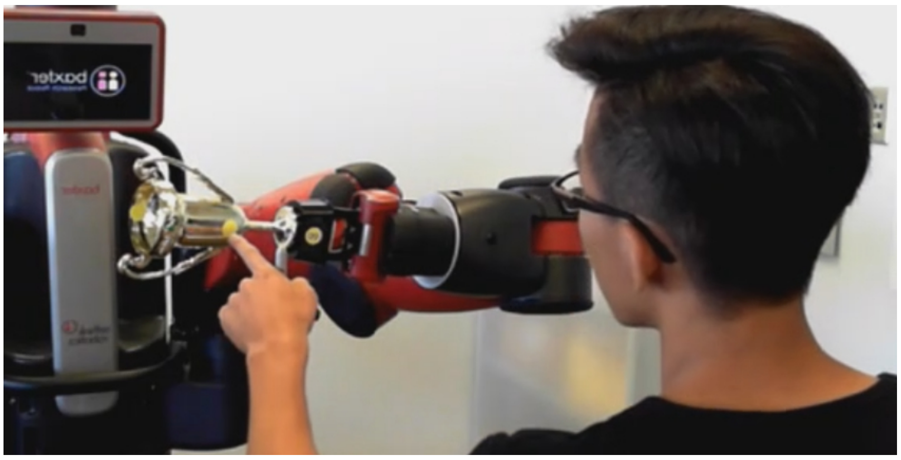


Figure 33: VC+ReMa – Group 2. Larry (MS) points at the trophy confirming the sticker position, knowing that Susan (TS) can see the gesture through the video

*says: "It's gonna take a sec." While the robot is rotating Larry's object, Susan watches the robot through the video chat. All the while, Larry is watching the robot and the object carefully. Once the robot completed the reorientation, Susan says "So... YES, at the back of the trophy there is a G." Larry, knowing Susan can see him in the VC, points at the sticker to confirm the sticker (see Figure 33). Finally, he peeks at the VC to confirm the orientation of the trophy before writing the letter.*

*Using the video channel for confirmation*

This vignette highlights three aspects of the interaction as it relates to the different channels. First, Larry's primary interest is on his immediate workspace: the proxy object, held by ReMa. Most of his visual attention is here — he waits for the proxy object to settle into position, and once its position is stable he writes on the project object. Second, Larry generally does not use video chat, with the exception of understanding Susan's gestures and ensuring that his object roughly matches Susan's. Similarly, Susan's primary use of the video chat is to watch as Larry's object rotates into place — she uses the video to confirm that ReMa has executed her rotation act properly. Ultimately, Susan waits for visual confirmation that Larry has completed the task before she moves onto the next marker. Thus, the video and proxy object each play distinct roles in supporting the interaction; we see this in the absence of one channel during the VC-only and ReMa-only trials.

While seven out of eight pairs used video in this way, G8 was an outlier. The MS participant used the video instead of the robot as a primary visual reference for solving the task. The pair communicated almost strictly via video chat, the MS determining the position and solution letter for blank stickers, going to the robot with the object ready in the right orientation, and finally writing the letter on the marker.

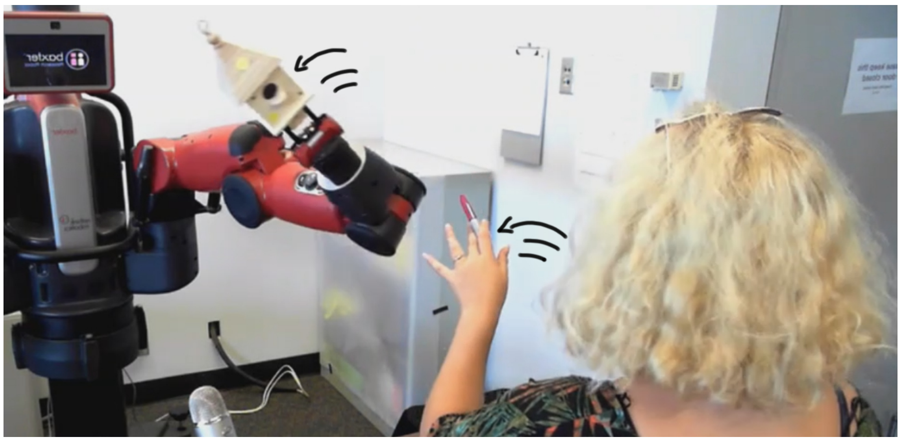


Figure 34: VC+ReMa – Group 1. Clara (TS) uses spatial hand gesture to describe the movement Lina (MS) should execute (annotated for clarity)

*ReMa-only does not support gestures*

*Using Gestures to Communicate Re-Orientation Acts:* As in Study 1, the MS participants were severely disadvantaged in communicating back to the TS participant. The MS participant could not physically manipulate their own object. Thus, to communicate how an object ought to be rotated, MS participants frequently used hand gestures to rotate an imaginary object in midair, providing a verbal description alongside the gesture. The TS participants could only rely on the video chat channel to understand what was intended by MS's description. TS then rotated the object as they interpreted the instructions, and MS confirmed based on the rotation of the physical proxy in front of them.

***Vignette 8:*** *Group 1, VC+ReMa. Clara (TS) wants to describe the right orientation of the object for Lina (MS). First, she describes the position of the sticker: "In the left bottom corner" Lina confirms and asks: "Yes, bottom left... what should I do?" Clara is uncertain how to describe the movement: "Just move it... left... 90 degrees to the left". She uses hand movements to show how Lina should move the object (see Figure 34). While Clara explains and gestures the movement Lina is watching the video chat to better understand Clara's gestures.*

*VC-only findings*

*VC-Only Trials:* Compared with the VC+ReMa trials, the pairs' main challenge was to effectively describe re-orientation actions to their partner, or to interpret those instructions (and carry them out properly). While they could use the video chat to observe the remote site, and interrupt when problems occurred, the presence of the video chat did not prevent these mistakes from happening.

