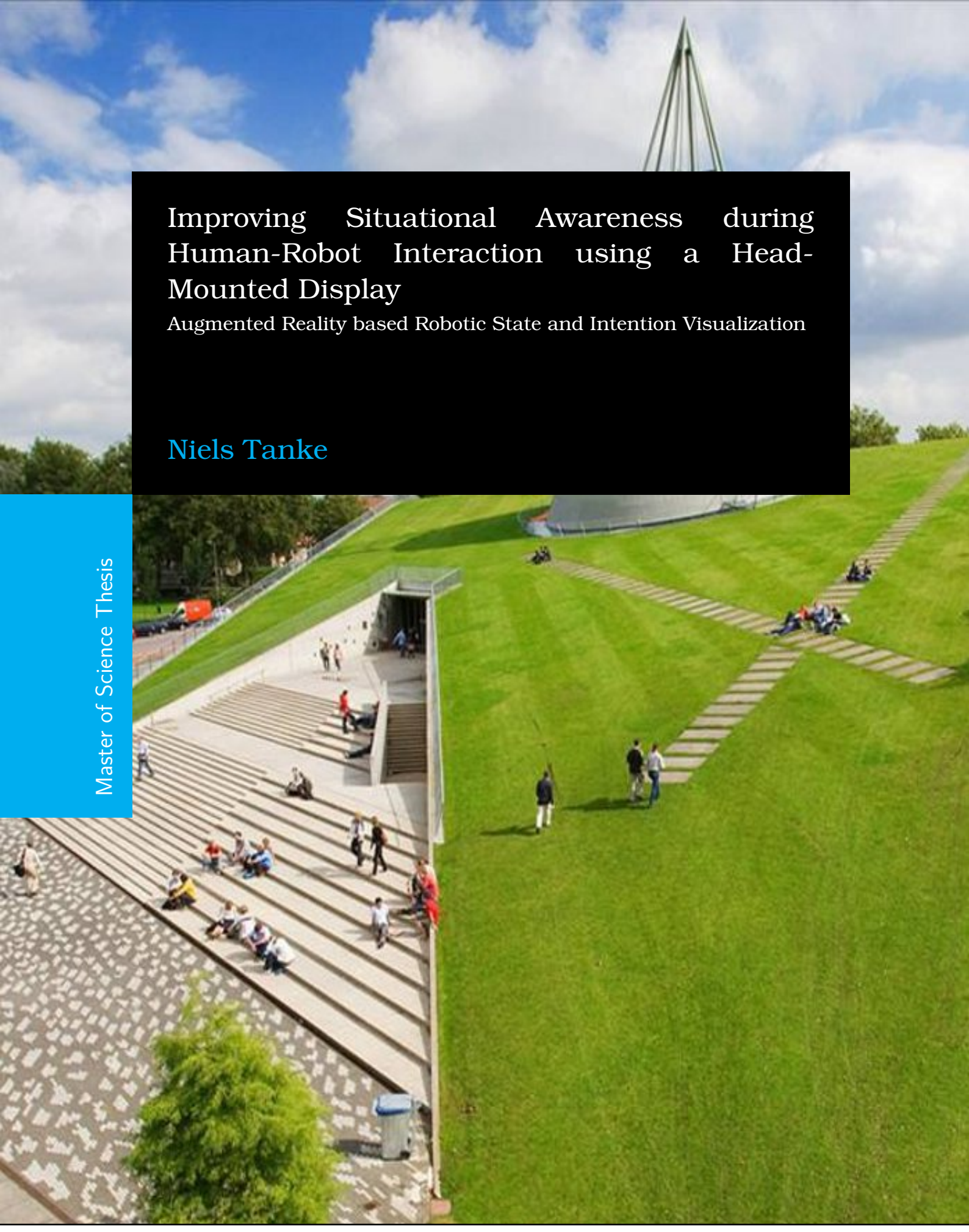


Improving Situational Awareness during Human-Robot Interaction using a Head-Mounted Display

Augmented Reality based Robotic State and Intention Visualization

Niels Tanke

Master of Science Thesis

TU Delft Delft University of Technology

Biomechanical Engineering (BmechE)

Improving Situational Awareness during Human-Robot Interaction using a Head-Mounted Display

Augmented Reality based Robotic State and Intention Visualization

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering at Delft University of Technology

Niels Tanke

September 2, 2013

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

A novel approach was proposed to improve situational awareness using a head-mounted display when working together with an industrial robot arm. With the introduction of affordable intrinsically safe industrial robot arms, research into human-robot interaction has increased. When robot arms and humans are working together, it is not only essential that the robot arm is aware of humans, but also that the human knows what the robot arm is doing and intending to do. Situational awareness is value representation for the comprehension of a situation used to infer what is going on and what is going to happen. The situational awareness of a human was improved by showing extra virtual information using augmented reality. The virtual information was shown using a virtual model of the robot arm on a head-mounted display. Two categories of information to improve situational awareness were found by analysing the human-robot interaction in orange repacking: state and intention information. From each category, one information type with a high potential to improve situational was implemented and explored. Two within-subject experiments were conducted in which the test subjects were asked to wear a head-mounted display and work in a designed robot working cell. During the first experiment the test subject were presented with visual cues about the state of the robot arm and during the second experiment with visual cues about the intended goal of the robot arm. In each experiment the test subjects were evaluated on the reaction time to the visual cues. Each experiment was done twice, once with the presence of virtual information and once without, but both while wearing the head-mounted display. Compared to the case without virtual information, the results showed a decrease in reaction times to state and intention information when presented with the virtual information. This decrease in reaction time was a measure for an increased situational awareness of the human and showed that situational awareness could be improved by using a head-mounted display to show extra, not observable, information about the robot arm. This work contributes to help humans work more efficiently together with industrial robot arms by using a head-mounted display to present information in a more natural manor.

Table of Contents

Glossary

List of Acronyms

3D	Three Dimensions
AR	Augmented Reality
AV	Augmented Virtuality
BGR	Blue-Green-Red
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
CRC	Cyclic Redundancy Check
DOF	Degree Of Freedom
ECG	Electrocardiography
EMG	Electromyography
EST	Expansive Space Trees
FPS	Frames Per Second
FOV	Field of View
GUI	Graphical User Interface
HMD	Head-Mounted Display
HRI	Human-Robot Interaction
HSV	Hue-Saturation-Value
KPIECE	Kinodynamic Motion Planning by Interior-Exterior Cell Exploration
LAN	Local Area Network
LED	Light-Emitting Diode

MP	Megapixel
MR	Mixed Reality
OMPL	Open Motion Planning Library
OpenCV	Open Computer Vision
OpenRAVE	Open Robotics Automation Virtual Environment
OST	Optical See-Through
PRM	Probabilistic Roadmap
RGBA	Red-Blue-Green-Alpha
ROS	Robotic Operating System
RPP	Robust Planar Pose
RRT	Rapidly-exploring Random Trees
RSD	Retinal Scanning Display
SA	Situational Awareness
SBPL	Search Based Planner Library
SC	Skin Conductance
SME	Small and Medium-sized Enterprise
STOMP	Stochastic Trajectory Optimization for Motion Planning
ToM	Theory of Mind
URDF	Unified Robot Description Format
USB	Universal Serial Bus
UTP	Unshielded Twisted Pair
V4L	Video4Linux
VST	Video See-Through

Acknowledgements

I would like to thank the following people for their help and contributions to make this work possible.

Martijn Wisse, Jouke Verlinden and Rob Prevel for helping me as daily supervisors, for their time spend discussing my work and for their guidance, comments and insights. I also would like to thank Pieter Jonker for heading the examination committee and supervising the graduation process and Kanter van Deurzen for participating as external committee member.

Thanks to Wouter Caarls' help and patience, I was able to solve my programming problems quickly and thanks to Floris Gausser and Machiel Bruinink I was able to solve the orange detection problem.

The experiments could not be done without test subjects, I would like to thank all of them of participating in the experiments.

Thanks to Brando Maathuis, Bart de Vette, Rob Vellekoop, Anne van Weije, Martijn Wisse, Pieter Jonker, Wouter Caarls, Rob Prevel and Kanter van Deurzen for proofreading my thesis.

Last but not least, I would like to thank my parents for supporting me during my time at Delft University of Technology.

Delft, University of Technology — Niels Tanke
September 2, 2013

Chapter 1

Introduction

1-1 Motivation

When thinking of robot arms for industry, the image is often of big, powerful, fast moving and dangerous robot arms. Because of these characteristics, safety measures are required to prevent injuries to humans [1]. The most common of these safety measures is a safety fence, a barrier between the robot arm and humans. This barrier prevents the robot arm from injuring people, but inhibits human-robot interaction, because when the safety fence is opened, the robot needs to stop to prevent hurting humans. Recent robot arms are made intrinsically safe and do not require the use of a safety fence as a barrier between humans and the robot arm. These arms have additional safety features such as a maximum dynamic power $\leq 80W$, or a maximum static force $\leq 150N$, that enable them to work around humans, without safety fences and be physically safe [2]. With these intrinsically safe robot arms, human-robot interaction is a possibility so that robots and humans can work together in the same physical space on the same project [3, 4]. This human-robot interaction allows production processes to be designed more efficiently, by playing to the specific strengths of both humans and robots. Robots are better at doing repetitive and heavy tasks all day long, while humans are able to adapt very quickly to changing surroundings and are better at doing complex and dexterous tasks. An industrial sector that has active research on the introduction of intrinsically safe robot arms is the fruit and vegetables repacking industry. Robot arms are used to do the easy and repetitive tasks, while the humans do the more complex and dexterous tasks. By letting humans and robots work together tasks can be done more efficiently, however to be able to work efficiently together an adequate Situational Awareness (SA) is needed by the operator [5].

SA is an important factor in human dynamic decision making and task performance [5]. When working together, robots and humans need to be aware of each other, their state (what is happening now) and intentions (what is happening next). A lot of research effort is being put into making robots aware of, and react to humans [6, 7, 8]. Sensors are added to the robot to detect humans that are in the working space of the robot or verbal communication and

non-verbal instructions are used to inform the robot [9]. However to have efficient human-robot interaction it is also important that the human knows what the robot is 'thinking', doing and intending to do [10]. This is needed so the human will feel more at ease, safe, less stressed [5, 11, 12, 13] and also to improve task execution performance [5, 14, 15]. Compared to the research being done to help robots understand humans, the research done to help humans understand robots is neglectable.

Seen from a human point of view, a robot arm is a black box, so the ability of the human to figure out what the robot is doing and intending to do is impaired. The ability to build an internal representation of the environment is very important to the human to figure out what is going on and what is going to happen [5]. To compensate the lack of this SA, various solutions were developed. Improving SA is dependent on the way the environment is able to provide information and what kind of information is provided. The provided information also needs to be as compatible as possible with human information processing abilities [5]. In this context information was defined as various cues to improve observations and make them clearer. To improve the SA of the human when working with an industrial robot arm it is important to show the information from the robot to the human co-worker in a clear and unambiguous way. What the robot is currently doing, the current state, is most obvious to show, because the robot itself can give a lot of cues, but the intended state, like actions done in the future or motions, are more difficult to show.

Various solutions are found in literature to show current state and intentions of a robot to the human. Social robots use speech, facial expressions and body language to communicate their state and intentions [16, 17]. Natural looking motions of an industrial robot arm are used to indicate intentions [13] and repeated motions of a mobile robot are used to indicate that the robot needs help from the human [10, 18, 19]. Visual approaches are also used, for example a 3D model of a robot arm, shown on monitor, to indicate the intended motion [11, 12] or a projector on a mobile robot to project the intended path on the ground [14, 15].

For this thesis a novel approach was chosen in the form of a Head-Mounted Display (HMD) combined with Augmented Reality (AR), this display was used to display state and intention information from the robot to the human. AR is used to facilitate communication between humans and robots and shows promise [20]. A HMD is a display worn on the head that displays visual information directly in front of the wearer's eyes. Recent advances in HMD technology have made them suitable for real-time use, because the image resolution and computation speed of computers have increased. Recent studies showed various advantages of using HMDs over traditional visualization techniques, such as lights or displays. Using a HMD, in combination with AR, reduces head-eye movement and attention switching, which increases eye-on-work time and overall performance [21].

To test if showing state and intentions information from an industrial robot using a HMD is beneficial for the SA of the human, an experiment was designed. This experiment was designed as a mock-up of a real world scenario and focusses on the handling and repacking of oranges. A mock-up scenario was used, instead of an abstract scenario, to improve fidelity for the test subjects [22]. The robot and human are working together in the same physical space to accomplish a shared task. The robot arm is working continuously, but requires human help and intervention to function better. The goal of the robot arm is to substitute human labor: make the process cheaper and faster. This setup was chosen because of knowledge and focus of recent proposals from the university, to bring robotics to the Small and Medium-sized

Enterprise (SME) sector, in particular to the repacking of fruit and vegetables sector.

1-2 Research question

To test if Situational Awareness (SA) can be improved by using a Head-Mounted Display (HMD) and Augmented Reality (AR) in an industrial setting, it is necessary to test if it is a good solution to show information from the robot to the human. The research question is:

Does showing state and intention information of an industrial robot arm, using a head-mounted display, help the human co-worker to improve situational awareness?

Based on a preliminary analysis of the task interaction (as described in Chapter 3) the following hypothesis were formulated:

- Visualizing the internal state of the robot arm using a head-mounted display reduces the reaction time of the human co-worker when the robot arm changes internal state.
- Visualizing the intended goal position of the robot arm using a head-mounted display reduces the reaction time of the human co-worker when reacting to a specific destination position of the robot arm.

1-3 Thesis organization

To answer the research question the following approach was taken:

1. HMD design: A Head-Mounted Display (HMD) was programmed to allow the showing of virtual information from an industrial robot arm using Augmented Reality (AR). Section 2-3 introduces the important factors involved with HMDs and Appendix A gives a detailed overview of the implementation, both hardware and software.

2. Analysis: The interaction between humans and robots when handling oranges was analysed. Chapter 3 gives an overview of the important parameters of Human-Robot Interaction (HRI) and Situational Awareness (SA) when working with a robot arm to handle oranges. Section 3-1 describes the pilot study that was done to discover the important parameters of HRI. Section 3-2 gives an overview of the information from the robot that is useful for humans and Section 2-1-2 describes the metrics used to measure SA.

3. Experiment design: An experimental robot working cell was designed to allow HRI and the use of a HMD to display virtual information from the robot to the human. Chapter 4 describes the design of the robot working cell (Section 4-1) and the design of the virtual information shown on the HMD (Section 4-4). Appendix B gives a more detailed overview of the robotic working cell, both hardware and software.

4. Experiment: An experiment was done to test if the virtual information shown on the HMD improves SA, Chapter 5 describes the experiment.

5. Evaluation: The experiment was evaluated to identify the important parameters associated with visually presenting state and intention information and human-robot interaction to draw conclusion from the results. Chapter 7 discusses the results of the experiments and Chapter 8 concludes this thesis.

Chapter 2

Background

To get a better understanding of the approach, solutions and choices made in this thesis, some background knowledge and information is needed. This chapter explains important background information regarding Situational Awareness (SA), Head-Mounted Display (HMD), Augmented Reality (AR) and prior research done into making humans aware of robots.

Section 2-1 describes the concepts of SA and explains the importance of the concept with respect to human decision making and describes how to measure it. Section 2-2 gives an overview of prior research done with respect to SA when working with robots. Section 2-3 describes what AR is, what the important challenges are and how they were solved in this thesis. A more detailed explanation of the solution with respect to the HMD hardware and AR software is given in Appendix A.

2-1 Situational awareness

Kobayashi and Yamada used ToM as the basis for their research into Human-Robot Interaction (HRI) [18]. They argued that ToM is an important factor for humans to be able to work efficiently together with a robot. ToM is a theory used in psychology and neuroscience and is the ability to assign mental states (beliefs, intentions, desires, knowledge, etc.) to oneself and to others [23, 24, 25]. ToM is built by using social interaction, observations of behavior, actions and emotions and by the use of natural language. Having a ToM allows a person to attribute thoughts, desires, and intentions to others. With these attributes a person is able to predict or explain actions of others and to infer their intentions [26].

A lot of concepts of ToM are not applicable to industrial robots, but are more applicable to social robots [16], because they are specific to social interaction. That is why ToM is not the proper basis to view state and intention sharing in HRI for industrial settings. A concept that is a better basis for state and intention sharing in HRI is Situational Awareness (SA), because it describes HRI from a broader perspective, a situation [27]. Endsley defines SA as: "the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future" [5]. A

simpler definition is given by Adam: "knowing what is going on so you can figure out what to do" [28]. This theory is about building an internal representation of the environment and use to make decisions.

2-1-1 Situational awareness model

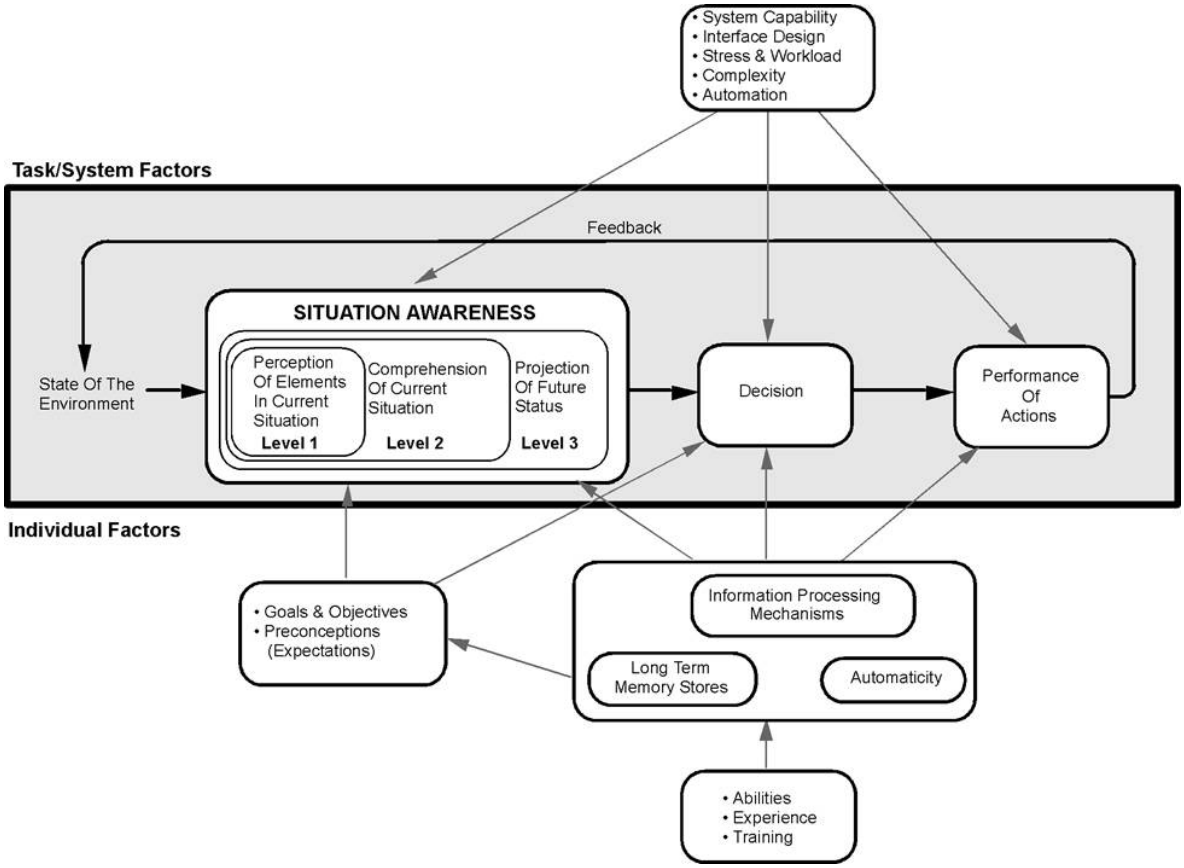


Figure 2-1: Model of Situational Awareness (SA) in human dynamic decision making [5]. At the heart of the dynamic decision making is the SA of a person. SA is divided into three levels, each levels corresponds to a more complete internal representation of the environment. Together with the goals, objective, skill, memory, information processing ability and expectations, a decision is made. This decision effects the environment and the observations from this environment are fed back to update the SA. Figure was extracted from Wikipedia [29].

Human decision making is a complex and an active area of research. This section explains the importance of SA with respect to human decision making by using the SA model as defined by Endsley (Figure 2-1). In this model all important factors involved with human decision making are modeled. As can be seen in Figure 2-1 the heart of the model is SA, which has three levels. From the SA, goals, objectives and expectations, decisions are made. These decisions effect the environments and the observation of this environment is fed back to make new decisions about the altered environment. Other important factors effecting decision making are: skill, prior training, memory and observation and information processing ability. The three SA levels correspond to a different level of completeness of the internal representation

of the environmental. The completeness of this representation effects decision making: when a person has a more complete internal representation of the environment, better decision can be made.

Level 1 SA: Perception

The first step in gaining SA is to perceive the status, attributes and dynamics of relevant parts of the environment. Examples are size, location, color and speed of objects. It is important to have a complete as possible representation of the environment. What kind of attributes are needed to understand the environment is dependent on the function of the human and on the task performed.

Level 2 SA: Comprehension

The next step is to use the perceived attributes of level 1 and not only be aware of them, but try to understand their relationship and significance to the environment and task. The ability to comprehend the perceived attributes of level 1 is related to the experience of the human. A less well trained or less knowledgeable human might be able to get to the same level of perception as a more trained or knowledgeable human, but fails to reach the same level of comprehension. With a lower comprehension, the build-up of the internal representation of the situation is impaired.

Level 3 SA: Projection

When good comprehension of the environment is achieved, predictions about the future states can be made. The ability to predict the future states requires good knowledge about the status and dynamics (level 1 and 2 SA) of the environment and allows for better decision making. The better you can predict future state, the better you can adapt your actions to meet your goal.

2-1-2 Measuring situation awareness of human-robot interaction

This section describes what methods and metrics were used to measure an improvement of SA. Measuring SA in HRI with an industrial robot arm is not a trivial task and in literature various methods are proposed. This is because SA is very specific to a situation. To come to a good method with proper metrics, first an overview of the most common evaluation methods is given. With this overview a suitable method with proper metrics was chosen to measure SA in HRI with an industrial robot arm used for handling oranges.

Research into HRI and SA has been growing in the recent years, not only in the design of robot systems, but also in the evaluation thereof. SA is a very complex field, that is why various evaluation methods have been developed and used, all with their advantages and disadvantages. According to Bethel and Murphy [22] and Endsley and Garland [30] there are four methods to evaluate SA: self-assessment, interview, behavioral measures and performance measures. These methods are the intersection between the methods proposed by both research teams, because the methods rely on the same principle.

Self-assessment: self-assessments are subjective methods and are based on the personal assessment of the test subject. Examples of self-assessment are: psychometric scales, questionnaires and surveys. The problem with these methods is that there is a bias, because the evaluation is done after the experiment, therefore the test subjects have had time to process what they have done and reason about it.

Interviews: an interview is also subjective method and is normally done by interviewer asking questions. This method is also prone to a bias, because the interviewer can influence the answers of the test subjects. The test subjects are can answer questions based on what the interviewer wants to hear instead of what the test subjects actually experienced.

Behavioral measures: behavioral measures are objective methods and are gathered by observing the test subjects during the actual interaction with the robot. These methods depend on the interpretation of the observer to map the behavioral responses of the test subject to the SA of the test subject.

Performance metrics: performance measures determine how well a person performs a task, reacts to an event or the amount of errors made. These measures give an objective measurement that don't rely on the interpretation of observation, but assume a relation between SA and performance. An advantage of performance metrics is that they can be compared between persons.

To answer the research question it is sufficient to only look at performance metrics. Performance metrics were chosen, because they are objective and can be compared straightforward between people. Which performance metrics were used is further discussed in Section 3-3.

2-1-3 Conclusion

As can be seen there is a difference between the usefulness of level 1 SA, level 2 SA and level 3 SA in human decision making. For good decision making and thus good human performance it is important that the human's level of SA is as high as possible, but gaining a better SA requires time and training. By helping the human in gaining a better comprehension and building of an internal representation of the situation, less time has to be spent trying to learn a the important parameters and dynamics of the situation. An improved understanding of the environment results in better and faster decision making. In this thesis specific information about the state and intention of the robot arm, which help improve comprehension (level 2 SA) and projection (level 3 SA), are shown on a HMD. An improvement in SA was measured by performance metrics and further discussed in Section 3-3.

2-2 Prior research done to improve situational awereness

This section gives an overview of the literature and prior research done to improve SA in human-robot interaction and describes why a visual based method was used in this thesis. Section 2-2-4 gives a conclusion of the literature discussed in this chapter.

In literature various solutions are presented that improve SA. The provided solutions, in the form of various cues presented to the human, are used to show what a robot is 'thinking', doing and intending to do. This was done to help the human gain better comprehension of the situation (level 2 SA and level 3 SA). There are three categories of cues defined using two major senses, seeing and hearing [7]:

Motion cues: motions of the robot were used to indicate state and intention of the robot. Section 2-2-1 gives an overview of the literature that used motion cues.

Auditory and natural language cues: audio cues from a robot were used to inform a human about the state and intention of the robot. Section 2-2-2 gives an overview of the literature that used auditory cues.

Visual cues: state and intention of the robot was also encoded into visual cues. Section 2-2-3 describes the literature that used visual cues.