*Dialogue from these trials were fragmented:* Participants used step-by-step language to describe their actions and stay aligned with the remote person. As a participant provided instructions, s/he would watch the video chat to see if/whether the instructions had been understood, repairing the interaction as necessary. In one instance, participant [G1-P2] needed three tries before he is satisfied with the outcome: *"And don't move the house...in this position there is. . . Wait, just move the house 90 degrees to the left"* In another example, the participant [G5-P10] realizes that the instruction he just gave is ambiguous, and tries to repair it twice, all while watching his remote partner struggle: *"Just rotate it. . . That means you just keep the tip of the house at the table and the base of the house upwards... facing the roof."*

Because participants could see each other's object in the video chat, they could catch errors quickly; however, this interaction was far from smooth. Just as in Study 1, many pairs defined a "start" position at the beginning of the task to avoid orientation confusion, and returned to this position when their objects became misaligned. In contrast, the VC+ReMa trials were smoother: TS participants did not need to verbally convey reorientation actions to their MS, resulting in dialogue that was much clearer and focused on the markers themselves. Confirmation that the proxy object had moved correctly, or that MS participant had completed the action correctly relied on the video chat channel.

*ReMa-only findings*

*ReMa-Only Trials:* Due to the absence of a video chat channel, participants needed to communicate verbally or through the orientation of the proxy object (via ReMa). This presented challenges for both the TS and MS, but in different situations. As in Study 1, the TS relied on verbally confirming with the MS about whether the ReMa had finished its movements, and whether the MS participant had finished his/her actions (i.e. writing the symbol on the sticker). We observed participants regularly and explicitly asking for verbal confirmation (e.g. *"Are you ready?"* [G1-TS] or *"You got it?"* [G3-TS]), because they had no other way to know the current state at the remote site. Where the video chat acted as a feedback mechanism in the VC+ReMa conditions, its absence in VC-only conditions markedly increased verbal confirmation cues.

Because the MS participant could not manipulate the TS participant's workspace, s/he described re-orientation steps for the TS object verbally (as in Vignette 8). Interestingly, MS participants still used spatial hand gestures to describe rotations (as in the VC+ReMa condition); of course, these hand gestures were not visible to the TS participant, and MS participants confirmed they were aware that TS participants could not see actions in a video stream. Instead, these "rehearsal" acts seemed let participants explain actions from a first-person perspective.

All TS participants oriented the proxy object such that the correct side of the object faced the participant, and that side was oriented such that the MS participant could easily write the letter "right side up." This contrasts with Study 1, where only two groups oriented the object such that the markers were placed "right side up"; most groups in Study 1 kept the object oriented "right side up", even if that meant the MS needed to affix the sticker on upside down. We suspect that the revised task and markers influenced this change — in Study 2, the MS had to write a symbol, whereas in Study 1, MS only needed to affix the sticker.

*No Pausing Necessary:* As in Study 1, we provided a pause functionality; however, no groups used this function in any trial. Given that the bulk of its use in Study 1 was during Opposing perspective conditions to mimic a Shared perspective, this is lack of use is perhaps unsurprising since Study 2 only used the Shared perspective. No participants complained about the use of a Shared perspective, and found it very straightforward and easy to use: *"You can just look at it and you see whatever the other one is seeing..."* [G7-P14] or *"It is easier to understand what the other person is really looking at"* [G8-P16].

*Summary of the second study*

In summary, study 2 focuses on how people use video chat and ReMa differently given the presence of both channels. ReMa was used primarily for orienting the shared workspace. Meanwhile, the video chat let people visually confirm what happened to the proxy object, and offered a means to gesture at the proxy object.

# 9 DISCUSSION & FUTURE WORK

In this chapter we discuss our results of the technical evaluation as well as the user studies, and we provide implication for future work. We start with the technical discussion and future work.

*Discussing our alternative approach of mapping object orientation to a robot end-effector*

*Elaborate executed robot movements:* In Chapter 7 we argued why often robot movements are "unnatural" and bigger than excepted. In addition to that, it is also a result of the inverse kinematics problem and the fact that we tracked the position and orientation of the object. Similar projects tracked the arm/hand position and mapped human wrist, elbow joints to the robot joints. However, our focus was on the object and because previous technial projects addressed hand/arm tracking with technologies such as Microsoft Kinect[1], Leap Motion[2] or Myo wristbands[3] we wanted to explore additional ways. Furthermore, grasping/holding/manipulating of objects is challenging when tracking the hands, because Baxter does not have a "hand (gripper)" similar to humans and has different kinematics and speed capabilities [51]. We also observed that people played with the object (i.e. manipulated it with both hands), which was not a problem with our system, but is difficult to realize with a direct mapping between human hand and robot end-effector. Future system designer/developer should consider whether the *use-case* of the system requires a direct mapping or not.

*Suggesting another alternative*

*Robotic limitations:* Independently of object - or hand/arm tracking, Baxter's capabilities are heavily limited. We do not have experience with other robots in order to evaluate it, but we think for reproducing object manipulations a simple robotic arm with 4DoF would be more effective. We think the arrangement, complexity and size of Baxter's seven different joints makes it (a) more difficult to find IK solutions and (b) needs a lot of time to physically reproduce the manipulation. Except for the robot initialization, our system could be easily re-used with other robots that provide the URDF, addressing future research in that field.

Below we move from the technical context to reflections relating to the design of future object-focused collaborative interfaces.

*Perspective Shifts:* Non-collocated collaborators have the flexibility to independently orient themselves (or an object) during an object-

1 https://www.xbox.com/en-US/xbox-one/accessories/kinect
2 https://www.leapmotion.com/
3 https://www.myo.com/

focused discussion. Yet, both studies demonstrated that a Shared perspective is useful and powerful when executing tasks with the object. The Shared perspective allowed participants to discuss the object without mentally rotating the object, and to describe parts of the object that are not necessarily visible from their perspective. Designers need to carefully consider camera placement and object/workspace orientation in systems like ReMa to reduce people's need to mentally rotate objects.

*Impact of perspective*

*Uses of Physical Object and Rotations:* The ReMa system and study confirm that indeed, rotations are an important part of object-focused collaboration. In prior work [33], rotations of flat 2D objects can play a role in both communicating (i.e. explaining something to someone else), as well as coordinating action (i.e. whose turn is it to do something). We saw a similar effect in our study: TS rotations of the object functioned as demonstrations of where to place a sticker, or annotate the object. Rather than requiring the TS to verbally describe how to turn the object, or demonstrating via video, ReMa simply performed the rotation. Performing the rotation both drew attention to a specific side of the artefact (communication), and signaled to the MS participant that something needed to be done at that site (coordination).

*Using object orientation for communication*

*Handling for Comprehension:* ReMa does not explicitly address the use of rotating an object for comprehension (i.e. exploring the object, or taking time to understand the object). The MS participant cannot explicitly hold or manipulate their object; when the MS wanted to explore their object, they had to explicitly ask the TS to re-orient their object for them (Vignette 8). As a result, the MS participant does not get to manipulate and explore the object for themselves—all object interactions are mediated by cumbersome dialogue with TS. We also observed that TS participants were more guarded in exploring their own object. As we saw in Vignette 5, when TS participants realized that their actions were immediately reflected at the MS (and might potentially disrupt the MS participant's actions), they avoided excess object manipulations. In contrast, participants in video chat-only conditions were free to manipulate their object, but at a cost of coordination. Video chat-only participants' objects easily became unsynchronized, necessitating the "start position" strategy.

*ReMa prevent exploration of the physical object*

In Study 1, we tried to design for comprehension acts by providing "Pause" functionality, which temporarily disabled the TS from the MS. In principle, this allowed the TS participant to explore the object without changing the MS participant's view of the object. However, few participants used it for this purpose; participants mainly used "Pause" to recreate a Shared perspective during Opposing perspective conditions. Future designs need to consider different ways to move between synchronized and un-synchronized remote objects: for example, through a clutching mechanism activated by proximity (i.e. only tracking a "shared" workspace, leaving personal workspace for independent manipulation), or manual clutching (similar to the pausing mechanism).

*Expanding the Manipulator Space:* We are interested in expanding the capabilities of the manipulator site, particularly to capture more degrees of freedom and object movement paths. This additional information is important for object-focused collaborations to describe: relationships between different objects; how an object should be used or oriented, or how a multi-part object might be assembled. Capturing the timing of a movement path is important, too. The current ReMa implementation is limited to manipulating an object's orientation (at 90 degree angles), which was sufficient to address our current research questions. We are interested in further developing the ReMa infrastructure such that more flexible and rich movements, positions and timing are accurately rendered. This would allow even subtle gestures or manipulations involving the object to be conveyed at a distance (e.g. [28]).

*Going beyond the current cababilities*

*Capturing and Rendering Manipulator Gestures:* Prior work focused on providing collaborators mechanisms with gesturing at objects in the workspace, or at areas of the workspace [10, 14, 15, 20, 28, 29, 35, 39, 54, 55]. Our study participants used gestures—particularly in the video chat conditions—to point at various parts of the object. When this capability was taken away in the ReMa conditions, this presented challenges for participants. Future research needs to develop new ways to both capture gestures (such as deictic or hand gestures) for object-focused collaboration, as well as determining how to render these gestures at a remote site for interpretability. While video is a reasonable stop-gap solution, it ignores the subtleties of gesturing at partially obscured or difficult-to-view locations on an object. It also misses the entire production of the gesture, which may be important for interpreting the meaning of a given gesture [39].

*ReMa does not support gestures*

*Bidirectional Capture and Manipulation:* While interaction with the physical object in our study was strictly unidirectional, we are also interested in bidirectional scenarios. As illustrated in [6], while bidirectional physical objects present compelling experiences, they also present new questions. Most notably: how should conflicts be resolved? One approach to resolving conflicts is to relax what would otherwise be strict synchronization. In this "relaxed" synchronization mode, a collaborator could choose whether to follow the remote site's object depending on his/her situation. We envision a mechanism that would allow a collaborator to explore their own interaction path with an object, and resynchronize with their remote collaborator when needed with little penalty to either.

*The eventual goal of a bidirectional system*

*Does the humanoid robot affect our results?*

*ReMa, and Human-Robot Interaction (HRI):* Due to opportunistic reasons we used a humanoid robot for both ReMa studies, and (in Study 1) we enabled head movements and different "facial" displays to provide feedback on ReMa's movements. Extensive past HRI work investigating the impact of anthropomorphism on interaction suggests that our choice may have affected our results, and that realizing ReMa with a more generic robotic arm could have potentially created different biases (e.g. [11, 27]). However, we found little evidence of the humanoid form effects, for example none of our participants recognized the "facial" displays in the post-study interview: *"I was so focused on the task and the object. I did not see [the face] at all"* [G4-P7]. We removed these feature for Study 2. Future ReMa-like systems should include and evaluate the effect of replacing the humanoid with a simple robotic arm implementation.

# 10 CONCLUSION

This thesis explains the development of a novel system ReMa, which automatically orients the proxy object to reflect the orientation at a remote location. We evaluate the technical parts of the system components and we also provide a ROS package making it accessible for future research. We built and studied the Remote Manipulator (ReMa) and explored the challenges of coordinating object-focused collaboration when collaborators are remote from one another. Specifically, we considered how collaborators' perspectives on an object affects the way in which they coordinate activity. We found that a shared perspective on the object is easier for people to manage compared to the default Opposing perspective offered by conventional video chat. We also found that ReMa can be a useful aid to collaboration, easing the pressure of describing and reproducing verbal reorientation cues on an object. Finally, our analysis shows that ReMa and a video channel complement each other when used together, giving people more effective tools to coordinate their actions in object-focused collaboration. Looking forward, our results suggest ways that researchers should consider new workspaces that improve object-focused collaboration, including supporting simultaneous object manipulation and remote gesture, managing synchronized and unsynchronized object manipulation, and handling bidirectional capture and manipulation.