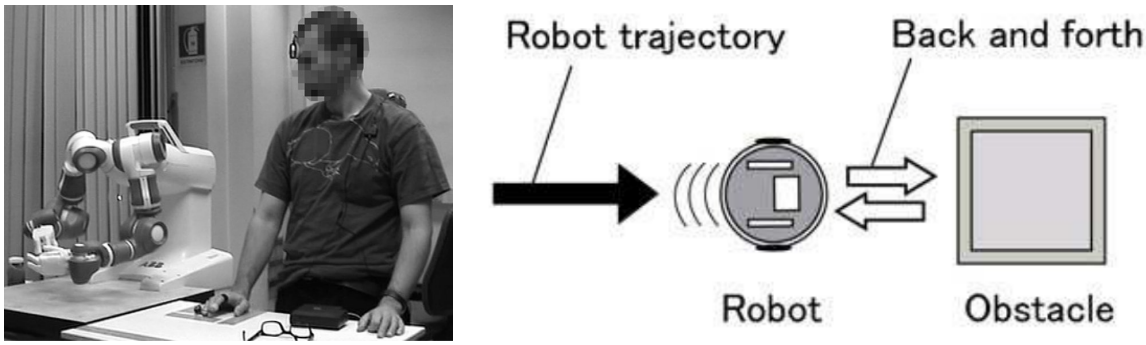
2-2-1 Motion cues

Zanchettin et al. [13] researched what the effect of natural looking motions of an industrial robot arm is on the psychophysical response of the human. A two arm setup (Figure 2-2(a)) was used to do an assembly task together with a human. They measured psychophysical responses in the form of heart rate, Electrocardiography (ECG) signals, Skin Conductance (SC) and Electromyography (EMG) signals and concluded that more natural looking motions of the robot arm resulted in less emotional arousal, less robot-induced stress, more ergonomic interaction and better prediction of intentions.

These natural looking motions are better adapted to human information perception and processing and allow humans to better infer intentions. When viewed from the eyes of the SA model, this allows a speed up of level 2 and 3 understanding of the environment.

Kobayashi and Yamada [18, 19, 10] used motions of their mobile robot to express its internal state and intentions which they called *expressing the robot's mind*. They used a back and forth going repeated motion to indicate that the robot needed help from a human (Figure 2-2(b)), the robot signalled the human when an obstacle needed to be removed. They compared the back and forth going motion cue with a light cue (Light-Emitting Diode (LED)) and a sound cue (buzzer). The conclusion was that a non-verbal cue in the form of motion was used to allow more variation in the cues and induce less stress by the human.

This research uses back and forth going repeated motion of a mobile robot to make the intentions of the robot clear to the human. By using the motion cues the human has a better SA compared to no motion cues, because level 2 and 3 information was given.



(a) Experimental setup used by Zanchetting et al.

(b) Experimental setup used by Kobayashi and Yamada.

Figure 2-2: Figure 2-2(a) shows the experimental setup used by Zanchetting et al. [13] in their experiment to research the psychophysical effects of natural looking motions on the human. Figure 2-2(b) shows the experimental setup used by Kobayashi and Yamada [18, 19, 10] to research the effect of motion cues by the robot to indicate that it needs help from the human.

2-2-2 Auditory and natural language cues

Fong et al. used natural language to communicate between humans and robots [31, 32] to communicate that the robot needed help from the human. They performed an experiment where an autonomous mobile robot had to make a decision and asked the human for help (Figure 2-3). This way the human knew what was going on, because the robot informed the human what decisions it was making.

By informing the human about its decisions, the human doesn't have to infer from observations what the robot is doing and going to do and allows the human to gain better SA.

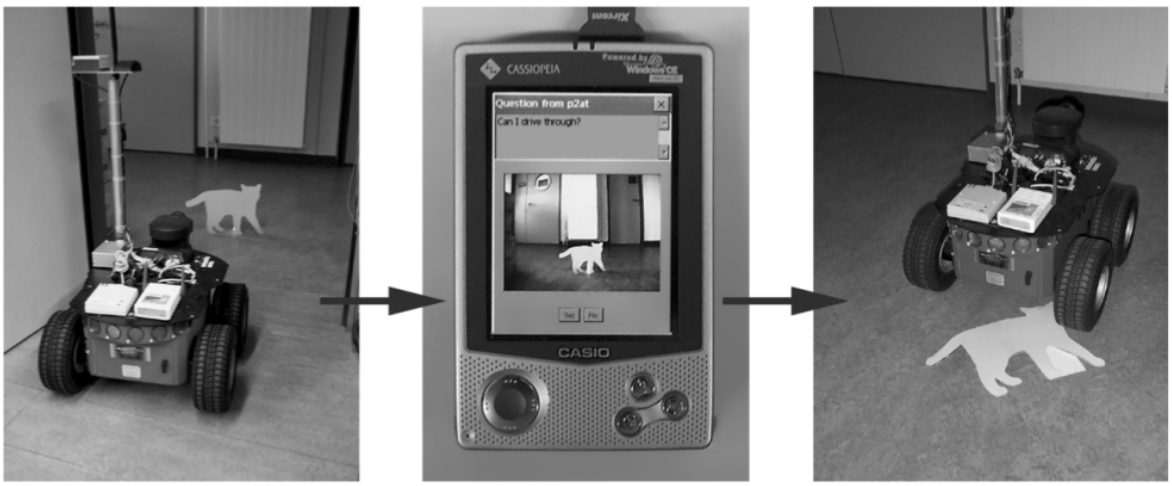


Figure 2-3: Teleoperation experiment done by Fong et al., where the robot had to make a decision and asked the human for help [31, 32].

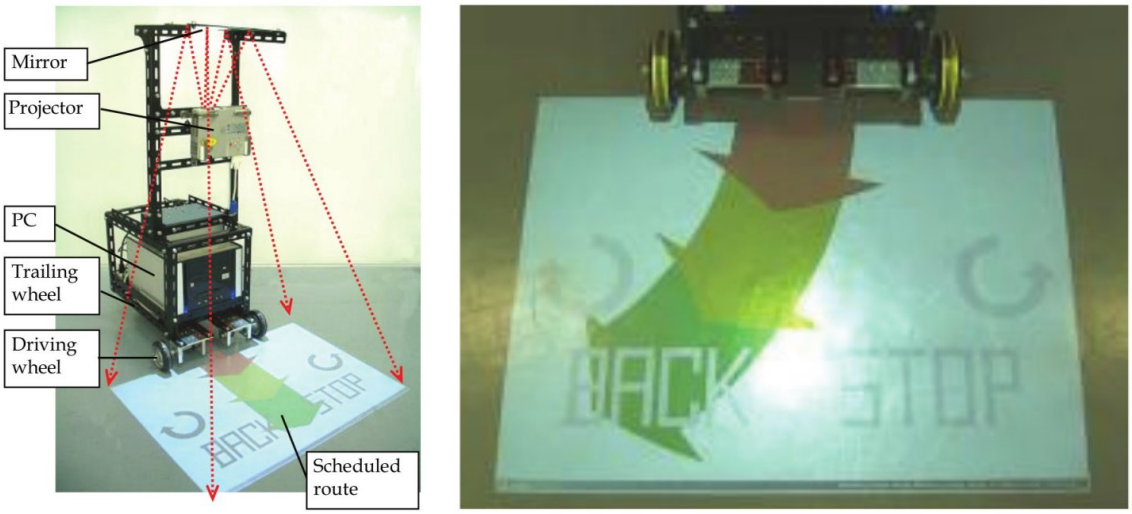
Auditory cues are mostly used in social robots in the form of speech. Speech allows the robot to communicate to the human in a way that is adapted to the information observation and

processing mechanisms often used by humans.

2-2-3 Visual cues

Matsumaru et al. used a projector and a mobile robot as a basis to implement their system to help humans gain SA [14, 33, 34, 35, 36]. They named it *preliminary announcement* and projected the state and intentions of their mobile robot onto the floor so it can be seen directly by humans (Figure 2-4). Matsumaru et al. researched the types of information that needed to be displayed and the way to display this information. In their research they compared the use of a projector with the use of a blowout, lamp and light ray to indicate the intended path of which the projector was best. They argued that two types of information were important to display: the speed and the direction of their mobile robot. The experimental results gathered from the *passing each other* experiment (Figure 2-5(a)) concluded that preliminary-announcement of the state and intention of the robot had a positive effect on the way humans judged the actions of the robot [15].

By projecting the state and intentions humans can directly observe the what the robot is intending to do, build a better model of the environment, improve SA and are thus better judge what the robot is doing and going to do.



(a) Mobile robot with projector to display intended path used by Matsumaru et al.

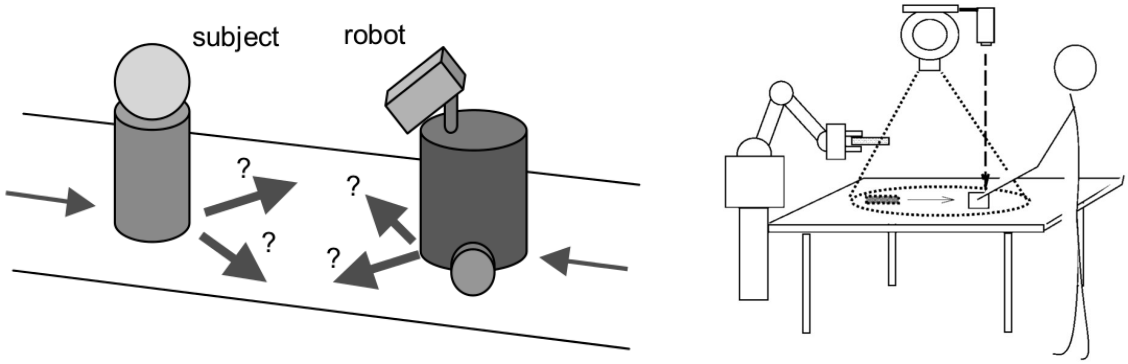
(b) Close-up of the projection of the intended path and actions of the mobile robot.

Figure 2-4: The solution by Matsumaru et al. to inform the intended motion of a mobile robot to a human [14]. Figure 2-4(a) shows the total mobile robot with projector. Figure 2-4(b) shows a close-up of the projection on the ground. The projection indicates the intended motion, speed and actions of their mobile robot.

Wakita et al. used a stationary robot arm to work together with a human [11, 12, 37]. They researched information sharing between human and robot by enabling the robot to show its intentions and requests to the human which they called *knowledge projection*. This knowledge projection was done by displaying a 3D model of the robot arm on a monitor and projecting the intended motion on that monitor. They also projected the hand position of the human

on the workspace using a projector, so that the human knows he was being observed and recognized. Figure 2-5(b) shows a schematic overview of the knowledge projection during the cooperation task between human and robot done by Wakita et al.

By showing the intended motion and the observation of the hand the human gets better insights in the mind of the robot, environment and task. This insight leads to better SA.



(a) *Passing each other* experiment done by Matsumaru et al.

(b) Knowledge projection experiment done by Wakita et al.

Figure 2-5: Figure 2-5(a) shows the *passing each other* experiment done by Matsumaru et al. [15]. The mobile robot projects its intended motion on the ground, this way the human is better able to judge where the robot is going. Figure 2-5(b) shows the knowledge projection experiment using a projector the display the intended motion of the robot arm and the position of the human's hand [12]

2-2-4 Conclusion

All solutions presented in Section 2-2 are used to improve SA of humans and all yielded positive results. But to be used in an industrial setting, some of the presented solutions would not work. When working with an industrial robot in an industrial environment, auditory cues can be confusing, hard to hear, stressing and attract attention away from the task of the humans [10]. Also can be confusing to encode the intentions of a robot using audio cues. Speech is more flexible, but again could be hard to hear and depends on language [18]. Motion cues could be used to display the state of a robot, but the intentions are harder to encode, because the movement of an industrial robot is too limiting and may interfere with the execution of the current task. Visual cues don't have these limitations and provide a good opportunity to show state and intention from the robot to the human [38], therefore in this thesis the use of visual cues will be explored further.

2-3 Head-mounted display

To show the visual cues, a display was needed. In this thesis a Head-Mounted Display (HMD) was used to visually present the state and intentions of an industrial robot arm. Section 2-3-1 describes Augmented Reality (AR), the technique used to visualize the state and intentions of the robot on the HMD. Section 2-3-2 describes the choice for a Video See-Through (VST) HMD and Section 2-3-3 discusses the choice of a marker based head pose tracker to enable AR.

2-3-1 Augmented reality

AR is a technique that allows virtual information to be shown in the real world and in this thesis it was used to visually present virtual information from the robot to the human. A lot of research was done on AR in combination with an industrial environment, for example to show virtual information in a HMD to help humans perform an assembly task better [21]. AR was also used in literature to display information in HMD to help humans during the programming of an industrial robot arm [39]. AR has been used in combination with unmanned aerial vehicles to improve SA [40]. A HMD in combination with AR has never been used to display the state and intentions of an industrial robot arm.

Augmented reality is a research area that focuses on enhancing reality by adding virtual objects to it. Augmented reality is part of the reality-virtuality continuum (Figure 2-6) as defined by Milgram and Kishino [41]. This model consists of four general areas: real environment, Augmented Reality (AR), Augmented Virtuality (AV) and virtual environments. AR and AV combined are defined as Mixed Reality (MR).

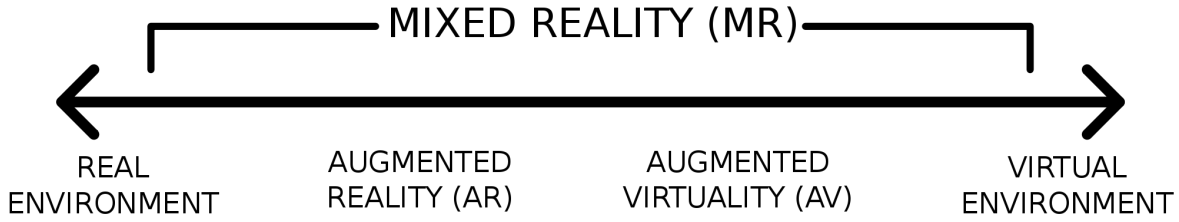


Figure 2-6: Reality-Virtuality continuum [42] as defined by Malgram and Kishino [41]. This model ranges from the real world to a full virtual environment, where Augmented Reality (AR) and Augmented Virtuality (AV) are a mixture of virtual information and the real world.

In the reality-virtuality continuum, the real environment is the non altered physical world and the virtual environment is the complete virtualization of the real world or an imaginary environment. AR is closer to the real world and AV is closer to the virtual world. With AR virtual objects are added to the real world and with AV real objects are added to the virtual environment. An AR system is defined by Krevelen and Poelman [43] which is a slightly altered version of the definition by Azuma et al. [44, 45]. An AR system:

- Combines real and virtual objects in a real environment.
- Registers (aligns) real and virtual objects with each other.
- Runs interactively, in Three Dimensions (3D) and in real time.

Conclusion

To use AR to visually present virtual information to a human when working with an industrial robot arm, an AR system had to be designed. When looking at the definition of AR it is clear that some problems need to be solved before AR can be used:

- How to combine and display the real and virtual objects (Section 2-3-2).
- How to align the real and virtual objects with each other (Section 2-3-3).
- How to run the system and display virtual information in 3D and in real time (Appendix A-2).

2-3-2 Video see-through head-mounted display

To combine and display real and virtual objects together, a display was needed. In a literature study, done prior to this thesis, an overview of all displays used for AR was given [46]. The displays were categorized into Head-Mounted Display (HMD), spatial displays and hand-held displays. All displays were judged against criteria specific to an industrial setting: hands free operation, mobility, investment costs, viewability by multiple users, safety, ease of deployment and the ability to display virtual information. The results of the literature study are shown in Table 2-1.

	HMD			**Spatial display**			
Criteria	**VST**	**OST**	**RSD**	**Screens**	**Projective**	**Holographic**	**Hand-held**
Hands free	++	++	++	++	++	++	-
Mobility	++	++	++	- -	- -	- -	++
Investment costs	+	0	0	0	+	- -	++
Multiple users	- -	- -	- -	++	++	- -	0
Ease of use	0	+	++	++	++	++	+
Safety	0	+	+	++	++	++	+
Ease of deployment	++	+	+	0	0	-	+
Virtual information	++	++	++	0	0	- -	++

Table 2-1: Evaluation of Augmented Reality (AR) displays. The category Head-Mounted Display (HMD) consists of: Video See-Through (VST), Optical See-Through (OST) and Retinal Scanning Display (RSD). The spatial displays are: projective displays, screen based displays and holographic displays. The last category is the hand-held displays like a mobile phone or tablet. All displays were judged based on the criteria specific to an industrial setting and rated on a scale ranging from '- -' (very negative) to '++' (very positive), with '0' being neutral.

Conclusion

The conclusion was that to combine and display real and virtual objects, an Optical See-Through (OST) HMD was considered best to use in an industrial setting, but that a Video See-Through (VST) HMD could be used as an alternative in a research setting, because they are cheaper and easier to program. Figure 2-7 shows a diagram of how a VST HMD works. In this thesis a VST HMD was used as a display and is shown in Figure 2-8. Appendix A-1 gives a detailed description of the hardware and software of the HMD.

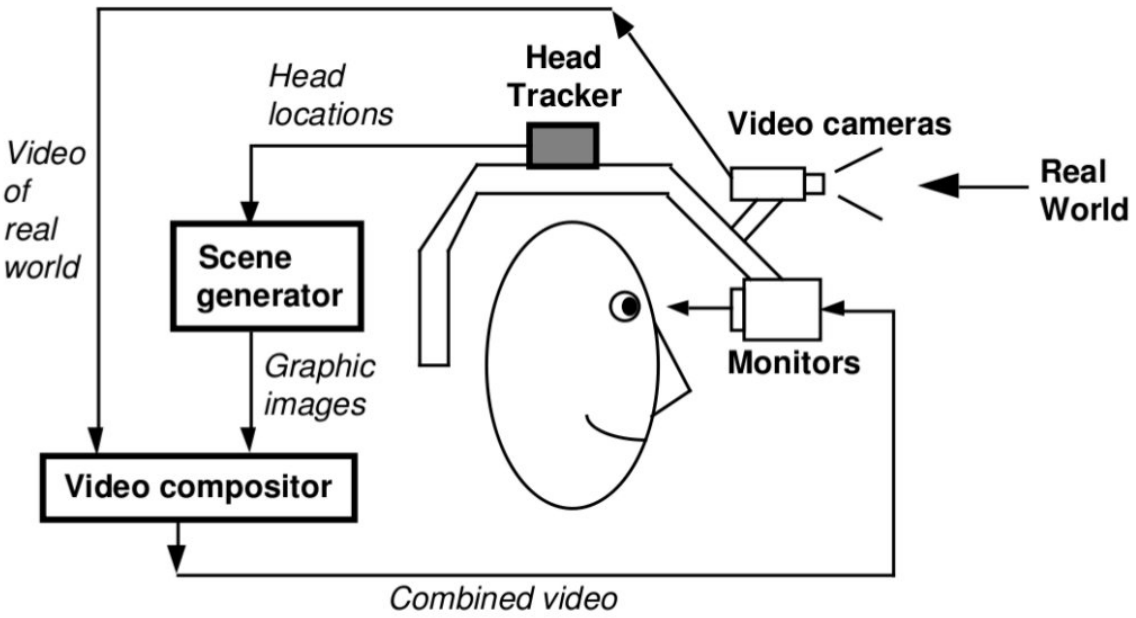


Figure 2-7: A diagram of how a Video See-Through (VST) HMD works [44]. Cameras capture images of the real world. These images are processed by a computer. A tracker determines what the user is looking at, so the correct virtual information can be rendered on the correct position. The computer then combines the real and virtual data and displays the image on the screens in front of the user's eyes.

(a) Head-mounted display nicknamed *Marty*.

(b) *Marty* being used.

Figure 2-8: Figure 2-8(a) shows the HMD (nicknames *Marty*) that was used. Figure 2-8(b) shows *Marty* being used.

2-3-3 Head pose estimation

To show the correct virtual information and overlay it with the real world on the VST HMD it is necessary to know the position and orientation of the display. To estimate the position and orientation of the display, it is necessary to track objects in the environment and the viewpoint of the user. What needs to be rendered is dependent on where the user is standing and what the user is looking at. A tracker was used to track the positions of objects and the head in 6 DOF. Tracking in 6 DOF, also called pose estimation, means tracking in 3 positional dimensions (x, y, z) and 3 orientation dimensions $(roll, pitch, yaw)$. This tracking needed to be done in a robust and accurate way and in real-time.

Various techniques have been developed to track objects and viewpoints in 3D [43, 47, 48, 49]. These techniques rely on various measurable physical phenomena to track objects and are divided into the following categories:

- **Magnetic**: uses the magnetic flux generated by coils moving through a magnetic field to calculate pose.
- **Acoustic**: uses sound waves to measure pose. The time it takes a specific wave to travel from the source to the object and back is a measure for the distance of the object.
- **Capacitive**: relies on the change of capacitive charge between two plates. This phenomenon is used by for example inertial trackers.
- **Mechanical**: uses a mechanism to track pose. An example of a mechanical tracking system is a joystick.
- **Optical**: uses light to calculate pose. A good example of an optical tracker is one that uses a camera image.

Besides a measurable phenomenon to do the tracking, other requirements for the tracker were defined by Azuma [50]:

- **6 Degree Of Freedom (DOF)**: the user should be able to walk around freely and not be limited by a position or orientation. The tracker needs to do head tracking and the tracking of objects in 6 DOF.
- **Accuracy**: the tracker must be able to do the tracking accurately, otherwise the projection of the virtual objects will be off. According to Azuma the accuracy of the tracker must be within a small fraction of a degree in orientation and a few millimeters in position.
- **Robustness**: the tracker must be robust with respect to for example changing light conditions.
- **Real-time**: the tracker must operate in real-time to allow good human-robot interaction with low latency.

All tracking principles have their advantages and disadvantages and are effectively in a experiments and setups. Unfortunately most trackers require the addition of sensors to the HMD or objects, to track it. To ensure maximal mobility, this added complexity is undesired. Optical tracking doesn't have those drawbacks and enables the tracking of multiple untethered objects through computer vision. Optical tracking is becoming more popular because of lower costs, increased camera quality and more computational power. Optical tracking is divided into two categories, marker based tracking and natural feature tracking.

Natural feature tracking

Natural feature tracking is a pure optical tracking method, therefore it doesn't rely on tethered objects in the environment. It works by analysing interesting (feature) points in the image, for example textures or corners, and tracking them through various frames. These methods work in three steps: feature detection, feature tracking and pose estimation.

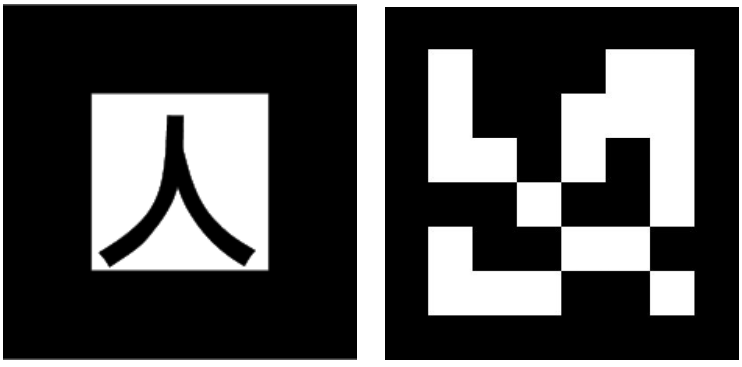
Feature detection: feature detection works by analysing images to detect so called feature point. For every feature an array of information (descriptor) is extracted using image analysis tools. Examples of features are, color, intensity, derivative, etc. The location and description of these feature points is stored for every frame.

Feature tracking: to track the feature point between frames, a tracker is needed. The tracker compares the detected feature points to a database of feature points. When a match is found, the tracker can map the new feature point to the already detected feature point and make a 3D map.

Pose estimation: pose estimation is done with the 3D map, the relation between the camera and the 3D map can be calculated to identify how to render virtual objects.

Marker based tracking

Various marker based tracking solution are used in literature, but by far the most used for AR is a fiducial marker based tracker. Popular trackers are: ARToolKit [51], ARToolKitPlus [52], ARTag [53] and AprilTag [54]. Fiducial marker based tracking works by analysing images to detect special markers (Figure 2-9). These markers are placed in the environment on known locations, so not only the relation between the marker and camera can be calculated, but also between the camera and objects in the environment.



(a) Marker used by ARToolKit. (b) Marker used by ARToolKitPlus.

Figure 2-9: Various fiducial markers used by ARToolKit [51] and ARToolKitPlus [52]

All trackers offer 6 DOF, accurate and real-time tracking of fiducial markers. ARToolKitPlus was chosen, because it offers robuster tracking compared to ARToolKit [52] and offers a C++ interface. ARTag offers even more robust tracking but was not freely available and AprilTag was written in Java, therefore not compatible with the implemented system.

Conclusion

To align the real and virtual objects a head pose tracker is needed. The quality of natural feature based trackers are getting better and the advantage is that they do not need tethered objects in the environment to be able to do the tracking which make them highly flexible in unknown environments. The disadvantage is that the natural feature tracking algorithms have to be combined with object recognition techniques to identify objects in the environment.

In this thesis marker based tracking was chosen, because the tools available are mature, optimized and robust and allow tracking in real-time. Another advantage of marker based tracking is that object recognition is not needed, because the relation between the markers and fixed objects in the environment is known. ARToolKitPlus [52] was used and Appendix A-2 describes how the marker based tracking was done.

Chapter 3

Analysis

The HMD of Section 2-3 was designed to be able to show visual information from the robot directly in front of the user's eyes. The HMD was used to improve SA when working with an industrial robot arm. The research question was:

Does showing state and intention information of an industrial robot arm, using a head-mounted display, help the human co-worker to improve situational awareness?

From the research question, two subquestions arise:

1. What kind of information about the robot arm is needed to improve Situational Awareness (SA)?
2. How to measure and quantify an improvement of SA?

Because SA is a broad concept which differs from situation to situation, it is essential to know the important factors of SA specific to the case where humans and robots are working together to repack oranges. Section 3-1 describes the pilot study that was done to analyse Human-Robot Interaction (HRI) in the case of orange repacking. From this analysis the important types of information were extracted that could lead to a potential improvement of SA. Section 3-2 gives an overview of what kind of information about the industrial robot arm was useful to show to the human using a HMD to improve SA. The types of information were analysed based on the SA level model and the potential for improvement by using a HMD.

To measure and quantify the effect of the information types shown to the human to improve SA, a measurement method and performance measure was needed. Section 3-3 gives the hypotheses and an overview of how SA was measured.

3-1 Pilot study

What kind of information from the robot arm and situation that is useful to display using a HMD is dependent on the situation and task. To analyse what the important parameters and information types were in the case of repacking oranges, a pilot study was done. In this study two test subjects were given a HMD and were asked to work together with the robot arm to repack oranges.

3-1-1 Purpose

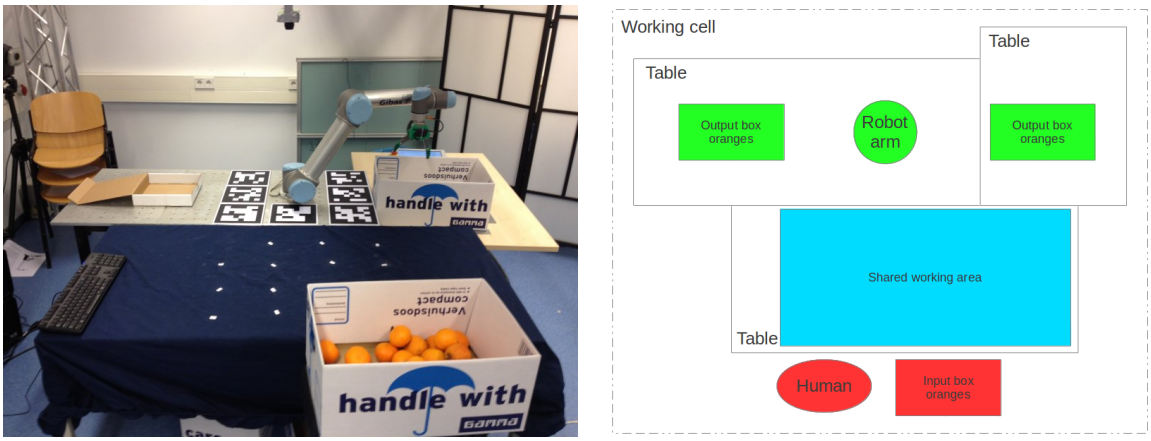
There is no literature available or prior research done to analyse the important parameters involved with HRI specific to the repacking of oranges. This is because the situations, where humans and robot arms are working together in the same physical space and on the same task, are not present in the industry. This pilot study was done to get the important parameters needed to improve SA of the human and to get an overview of the possibilities and limitations of the robot arm and human. The following questions were identified prior to the pilot study and were answered using an analysis of the pilot study:

- Is there an indication that showing the state and intentions of the robot could improve SA?
- What are characteristics of the HRI?
- Is the human able to do his/her task when working with an industrial robot arm?
- What is the effect of the HMD on the human?
- Is the robot able to do its task?

The answers were used to better design the virtual information displayed in the HMD and help design the experiment.

3-1-2 Experimental setup

Figure 3-1 shows the experimental setup where oranges were presented in the *input* box. The goal of the human was to present the oranges to the robot so that it could pick them up and put them in *output* box. A more detailed overview of the experimental setup is given in Section 4-1.



(a) Pilot study setup.

(b) Diagram of pilot study setup.

Figure 3-1: Figure 3-1(a) shows what the pilot study setup looked like from the viewpoint of the test subjects. Figure 3-1(b) shows a top view diagram of the pilot study setup. The human and *input* box are shown in red. The *shared working area* is shown in blue and the robot with *output* boxes are shown in green.

3-1-3 Tasks of human and robot arm

The test subjects worked together with the robot arm to repack oranges. The procedure of the test subjects is described in Algorithm 1. They were asked to put oranges on the *shared working area* and check them on quality, put a sticker on them and spreading them out. When the robot arm had filled the *output* box, the test subjects were asked to switch the box with an empty one.

Algorithm 1 Pilot study procedure test subjects

```
repeat
    Put oranges on shared working area
    Check orange on quality
    if orange = bad then
        Remove orange
    end if
    Put sticker on orange
    Spread out oranges
    if output box = full then
        Switch output box with empty box
    end if
until Done checking and stickering oranges
```

The procedure for the robot arm is described in Algorithm 2. The robot was picking up oranges and putting them in the *output* box. When 5 oranges were put in the *output* box, the robot waited for the test subject to switch the box.

Algorithm 2 Pilot study procedure robot arm

```
repeat
    if Detected oranges > 0 then
        Move to orange
        Pick up orange
        Put orange in output box
        if Output box = full then
            Wait for box switch
        end if
    else
        Wait for oranges
    end if
until forever
```

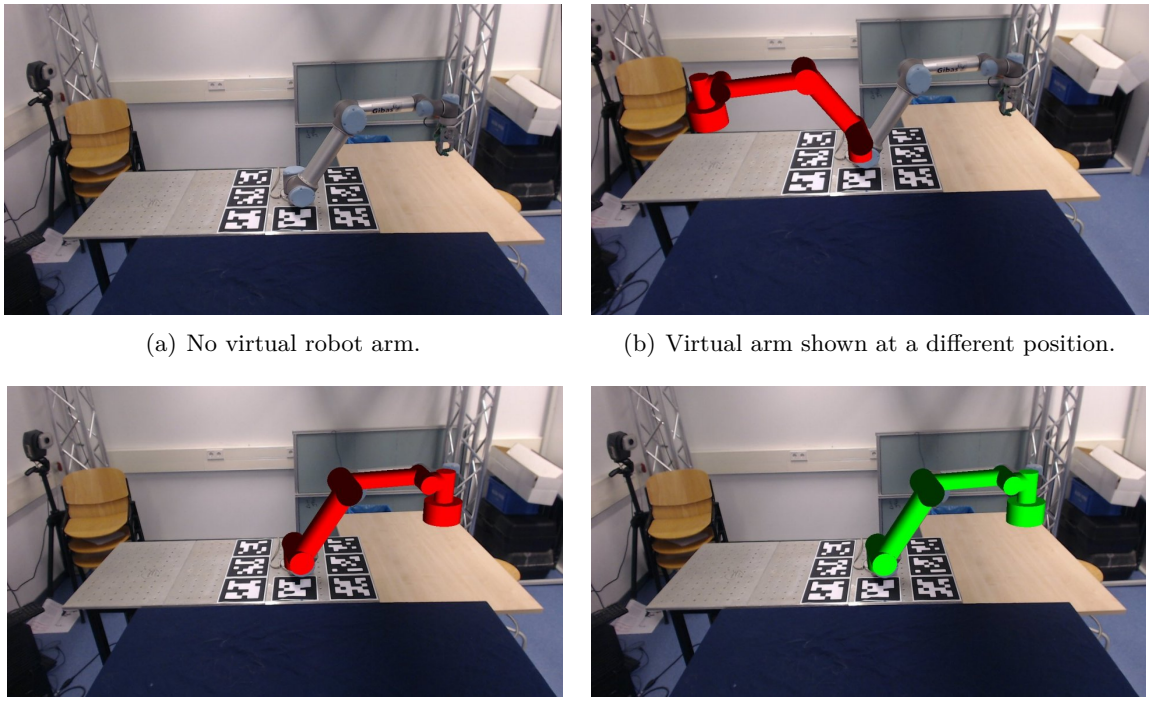
3-1-4 Virtual information

During the experiment the test subjects were presented with virtual information shown in the HMD. All virtual information was encoded using a virtual model of the robot arm (Section 4-4). To use the HMD during the pilot study, some information was shown to the

user about the robot. For the purpose of the pilot study it was opted to show a single piece of information from both the state category and the intention category (Figure 3-2). These pieces of information were selected arbitrarily. Further analysis of information that can be shown was conducted at a later stage and is presented in Section 3-2.

State: two states were distinguished in this pilot study. The first state is when the robot arm was moving. In this state the color of the virtual robot arm is red (Figure 3-2(c)). The seconds state is when the robot arm was standing still, specifically when the *output* box was full. In that case the robot arm was waiting for user intervention and would signal this to the test subject by changing the color of the virtual robot arm to green.

Intended motion: the motion, in the form of a trajectory, could be useful to the human. In the pilot study the virtual robot arm indicated the intended motion of the real robot arm by changing its position (Figure 3-2(b)). The whole trajectory was shown using the virtual robot arm, before the real robot arm would move. This trajectory was shown by lowering the interval between waypoints and displaying the virtual robot arm.



(a) No virtual robot arm.

(b) Virtual arm shown at a different position.

(c) Red virtual robot arm.

(d) Green virtual robot arm.

Figure 3-2: Virtual information shown in the HMD. Figure 3-2(a) shows the view of the experimental setup when viewed without wearing the HMD, Figure 3-2(b) shows the virtual robot arm at a different position when viewed with the HMD, Figure 3-2(c) and Figure 3-2(d) show the virtual robot arm with different colors.

3-1-5 Test procedure

Three test subjects, all male students or faculty staff from Delft University of Technology, were asked to do the experiment. Each test subject was asked to do the experiment twice,

once without wearing the HMD and once while wearing the HMD.

The experiment was observed and video taped. The observations done during the experiment and with the video tapes were used to check the behavior of the test subjects when working with the robot. Collisions with the robot were checked for and if the test subjects were able to do their task while wearing the HMD.

After the experiment the test subjects were interviewed. The test subjects were asked how the experienced the cooperation with the robot arm and how they experienced the HMD and the virtual information.

3-1-6 Discussion

Showing a state change by changing color of the virtual robot arm is useful, the test subjects were not hesitating and guessing why the robot is not doing anything, but immediately know why. An indication was given that showing state changes using virtual information could improve the SA of the human.

Showing the whole trajectory is not very useful, because there is no need for that information in this scenario. Although the test subjects did respond positive to information about the final position of the intended motion. If the test subjects knew the final position and saw that the robot was going to pick up an orange that was ready to be picked up, they could continue with their work and not keep an eye on the robot.

While doing the experiment with the HMD, the test subjects commented that that wearing the HMD was very limiting. The HMD limited the Field of View (FOV), eye-hand coordination and freedom to move. Because the HMD was tethered to a computer, by a thick cable, moving 2 meters was possible but not ideal, so changing the output box with an empty one was challenging. The eye-hand coordination was also impaired, because of lag in the visualization of the environment. This lag introduces sensory mismatches between what was seen on the display and what was done with the hands. Also the camera's auto adjusted to changing light conditions, which means that sometimes the images were very bright and the oranges could not be seen clearly.

The robot system was able to pickup and handle oranges, but the operating speed of the robot arm was too low. The human was able to handle oranges faster than the robot arm could move them. An increase in operating speed was needed.

3-1-7 Conclusion

From the pilot study could be concluded that indicating state changes and intended goal could improve SA, but that showing intended trajectory is not beneficial to improve SA. Furthermore the use of the HMD is possible, but limits the mobility and eye-hand coordination of the user.

3-2 Information to improve situational awareness

From the pilot study and an analysis of the HRI, information that would improve SA was found. Two main categories were found that enable the human to get a better understanding of the situation and environment: state and intention information. This section gives an analysis and overview of the state (Section 3-2-1) and intention (Section 3-2-2) information useful to show using a HMD to improve SA.

3-2-1 State information

Information about the state of the robot arm can improve SA, because it helps the human get a better understanding of what the robot arm is currently doing. By adding extra indications of the state, a better understanding of the robot can be achieved. Important indications of state are: position, velocity, actions done by robot arm and internal state information.

Position

The position of the robot arm is the most trivial observation and is important to reason about the situation and task execution. Because without knowing where the robot arm is, it is difficult to take it into account. The position of the robot can be observed directly, because the physical robot is present in the environment. This means that according to the SA model, it is a level 1 attribute, no extra comprehension about the environment is needed.

In the orange repacking scenario the robot is working in close proximity to the human. Because the position is directly observable, improving the observation by adding virtual information has a low potential for improving SA.

Velocity

Velocity information is needed to reason about the robot arm and predict where the robot arm will go next. Because velocity has a direct relation to position, it can also be directly observed. Velocity is therefore a level 1 attribute in the SA model, because no comprehension about the environment or situation is needed.

Because humans are better in judging position than velocity, adding extra information about velocity has some potential of being useful, but not a lot.

Action

Robot arms are used in industry to manipulate the environment and are equipped with tools to do that. For example if the robot arm is equipped with a gripper, it can be used to manipulate objects. Actions that are being performed by the robot can also directly be observed and are therefore a level 1 attribute in the SA model. Actions performed by the robot are important, because they are the way the robot interacts with the environment. That is why actions performed by the robot are also important to the human. When working with the robot arm, the human can adjust their own actions depending on the actions of the

robot arm. These actions can be made clearer by adding virtual information using a HMD, but the effect would be low. By adding virtual information no new information is given, only extra information.

In the orange repacking scenario the only action performed by robot arm was the gripping action. The robot was gripping the orange when it was positioned above it and this can be clearly seen. Adding extra indication that the robot is gripper has therefore a low potential for improving SA.

Internal state

Industrial robot arms also have an internal state. Some states are very clear when observed by the human, for example when the robot arm is moving. Sometimes the robot arm has a state where it is for example not moving, in those cases it is less clear to the human what the internal state is. Possibilities are that it is turned off, that there is an error or that it is waiting for human intervention. Internal state information has a high potential for improvement, because a human can't see on the outside of the robot what its current state is. So by adding information about the internal state of the robot arm, new information is provided. When the robot arm is not moving, it is important to know why it is not moving, since based on that knowledge, the human can decide to take appropriate actions. This kind of information is useful to the human to improve SA and is level 2 information according to the SA model. Extra information about the internal state shown in a HMD can improve SA of the human.

When working with a robot arm to repack oranges, two internal states could be identified.

Waiting for user: when the *output* box is filled with oranges, the robot arm will wait for the user to replace the box. Adding an extra indication that the robot is waiting for user intervention is helpful, because the human then doesn't have to guess why the robot is not moving, therefore it has a high potential of improving SA.

Waiting for oranges: when the robot arm is not detecting oranges, the robot arm is not moving. Adding extra information about this state is useful, because then again the human doesn't have to infer why the robot is not moving.

3-2-2 Intention information

In the SA model of Endsley (Section 2-1-1) level 3 SA states that an important part of human decision making is projecting the future. To predict what is going to happen next is difficult and requires a good internal model of the situation. To build this model it takes time and experience.

Showing what the robot arm is intending to do before it actually does it could be useful to help the human reason on level 3 about the situation. From direct observations of the robot arm, there is no way for the human to know where the robot arm is going to move, that is why providing this extra information has a high potential for improvement.

Knowing where the robot arm is going consists of four parts: the goal position and the trajectory of the movement, the intended action and intended internal state.

Intended goal position

The intended goal position of the robot is important information to the human, because it can be used to make better decisions. For example if the human knows that the robot is going to move to a specific area, he can decide to not go there and do other tasks. This information is about projecting the future, so it is level 3 knowledge. Because it is not directly observable by the human, it has a high potential for improvement.

The robot arm can go to various positions to handle oranges, it is therefore useful for the human to know where the robot is going to move.

Intended trajectory

Not only the information about the intended goal, but the whole motion could be important for humans to reason about the robot. Knowing how it is going to move through space could be useful to anticipate to the robot. This information is also about projecting the future and therefore it is also level 3 knowledge.

From the pilot study was known that in the case of repacking oranges, knowing the trajectory of the intended motion is not useful with the chosen implementation.

Intended action

Information about the intended actions of the robot arm relates to the information about current action performed by the robot, only projected into the future. By knowing the intended action of the robot the human could better anticipate to it. The intended actions can't be directly observed by the human and therefore it has a high potential for improvement.

In the case of working with a robot arm to repack oranges, only one action is available to the robot arm. The human already knows what action the robot is going to take then, therefore, in this implementation, it has a low potential for improving SA.

Intended internal state

Information about the intended state of the robot arm relates to information about the current internal state of the robot, only projected in the future. Knowing the intended state could be useful for humans to anticipate. This information is level 3 knowledge, because it needs to be inferred from other observations.

When working with the robot arm to repack oranges, the difference between current internal state and intended internal state is the time the information is known beforehand, therefore the potential to improve SA, in this implementation, is medium.

3-2-3 Conclusion

Table 3-1 gives an overview and summery of the categories and types of information that were determined to be important to a human to improve SA. The content of the table was gathered

during the research done on this project and from the pilot study. All information types were given a SA level corresponding to the SA model. Based on the analysis, all information types were given an *improvement potential*, this indicates how much the human would benefit if that information is available when working with the robot arm.

Category	Type	SA level	Improvement potential
State	Position	Level 1	Low
	Velocity	Level 1	Low
	Action	Level 1	Low
	Internal state	Level 2	High
Intention	Position	Level 3	Very high
	Trajectory	Level 3	Very high
	Action	Level 3	Very high
	Internal state	Level 3	Very high

Table 3-1: Overview of information to improve SA. This table was constructed during the research and shows per information type the level in the SA model and its potential for improvement to the SA of the human. Position, velocity and trajectory are information about the way the robot moves through the space. The actions are done by the robot, for example gripping. Internal state represents what the robot is doing, for example waiting for actions by the user.

The state information from the robot with the highest potential to improve human SA is the internal state. In the category of intention all information types have a high potential to improve SA. To answer the research question it is sufficient to only test one information type from each category, therefore only internal state and intended position were chosen to be implemented and evaluated further.

3-3 Hypotheses

The results of Section 2-1-2 and 3-2 were combined to formulate hypotheses to measure an improvement of SA. Section 2-1-2 stated that performance metrics were used to evaluate SA, this section elaborates on that and combines a performance metric with the two chosen information types to formulate two specific hypotheses.

Performance metrics

In literature various performance metrics are defined and can be divided into three classes: time based, error based and coverage based [55, 56].

Time based: these metrics are used to measure the speed of task completion or reaction times.

Error based: these metrics are used to measure mistakes during task execution.

Coverage based: these metrics tell something about how much of the larger goal is completed.

A time based metric was the chosen metric to measure SA. As a time based metric, the time the human needed to judge a situation is used. By getting the extra information through the

HMD about internal state and goal position, the human can judge and reason faster about the situation, therefore the reaction time of the human were used.

When combining the types of information that are useful to display and the performance metric, the following hypotheses were formulated:

- Visualizing the internal state of the robot arm using a head-mounted display reduces the reaction time of the human co-worker when the robot arm changes internal state.
- Visualizing the intended goal position of the robot arm using a head-mounted display reduces the reaction time of the human co-worker when reacting to a specific destination position of the robot arm.

Chapter 4

Design

To test if the hypotheses of Section 3-3 are valid, an experiment was done. This chapter describes the design of the experimental setup that was used to enable Human-Robot Interaction (HRI), and the design of the virtual information shown on the HMD. Section 4-2 gives a general overview of the layout of the robot working cell and Section 4-3 describes the workings of the vision system, the robot arm and the gripper. Section 4-4 describes the way the information of Section 3-2 was encoded into visual cues shown on the HMD.

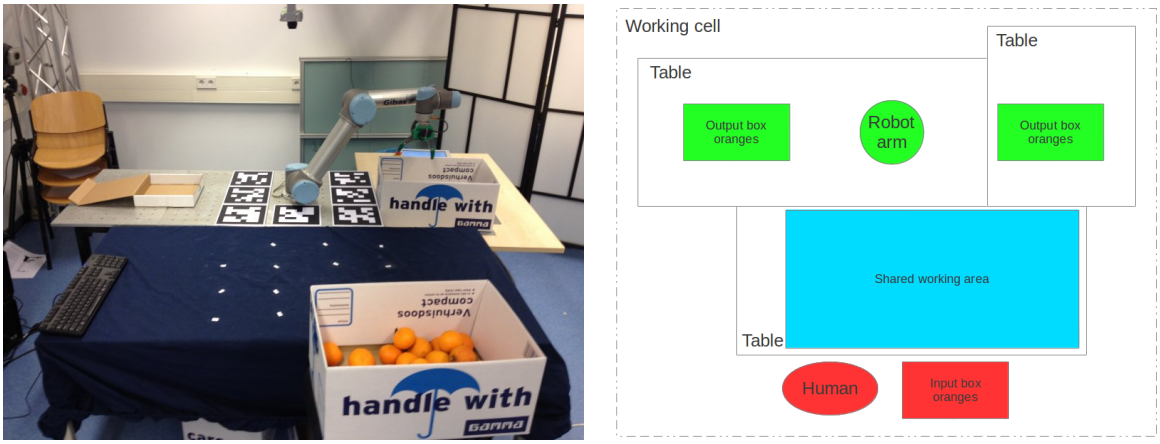
4-1 Experimental design

The experimental setup was modelled to a real world scenario. Bethel and Murphy [22] state that a HRI experiment should reflect the application domain as realistically as possible. The use of a robot arm to help with repacking of fruit and vegetables, in this case oranges, was chosen. Oranges get delivered from the farms in big crates (about 60 pieces per crate) and need to be repacked into so-called display boxes (about 20 pieces per box in a single layer). The boxes are then ready to be displayed in supermarkets and grocery stores. During repacking the oranges also need to be checked for quality and a sticker needs to be put onto them. This repacking, quality checking and placing of the sticker is largely done by hand, but could potentially be automated with the use of a robot arm. The experimental setup, modelled as a robot working cell, was designed in such a way that the human and robot are working in the same working area.

4-2 Robot working cell

As stated before, the robot working cell was designed to handle oranges. Oranges were chosen, because they have a bright orange color and can be detected accurately using a detection algorithm. The second reason for choosing oranges was that they are small and round, so that a gripper can pick them up reliably. The robot working cell is shown in Figure 4-1(a)

and a top view diagram of the robot working cell is shown in Figure 4-1(b). To get a better understanding of the working of the cell, it can be divided into two parts: a human part and a robot part (Figure 4-2).



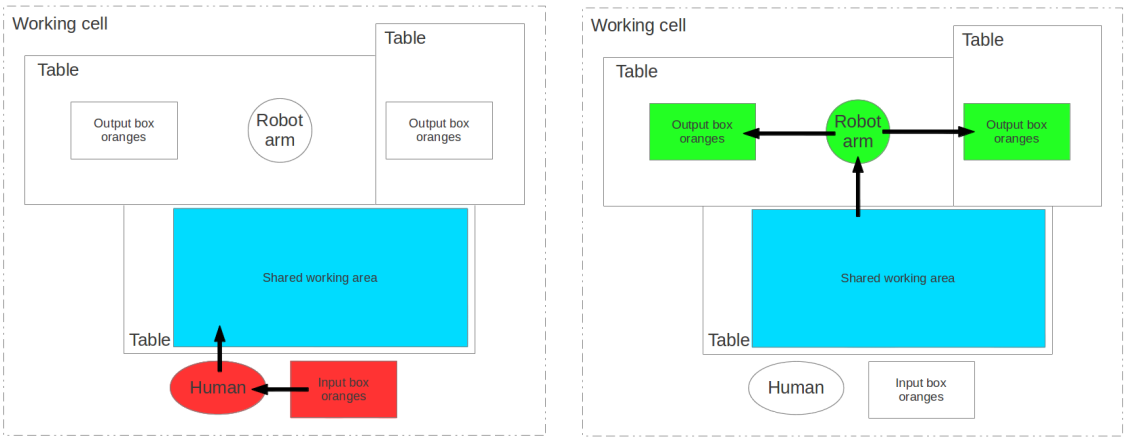
(a) Photo of the experimental setup.

(b) Top view diagram of the experimental setup.

Figure 4-1: Experimental setup overview. Figure 4-1(a) shows a picture of the experimental setup and Figure 4-1(b) shows top view diagram of experimental setup. The core of the setup was the *robot arm*, in front of the robot, covered in a blue cloth, is the *shared working area*. The oranges were taken from the *input* box, handled by the human and put on the *shared working area*. The robot system detected the oranges, picked them up and placed them in an *output* box.

Figure 4-2(a) shows a top view diagram of the human part of the robot working cell. The human interacted with the robot cell on two locations: the *input* box and the *shared working area*. The task of the human was to stands in front of the *shared working area*, take oranges from the *input* box, do quality checks, put a sticker on them and place them on the *shared working area*. The second task of the human was to help the robot arm. This was done by spreading out the oranges that are put on the table, so that the robot arm can detect them and pick them up. This was done by placed the oranges on a grid (shown by the little white stickers on the blue table cloth in Figure 4-1(a)). The human also needed to keep an eye out for the robot, because sometimes the robot needed help from the human.

Figure 4-2(b) shows a top view diagram of the robot part of the robot working cell. The robot was the core of the setup and interacted with it on two locations: the *shared working area* and the *output* boxes. The robot stood on opposite to the human, on the other side of the *shared working area*. The task of the robot was to detect the oranges on the *shared working area*, pick the oranges from the table and place them in one of the *output* boxes. When no oranges were detected it waited until oranges were detected and when the *output* box Was full it waited for the human to switch the box with an empty one. These waiting moments were the *internal states* of the robot arm. There are no external displays or other methods for the human to know what the robot is doing, so the human is relying solely on their own observations to infer the *internal states* of the robot arm.