# APPENDIX A

This chapter provides supplementary materials for this thesis. In Section A.1 we provide more background information about the mathematical fundamentals, the gimbal lock problem as well as quaternions. Next, Section A.2 shows the OptiTrack settings we chose for our prototype system ReMa. Finally, Section A.3 provides the questionnaires and the interview questions we used for both user studies.

## A.1 MATHEMATICAL

Here, we elucidate a special matrix, called a rotation matrix. Rotation matrices are used to describe for instance a rotation of a vector $\vec{v}$ in Euclidean space. Rotating a vector $\vec{v}$ can be achieved through a matrix multiplication $Rx\vec{v}$, where $\vec{v}$ must be a column vector. Here, we only consider rotations in three-dimensional space.

*Rotation matrix for describing rotations in 3D space*

*Note:* Rotation matrices are square matrices with $\theta \in \mathbb{R}$. In three dimensional space we have a 3x3 rotation matrix to describe a rotation around a specific axis. In particular, the rotation matrix is an orthogonal matrix with $\det |R| = 1$, and $R^T = R^{-1}$.

The following three matrices are used for basic rotations, or elemental rotations with an angle $\theta$ around the $x$,$y$ or $z$ axis.

$$\mathbf{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \mathbf{R}_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$\mathbf{R}_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

It is clear that vector $\vec{v}$ must be a column vector. Otherwise, we could not multiply it with the rotation matrix $R$. Summarized, we have a 1x3 vector $\vec{v}$ and a 3x3 rotation matrix $R$, which we can multiply. These rotation matrices are valid for left-handed and right-handed Cartesian coordinate system. However, we do not only perform a single rotation. To achieve that, we can multiply the elemental rotation matrices. For instance, we want to rotate our vector $\vec{v}$ 90 degrees around the x-axis, and after that another 90 degrees around the y-axis. We can simply multiply the rotation matrices $R_x(\theta)$ and $R_y(\lambda)$ with $\theta$,$\lambda$ = 90 degrees. The resulting rotation matrix $R_{x,y}(\theta, \lambda)$ describes the ori-

| **x** | **1** | ***i*** | ***j*** | ***k*** |
|---|---|---|---|---|
| **1** | 1 | *i* | *j* | *k* |
| ***i*** | *i* | -1 | *k* | -*j* |
| ***j*** | *j* | -*k* | -1 | *i* |
| ***k*** | *k* | *j* | -*i* | -1 |

Table 4: Quaternion multiplication

entation which we finally multiply with vector $\vec{v}$.

$$R_{xy}(90,90) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(90) & -\sin(90) \\ 0 & \sin(90) & \cos(90) \end{pmatrix} \begin{pmatrix} \cos(90) & 0 & \sin(90) \\ 0 & 1 & 0 \\ -\sin(90) & 0 & \cos(90) \end{pmatrix}$$
$$= \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

*Similarity to quaternions*

As we know matrix multiplication is non-commutative (like quaternions). Conclusively, when we use the rotation matrix to determine the rotation of a vector $\vec{v}$, we must consider the sequence of the rotations.

*Quaternions into rotation matrix* (only works with normalized quaternions Section 2.1): $q = qa + i * qb + j * qc + k * qd$

$$\begin{pmatrix} 1 - 2 * qc^2 - 2 * qd^2 & 2 * qb * qc - 2 * qd * qa & 2 * qb * qd + 2 * qc * qa \\ 2 * qb * qc + 2 * qd * qa & 1 - 2 * qb^2 - 2 * qd^2 & 2 * qc * qd - 2 * qb * qa \\ 2 * qb * qd - 2 * qc * qa & 2 * qc * qd + 2 * qb * qa & 1 - 2 * qb^2 - 2 * qc^2 \end{pmatrix}$$

*Describing orientations using Euler angles*

Another common approach to describe rotations is Euler angles, especially in physics and aerospace engineering. Every possible rotation can be achieved as a result of three elemental rotations. Euler angles are made of three angles, each angle around an axis in the standard coordinate system. Following that, we have three matrices, one for each axis and we multiply each 3x3 matrix. Even with Euler angles the sequence of the rotations are very important, because they result in a different solution for the rotation matrix. Basically, there are six different possibilities, because we have three different rotation axes $3! = 6$. Often, these rotations are also called Roll, Pitch, and Yaw, where Roll would be a rotation around the x-axis , Pitch a rotation around the y-axis and Yaw around z. Following is an example from the above introduced rotation matrix with Euler angles. Roll, Pitch and Yaw is the sequence of our rotations. That means we have to multiply $R_x$ and $R_y$, and after that $R_{xy}$ with $R_z$. The resulting matrix is:

Euler-Angles: $\phi = 90$; $\theta = 90$; $\psi = 0$

$$\mathbf{R}_{\mathrm{RPY}}(\phi, \theta, \psi) = \mathbf{R}_{\mathrm{RPY}}(90, 90, 0) = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

As we see, when we consider the sequence of the rotations we get the same result with Euler angles. However, there are disadvantages with Euler's representation of orientations. The greatest issue with Euler angles is *gimbal lock* which we briefly explain in this section.

*What is a gimbal lock?*

*Gimbal Lock:* Gimbal lock describes the loss of one degree of freedom. Figure 35 (left) shows the three different gimbals red, green and blue for rotations in Euler representation. We can generate a gimbal lock by a 90 degree rotation around one of the coordinate axes Yaw,Pitch or Roll. For instance, in Figure 35 (right) we see that all three gimbals are in the same plane. Conclusively, a rotation around one axis is not possible anymore. Although we are able to rotate around each axis, two axis will always result in the same position.

In robotics, a similarity to gimbal lock is called *wrist flip*. However, we will not discuss it in this thesis.

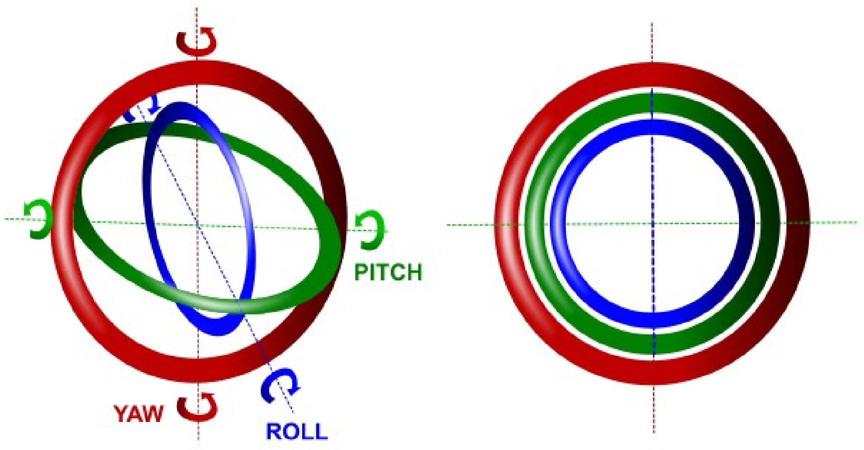


Figure 35: Gimbal lock problem[1]

## A.2 TRACKING SETTINGS

*Detailed configuration settings of the cameras*

The Motive tool provides users various configuration opportunities making it flexible and applicable for many situations. We used the documentation and a trial and error approach to find the best settings for our scenario. We used six Flex 13 cameras capturing the volume with 120 frames per second (FPS). We chose a resolution of 1280 x 1024 corresponding with 1.3 mega pixel. The camera exposes per frame (EXP) which is measured in microseconds was 500 of 7500. If we increase the EXP, it improves the visibility for smaller markers.

1 http://documentation.quest3d.com

On the other hand a higher EXP results in a higher probability of errors because it will recognize even smaller reflections, and eventually interpret them as markers. We set the threshold to 200 of 255 to determine the required brightness of a reflective marker for recognition. It depends on the distance between our cameras and markers. We chose a relatively high number because we operated very close the cameras, and we wanted to avoid interference with external light. The LED Illumination (LED) allowed us to set the brightness level for the camera IR LED ring. As mentioned, we operate very close to the camera, therefore we used a lower setting for the LED. At the beginning we had the highest possible brightness, but we saw a lot of interference with even our hands, in particular with human fingernails. After many tests, we finally decided to use the setting 3-4 (depending on the day) of 15, where 15 is the brightest configuration. Finally, we used the standard settings for "short range" and "precision mode" in the camera configuration menu.

## A.3 STUDY MATERIALS

The following pages provide the questionnaire from the first and the second study, as well as the interview questions.

Group #____________, Participant # ____________

**Questionnaire study 1**

A) Gender

__ Female __ Male __ Prefer not to answer

B) Age _____

C) Professional Background/Major: ____________________________________________

D) How often do you use video conferencing software such as Skype, Google Hangouts Video, FaceTime etc.?

__ I have not used any video conferencing software before

__ less than once a week

__ once a week

__ several times a week

__ daily

E) What type of activities have you used video conferencing for? Select all that apply.

__ I have not used any video conferencing software before

__ chat 1-on-1

__ chat with multiple people

__ share video of a product or object

__ get help with a task

__ give help to others on a task

__ other (please describe):

Group #____________, Participant # ____________

F) Overall impression of the system: Given the choice, which of these systems would you use

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baxter: Face to face | | | | | | | | | | | |
| Baxter: Over the shoulder | | | | | | | | | | | |
| Skype: Face to face | | | | | | | | | | | |
| Skype: Over the shoulder | | | | | | | | | | | |

Very much prefer — Would avoid

G) Ease of use: How easy was each system to use?

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baxter: Face to face | | | | | | | | | | | |
| Baxter: Over the shoulder | | | | | | | | | | | |
| Skype: Face to face | | | | | | | | | | | |
| Skype: Over the shoulder | | | | | | | | | | | |

Very easy — Very difficult

H) Communication: How easy was it to communicate with your partner?

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baxter: Face to face | | | | | | | | | | | |
| Baxter: Over the shoulder | | | | | | | | | | | |
| Skype: Face to face | | | | | | | | | | | |
| Skype: Over the shoulder | | | | | | | | | | | |

Very easy — Very difficult

I) Perspective: How easy was it to know what your remote partner could see?

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baxter: Face to face | | | | | | | | | | | |
| Baxter: Over the shoulder | | | | | | | | | | | |
| Skype: Face to face | | | | | | | | | | | |
| Skype: Over the shoulder | | | | | | | | | | | |

Very easy — Very difficult

J) Remote Perspective: How easy was it for your partner to see what you could see?

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baxter: Face to face | | | | | | | | | | | |
| Baxter: Over the shoulder | | | | | | | | | | | |
| Skype: Face to face | | | | | | | | | | | |
| Skype: Over the shoulder | | | | | | | | | | | |

Very easy — Very difficult

Group #____________, Participant # ____________

**Questionnaire study 2**

A) Gender

__ Female     __ Male     __ Prefer not to answer

B) Age _____

C) Professional Background/Major: ______________________________________________

D) How often do you use video conferencing software such as Skype, Google Hangouts Video, FaceTime etc.?

__ I have not used any video conferencing software before

__ less than once a week

__ once a week

__ several times a week

__ daily

E) What type of activities have you used video conferencing for? Select all that apply.