(a) Human part of the robot working cell.

(b) Robot part of the robot working cell.

Figure 4-2: Human and robot part of robot working cell. Figure 4-2(a) shows a diagram of the human part of the robot working cell. The areas highlighted were the interaction points of the human: oranges were taken from the *input* box, handled and placed on the *shared working area*. Figure 4-2(b) shows a diagram of the robot part of the robot working cell. The highlighted parts were interaction points of the robot arm: oranges were taken from the *shared working area* and placed in one of the *output* boxes. Arrows indicate the path the oranges take.

4-2-1 Conclusion

The robot working cell was designed to mimic a real world scenario, repacking of oranges. The human was working with the robot to repack the oranges, do quality checks and put a sticker on the oranges. The robot working cell was designed specifically to ensure that the human and robot are working in the same working space.

4-3 Robotic system implementation

To be able to work with a human, the robot system needed hardware and software to operate. It needed software to detect the oranges, operate the robot arm and the gripper. Hardware is needed to pick up the oranges. This section describes how the system was designed as explained in Section 4-2. Figure 4-3 gives an overview of the main parts of the robot system.

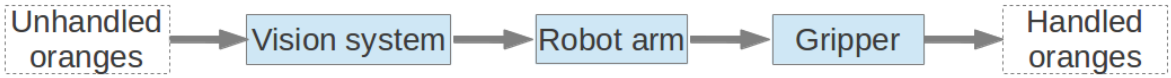


Figure 4-3: Robot system overview. A vision system is used to detect the oranges, which are then picked up with a gripper mounted on the industrial robot arm.

Section 4-3-1 lists all the materials used to build the robot working cell. Section 4-3-2 describes how the robot system is combined with the HMD. Section 4-3-3 describes the orange detection system used to detect the oranges on the *shared working area* and Section 4-3-4 describes the hardware and software to control the robot arm.

4-3-1 Materials

The materials used to design the robot working cell were:

Univeral Robot UR5: the robot arm that was used is a Universal Robots UR5, a 6 DOF robotic manipulator (Appendix B-1-2).

Lacquey Feth Hand: to manipulate objects in the environment, the UR5 was equipped with an end-effector (Appendix B-1-3).

Table: the shared working space of the robot and the human was a normal table of about $1.8m$ by $0.8m$. It was put on wooden blocks of $5cm$ to level it with the robot arm socket. The table was covered with a dark blue cloth, to make the oranges better detectable.

Logitech C905 webcam: this webcam was used for the orange detection algorithm (Appendix B-1-1).

Oranges: oranges ($\pm$ 50 pieces) with a diameter of about $6cm$ were used to do the experiment. Bigger oranges could not be used, because the gripper could not pick them up reliably. This was because the gripping force was to low, therefore the orange would slide out of the gripper.

Head-mounted display: the HMD was used to display the virtual information from the robot system (Appendix A-1).

Boxes: as *input* and *output* boxes, simple moving boxes were used.

Laptop: a HP Elitebook 8540w with a quad-core Intel i7 was used to operate the HMD and run the AR software.

Desktop computer: a desktop computer was used to operate the robotic system with the control software.

LAN switch: to connect the two computer and the UR5 a LAN switch (Linksys WRT54g) was used.

UTP cables: to connect the computer and UR5 to the switch, UTP cables of $10m$ were used.

Video camera: to records the experiment a Sony Cyber-shot with tripod was used.

4-3-2 Robot working cell combined with HMD

This robot working cell was combined with the HMD of Section 2-3-2. These were combined to show information from the robot arm on the display in real-time. Two computers were used to operate the combined system, one computer was connected to the robotics system and one computer was connected to the HMD. They communicated with each other using a LAN-connection. Figure 4-4 shows the connection scheme of the computers. Two computers were used because of bandwidth issues when connecting three USB cameras to on computer.

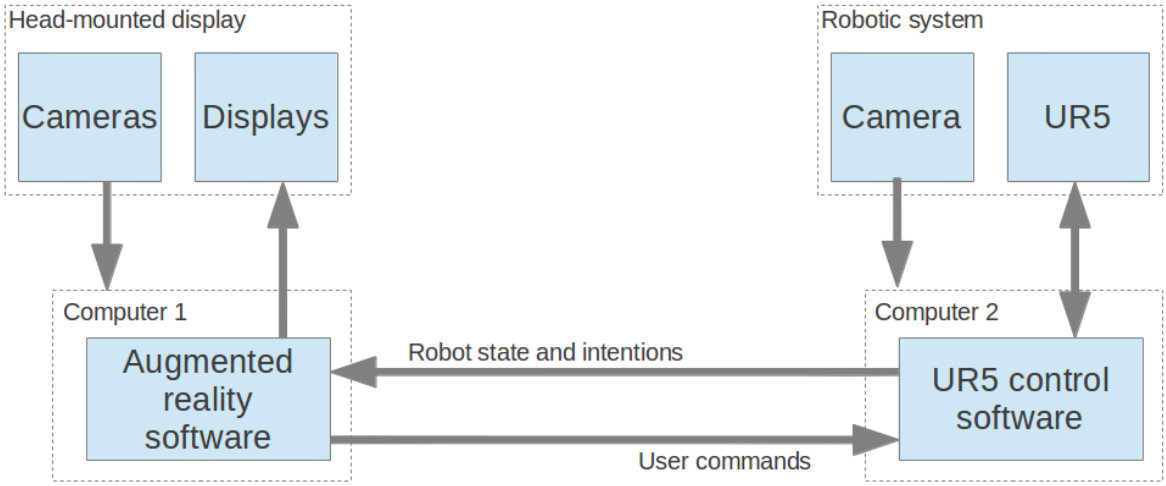


Figure 4-4: Connection scheme of the computers used in the experiment. Computer 1 is used to operate the HMD and is connected to it using a DisplayPort cable for the screens and two USB cables for the cameras (Section 4-4). Computer 2 is used to operate the robotic system. It connects to the UR5 using an UTP cable and is connected to the orange vision camera using an USB cable.

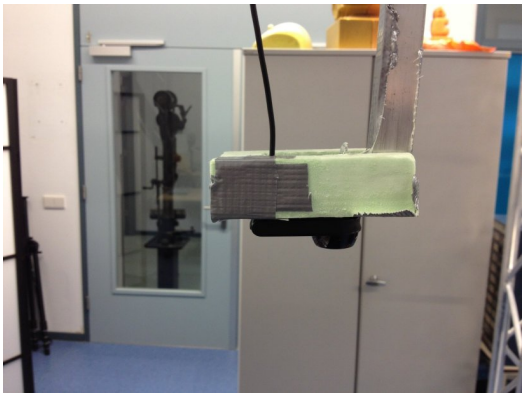
4-3-3 Vision system

To be able to detect oranges on the *shared working space*, a vision system was needed. This section gives a brief overview of the important hardware, software and capabilities of the orange detection system. The vision system was specifically designed to detect oranges on the *shared working area*. Appendix B-1-1 describes the hardware components and Appendix B-2-2 describes the software in more detail.

Hardware

Figure 4-5 shows the camera used to detect the orange. This camera was mounted $0.9m$ above the *shared working area* in a foam enclosure. This enclosure housed the camera and made it face downwards. The camera captured images of the table with a frequency of $2Hz$. $2Hz$ was used to save computational researches and because the orange handling speed of the robot arm is about 1 orange per 4 seconds, so a higher update frequency doesn't give more information to the robot.

(a) Logitech C905 consumer grade webcam.



(b) Foam enclosure to mount the camera.

(c) Aluminium rod mounted on a frame above the robotic system.

Figure 4-5: Camera and mounting frame used to detect oranges. Figure B-2(a) shows the Logitech C905 consumer grade webcam that was enclosed in foam (Figure B-2(b)) and mounted using an aluminium rod to a frame (Figure B-2(b)) at $0.9m$ above the table with oranges.

Software

Figure 4-6 gives an overview of the software to detect oranges.

Image capture: the images from the camera needed to be capture (using OpenCV [57]) on the computer and stored in an appropriate format to be analysed. The images were captured at a frequency of $2Hz$ and stored in Blue-Green-Red (BGR) format. The input image in Figure 4-7 shows the view of the camera when facing down at the *shared working area*.

Orange detection: the images captured by the camera were analysed by a Gaussian Mixture Model algorithm (designed by Floris Gaisser). Figure 4-7 shows the steps of the algorithm.

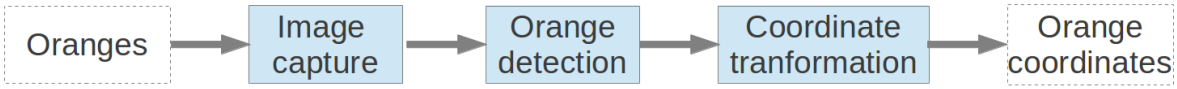


Figure 4-6: Schematic overview of the orange detection software. The images taken by the camera were captured and stored for analysis. A Gaussian Mixture Model algorithm detected the oranges in the images. For the robot arm to be able to pick up the oranges, the coordinates needed to be converted from image frame coordinates (pixels) to real world coordinates (meters).

The *input image* was stored to be analysed. The first step was to extract the *hue image*. The *hue image* was used, because there is a big difference between the hue value for blue (background) and orange (objects), therefore a clear distinction could be made. A *histogram* was made of the *hue image*, in the *histogram* two peaks can be detected, one for each hue-value (blue and orange). The Gaussian Mixture Model algorithm iterates over the *histogram* to fit two Gaussian distributions over the peaks. These Gaussian distributions were used to segment the *input image* into a binary image, also called a *mask*. In the *mask* the background is represented a pure black and the oranges are pure white. After a binary morphology step a *filled masked* was used to fit ellipses to get the coordinates of the oranges.

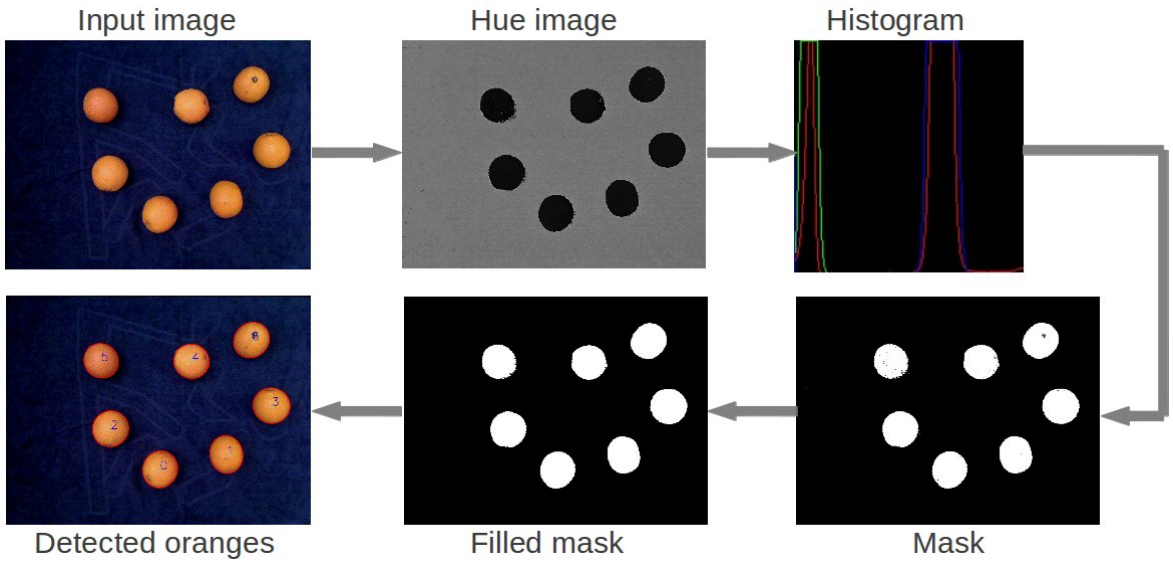


Figure 4-7: Overview of Gaussian Mixture Model detection algorithm. From the *input image* the *hue image* was extracted. A *histogram* was calculated from the *hue image* and two Gaussian distributions were fit over the *histogram*. Segmentation was done using the two Gaussian distributions to get a *mask*. Ellipses were fit over the *filled mask* to get the coordinates of the *detected oranges*.

Coordinate transformation: the *detected oranges* were defined in image coordinates (u, v) in pixels, with $(0, 0)$ defined as the upper left corner. For the robot to be able to pick them up, these coordinates needed to be converted form image coordinates to real world coordinates (X_c, Y_c, Z_c) in meters (Equation 4-3). The output of the orange detection algorithm was a list of oranges with their coordinates in the image frame (u, v) and coordinates in the world frame (X_c, Y_c, Z_c).

$$X_c = \frac{(u - c_x)}{f_x} z_c \tag{4-1}$$

$$Y_c = \frac{(v - c_y)}{f_y} z_c \tag{4-2}$$

$$Z_c = 0 \tag{4-3}$$

In this transformation (c_x, c_y) was the principle point of the image and f_x and f_y and the focal lengths. z_c was the height of the camera above the table, which is constant. Figure 4-8 shows the relation between the two coordinate frames. The coordinates of the detected oranges were converted (X, Y) world coordinates using Equation 4-3, but the Z-coordinates was missing. The Z-coordinate of the oranges was defined as 0, because the *shared working area* was at the same height as be base of the robot. So from the robots point of view, all oranges were at a height of 0.

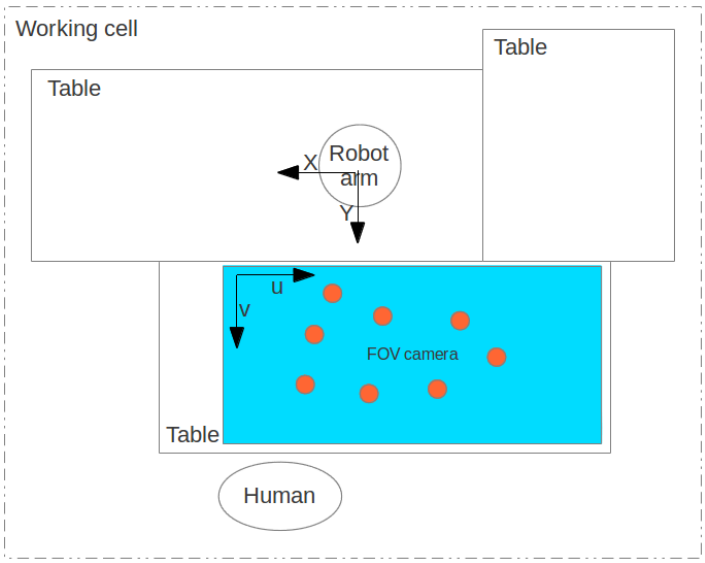


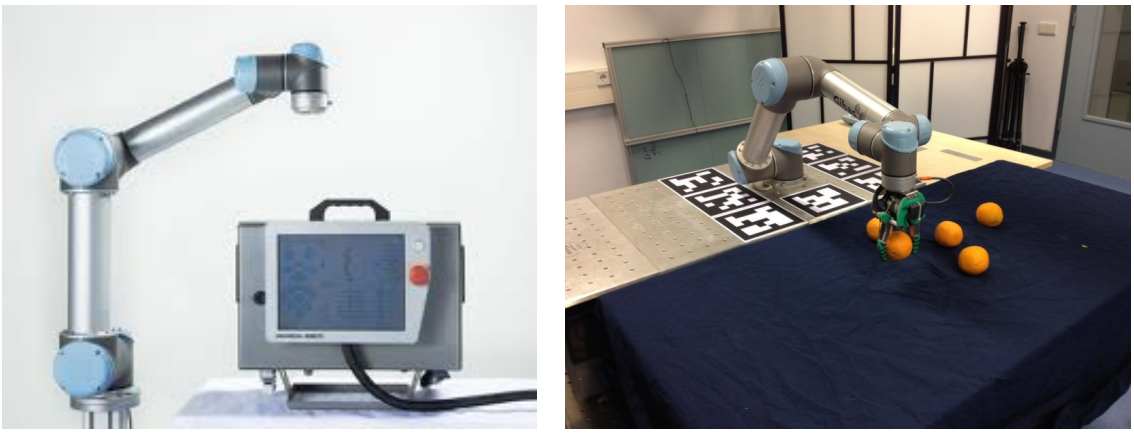
Figure 4-8: Image and robot coordinate frames. The image coordinate frame (u, v) was the local coordinate frame of the captured images, with (0, 0) defined as the top left corner of the image. The world coordinate frame (X_c, Y_c, Z_c) was defined at the base of the robot arm (0, 0, 0).

4-3-4 Robot arm

With the coordinates of the oranges known, the robot arm was used to pick up the detected oranges and move them. This section gives an overview of the hardware and software used to pick and place the detected oranges. Appendix B-1-2 describes the hardware and Appendix B-2 describes the software is more detail.

Hardware

Figure 4-9 shows robot arm that was used and how it was used in the experimental setup. It is a commercial 6 DOF intrinsically safe industrial robot arm, developed by Universal Robots. An intrinsically safe robot arm, means that it doesn't have enough power to physically hurt a person and therefore no fences have to be around it [2]. The joints of the UR5 have voltage sensors, if the measured voltage exceeds the predicted voltage, the robot will switch off. The robot arm is $1m$ high, has a reach of $0.85m$ and can handle a maximum payload of $5kg$. The specifications of the UR5 are shown in Table 4-1.



(a) Universal Robots UR5 with control unit. (b) Universal Robot UR5 in experimental setup.

Figure 4-9: Figure 4-9(a) shows the UR5 with control unit and Figure 4-9(b) shows the UR5 in the experimental setup handling oranges.

Parameter	Value
Weight [kg]	18.4
Payload [kg]	5
Reach [m]	0.85
Joint ranges [rad]	-2π to 2π
Joint speed [rad/s]	π
Maximum tool speed [m/s]	1
Repeatability [mm]	$\pm$ 0.1
DOF [-]	6
Power consumption [W]	200
UR5 connection [-]	UTP

Table 4-1: Universal Robots UR5 specifications. This table shows all relevant specifications of the Universal Robots UR5 industrial robot arm.

Software

To make the robot do what we want, control software is needed. The Universal Robots UR5 has its own control box but unfortunately the functionality is sufficient. The software

provides all the components needed to actuate the robot and make it move to any position, but unfortunately the motion planning and inverse kinematic software is a black box: there is no way to extract the motion plan before execution. An alternative control software package was found in Robotic Operating System (ROS) [58]. This package allows the motion plan to be extracted before the motion plan is executed.

All software was designed on the ROS framework, version *Fuerte*. ROS is a framework that focuses on delivering a flexible, versatile and complete set of tools needed to program a robot. It is widely used in academic community because it makes it rapid development of relative complex pieces of software possible. Figure 4-10 shows an overview of all ROS components needed to operate the robotic working cell.

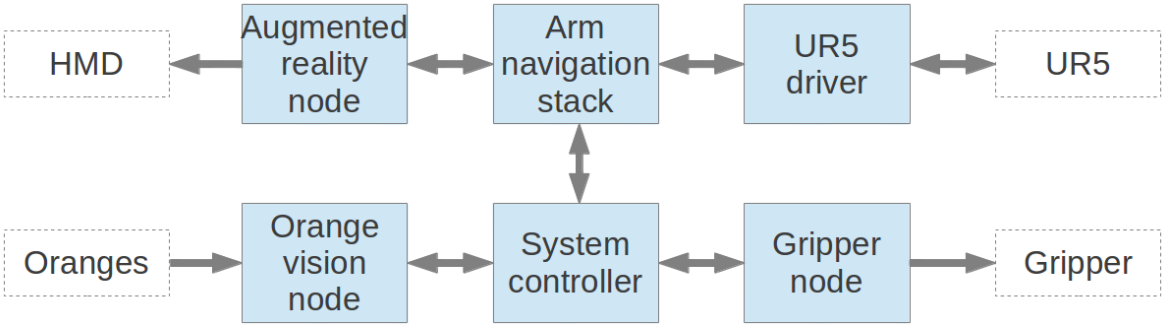


Figure 4-10: Overview of all the ROS software components. The *system controller* is the main hub of the robotic system. This node requests the coordinates of the orange from the *orange vision* node. With the orange coordinates a motion plan request is send to the *arm navigation stack*. This stack is a collection of nodes responsible for calculating a proper trajectory for the robot arm. The *arm navigation stack* sends the calculated trajectory to the *UR5 driver*, which activates the real robot arm. The *system controller* also activates the gripper via the *gripper node*. The *UR5 driver* and *arm navigation stack* both send robot data to the *augmented reality node* (Section 4-4).

To operate the robot, the *system controller*, *arm navigation stack* and *UR5 driver* are important. The *system controller* is a piece of software responsible for connecting all software and hardware components, so basically the core of the robot working cell. The *arm navigation stack* is responsible for trajectory planning to allow the robot arm to move to the desired destination without colliding with the environment. The *UR5 driver* sends the generated trajectory to the robot arm for execution and receives real-time state information from the robot arm. The *augmented reality node* is further described in Section 4-4.

System controller

The system controller was designed as the core component of the robot working cell, it coordinated all other software and hardware components to make sure that the system ran correctly. This node received the coordinates of the oranges (X_c, Y_c, Z_c) and converted those to coordinates in the robot coordinate frame (X_r, Y_r, Z_r). With the coordinates in the robot coordinate frame, a motion plan was generated to pick up the orange, this was done by the *arm navigation stack*. Once a confirmation was given by the *arm navigation stack*, the gripper was operated.

Arm navigation stack

The *arm navigation stack* was provided by ROS. To get it to work properly it had to be

customized and tweaked, because the *arm navigation stack* is generic and is designed to work on every robot arm (with some modifications)(Appendix B-2-5). Figure 4-11 give an overview of the components involved with enabling the robot to pick up oranges with the *arm navigation stack*.

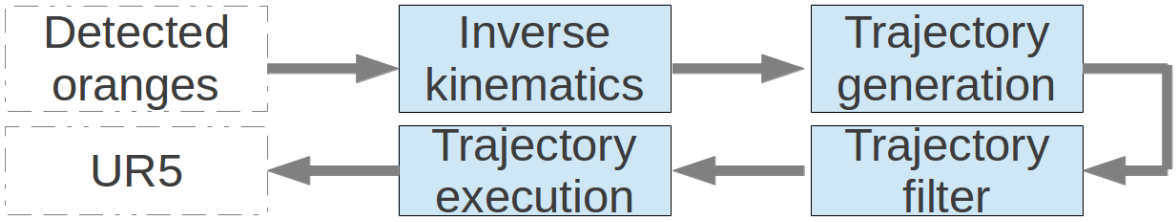


Figure 4-11: Schematic overview of the steps needed to pick the detected oranges. The first step was to calculate if the detected orange could be picked up, this was done by an inverse kinematic calculation. If the orange could be picked up, a trajectory was generated and filtered. The last step was to execute the calculated trajectory.

Inverse kinematics: the detected oranges needed to be reachable by the robot arm. To check this, an inverse kinematic calculation was done to determine if the orange could be picked up and in what position the robot arm needed to be. This final position was also checked for collisions with the environment and for self-collision.

Trajectory generation: when the orange could be reached by the robot arm, a trajectory was generated by the motion planner. The motion planner calculated a trajectory that did not violate the environmental constraints and did not have self-collisions.

Trajectory filter: the result of the motion planner was a very sparse motion plan. To be able to execute the motion plan on the robot, the trajectory needed to be smooth. The trajectory filter interpolated the trajectory and outputted a smooth motion plan.

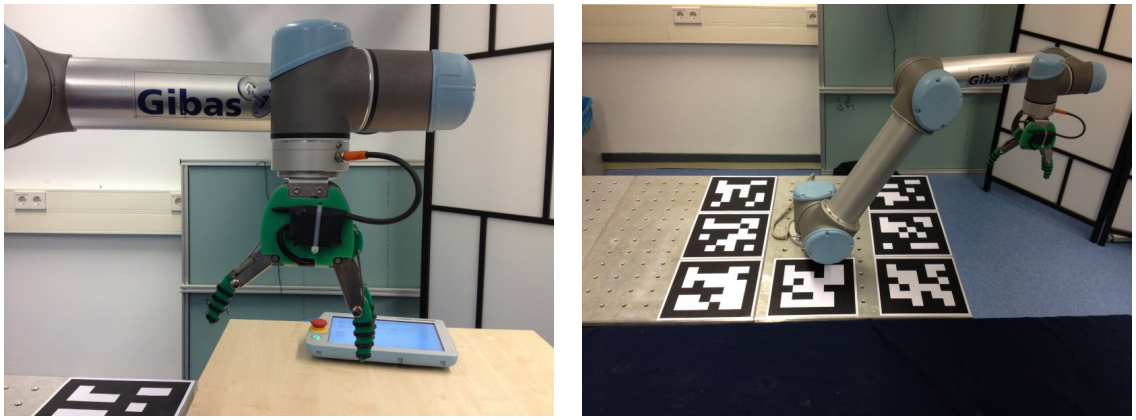
Trajectory execution: the filtered motion plan was executed by a driver. This driver communicated with the UR5 robot arm to execute the motion plan. It also got the real time position and velocity from the robot arm.

UR5 driver
The driver checked the calculated trajectory again for invalid data and interpolated the data points again for a smooth execution. It sent the joint angles and velocities to the robot and received the current joint angles and velocity from the robot at a frequency of $100Hz$. The communication to the robot was done over UTP.

4-3-5 Gripper

The end-effector in the setup was a Fetch Hand from Lacquey. This is a 6 DOF underactuated robotic hand developed as an affordable alternative to existing industrial robotic hands. It has three fingers with each two DOF. All 6 DOF are actuated by one motor, this makes in underactuated. The intelligence of the hand is all in the mechanics, all DOF follow the way of the least resistance. That means that one DOF feels resistance, the power goes to the other DOF, this way all finger enclose the object and the gripping force is divided over the fingers. Figure 4-12 shows the gripper mounted on the Universal Robots UR5 in the

experimental setup. It is connected to two digital outputs ($24V$) of the robot arm, located on the end-effector. Both digital outputs are connected to a relay, one enables/disables the gripper and one opens/closes the gripper.



(a) Lacquey Fetch Hand close-up.

(b) Lacquey Fetch Hand.

Figure 4-12: Lacquey Fetch Hand mounted on the Universal Robots UR5 in the experimental setup.

4-3-6 Conclusion

The robot working cell enabled humans and the robot to repack oranges together. A vision system was used to detect the oranges and send the coordinates to the control software. The control software operated the robot arm and gripper to pick up oranges without colliding with the environment. It also provides the needed information to be able to show the current state and intended motion to the user using a HMD. The position and velocity were received from the UR5 using the *UR5 driver* in real-time, the intended goal position and motion plan were extracted from the motion planner in the *arm navigation stack* and the current internal state were extracted from the *system controller*. These five groups of information (Figure 4-13) were used to show the state and intentions of the robot arm using a HMD.

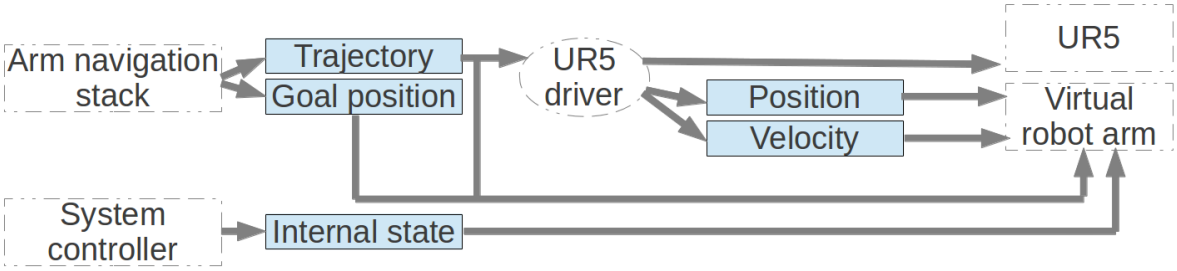


Figure 4-13: State and intention data overview. The motion planner in the *arm navigation stack* generated a motion plan and outputted the trajectory and goal position. The *UR5 driver* made sure the trajectory was executed on the robot arm and received the position and velocity in real-time from the robot arm. The current internal state, trajectory, goal position, current position and current velocity were also sent to the virtual robot arm.

4-4 Virtual robot arm implementation

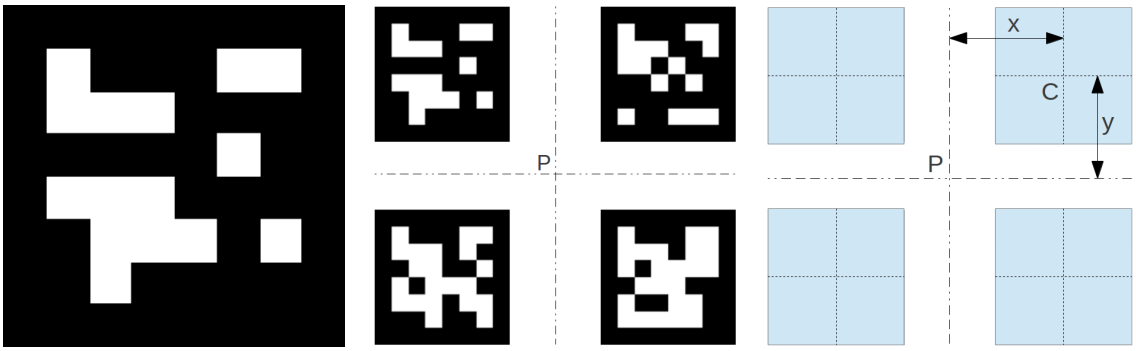
With the state and intention information available from the control software, it could be displayed on the HMD. To display the information two design problems were solved:

- Calculate what the user is looking at, also called head pose estimation.
- Render the state and intention information on the HMD.

Section 4-4-1 describes how the head pose estimation was done by placing markers in the robot working cell. Section 4-4-2 and 4-4-3 describes how the state and intention information was rendered on the HMD using a virtual model of the robot arm.

4-4-1 Head pose estimation

To make sure the correct virtual information was shown in the correct place, head pose estimation was needed. As stated in Section 2-3-3, ARToolKitPlus was used for the head pose estimation. This is a fiducial marker based tracker and estimates the head pose by detecting markers in the captured images. The transformation matrix between the cameras and the markers was used to calculate the 6 DOF head pose. Figure 4-14(a) shows what a marker look like, it is a square black and white image with known dimensions. The unique pattern in the middle of the marker is used to identify it.



(a) Fiducial marker.

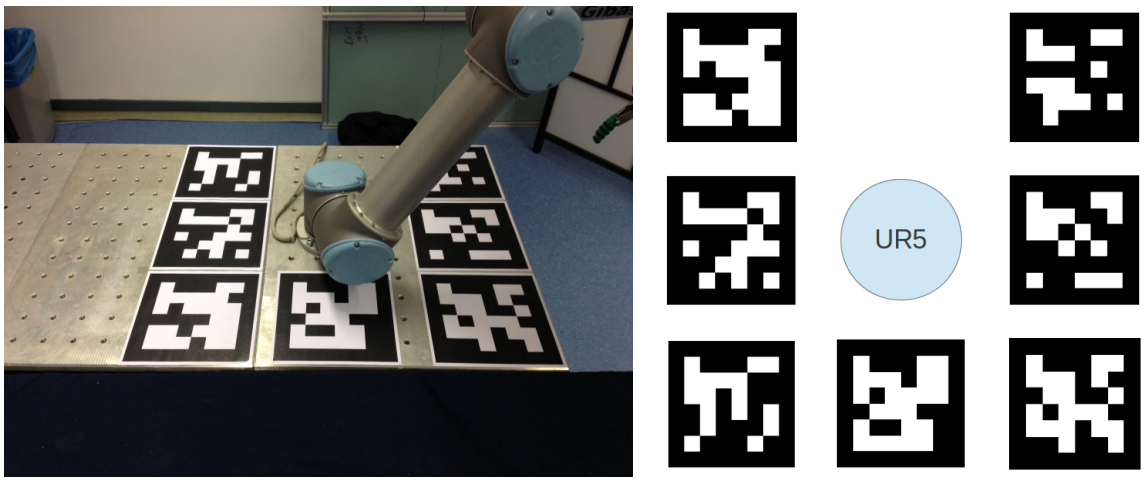
(b) Marker board with multiple markers to identify a shared central point P.

(c) Diagram of marker board. x and y are the distance from the center of marker C to the shared central point P.

Figure 4-14: Figure 4-14(a) shows what a marker used by ARToolKitPlus looks like. Figure 4-14(b) shows how multiple markers can be used to identify a shared central point and Figure 4-14(c) shows a diagram of how a marker board is defined.

The marker detection algorithm used by ARToolKitPlus provided a good and fast way to detect markers in an image and was be used to track the head pose of a human. But there were some limitations and drawbacks to using markers. One limitation was that the camera had to have a marker in its FOV to detect it. When the marker was outside the FOV or when it was occluded, no marker could be detected. This was solved by using multiple markers defined as a marker board (Figure 4-14(b)). A marker board uses a known set of markers in a predefined position and with a known relation to each other. The advantage of a marker

board was that only one marker of the whole marker board needed to be detected to know the position of the whole board. To accurately detect a marker in the image, it needed to be of a minimal size. To ensure that the marker was detectable when viewed from about 2 m, a marker size of 0.2 m was used. This means that the marker has a width and a height of 0.2 m. Figure 4-15 shows the marker board in relation to the robot working cell.



(a) Markerboard surrounding the UR5. (b) Diagram of the markerboard.

Figure 4-15: Multiple markers defined to represent the same central point is used to overcome the shortcomings of using one marker. Figure 4-15(a) shows the marker board surrounding the robot arm and Figure 4-15(b) is a diagram of the marker board.

The system needed to track the position of the head in 6 DOF to know what the user was looking at. When a marker is detected, four coplanar and not collinear points of the marker were used to calculate the transformation matrix between the camera and the marker. Figure 4-16 shows this relationship.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} V_{11} & V_{12} & V_{13} & W_x \\ V_{21} & V_{22} & V_{23} & W_y \\ V_{31} & V_{32} & V_{33} & W_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = \begin{bmatrix} & V_{3x3} & & W_{3x1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = T_m \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \quad (4\text{-}4)$$

In Equation 4-4 T_m is the transformation matrix between marker and camera [51]. This transformation matrix consists of a 3x3 rotation matrix (V_{3x3}) and a 3x1 translation matrix (W_{3x1}). These matrices were found with ARToolKitPlus by solving the equations using every corner point of the marker (x_c, y_c). The coordinates were also corrected by the camera calibration parameters (Appendix C).

Conclusion

ARToolKitPlus was used to calculate what the user was looking by calculating a transformation matrix (T_m). This matrix was the relationship between the marker board coordinate

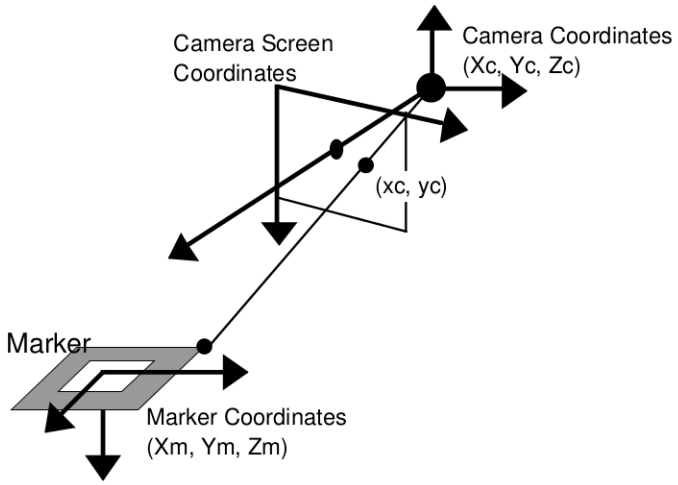


Figure 4-16: Relationship between camera and marker coordinate frames [51]. The marker coordinate frame (X_m, Y_m, Z_m) needs to estimated by calculating a transformation matrix using all corner points (x_c, y_c) in the camera coordinate frame (X_c, Y_c, Z_c).

frame (X_m, Y_m, Z_m) and the camera coordinate frame (X_c, Y_c, Z_c). The marker coordinate frame was defined using a marker board, which was a combination of different markers to identify one specific point. A marker board was used to overcome the shortcomings when using one marker, for example occlusion of the marker. By using a marker board the chance that one marker was detected by ARToolKitPlus increased and when multiple markers were detected the accuracy of the transformation matrix also increased.

4-4-2 Virtual robot arm rendering

To help the human gain better SA when working in the robot cell of Section 4-2, extra information (Chapter 3) was added using a HMD. All types of information from the robot were visualized with a virtual robot arm. This virtual robot arm was rendered over the real robot arm using the transformation matrix calculated by the marker tracker. This virtual robot was used to show the state and intentions of the real robot arm.

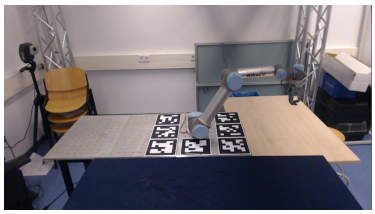
A virtual representation of the robot arm was chosen, because of human spatial reasoning. By spatially relating information to physical objects and locations in the real world it can help the human improve SA [59, 60]. The virtual arm was a direct copy of the real version and had the following properties that can be changed to display specific information.

Color: the color of the arm could be changed to tell something about the robot. Matsumaru uses colors to distinguish between various intended positions of his mobile robot [14], because color is an important property of human observations. Humans can clearly distinguish between colors and colors invoke response in humans, for example warm colors (red) are more arousing then cool colors (blue) [61, 62]. Therefore various colors were used to signal information to the human. Figure 4-17(a) to Figure 4-17(d) show various colors of the virtual robot arm.

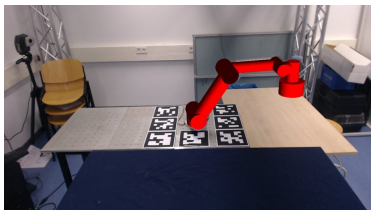
Transparency: by changing the transparency the attention of the human could be grabbed.

A more transparent arm could indicate less important information, but can also be used to make the arm flash by gradually changing between high and low transparency. Figure 4-17(e) to Figure 4-17(h) show various transparency levels of the virtual robot arm.

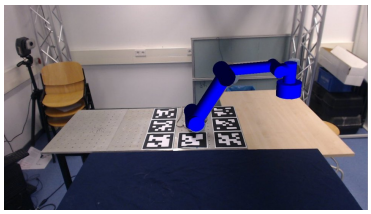
Position: the position of the virtual robot arm could be changed to indicate relations to the real robot arm. By having a virtual model at a different position than the real robot arm the human could infer a relation, for example intended goal position. Humans relate to spatial positions of object and that can help them reason about the environment [59, 60]. Figure 4-17(i) and Figure 4-17(j) show the virtual robot arm at different positions.



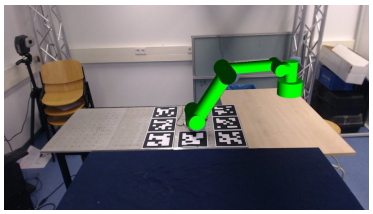
(a) No virtual robot arm.



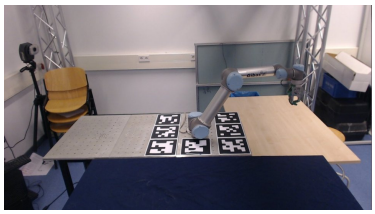
(b) Red virtual robot arm.



(c) Blue virtual robot arm.

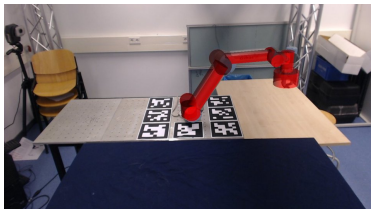


(d) Green virtual robot arm.

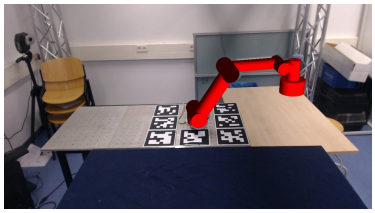


(e) Full transparency.

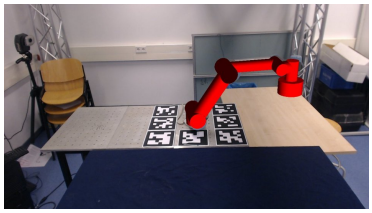
(f) High transparency.



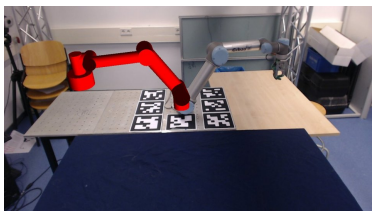
(g) Low transparency.



(h) No transparency.



(i) Virtual arm projected over real robot arm.



(j) Virtual arm projected at a different position.

Figure 4-17: Various rendering principles for the virtual robot arm. Figure 4-17(a) to Figure 4-17(d) show various colors of the virtual robot arm, Figure 4-17(e) to Figure 4-17(h) show various transparency levels of the virtual robot arm and Figure 4-17(i) and Figure 4-17(j) show the virtual robot arm at different positions.

4-4-3 State and intention encoding

In Chapter 3 was shown that various types of information have a high potential for improving SA when working with an industrial robot arm. To answer the research question only two of these were chosen to be research further: internal state, intended goal. From the pilot study was knows that implementation of showing the intended trajectory was not beneficial, but it was implemented, therefore this section also describes how intended trajectory was shown. Each of these types was shown in the HMD using a virtual robot arm, so in total three virtual robot arms were rendered, which could be enabled and disabled individually to show only the needed virtual robot arm. This section describes the functions of the virtual robot arms and where they get the data from.

Virtual model for state information

Figure 4-18 shows the inputs of the virtual model which are needed to display *internal state* information.

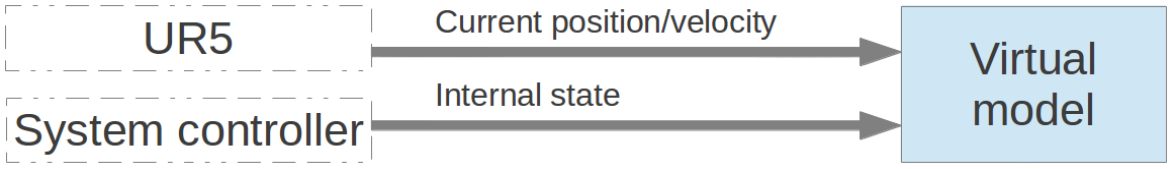


Figure 4-18: Inputs to the virtual model to display *internal state* information. The real robot arm, using the *UR5 driver*, provided the real-time position and velocity. The *control node* set the *internal state* of the virtual robot arm.

In the case of *internal state* information, the virtual model was rendered directly over the real robot arm at all times. This way when the virtual model changed color, it looked like the real robot arm changed color. Because it is rendered directly over the real robot arm, the real time position of the real arm must be known. These positions were provided by the real robot arm and were updated in real time at $100Hz$. The information about the *internal state* of the robot was provided by the *control node*. This node gave a signal when the real robot arm changed *internal state* to trigger a change in the virtual model and was either transparency or color of the arm.

Virtual model for intended goal information

Figure 4-19 shows the input of the virtual model which are needed to display *intended goal* information.

Figure 4-19: Inputs to the virtual model to display *intended goal* information. The *motion planner* provided the joint angles to indicate the final goal position of the robot arm.

The *intended goal* information was used to show where the real robot arm was going to next. This information was provided by the motion planner and was available at the moment the real arm started moving.

Virtual model for intended trajectory information

Figure 4-20 shows the input of the virtual model which are needed to display *intended trajectory* information.

Figure 4-20: Inputs to the virtual model to display *intended trajectory* information. The *motion planner* provided the joint angles for the whole intended trajectory of the robot arm.

The *intended trajectory* information was intended to show the whole future motion of the robot arm. This information was provided by the *motion planner* and consisted of a matrix with joint angles of every point in time. This way the trajectory could be simulated by rendering the virtual arm every time step. Because the motion planner outputted a sparse set of positions, they needed to be interpolated to result in a smooth display of the trajectory. This interpolation was done by linear interpolation to get double the amount of points. The time needed to display the trajectory can be varied by changing the loop rate of the display loop and resulted in faster or slower playback of the trajectory.

Chapter 5

Experiment

With the designed system of Chapter 4 an experiment was done to validate the previously formulated hypotheses. These were:

- Visualizing the internal state of the robot arm using a head-mounted display reduces the reaction time of the human co-worker when the robot arm changes internal state.
- Visualizing the intended goal position of the robot arm using a head-mounted display reduces the reaction time of the human co-worker when reacting to a specific destination position of the robot arm.

The test subjects were asked to work in the robot working cell to repack oranges and while wearing a HMD on which virtual information was shown. The experiment was done to test if the added information is beneficial to the human, in accordance with the hypotheses. Two within-subject experiments were done, one experiment to test each of the hypotheses. A within-subject experiment compares measurements done on individual test subjects treated with two or more test conditions. Because reaction times were chosen to be measured, and there is a lot variability in reaction ability between people, it is better to do a within-subject experiment. In the first experiment information about the internal state of the robot was shown, in the second experiment information about the intentions of the robot was shown. Each experiment was done once with, and once without virtual information.

Section 5-3 explains the experiment done to test if showing internal state information using a HMD helps the human improve SA and Section 5-4 explains the experiment to test if showing intention information helps improve SA.

5-1 Eye-hand coordination limitations

From the pilot study was known that while wearing the HMD eye-hand coordination was difficult. Because a VST HMD was used, the whole scene had to be digitized. This introduced lag between the real world and the images shown on the display. The HMD reduced the FOV of the test subjects from about 178° to about 40°, because of the limited FOV of the cameras

and displays. The fidelity of the colors, auto correction of brightness levels and the reduced resolution of the images were also limiting to the test subjects. And finally there was a mismatch between the position of the eyes and the cameras and resulted in objects being shown further away than they in reality were. All these factors resulted in an impairment of eye-hand coordination.

To still be able to pick up oranges and put a sticker onto them, the test subjects were allowed to look under the HMD. While wearing the display, the screens did not fully cover the FOV of the test subjects, only the top 60%. This meant that a direct view of the hands was possible, which enable normal eye-hand coordination while wearing the HMD. When looking under the HMD during the experiments they had normal eye-hand coordination, while still being able to seen the virtual robot arm.

5-2 Participants

A total of 10 test subjects were tested (power analysis shown in Section 6-1). All but two were students or faculty staff of Delft University of technology. Each test subject did the two experiments, each with and without virtual information shown in the HMD. Test order and starting conditions were randomized.

Before starting the experiment, the test subjects were asked to calibrate the HMD. Because the HMD had to be worn by various people and because everybody is different, it had to be adjusted to the individual test subject. This calibration was done to ensure that the test subjects had a clear 3D picture without having to focus their eyesand was done before the experiment by looking at a marker on the table that showed the virtual robot arm. The test subject could adjust the alignment between the left and right image by pressing the arrow keys on the keyboard. By pressing the arrow keys the image of the left eye was moved one pixel at a time. The test subjects could adjust until a good 3D picture was visible.

After calibration, the test subject were asked to read the instructions (Appendix D). The test subject were asked to stand in front of the *shared working area*, take on orange from the *input* box, put a sticker on it and place it on the grid of the *shared working area*. Depending on the experiment, the test subjects had to react to *internal state* or *intended goal* information from the robot.

All test subject had about 5 minutes of training time get familiar with the task and robot arm.

5-3 Experiment 1: internal state information

As stated in Section 3-2 *internal state* information has a high potential to improve SA. Two *internal state* changes were identified: *waiting for user* and *waiting for oranges*. Because the *waiting for user* state occurs more often, that was chosen was the *internal state* change to react to. This internal state change happened when the robot was waiting for the human to switch a filled box, therefore the robot arm was not moving and the human needed to judge what was meant by that.

When the *internal state* change occurred, the test subjects were presented with a color and transparency change of the virtual robot arm. The transparency was changed to make it pulsate and thus attract the attention of the human. Figure 5-1 shows what the test subjects saw when looking at the robot arm through the HMD with and without virtual information.

(a) No virtual robot arm.

(b) Green virtual robot arm indicating an internal state change.

Figure 5-1: Figure 5-1(a) shows what the test subjects saw when looking at the robot through the HMD without virtual information and Figure 5-1(b) shows the view with virtual information. The *internal state* change was shown by changing the color and transparency of the virtual robot arm.

5-3-1 Procedure

In this experiment, pick and place of oranges, the test subjects and robot arm were working together to fill the boxes with oranges. The robot arm was picking up the oranges from the table and placing them in the *output* box. The test subjects were placing the oranges on the grid of the shared working area and putting a sticker on them.

When the robot had filled the *output* box with a random number of oranges (1, 2 or 3 to eliminate predictability) the box was regarded as full and needed to be switched by the test subjects. When a box was full the robot waited for user input and thus changed state to *waiting for user*. When the test subjects would notice the state change, the box switch was simulated by pressing the space bar on the keyboard. After the space bar was pressed, the robot arm would continue handling oranges. Algorithm 3 and 4 describe the procedure for the test subject and the robot arm.

Algorithm 3 Procedure human worker

```
repeat
    Take orange from input box
    Put sticker on orange
    Put orange on grid of the shared working space
    if output box = full then
        Press space bar on the keyboard
    end if
until 5 internal state changes
```

Algorithm 4 Procedure robot arm

```
repeat
    if Detected oranges > 0 then
        Move to orange
        Pick up orange
        Put orange is output box
        if Output box = full then
            Wait for user input
        end if
    else
        Wait for oranges
    end if
until forever
```

5-3-2 Measurements

In this experiment the independent variable was the presence of virtual information. Each test subject did the experiment once with virtual information and once without virtual information. To keep the test conditions between the experiments as equal as possible, both experiments are done while wearing the HMD.

During the experiment the time between the moment the robot changed *internal state* and the moment the test subject pressed the space bar was measured. This was a measure for the reaction time to a state change of the robot arm. The experiment was done until 5 state changes occurred.

5-4 Experiment 2: intended goal information

As stated in Section 3-2 *intended goal* information has a high potential to improve SA. While handling oranges the robot arm could move to various positions and depending on the *intended goal* position, different actions by the humans could be required.

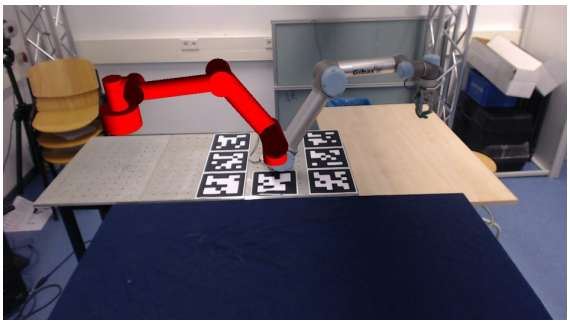
The *intended goal* of the robot arm was shown on the HMD by changing the position of the virtual robot arm (Figure 5-2).