__ I have not used any video conferencing software before

__ chat 1-on-1

__ chat with multiple people

__ share video of a product or object

__ get help with a task

__ give help to others on a task

__ other (please describe):

Group #_____________, Participant # _____________

F) Overall impression of the system: Given the choice, which of these systems would you use

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baxter Only | | | | | | | | | | | |
| Skype Only | | | | | | | | | | | |
| Baxter and Skype | | | | | | | | | | | |

Very much prefer Would avoid

G) Ease of use: How easy was each system to use?

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baxter Only | | | | | | | | | | | |
| Skype Only | | | | | | | | | | | |
| Baxter and Skype | | | | | | | | | | | |

Very easy Very difficult

H) Communication: How easy was it to communicate with your partner?

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baxter Only | | | | | | | | | | | |
| Skype Only | | | | | | | | | | | |
| Baxter and Skype | | | | | | | | | | | |

Very easy Very difficult

I) Perspective: How easy was it to know what your remote partner could see?

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baxter Only | | | | | | | | | | | |
| Skype Only | | | | | | | | | | | |
| Baxter and Skype | | | | | | | | | | | |

Very easy Very difficult

J) Remote Perspective: How easy was it for your partner to see what you could see?

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baxter Only | | | | | | | | | | | |
| Skype Only | | | | | | | | | | | |
| Baxter and Skype | | | | | | | | | | | |

Very easy Very difficult

For experimenter use only: [Skype: H | W] [Baxter: H | W]
Group #____________

# Interview questions

K) What kinds of challenges did you experience with the Baxter (only) system?

L) What kinds of challenges did you experience with the Skype (only) system?

M) Did you trust Baxter (Baxter only)?

N) Tradeoffs? Lose/Win – Advantages/Disadvantages

O) Use pause function yes/no? Why?

P) Reasons for Baxter/video – applications

Q) Scenarios physical representation useful

R) More attention to video/Baxter – Why? (Study 2)

# BIBLIOGRAPHY

[1] Ronald M Baecker. *Readings in groupware and computer-supported cooperative work: Assisting human-human collaboration*. Elsevier, 1993.

[2] Istvan Barakonyi, Tamer Fahmy, and Dieter Schmalstieg. "Remote Collaboration Using Augmented Reality Videoconferencing." In: *Proceedings of Graphics Interface 2004*. GI '04. London, Ontario, Canada: Canadian Human-Computer Communications Society, 2004, pp. 89–96. ISBN: 1-56881-227-2. URL: http://dl.acm.org/citation.cfm?id=1006058.1006070.

[3] Lukas Barinka. *Inverse Kinematics - Basic Methods*. Tech. rep. Czech Technical University, 2002. URL: http://old.cescg.org/CESCG-2002/LBarinka/paper.pdf.

[4] Patrick Beeson and Barrett Ames. "TRAC-IK: An open-source library for improved solving of generic inverse kinematics." In: *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE. 2015, pp. 928–935.

[5] Mark Billinghurst and Hirokazu Kato. "Collaborative Augmented Reality." In: *Commun. ACM* 45.7 (July 2002), pp. 64–70. ISSN: 0001-0782. DOI: 10.1145/514236.514265. URL: http://doi.acm.org/10.1145/514236.514265.

[6] Scott Brave and Andrew Dahley. "inTouch: A Medium for Haptic Interpersonal Communication." In: *CHI '97 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '97. Atlanta, Georgia: ACM, 1997, pp. 363–364. ISBN: 0-89791-926-2. DOI: 10.1145/1120212.1120435. URL: http://doi.acm.org/10.1145/1120212.1120435.

[7] Jed R Brubaker, Gina Venolia, and John C Tang. "Focusing on shared experiences: moving beyond the camera in video communication." In: *Proceedings of the Designing Interactive Systems Conference*. ACM. 2012, pp. 96–105.

[8] Herbert H Clark, Susan E Brennan, et al. "Grounding in communication." In: *Perspectives on socially shared cognition* 13.1991 (1991), pp. 127–149.

[9] Rosen Diankov and James Kuffner. "Openrave: A planning architecture for autonomous robotics." In: *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34* 79 (2008).

[10] Omid Fakourfar, Kevin Ta, Richard Tang, Scott Bateman, and Anthony Tang. "Stabilized annotations for mobile remote assistance." In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM. 2016, pp. 1548–1560.

[11] Julia Fink. "Anthropomorphism and human likeness in the design of robots and human-robot interaction." In: *International Conference on Social Robotics*. Springer. 2012, pp. 199–208.

[12] Christian Freksa. "Qualitative spatial reasoning." In: *Cognitive and linguistic aspects of geographic space* 63 (1991), pp. 361–372.

[13] Susan R Fussell, Leslie D Setlock, and Robert E Kraut. "Effects of head-mounted and scene-oriented video systems on remote collaboration on physical tasks." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2003, pp. 513–520.

[14] Susan R. Fussell, Leslie D. Setlock, Jie Yang, Jiazhi Ou, Elizabeth Mauer, and Adam D. I. Kramer. "Gestures over Video Streams to Support Remote Collaboration on Physical Tasks." In: *Hum.-Comput. Interact.* 19.3 (Sept. 2004), pp. 273–309. ISSN: 0737-0024. DOI: 10.1207/s15327051hci1903_3. URL: http://dx.doi.org/10.1207/s15327051hci1903_3.

[15] Steffen Gauglitz, Benjamin Nuernberger, Matthew Turk, and Tobias Höllerer. "In Touch with the Remote World: Remote Collaboration with Augmented Reality Drawings and Virtual Navigation." In: *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*. VRST '14. Edinburgh, Scotland: ACM, 2014, pp. 197–205. ISBN: 978-1-4503-3253-8. DOI: 10.1145/2671015.2671016. URL: http://doi.acm.org/10.1145/2671015.2671016.

[16] Ray C Goertz and William M Thompson. "Electronically controlled manipulator." In: *Nucleonics (US) Ceased publication* 12 (1954).

[17] Andrew Goldenberg, Beno Benhabib, and Robert Fenton. "A complete generalized solution to the inverse kinematics of robots." In: *IEEE Journal on Robotics and Automation* 1.1 (1985), pp. 14–20.

[18] Saul Greenberg, Carl Gutwin, and Mark Roseman. "Semantic telepointers for groupware." In: *Computer-Human Interaction, 1996. Proceedings., Sixth Australian Conference on*. IEEE. 1996, pp. 54–61.

[19] Pavel Gurevich, Joel Lanir, and Benjamin Cohen. "Design and Implementation of TeleAdvisor: A Projection-Based Augmented Reality System for Remote Collaboration." In: *Comput. Supported Coop. Work* 24.6 (Dec. 2015), pp. 527–562. ISSN: 0925-9724. DOI: 10.1007/s10606-015-9232-7. URL: http://dx.doi.org/10.1007/s10606-015-9232-7.

[20] Pavel Gurevich, Joel Lanir, Benjamin Cohen, and Ran Stone. "TeleAdvisor: A Versatile Augmented Reality Tool for Remote Assistance." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '12. Austin, Texas, USA: ACM, 2012, pp. 619–622. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2207763. URL: http://doi.acm.org/10.1145/2207676.2207763.

[21] Mary Hegarty and David Waller. "A dissociation between mental rotation and perspective-taking spatial abilities." In: *Intelligence* 32.2 (2004), pp. 175–191.

[22] Peter F Hokayem and Mark W Spong. "Bilateral teleoperation: An historical survey." In: *Automatica* 42.12 (2006), pp. 2035–2057.

[23] Berthold KP Horn. "Closed-form solution of absolute orientation using unit quaternions." In: *JOSA A* 4.4 (1987), pp. 629–642.

[24] Steven Johnson, Madeleine Gibson, and Bilge Mutlu. "Handheld or handsfree?: Remote collaboration via lightweight head-mounted displays and handheld devices." In: *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM. 2015, pp. 1825–1836.

[25] Brennan Jones, Anna Witcraft, Scott Bateman, Carman Neustaedter, and Anthony Tang. "Mechanics of camera work in mobile video collaboration." In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM. 2015, pp. 957–966.

[26] Brigitte Jordan and Austin Henderson. "Interaction analysis: Foundations and practice." In: *The journal of the learning sciences* 4.1 (1995), pp. 39–103.

[27] Takayuki Kanda, Takahiro Miyashita, Taku Osada, Yuji Haikawa, and Hiroshi Ishiguro. "Analysis of humanoid appearances in human–robot interaction." In: *IEEE Transactions on Robotics* 24.3 (2008), pp. 725–735.

[28] David Kirk, Andy Crabtree, and Tom Rodden. "Ways of the hands." In: *ECSCW 2005*. Springer. 2005, pp. 1–21.

[29] David Kirk and Danae Stanton Fraser. "Comparing remote gesture technologies for supporting collaborative physical tasks." In: *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM. 2006, pp. 1191–1200.

[30] David Kirsh and Paul Maglio. "On distinguishing epistemic from pragmatic action." In: *Cognitive science* 18.4 (1994), pp. 513–549.

[31] John Krakauer, M F Ghilardi, and Claude Ghez. "Independent learning of internal models for kinematic and dynamic control of reaching." In: 2 (Dec. 1999), pp. 1026–31.

[32] Johannes Maria Kraus, Florian Nothdurft, Philipp Hock, David Scholz, Wolfgang Minker, and Martin Baumann. "Human After All: Effects of Mere Presence and Social Interaction of a Humanoid Robot as a Co-Driver in Automated Driving." In: *Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications Adjunct*. ACM. 2016, pp. 129–134.

[33] Russell Kruger, Sheelagh Carpendale, Stacey D Scott, and Saul Greenberg. "Roles of orientation in tabletop collaboration: Comprehension, coordination and communication." In: *Computer Supported Cooperative Work (CSCW)* 13.5 (2004), pp. 501–537.

[34] Hideaki Kuzuoka, Toshio Kosuge, and Masatomo Tanaka. "GestureCam: A Video Communication System for Sympathetic Remote Collaboration." In: *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. CSCW '94. Chapel Hill, North Carolina, USA: ACM, 1994, pp. 35–43. ISBN: 0-89791-689-1. DOI: 10.1145/192844.192866. URL: http://doi.acm.org/10.1145/192844.192866.

[35] Hideaki Kuzuoka, Shinya Oyama, Keiichi Yamazaki, Kenji Suzuki, and Mamoru Mitsuishi. "GestureMan: a mobile robot that embodies a remote instructor's actions." In: *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM. 2000, pp. 155–162.

[36] Daniel Leithinger, Sean Follmer, Alex Olwal, and Hiroshi Ishii. "Physical Telepresence: Shape Capture and Display for Embodied, Computer-mediated Remote Collaboration." In: *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. UIST '14. Honolulu, Hawaii, USA: ACM, 2014, pp. 461–470. ISBN: 978-1-4503-3069-5. DOI: 10.1145/2642918.2647377. URL: http://doi.acm.org/10.1145/2642918.2647377.

[37] Juxi Leitner, M Luciw, Alexander Foerster, and J Schmidhuber. "Teleoperation of a 7 DOF Humanoid Robot Arm Using Human Arm Accelerations and EMG Signals." In: June 2014.

[38] Christian Licoppe, Paul K. Luff, Christian Heath, Hideaki Kuzuoka, Naomi Yamashita, and Sylvaine Tuncer. "Showing Objects: Holding and Manipulating Artefacts in Video-mediated Collaborative Settings." In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: ACM, 2017, pp. 5295–5306. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025848. URL: http://doi.acm.org/10.1145/3025453.3025848.