5-4-1 Procedure

In this experiment, modeled as sorting of oranges, the test subjects and robot arm were working together to sort oranges and fill boxes. The robot was picking up the oranges and placing them in one of the *output* boxes. It randomly places an orange in left or right *output* box. The test subjects were placing the oranges on the grid of the shared working area and putting a sticker on them.

When the robot arm would move to the left *output* box, the test subjects needed to take action. When the test subject would notice the movement to the left, this action was simulated by pressing the space bar on a keyboard. Algorithm 5 and 6 describe the procedure for the test subject and the robot.

(a) No virtual robot arm.



(b) Virtual robot arm shown at a different position indicating intended goal of the robot arm.

Figure 5-2: Figure 5-2(a) show what the test subjects saw when looking at the robot through the HMD without virtual information and Figure 5-2(b) shows the view with virtual information. The *intended goal* of the robot arm was shown by changing the position of the virtual robot arm.

Algorithm 5 Procedure human worker

```
repeat
    Take orange from input box
    Put sticker on orange
    Put orange on grid of the shared working space
    if Robot moves to left goal position then
        Press space bar on the keyboard
    end if
until 5 positional changes
```

Algorithm 6 Procedure robot arm

```
repeat
    if Detected oranges > 0 then
        Move to orange
        Pick up orange
        Randomly determine drop position
        Put orange in output box
    else
        Wait for oranges
    end if
until forever
```

5-4-2 Measurements

In this experiment the independent variable was the presence of virtual information. Each test subject did the experiment once with virtual information and once without virtual information. To keep the test conditions between the experiments as equal as possible, both experiments are done while wearing the HMD.

During the experiment the time between the moment the robot arm would start moving to the left goal and the moment the test subject presses the key was measured. This was a

measure for the reaction time to an *intended goal* change of the robot arm. The experiment was done until 5 oranges were moved to the left *output* box.

5-5 Evaluation

Besides the reaction times (performance metrics) measured during the experiments, all experiments were recorded using a video camera for analysis. The video data was used to count the number of oranges handles by the test subjects. The number of oranges handled per minutes is a measure for the task performance of the human. The verbal remarks of the test subjects during the experiment were also recorded on the video.

Chapter 6

Results

This chapter lists the results gathered from the experiments. Section 6-1 explains the statistical method used to analyse the data gathered from the experiment. Section 6-4 explains the video analysis done. Section 6-2 and 6-3 show the analysed results of the state and goal experiment.

6-1 Statistical analysis

The experiments were spread out over 2 days. On the first day 3 test subjects were tested and the results from those experiments were used to determine a sample size using G*Power3 [63]. The calculation of the sample size was based on the significance level (α), statistical power ($P = 1 - \beta$) and effect size. A significance level of 5% and a statistical power of 80% were chosen [22]. This analysis resulted in a sample size of 8, but a total of 10 test subject were tested to take measurement errors and equipment failure into account.

Every test subject did two experiments with each two conditions, for which 5 data points were gathered. The 5 data points were averaged to get one average per test condition and test subject. Because a within-subject experiment with two conditions was done, a paired t-test using Matlab 2013b was used to analyse the data. Appendix F-1 shows the Matlab code used to analyse the data.

6-2 Experiment 1: internal state information

These results of this experiment show an significant difference between the reaction times of the test subject with and without the presence of virtual information. Table 6-1 shows the results of the internal state experiment. Figure 6-1 shows a boxplot of the date of Table 6-1. The value for each subject is the average of 5 measurements. All test subjects show an improvement in reaction time, which range from almost nothing (0.10 seconds) to a couple of seconds (2.37 seconds). The average reaction time without internal state information

is 2.49 seconds and with internal state information it is 1.55 seconds. Also the standard deviation decreases with state projection. It decreases from 0.75 seconds to 0.44 seconds. These measurements were significant with a p-value < 0.01.

	Reaction times [s]		
Subject	**w/o virtual information**	**w/ virtual information**	**difference [s]**
1	3.352	2.336	1.016
2	3.684	1.308	2.376
3	2.240	2.142	0.098
4	2.876	1.312	1.564
5	1.774	1.258	0.516
6	2.328	1.318	1.010
7	1.624	1.136	0.488
8	3.266	2.152	1.114
9	1.836	1.704	0.132
10	1.952	1.554	0.398
Mean	2.493	1.622	0.871
SD	0.745	0.438	0.706

Table 6-1: Reaction time results of the experiment 1: internal state information. 10 test subjects did the experiment with each two test conditions: with and without the presence of virtual information. For each condition 5 reaction measurements were gathered and averaged. The difference in average reaction times is significant with a p-value < 0.01.

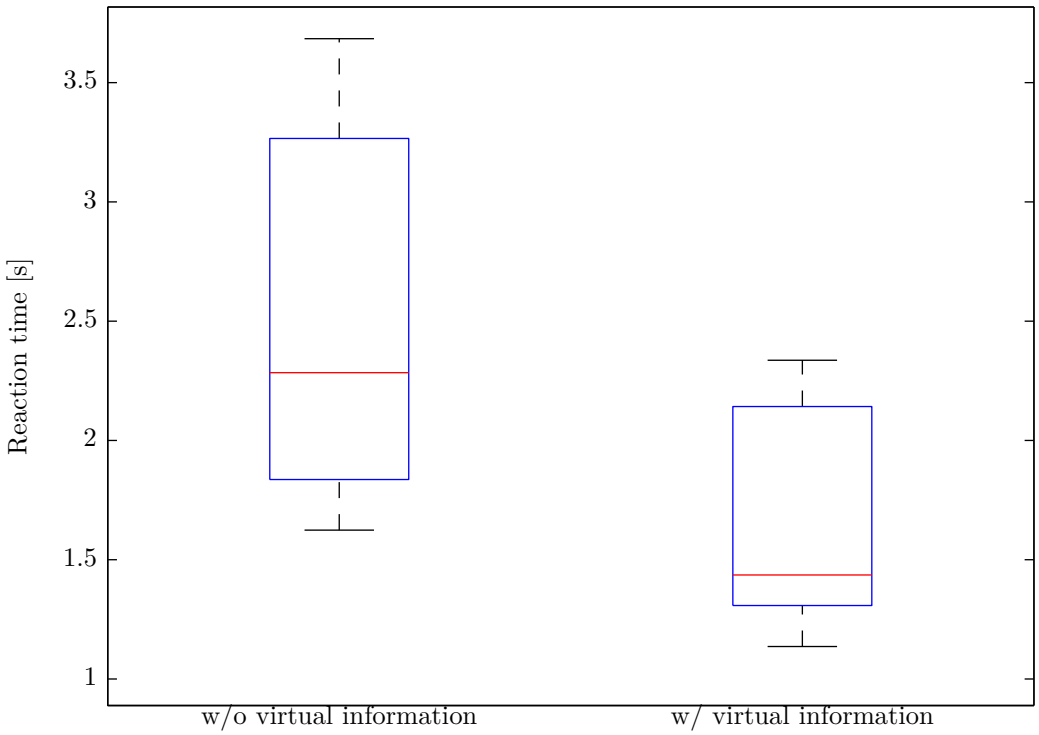


Figure 6-1: Boxplot of the data from Table 6-1. The left boxplot shows the reaction times when the test subjects were not presented with virtual information and the right boxplot shows the data when the test subject were presented with virtual information. The difference between the two means is significant with a p-value < 0.01.

6-3 Experiment 2: intended goal information

Table 6-2 shows the results of the intended goal experiment. Figure 6-2 shows a boxplot of the date of Table 6-2. The value for each subject was the average of 5 measurements. All test subjects show an improvement in reaction time, which range from almost nothing (0.03 seconds) to 0.77 seconds. The average reaction time without goal projection was 1.68 seconds and with goal projection it was 1.42 seconds. Also the standard deviation decreases with goal projection decreased from 0.46 seconds to 0.31 seconds. These measurements were significant with a p-value of 0.013.

	Reaction times [s]		
Subject	**w/o virtual information**	**w/ virtual information**	**difference [s]**
1	1.104	0.934	0.170
2	1.824	1.222	0.602
3	1.364	1.298	0.066
4	2.840	2.066	0.774
5	1.554	1.148	0.406
6	1.486	1.420	0.066
7	1.568	1.526	0.042
8	1.984	1.612	0.372
9	1.414	1.368	0.046
10	1.608	1.574	0.034
Mean	1.675	1.417	0.258
SD	0.475	0.308	0.267

Table 6-2: Reaction time results of the experiment 2: intended goal information. 10 test subjects did the experiment with each two test conditions: with and without the presence of virtual information. For each condition 5 reaction measurements were gathered and averaged. The difference in average reaction times is significant with a p-value of 0.013.

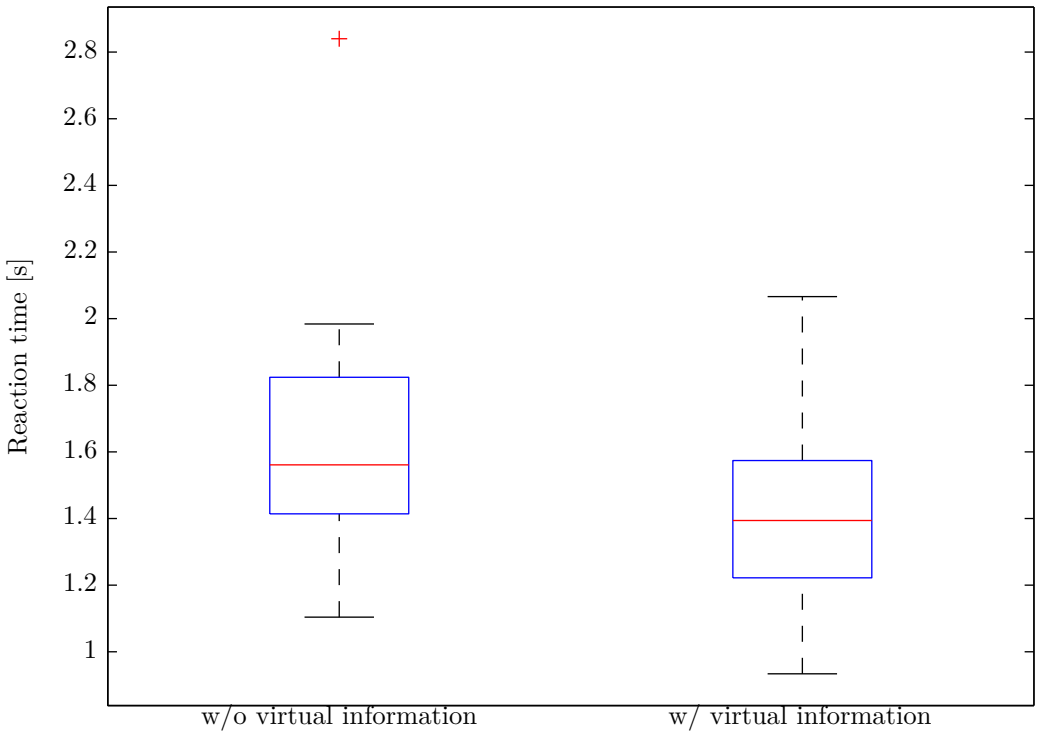


Figure 6-2: Boxplot of the data from Table 6-2. The left boxplot shows the reaction times when the test subjects were not presented with virtual information and the right boxplot shows the data when the test subject were presented with virtual information. The difference between the two means was significant with a p-value of 0.013.

6-4 Video analysis

The recorded videos were used to analyse the orange handling performance of the test subjects and observe the test subjects while they were doing the experiment.

All test subject were initially reserved during the first minutes when working in the robot working cell. They showed signs of hesitation while handling oranges, because they were watching the robot arm. An improvement in orange handling speed was shown after some time (Figure 6-5). Occasionally a collision between human and robot would occur. The test subjects that had a collision with the robot arm stopped working for a second and then continued working. During the experiment the robot arm would sometimes make a strange, unpredictable and fast movements due to bad motion planning. At those times the test subject would stop working, do a step back and look at the robot.

6-4-1 Orange handling

Figure 6-3 and 6-4 show the number of oranges handled by the test subjects. No significant difference with of without internal state projection was measured in the amount of oranges handled by the test subjects ($p = 0.20$). Also no significant difference between the amount of oranges handled with of without goal projection was measured ($p = 0.38$).

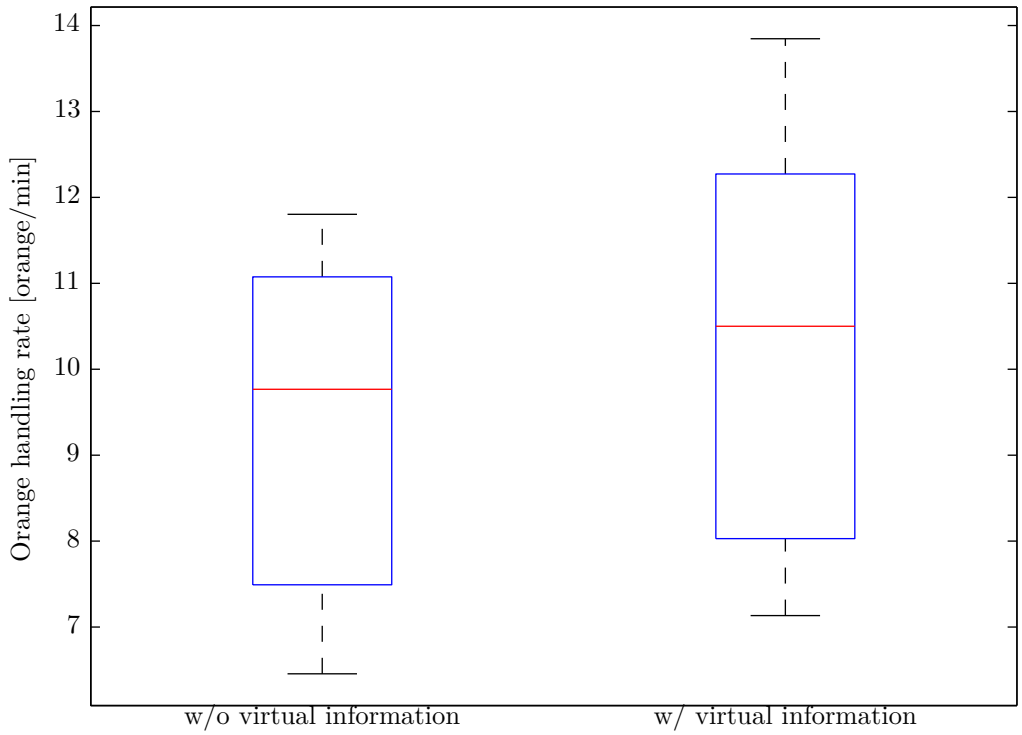


Figure 6-3: Boxplot of the orange handled per minute [oranges/minute] during the internal state information experiment. The left boxplot shows the oranges handled per minute without virtual information and the right boxplot shows the number of oranges handled with virtual information presence. There was no significant difference between the means ($p = 0.20$).

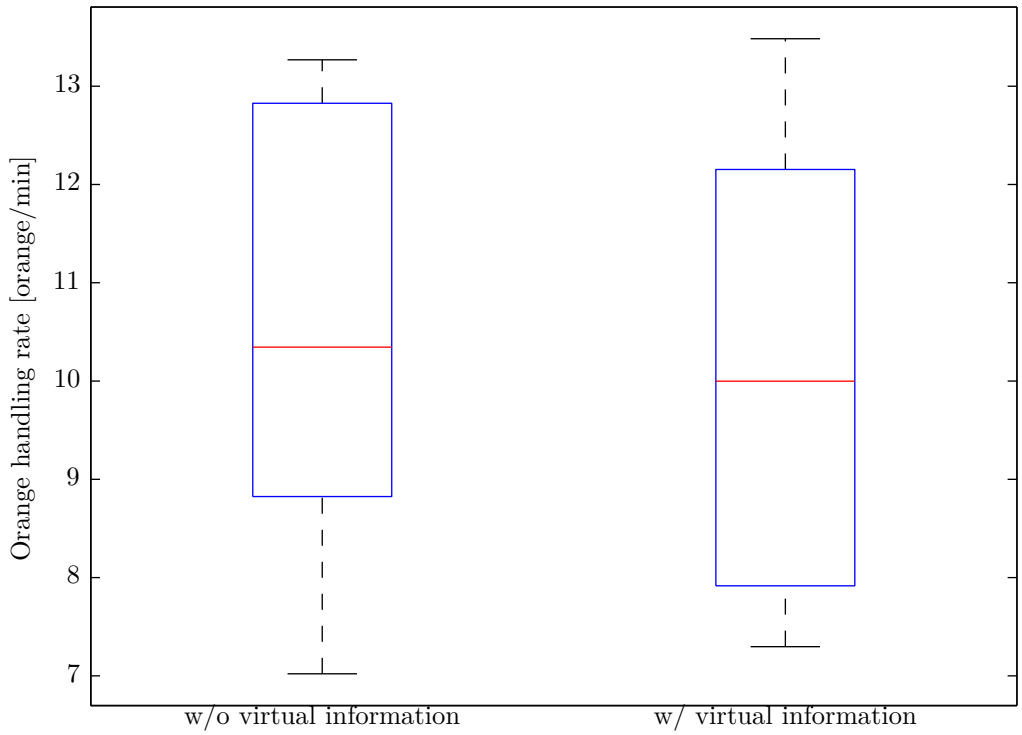


Figure 6-4: Boxplot of the orange handled per minute [oranges/minute] during the goal information experiment. The left boxplot shows the oranges handled per minute without virtual information and the right boxplot shows the number of oranges handled with virtual information presence. There was no significant difference between the means ($p = 0.38$).

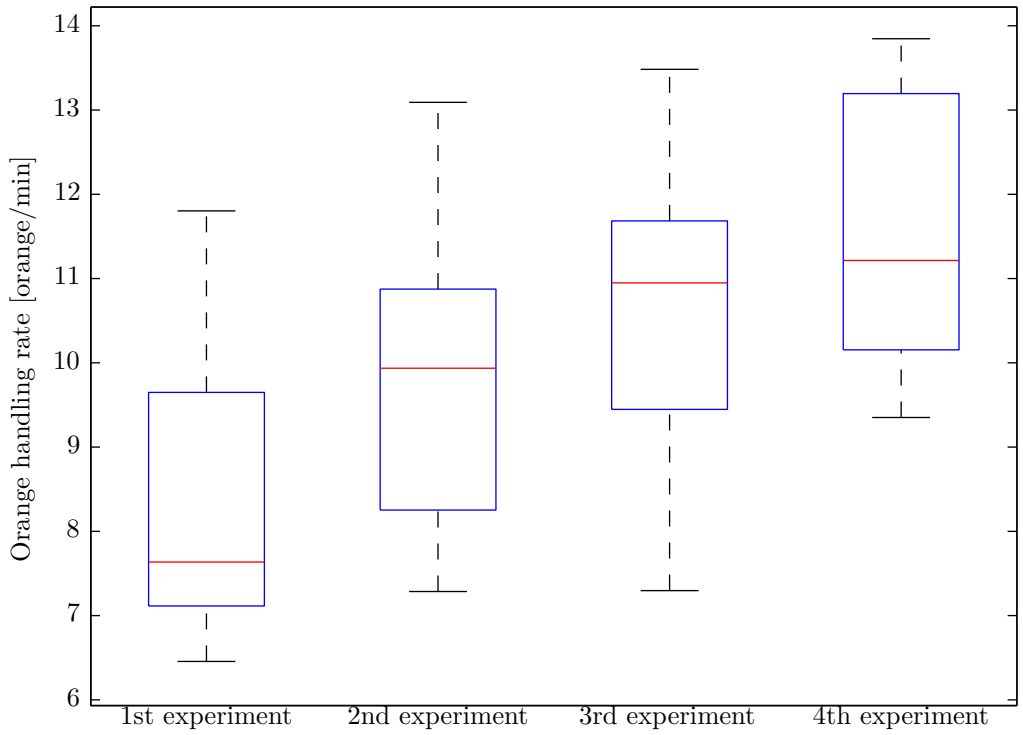


Figure 6-5: Boxplot of the number of oranges handled per minute [oranges/minute] for the first, second, third and forth experiment. Because the experiment order was randomized, the data is grouped into the order of the experiments. This graph shows a learning curve, the more the test subjects did their task, the better they were able to do it.

Chapter 7

Discussion

As stated in Chapter 1 the research question was formulated as:

Does showing state and intention information of an industrial robot arm, using a head-mounted display, help the human co-worker to improve situational awareness?

In this chapter an evaluation of the robot working cell (Section 7-1), Head-Mounted Display (HMD) (Section 7-2) and experiments (Section 7-3) is given. These evaluations are then discussed to answers the research question.

7-1 Robot working cell

A robot working cell was designed to be able to test human-robot interaction. A mock-up of an orange handling line was chosen, which consisted of a Universal Robots UR5 arm that, in combination with a vision algorithm, could pick up oranges and put them in a box. The human co-worker would handle the oranges and put them on the shared working area so that the robot arm could pick them up. The designed robot working cell was a good way to test human-robot interaction, because it forcing humans and robots to work in the same working space to accomplish a shared task. This way the strengths of both the robot and the human could be used to make the task execution more efficient. Humans did the complex and dexterous tasks and the robot arm the repetitive tasks.

When working with the robot the test subjects were initially reserved. The robot was new to them and were not really sure what it was going to do. But after working with the robot for a couple of minutes, they felt more comfortable and dared to be closer to the robot. The test subjects also didn't keep a close eye on the robot arm after a while, because they were building an internal representation of the environment and robot arm, they were gaining better SA. This internal representation made the robot predictable so that the test subjects felt more comfortable. Video analysis showed that sometimes the robot would make a strange, unpredictable and fast movements due to bad motion planning. At those times the test subject would stop working, do a step back and look at the robot. The robot

became unpredictable and the test subjects felt uncomfortable. This behavior showed that predictability is important when working with a robot.

Occasionally a collision between a test subject and robot would occur. The test subjects that had a collision with the robot arm stopped working for a second and then continued working. This shows that the robot is intrinsically safe.

To work together with a human, the robot system doesn't have to be engineered to work 100% reliable, because the human can take over tasks that are too complex or take too long to implement on the robot. A human can even work together with multiple robot arms. All robot arms can inform the human when help is required. In the case of the designed orange repacking setup, the human works faster than the robot arm, so multiple robot arm could be attended to. This way a strength of robots, continues operations, could be exploited.

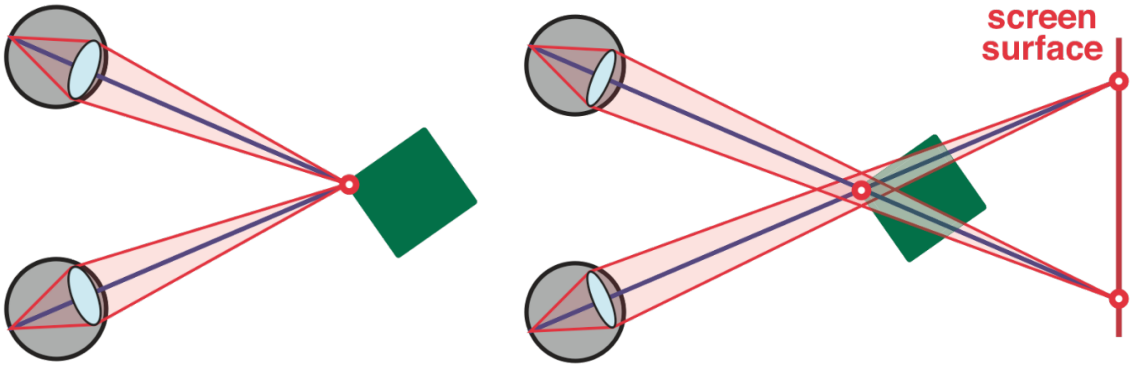
7-2 Head-mounted display

When working with a robot, information sharing is important and various solutions were presented in literature to facilitate this information sharing. This thesis showed that a HMD with the use of AR can be used to show information from an industrial robot arm to the human. The combination of a HMD and AR allows for real-time virtual information to be shown directly in front of the test subject's eyes. This way the test subjects could keep their attention focused on their own task and the robot arm, without being distracted by having to switch attention. Showing information about the robot could potentially be done by more simple solutions, for example a light or buzzer, but as the robot system and the interaction becomes more complex, also more complex information needs to be delivered to the human. In those cases a simple light or buzzer doesn't have the versatility to encode the increased information load. By using a HMD more complex information could be shown to the user.

The fiducial marker based tracker used by ARToolKitPlus provided good real-time tracking capabilities. It was robust, fast, accurate and allowed the virtual robot arm to be displayed over the real robot arm. A drawback of using markers is that they have to be in the field of view of the cameras, therefore if the markers were occluded or totally not visible, the tracker couldn't calculate the head pose of the user and virtual information couldn't be displayed. In the experimental setup this doesn't pose a big problem, because it was designed in such a way that the human and robot were always in close proximity to each other. But when using this tracking in unknown and unstructured environments, this could pose a problem. This is because prior placement of markers is needed to enable tracking and object recognition. This could potentially be solved by using natural feature tracking and model based object recognition.

Unfortunately there were some limitations with using the HMD in this thesis, because a VST HMD was used a direct view of the world was impaired. Instead the world was viewed via a pair of displays. Because there is one display of each eye, stereoscopic 3D view was still possible, but with a limitation. The human has to accommodate to the screen because that is where the light is coming from, but at the same time converge to the virtual object, that might be in front of the screen [64] (Figure 7-1).

This mismatch between convergence and accommodation could potentially cause discomfort in the form of headaches. This means that long term use of a VST HMD is not possible.