[39] Paul Luff, Christian Heath, Hideaki Kuzuoka, Jon Hindmarsh, Keiichi Yamazaki, and Shinya Oyama. "Fractured ecologies: creating environments for collaboration." In: *Human-Computer Interaction* 18.1 (2003), pp. 51–84.

[40] Guan-Chun Luh. "Intuitive Muscle-Gesture based Robot Navigation Control Using Wearable Gesture Armband." In: July 2015.

[41] Filip Maric, Ivan Jurin, Ivan Markovic, Zoran Kalafatic, and Ivan Petrovic. "Robot arm teleoperation via RGBD sensor palm tracking." In: May 2016, pp. 1093–1098.

[42] Terrance Mok and Lora Oehlberg. "Critiquing Physical Prototypes for a Remote Audience." In: *Proceedings of the 2017 Conference on Designing Interactive Systems*. ACM. 2017, pp. 1295–1307.

[43] João Pedro Morais, Svetlin Georgiev, and Wolfgang Sprößig. "Quaternions and Spatial Rotation." In: *Real Quaternionic Calculus Handbook*. Basel: Springer Basel, 2014, pp. 35–51. ISBN: 978-3-0348-0622-0. DOI: 10.1007/978-3-0348-0622-0_2. URL: https://doi.org/10.1007/978-3-0348-0622-0_2.

[44] Carman Neustaedter, Gina Venolia, Jason Procyk, and Daniel Hawkins. "To Beam or Not to Beam: A Study of Remote Telepresence Attendance at an Academic Conference." In: *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. CSCW '16. San Francisco, California, USA: ACM, 2016, pp. 418–431. ISBN: 978-1-4503-3592-8. DOI: 10.1145/2818048.2819922. URL: http://doi.acm.org/10.1145/2818048.2819922.

[45] Jacki O'Neill, Stefania Castellani, Antonietta Grasso, Frederic Roulland, and Peter Tolmie. "Representations can be good enough." In: *ECSCW 2005*. Springer. 2005, pp. 267–286.

[46] Richard P. Paul. "Robot Manipulator: Mathematics, Programming and Control." In: (Jan. 1981).

[47] Natural Point. "Optitrack." In: *Natural Point, Inc.,[Online]. Available: http://www. naturalpoint. com/optitrack/.[Accessed 22 7 2017]* (2017).

[48] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. "ROS: an open-source Robot Operating System." In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe. 2009, p. 5.

[49] Irene Rae, Bilge Mutlu, and Leila Takayama. "Bodies in motion: mobility, presence, and task awareness in telepresence." In: *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM. 2014, pp. 2153–2162.

[50] Irene Rae, Leila Takayama, and Bilge Mutlu. "In-body Experiences: Embodiment, Control, and Trust in Robot-mediated Communication." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. Paris, France: ACM, 2013, pp. 1921–1930. ISBN: 978-1-4503-1899-0. DOI: 10.1145/

2470654.2466253. URL: http://doi.acm.org/10.1145/2470654.2466253.

[51] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. "A Motion Retargeting Method for Effective Mimicry-based Teleoperation of Robot Arms." In: *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. HRI '17. Vienna, Austria: ACM, 2017, pp. 361–370. ISBN: 978-1-4503-4336-7. DOI: 10.1145/2909824.3020254. URL: http://doi.acm.org/10.1145/2909824.3020254.

[52] Kyle B. Reed and Michael A. Peshkin. "Physical Collaboration of Human-Human and Human-Robot Teams." In: *EEE Trans. Haptics* 1.2 (July 2008), pp. 108–120. ISSN: 1939-1412. DOI: 10.1109/TOH.2008.13. URL: http://dx.doi.org/10.1109/TOH.2008.13.

[53] Ruben Smits, H Bruyninckx, and E Aertbeliën. "Kdl: Kinematics and dynamics library." In: *Avaliable: http://www. orocos. org/kdl* (2011).

[54] Anthony Tang, Michael Boyle, and Saul Greenberg. "Understanding and mitigating display and presence disparity in mixed presence groupware." In: *Journal of Research and Practice in Information Technology* 37.2 (2005), pp. 193–210.

[55] Anthony Tang, Michel Pahud, Kori Inkpen, Hrvoje Benko, John C Tang, and Bill Buxton. "Three's company: understanding communication channels in three-way distributed collaboration." In: *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. ACM. 2010, pp. 271–280.

[56] John C Tang. "Findings from observational studies of collaborative work." In: *International Journal of Man-machine studies* 34.2 (1991), pp. 143–160.

[57] John C Tang and Scott L Minneman. "VideoDraw: a video interface for collaborative drawing." In: *ACM Transactions on Information Systems (TOIS)* 9.2 (1991), pp. 170–184.

[58] John C Tang and Scott Minneman. "VideoWhiteboard: video shadows to support remote collaboration." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 1991, pp. 315–322.

[59] Peter Turpel, Bing Xia, Xinyi Ge, Shuda Mo, and Steve Vozar. "Balance-arm tablet computer stand for robotic camera control." In: *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*. IEEE Press. 2013, pp. 241–242.

[60] Jeffrey M Zacks, Jon Mires, Barbara Tversky, and Eliot Hazeltine. "Mental spatial transformations of objects and perspective." In: *Spatial Cognition and Computation* 2.4 (2000), pp. 315–332.

[61] Dingyun Zhu, Tom Gedeon, and Ken Taylor. "Exploring Camera Viewpoint Control Models for a Multi-tasking Setting in Teleoperation." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. Vancouver, BC, Canada: ACM, 2011, pp. 53–62. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1978952. URL: http://doi.acm.org/10.1145/1978942.1978952.

# DECLARATION

Ich versichere, dass ich die vorliegende Arbeit (bei einer Gruppenarbeit: den entsprechend gekennzeichneten Anteil der Arbeit) selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note „nicht ausreichend"zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

*University of Calgary, Alberta (Canada), April 2017 - September 2017*

Martin Feick

## COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

*Final Version* as of January 23, 2018 (`classicthesis`).