(a) Natural viewing: eyes accommodated and converged at the same distance.

(b) Stereoscopic 3D viewing: eye accommodated and converged at different distances.

Figure 7-1: The vergence-accommodation conflict in stereoscopic displays. Figure 7-1(a) shows natural viewing vergence and accommodation are to the same distance. Figure 7-1(b) shows the steroscopic3D case, vergence and accommodation must be decoupled for the viewer to have clear, single binocular vision.

Other limitation were that the FOV, resolution and color clarity are worse than when we view the world directly. So by wearing the HMD a lot of information is lost to the human. Another limitation is that the cameras are mounted about $5cm$ in front of the displays, so objects viewed through the HMD would appear father away than they in reality are. This combined with the lag introduced by the finite computational power of the computer, limits eye-hand coordination. Also the HMD is tethered to a computer using sturdy cable, which limits movement flexibility. These limitations were partially solved by looking under the HMD, to enable more natural eye-hand coordination (Section 5-1).

7-3 Virtual information and experiments

To research if the use of HMD in combination with AR could improve SA when working with an industrial robot to repack oranges, an experiment was done. Information to potentially improve SA was analysed, it was found that both state and intention information could improve SA of the human. When working with an industrial robot particular information can't be directly observed and level 2 and 3 SA information is therefore difficult to infer. Decent comprehension about the robot arm is needed to project future events. By helping the human by providing extra information, that were not directly observable, could help the human gain a better comprehension of the robot arm. From analysis was shown that there are two main categories of information: state and intention.

In the category state the information types that could potentially improve SA of humans were: position, speed, action and internal state. Of those only *internal state* had a high potential of improving SA, because internal state can't be directly observed and had to be inferred by looking at the environment and task. This indirect observation of internal state requires a good comprehension of the tasks to make sense of the observation. By showing a virtual model of the real robot arm to indicate a change in internal state, the test subjects

were able to detect this change faster. The experiment showed that a reduction in reaction time was measured, this is because of the direct observation of an indirect phenomenon. The hypothesis

Visualizing the internal state of the robot arm using a head-mounted display reduces the reaction time of the human co-worker when the robot arm changes internal state.

has to be accepted at a significance level < 0.01.

From the other category, intentions, the extra information that could potentially improve SA was: intended position, intended trajectory, intended action and intended internal state. This is again because the information provided can't be directly observed and requires decent comprehension about the environment and task. To validate the following hypothesis:

Visualizing the intended goal position of the robot arm using a head-mounted display reduces the reaction time of the human co-worker when reacting to a specific destination position of the robot arm.

only *intended goal position* was chosen to be implemented and tested. This hypothesis had to be accepted at a significance level of 0.01. The virtual information was shown at the same time as the movement of the robot begins. The test subjects had to react when they noticed that the robot would be moving to an alternative direction. This could be directly observed, because the robot was moving, so showing the intended goal of the robot arm at the same time as the start of the movement was not really useful to the human. It would be better to know the *intended goal position* a couple of seconds beforehand. Further research has to be done to determine the optimal time between the cue being shown and the actual movement of the robot. If it is too short the human doesn't have enough time to react and when it's too long, the human would already have forgotten it. The goal position of the robot arm could be useful to know when it indicates for example which object it is going to pick up.

It was shown that showing information about the *internal state* and *intended goal position* using a virtual model shown in a HMD improves SA of humans when compared to the case where no virtual information was shown. To generalize the conclusion, further research has to be done to check if the implemented method behaves differently that when not wearing a HMD or when other visualization methods are used.

Another potential useful type of information is the trajectory of the intended motion. This was not implemented in this experiment, because for a pick and place task of the robot it was not useful. This is because the motion of the robot arm was limited and most of the time the working area of the robot arm was free of obstacles. In the case of for example programming an industrial robot knowing the trajectory could be useful. By using a virtual representation of the robot arm and a HMD a more intuitive interaction between the human, robot arm and motion plan could be gained. This way the programmer can directly see the motion plan of the robot arm in the real environment, without using the real robot setup [39]. By not using the real robot setup, an increased safety and working speed can be achieved [65]. When working with mobile robots it could also be beneficial to know the trajectory of the robot [15].

When the test subjects were working with the robot arm and presented with the virtual information, there was no significant difference in the amount of oranges that were handled.

Chapter 8

Conclusion

A human-robot interaction experiment was done to test if showing virtual information about the state and intentions of an industrial robot arm using a head-mounted display could improve situational awareness. From each category, one information type with a high potential to improve situational was implemented and explored. It was found that showing information about the internal state of the robot using a head-mounted display is beneficial to the human to improve situational awareness, because the information shown can't be directly observed. Compared to the case without virtual information, the results showed a decrease in reaction times to state and intention information when presented with the virtual information. This decrease in reaction time was a measure for an increased situational awareness of the human and showed that situational awareness could be improved by using a head-mounted display to show extra information about the robot arm. The added internal state information was information about a non directly observable state change, so new information was added. In the case of showing intended goal information of the robot arm this information was extra, because the intended goal position could be inferred by looking at the initial movement of the robot arm. It would therefore be more beneficial to shown the intended goal information a couple of seconds before the robot arm started moving as to increase the difference between the two cues. This is because the human needs to be able to make a decision and react to the information.

8-1 Future work

Real-world robot working cell. The problem with human-robot interaction in an industrial setting is that there are almost no real world examples. Intrinsically safe robot arms became wide spread only a couple of years ago, so they aren't used often. Once more scenarios are created to let humans and robot work together, a more focussed and specific analysis can be done to improve this human-robot interaction. Robot can't only be used to do pick and place tasks, but can also support humans by lifting heavy objects during assembly or even work together with a human to assemble simultaneously. In those cases maybe other types of information have a higher potential to improve situational awareness.

Improve the Head-Mounted Display (HMD). A lot of the limitation with the HMD can be solved by switching to a different HMD. Optical See-Through (OST) HMD (Figure 8-1) don't have a lot the of drawbacks of VST HMDs, because they offer a direct view of the world. This makes them a good solution to be used in an industrial setting, because eye-hand coordination is not impaired and they can be used for a longer period of time. OST HMD are becoming more affordable and mainstream, for example the Epson BT-100 (Figure 8-1(a)) or even the Google Glass (Figure 8-1(b)). The Epson doesn't have cameras mounted on them, the Google Glass does, but by adding cameras to them, these can be used to facilitate AR. Because the whole scene doesn't have to be digitized, a lower bandwidth is needed to analyse the camera images. This lower bandwidth requirement makes the display wireless and therefore could greatly improve mobility of the user. Computation could then be done on a hand-held device, like a mobile phone.

(a) Epson BT-100 OST HMD. (b) Google Glass.

Figure 8-1: Figure 8-1(a) shows the Epson BT-100 OST HMD and Figure 8-1(b) shows the Google Glass.

Research virtual information display methods. In this thesis all virtual information was encoded using a virtual model of the robot arm and shown on a head-mounted display. Further research has to be done to gain a more complete insight into what method is best to encode and show information from a robot arm.

Appendix A

Augmented reality system

A Video See-Through (VST) Head-Mounted Display (HMD) was used as a display to view the real world and the virtual information. This appendix describes the hardware and software needed to use AR with a HMD. Appendix A-1 describes the VST HMD hardware and Appendix A-2 describes the tracking and visualization software.

A-1 Head-mounted display hardware

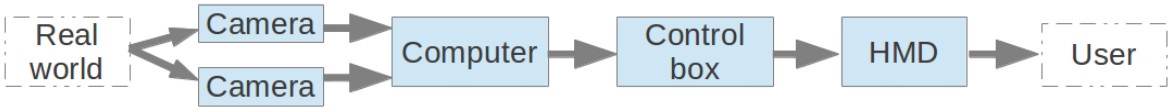


Figure A-1: Date flow of the HMD hardware. The cameras captured the real world and the computer processed the captured images to stitch them together and send them to the control box. The control box split the images to one image per eye, to get a stereoscopic 3D view. That two images were shown on the HMD so that they could be seen by the user wearing the HMD.

Figure A-1 gives an overview of the date flow of the HMD and consists of the following parts:

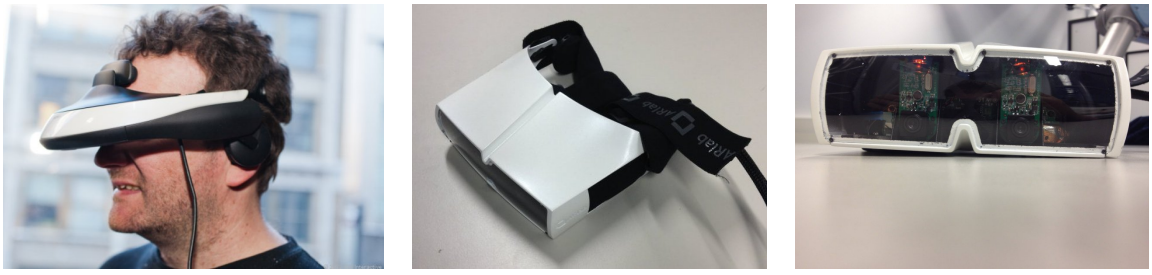
Cameras: the cameras mounted on the HMD captured the real world, one camera captured the image for one eye.

Computer: the computer processed the images captured by the cameras. Appendix A-2 goes into detail about the software used to process the images. The two images from the cameras were combined into one image to enable stereoscopic 3D view (Appendix A-1).

Control box: the processed images from the computer were handled by the control box. The control box was connected to the computer using HDMI or DisplayPort and split the combined images from the computer into two separate images for each eye. The split images were send, using a proprietary cable, to the HMD.

Head-mounted display: the HMD showed the two images on the displays, mounted in front of each eye. Appendix A-1 describes the HMD.

The HMD used was a modified Sony HMZ-T1 (Figure A-2(b)), nicknamed *Marty* and was designed by Niels Mulder. The Sony HMZ-T1 (Figure A-2(a)) is a commercial display for home and personal use. It is worn on the user's head as an alternative to a television, for example to watch a movie or play video games. The enclosure of the HMZ-T1 was stripped and two Logitech C905 webcams were mounted on it. A new enclosure was designed to incorporate the newly mounted cameras, making it a VST HMD. Table A-1 shows the specification of *Marty*.



(a) Original Sony HMZ-T1. (b) Top view of *Marty*. (c) Front view of *Marty*.

Figure A-2: Original (Figure A-2(a)) and modified Sony HMZ-T1 nicknames *Marty* (Figure A-2(b) and A-2(c)).

Parameter	Specification
Number of screens [-]	2
Screen resolution [pixels]	1280x720
Weight [g]	100
FOV [deg]	40
Connector HMD [-]	Proprietary
Connector control box [-]	HDMI/DisplayPort
Dimensions (l x w x h) [cm]	15cm x 15cm x 6cm
Number of cameras [-]	2
Inter ocular distance [cm]	6.5
Camera resolution [pixels]	1280x720
Camera FPS [fps]	30
Camera connector [-]	USB
Length cables [m]	5

Table A-1: Specification of *Marty*. This table shows all the relevant specification of *Marty*.

Stereoscopic 3D

When we view the world, it is seen in 3D, because we have two eyes. Two eyes let us view the world from two slightly different perspectives, which we can transform to a 3D image. When looking at a monitor we miss this stereoscopic effect and can't directly see a 3D image. Because *Marty* has two screens, one in front of each eye, the system can be designed to enable

stereoscopic 3D views. This is done in the control box, the HMD was connected to this control box, which supplies the display with power and visual information. The control box was connected to a computer via HDMI or DisplayPort. When a video source is supplied in the side-by-side or top-bottom format (Figure A-3), the control box split the video into one image for each eye, to enable stereoscopic view. The drawback of this is the loss of resolution. In this thesis sibe-by-side images were used, which cut the horizontal resolution in half.

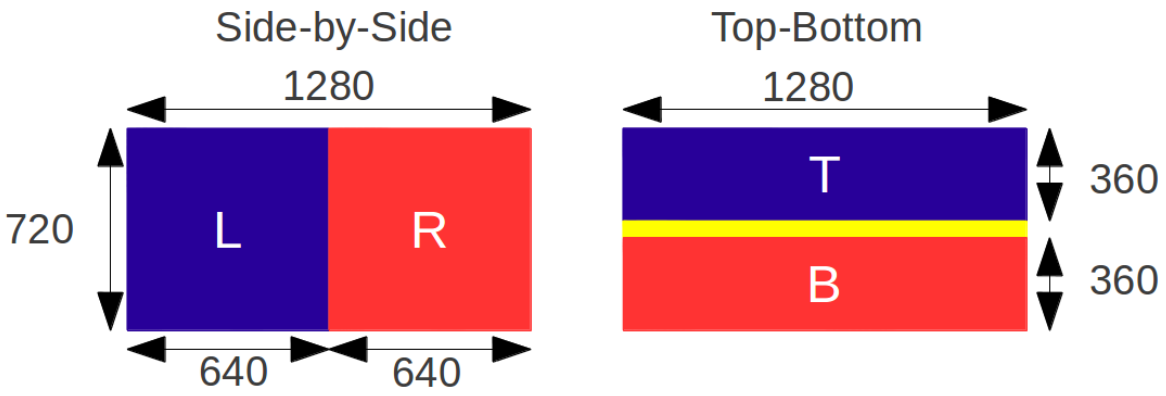


Figure A-3: HMDI format for stereoscopic 3D images with a resolution of 1280x720 pixels. The left part of this figure shows HDMI side-by-side and specifies that the image for the left and right eye are put side by side. This method reduces the horizontal resolution by half. The right part of this figure shows HDMI top-bottom and specifies that the images for the left and right eye are put on top of each other with a spacer in between. This method reduces the vertical resolution by half.

A-2 Augmented reality software

To use the HMD and show the correct information on it, software is needed. When looking at Figure A-1 in Appendix A-1 the computer block takes care of all software required for optical tracking and image processing. This section describes all software components, their function and implementation, to enable virtual information to be shown on the HMD using optical based marker tracking.

Figure A-4 gives an overview of the important steps involved with marker based optical tracking and rendering of virtual objects. The following parts are important:

Image capture: the cameras captured the real world. Because a VST HMD was used, the whole scene needed to be digitized and displayed on the screens of the HMD. The images from the camera needed to be captured and stored in a proper format to be analysed. Appendix A-2-1 describes how the image capture was done.

Marker detection: to be able to track what the user is looking at, markers needed to be detected in the images. From the captured images the markers in the scene were detected. Appendix A-2-2 describes the algorithms used to detect the markers in the image.

Marker identification: the detected markers were not only used to calculate what the user was looking at, but they also represented a specific location in the scene. To be able to use the

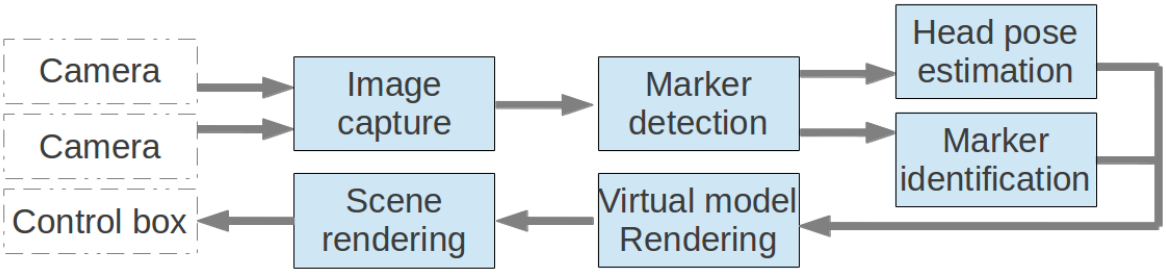


Figure A-4: Data flow of information for marker based optical tracking and virtual model rendering. The images from the cameras need to be captured and stored in a useful format. The images need to be analysed to detect and identify the markers. From these identified markers the transformation matrix between the camera and markers can be calculated. With this transformation matrix the virtual model can be rendered in the scene.

specific location of the marker, it needed to be identified, because different virtual information needed to be shown on different locations. Appendix A-2-2 describes the important parts of marker identification.

Head pose estimation: the most important goal of the marker tracking was to calculate what the user was looking at. From the detected markers the head pose was be calculated, by calculating the transformation matrix between the cameras and the markers. The transformation matrix between the cameras and the markers allow the virtual objects to be rendered in the correct pose. Appendix A-2-3 describes the algorithms used to calculate the transformation matrix.

Virtual model rendering: with the calculated transformation matrix, the virtual model could be rendering with their correct pose. Appendix A-2-5 describes the how the virtual models were rendered.

Scene rendering: the captured images also needed to be rendered and combined with the virtual model so that it could be displayed on the HMD properly. Appendix A-2-4 describes how the scene was rendered and how the images were combined.

A-2-1 Image capture

In order to show the real world on the HMD, the scene needed to be digitized, because VST was used. The software used to capture images from the cameras was OpenCV [57]. OpenCV is a mature and robust toolkit that is used for a wide variety of tasks, for example image manipulation, camera calibration, feature detection, object detection or complex matrix calculations.

OpenCV was used as a wrapper for Video4Linux (V4L) and captured the images at 30 Frames Per Second (FPS) in Blue-Green-Red (BGR) format at a resolution of 1280x720 pixels. Because the images were directly shown on the HMD and viewed by the user, the resolution and frame rate need to be as high as possible. The maximum resolution of the HMD is 1280x720 pixels per eye. To save computation power, the images were not captured at a higher resolution. To be able to see a smooth video feed of the real world, a frame rate of minimal 60 FPS is considered good enough. Unfortunately the cameras have a maximum

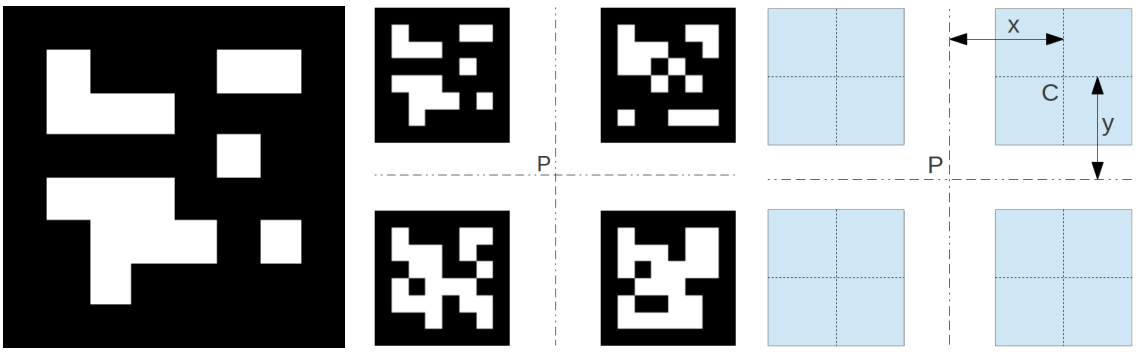
frame rate of 30 FPS, so 30 FPS was used. The images were stored in a 1280x720x3 matrix, with the first image the blue image, the second the green image and the third the red image.

Because of the high resolution and high FPS a lot of bandwidth and computational power is needed to handle the images. Because of the high bandwidth two separate USB2 controllers were needed or one USB3 controller. This ensures that both cameras can be used at a resolution of 1280x720 pixels and 30 FPS.

Not only is capturing images at high resolution bandwidth intensive, but also computationally intensive. A laptop with a quad core Intel i7 was used to capture the images. This CPU is not fast enough to capture two cameras in one thread, therefore two threads were used to capture and analyse the images. By using two threads the synchronization between the two cameras was also improved, because the two images could be captured at the same time, so that when the HMD was moved there was less lag between the left and right image.

A-2-2 Marker detection and identification

From the captures images, markers needed to be detected. This detection was done by ARToolKitPlus [52], an optimized an extended version of ARToolKit [51]. The advantage of ARToolKitPlus over ARToolKit are that it is a complex rewrite and stripped version, making it leaner and more focussed. ARToolKit is a complete solution to marker based AR and has software for video capture, marker detection and virtual object rendering. ARToolKitPlus only has the marker detection part and is completely rewritten in C++. Besides being a complete rewrite, many optimizations and better algorithms are incorporated. The algorithms make the detection faster, robuster and more reliable [52, 66].



(a) Fiducial marker used by ARToolKitPlus.

(b) Marker board with multiple markers to identify a shared central point P.

(c) Diagram of marker board. x and y are the distances from the center of marker C to the shared central point P.

Figure A-5: Figure A-5(a) shows what a marker used by ARToolKitPlus looks like. Figure A-5(b) shows how multiple markers can be used to identify a shared central point and Figure A-5(c) shows a diagram of how a marker board is defined.

The markers (Figure A-5) used in the tracking algorithm were of low complexity, but serve a concrete purpose. All markers were square with fixed and known dimensions. The boarders of the markers had a fixed width and were clearly recognizable from the rest of the scene. The pattern in the center of the marker has an unique combination (4096 different patterns)

of squares and can be decoded to represent a number ID. This way all detected markers could be easily distinguished.

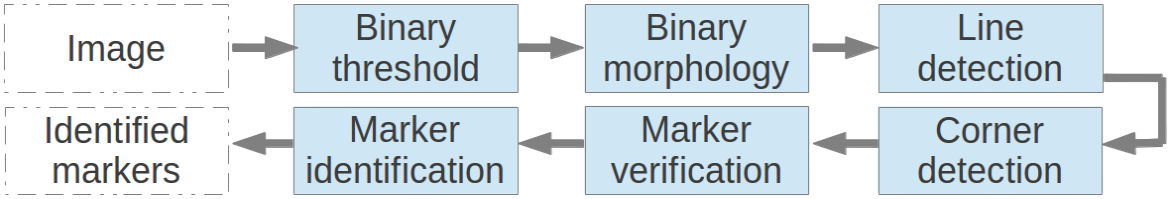


Figure A-6: ARToolKitPlus marker detection overview. The captured images were converted to a binary image. In the binary image, binary morphology was used clean up the image. In the cleaned image, lines were detected and at the cross sections, corner points were detected. The pattern inside marker was used to verify the marker and identification it.

Figure A-6 shows the steps used in the marker detection algorithm to detect markers in the images and consists of the following steps:

Binary threshold: ARToolKitPlus uses binary images to do the detection of markers on. To convert a BGR-image to a binary image a threshold needed to be determined. This BGR-image uses 8 bits per channel (0-255) to represent color. A binary image uses a 1 bit channel to represent either black (0) or white (255). Because the marker consists of a pure black and a pure white part, it survives after the threshold was applied. This threshold served as a divider to set the pixels to either black or white. Because of environmental conditions (lighting and camera quality) a fixed threshold causes problems, therefore an automatic threshold finder algorithm was used. This algorithm makes an educated guess, based on the histogram of the image, to come up with an initial threshold value. When no markers were detected in the image, another educated guess is made to determine the threshold. These steps were repeated a maximum of 3 times, to improve the chance of finding a marker.

Binary morphology: the newly created binary image is cleaned up by binary morphology. Binary morphology uses dilation, erosion, opening and closing to get rid of holes in the images. The result is clean up image.

Line detection: on the cleaned up image, line detection was used to find the edges of the marker. Because the markers are square and have clearly visible border, the detected lines represented the borders of the markers.

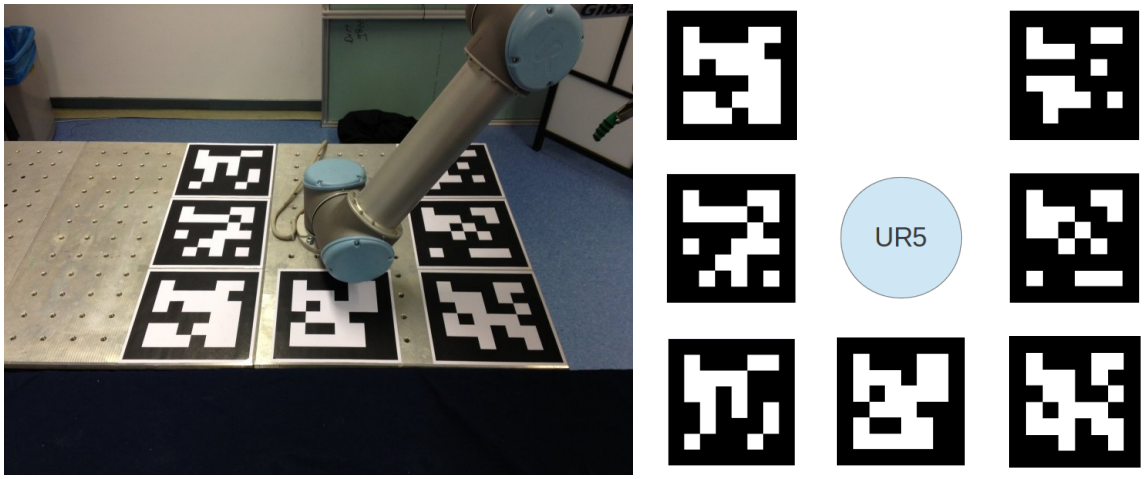
Corner detection: where the detected lines cross, a corner point could be made. This corner point corresponded to a corner of the marker.

Marker verification: when 4 corners were identified, a check must be done to ensure that the area surrounded by the 4 corner points was indeed a marker. Verification was done with the pattern inside the marker. This pattern corresponds to a unique ID and was checked with a Cyclic Redundancy Check (CRC) algorithm.

Marker identification: when a marker is verified, the ID was checked against a database of markers. When a match was found, the ID of the marker was returned.

The marker detection algorithm used by ARToolKitPlus provided a good and fast way to detect a marker in an image and was be used to track the head pose of a human. But there were some limitations and drawbacks to using markers. One big limitation was that the

camera had to have a marker in its FOV to detect it. When the marker is outside the FOV or when it is occluded, no marker was detected. This was solved by using multiple markers defined in a marker board (Figure A-5(b)). A marker board uses a known set of markers in a predefined position and with a known relation to each other. An advantage of a marker board is that only one marker of the whole marker board needs to be detected to know the position of the whole board.



(a) Marker board surrounding the UR5.

(b) Diagram of the marker board.

Figure A-7: Multiple markers defined to represent the same central point was used to overcome the shortcomings of using only one marker. Figure A-7(a) shows the marker board surrounding the robot arm and Figure A-7(b) shows a diagram of the marker board.

To accurately detect a marker in the image, it needed to be of a minimal size. To ensure that the marker is detectable when viewed from about 2 m, a marker size of 0.2 m was used. This means that the marker has a width and a height of 0.2 m.

A-2-3 Head pose estimation

The system needed to track the position of the head in 6 DOF to know what the user was looking at. When a marker is detected, four coplanar and not collinear points of the marker were used to calculate the transformation matrix between the camera and the marker. Figure A-8 shows this relationship.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} V_{11} & V_{12} & V_{13} & W_x \\ V_{21} & V_{22} & V_{23} & W_y \\ V_{31} & V_{32} & V_{33} & W_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = \begin{bmatrix} & V_{3x3} & & W_{3x1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = T_m \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \quad \text{(A-1)}$$

In Equation A-1 T_m is the transformation matrix between marker and camera [51]. This transformation matrix consists of a 3x3 rotation matrix (V_{3x3}) and a 3x1 translation matrix (W_{3x1}). These matrices were found with ARToolKitPlus by solving the Robust Planar Pose

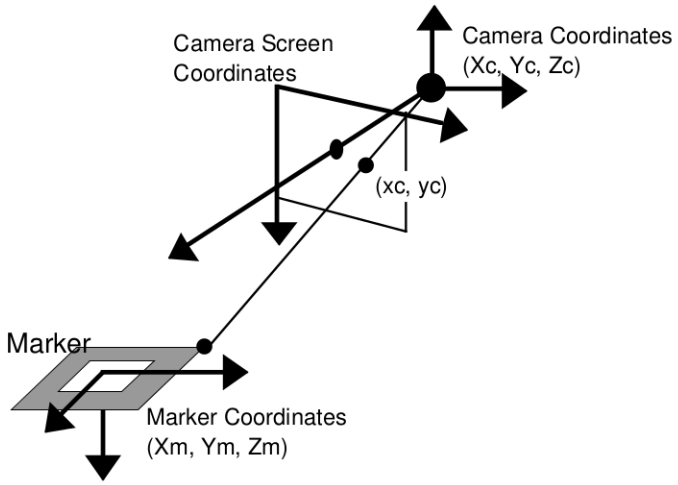


Figure A-8: Relationship between camera and marker coordinate frames [51]. The marker coordinate frame (X_m, Y_m, Z_m) needs to estimated by calculating a transformation matrix using all corner points (x_c, y_c) in the camera coordinate frame (X_c, Y_c, Z_c).

(RPP) [67] equations using every corner point of the marker (x_c, y_c). The coordinates were also corrected by the camera calibration parameters (Appendix C).

A-2-4 Scene rendering

Before the virtual model could be correctly rendered using the transformation matrix, the scene needed to be rendered. The scene is the background image and was directly captured by the cameras. The images from the cameras were rendered as an OpenGL texture using FreeGLUT. FreeGLUT is the free version of the OpenGL Utility Toolbox (GLUT) and provides application window control and allows for user input in the form of keyboard presses or mouse movement. To enable stereoscopic 3D images, the images from the cameras were provided to the control box using the HDMI side-by-side format.

A-2-5 Virtual model

After analysing the images, ARToolKitPlus outputs a $4x4$ transformation matrix in OpenGL [68] notation. This transformation matrix was used to render the virtual models with their correct pose. After the viewpoint is transformed, the local coordinates system of the marker was known.

The virtual model was designed to resemble the real robot arm. This was done by using simple cylinders to draw the model. Figure A-9 shows the real and the virtual robot arm side by side. The virtual model also has the same DOF as the real robot arm.

To enable movement and state changes of the virtual model, various properties of the virtual model were defined: position, velocity and internal state. Te be able to represent all the information needed for the experiment, three virtual robot arms were created and each model served a different purpose by reacting to different information. Each model could be switched

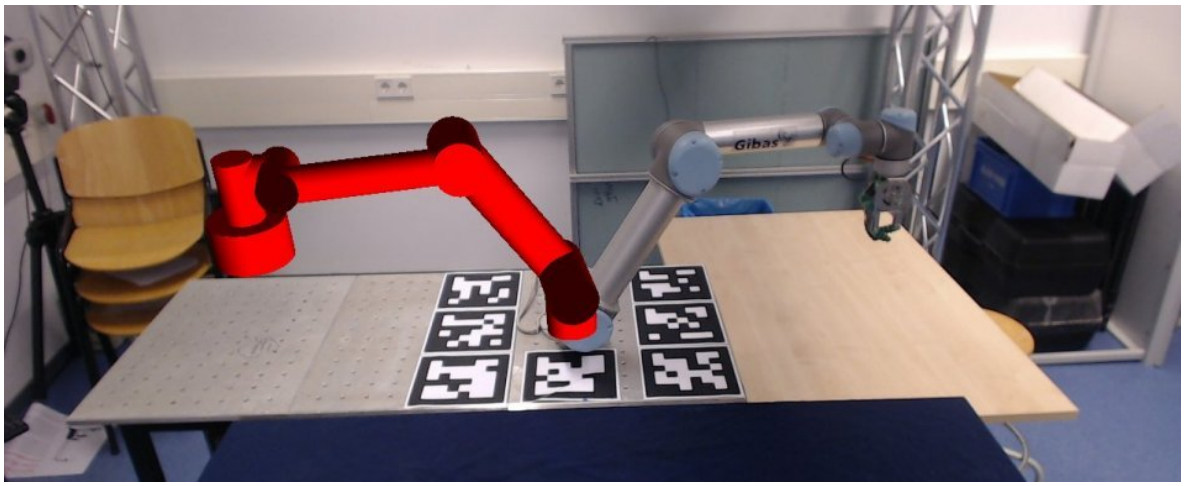


Figure A-9: The virtual model and the real robot arm. The virtual model has the same dimensions and is modelled to look like the real robot arm.

on and off to show only the model that was needed. The following sections describe what information was needed to represent state, goal and trajectory information. Figure A-10 shows where each type of information comes from.

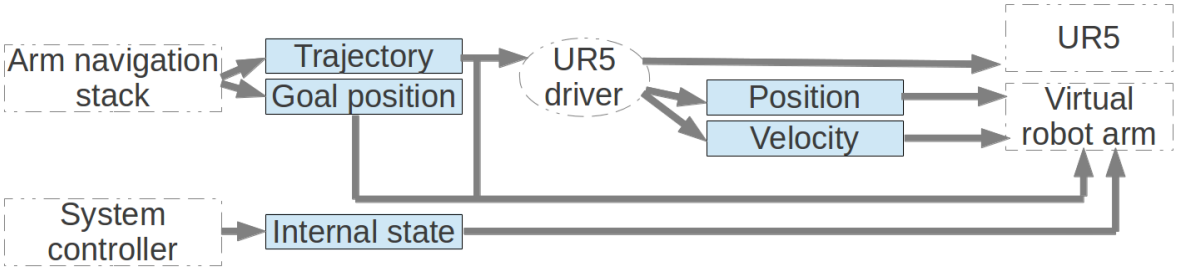


Figure A-10: State and intention data overview. The motion planner in the *arm navigation stack* generated a motion plan and outputted the trajectory and goal position. The *UR5 driver* made sure the trajectory was executed on the robot arm and received the position and velocity in real-time from the robot arm. The current internal state, trajectory, goal position, current position and current velocity were send to the virtual robot arm.

Position

The positions were formatted in a 6x1 vector containing a floating point value (radians) for every DOF of the robot arm. Because OpenGL works with degrees, the values needed to be converted.

Color and transparency

Color and transparency of the virtual arm was set by setting the RGBA property of the arm. The red, blue and green component made up the color and the alpha component was responsible for setting the transparency.

Virtual model for state information

Figure A-11 shows the inputs of the virtual model which are needed to display *internal state* information.

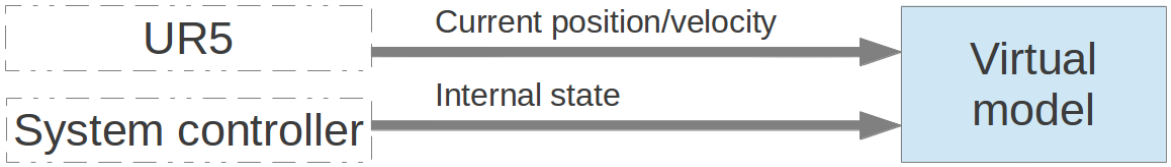


Figure A-11: Inputs to the virtual model to display *internal state* information. The real robot arm, using the *UR5 driver*, provided the real-time position and velocity. The *control node* set the *internal state* of the virtual robot arm.

In the case of *internal state* information, the virtual model was rendered directly over the real robot arm at all times. This way when the virtual model changed color, it looked like the real robot arm changed color. Because it is rendered directly over the real robot arm, the real time position of the real arm must be known. These positions were provided by the real robot arm and were updated in real time at $100Hz$. The information about the *internal state* of the robot was provided by the *control node*. This node gave a signal when the real robot arm changed *internal state* to trigger a change in the virtual model and was either transparency or color of the arm.

Virtual model for intended goal information

Figure A-12 shows the input of the virtual model which are needed to display *intended goal* information.

Figure A-12: Inputs to the virtual model to display *intended goal* information. The *motion planner* provided the joint angles to indicate the final goal position of the robot arm.

The *intended goal* information was used to show where the real robot arm was going to next. This information was provided by the motion planner and was available at the moment the real arm started moving.

Virtual model for intended trajectory information

Figure A-13 shows the input of the virtual model which are needed to display *intended trajectory* information.

The *intended trajectory* information was intended to show the whole future motion of the robot arm. This information was provided by the *motion planner* and consisted of a matrix with joint angles of every point in time. This way the trajectory could be simulated by rendering the virtual arm every time step. Because the motion planner outputted a sparse

Figure A-13: Inputs to the virtual model to display *intended trajectory* information. The *motion planner* provided the joint angles for the whole intended trajectory of the robot arm.

set of positions, they needed to be interpolated to result in a smooth display of the trajectory. This interpolation was done by linear interpolation to get double the amount of points. The time needed to display the trajectory can be varied by changing the loop rate of the display loop and resulted in faster or slower playback of the trajectory.

A-2-6 Software architecture

Figure A-14 gives a total overview of the software used to display the virtual information on the HMD. Vertical columns indicate the threads and arrows indicate exchange of information.

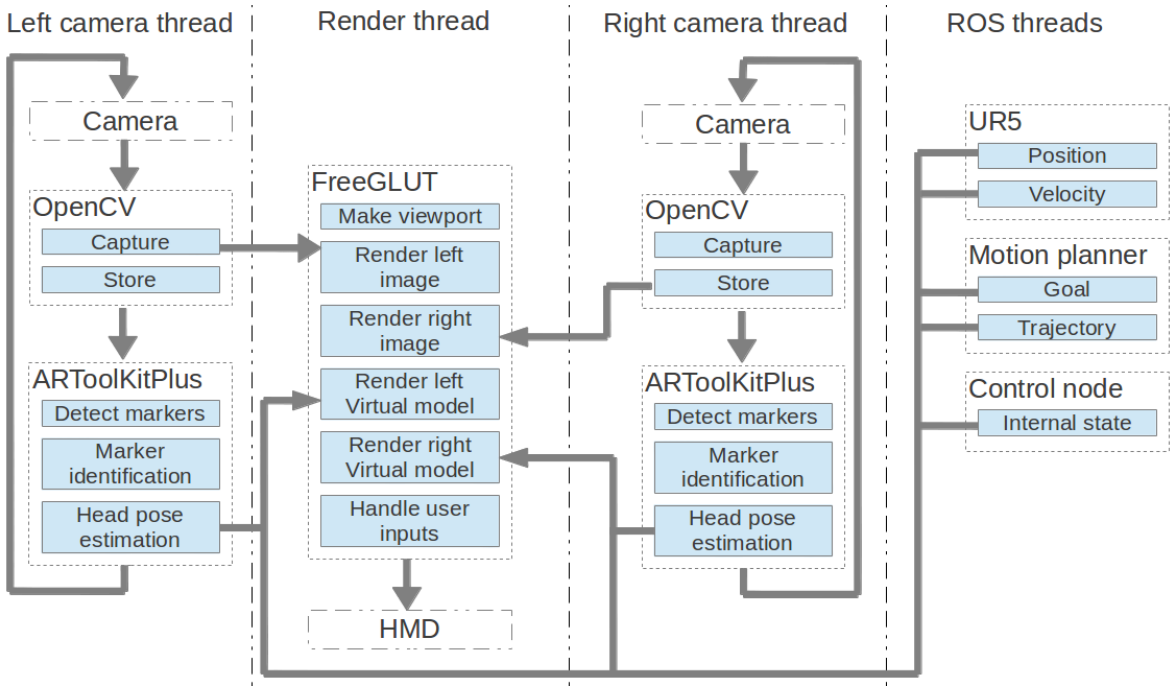


Figure A-14: Total overview of the HMD software. The software used four different threads, two for capturing and analysing the camera images, one for rendering the virtual models and the scene and one thread to communicate with various node in ROS.

Appendix B

Robotic system

This chapter describes the robotic system used to pick up oranges and do the experiments. Appendix B-1 describes all hardware systems and Appendix B-2 describes all software components.

B-1 Robotic hardware

Figure B-1 gives an overview of the hardware components of the robotic system. The hardware used to handle oranges are: a camera to detect oranges (Appendix B-1-1), an industrial robot arm (Appendix B-1-2) to move the oranges and an industrial gripper (Appendix B-1-3) to handle the oranges.

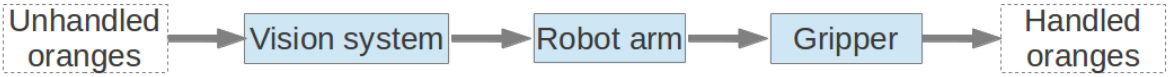
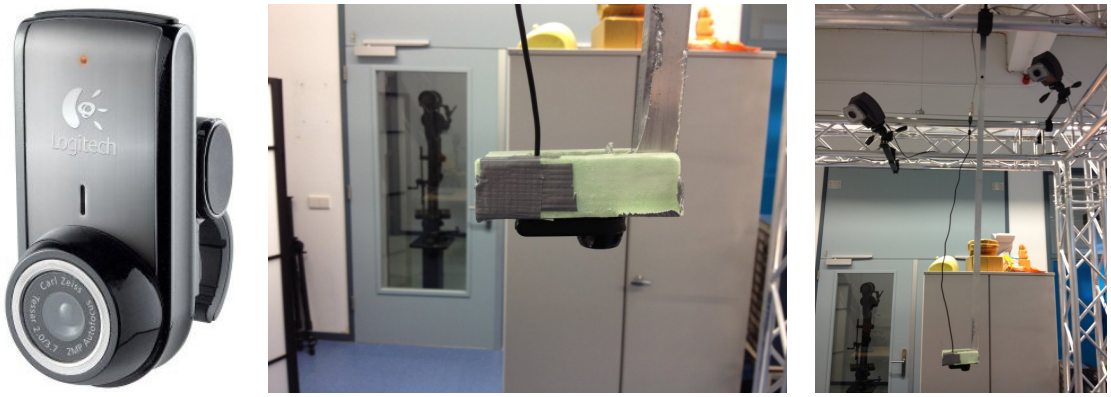


Figure B-1: Overview of the hardware used in the robotic system. A vision system was used to detect the oranges. The detected oranges were then pickup up with a gripper mounted on the industrial robot arm.

B-1-1 Orange detection camera

The camera used to detect the oranges was a Logitech C905 (Figure B-2(a)) consumer grade webcam with a maximum resolution of 2MP that can capture images at a maximum of 30 FPS. This camera was mounted on an aluminium rod (Figure B-2(b)) in a enclosure made of foam (Figure B-2(b)). This rod was mounted on a frame above the robot system with the camera parallel to the table at a height of $0.9m$ looking down at the oranges.



(a) Logitech C905 consumer grade webcam.

(b) Foam enclosure to mount the camera.

(c) Aluminium rod mounted on a frame above the robotic system.

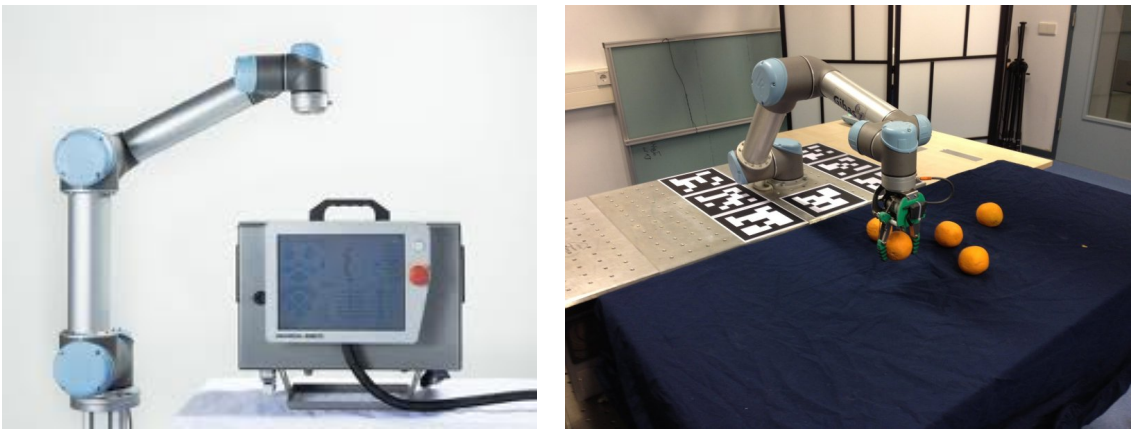
Figure B-2: Camera and mounting frame used to detect oranges. Figure B-2(a) shows the Logitech C905 consumer grade webcam that was enclosed in foam (Figure B-2(b)) and mounted using an aluminium rod to a frame (Figure B-2(b)) at 0.9m above the table with oranges.

B-1-2 Universal Robots UR5

The robotic arm was used to pick up the detected oranges and place them in a box. The arm that was used is a Universal Robots UR5 (Figure B-3), which is an intrinsically safe robot arm. This means that it doesn't have enough power to physically hurt a person and therefore doesn't need to have fences around it to protect humans [2]. The joints of the UR5 have voltage sensors to measure if the measured voltage exceeds the predicted voltage. When this happens, the robot will switch itself off. By using an intrinsically safe robot arm, humans can work in close proximity to it. The specifications of the UR5 are shown in Table B-1.

Parameter	Value
Weight [kg]	18.4
Payload [kg]	5
Reach [m]	0.85
Joint ranges [rad]	-2π to 2π
Joint speed [rad/s]	π
Maximum tool speed [m/s]	1
Repeatability [mm]	$\pm$ 0.1
DOF [-]	6
Power consumption [W]	200
UR5 connection [-]	UTP

Table B-1: Universal Robots UR5 specifications. This table shows all relevant specifications of the Universal Robots UR5 industrial robot arm.

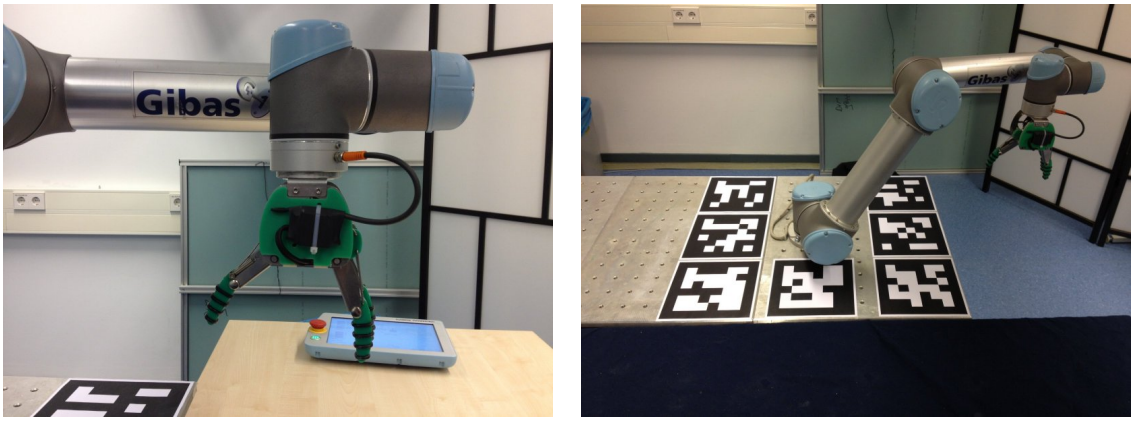


(a) Universal Robots UR5 with control unit. (b) Universal Robot UR5 in experimental setup.

Figure B-3: Figure B-3(a) shows the UR5 with control unit and Figure B-3(b) shows the UR5 in the experimental setup handling oranges.

B-1-3 Lacquey Fetch Hand

The end-effector in the setup was a Fetch Hand from Lacquey. This is a 6 DOF underactuated robotic hand developed as an affordable alternative to existing industrial robotic hands. It has three fingers with each two DOF. All 6 DOF are actuated by one motor, this makes in underactuated. The intelligence of the hand is all in the mechanics, all DOF follow the way of the least resistance. That means that one DOF feels resistance, the power goes to the other DOF, this way all finger enclose the object and the gripping force is divided over the fingers. Figure B-4 shows the gripper mounted on the Universal Robots UR5 in the experimental setup. It is connected to two digital outputs ($24V$) of the robot arm, located on the end-effector. Both digital outputs are connected to a relay, one enables/disables the gripper and one opens/closes the gripper.



(a) Lacquey Fetch Hand close-up. (b) Lacquey Fetch Hand.

Figure B-4: Lacquey Fetch Hand mounted on the Universal Robots UR5 in the experimental setup.

B-2 Robotic software

This section describes all software components of the robotic system. An overview of the implementation, design choices and capabilities is given.

B-2-1 Robotic Operation System

To make the robot do what we want, control software is needed. The Universal Robots UR5 has its own control box but unfortunately the functionality is sufficient. The software provides all the components needed to actuate the robot and make it move to any position, but unfortunately the motion planning and inverse kinematic software is a black box: there is no way to extract the motion plan before execution. An alternative control software package was found in Robotic Operating System (ROS) [58]. This package allows the motion plan to be extracted before the motion plan is executed.

All software was designed on the ROS framework, version *Fuerte*. ROS is a framework that focuses on delivering a flexible, versatile and complete set of tools needed to program a robot. It is widely used in academic community because it makes it rapid development of relative complex pieces of software possible. Figure B-5 shows an overview of all ROS components needed to operate the robotic working cell.

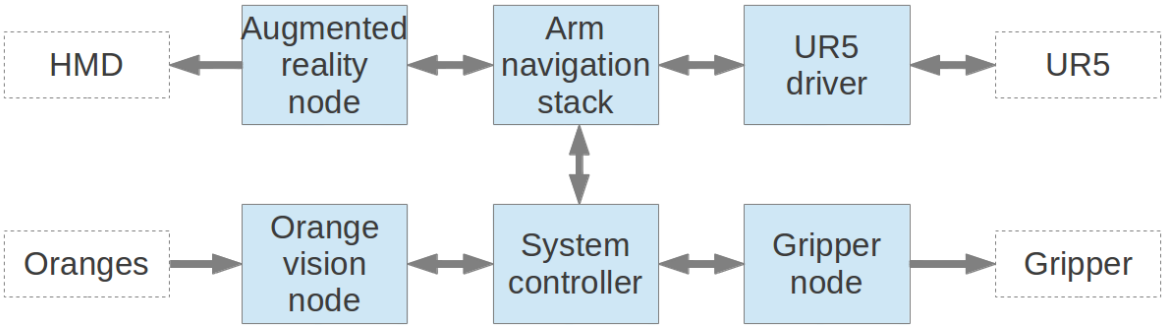


Figure B-5: Overview of all the ROS software components. The *system controller* is the main hub of the robotic system. This node requests the coordinates of the orange from the *orange vision* node. With the orange coordinates a motion plan request is send to the *arm navigation stack*. This stack is a collection of nodes responsible for calculating a proper trajectory for the robot arm. The *arm navigation stack* sends the calculated trajectory to the *UR5 driver*, which activates the real robot arm. The *system controller* also activates the gripper via the *gripper node*. The *UR5 driver* and *arm navigation stack* both send robot data to the *augmented reality node*.

The total system consists of the following software components:

Orange vision node: the orange vision node took care of the detection of oranges on the table. This was done by analysing images using a Gaussian Mixture Modelling algorithm. The coordinates of the detected oranges were then converted from image coordinates to world coordinates. Section B-2-2 describes the vision system in more detail.

System controller node: the system controller node was the main hub of the robot working cell. It was responsible for the coordination and communication between the other nodes.

Tasks include: requesting list or oranges from the *orange vision node*, generating a motion plan to send to the *arm navigation stack* and keep track of the *internal state* of the robot arm.

Gripper node: the gripper node sent commands to the Universal Robots UR5 to actuate the gripper.

Arm navigation stack: the arm navigation stack is a collection of nodes designed to calculate a collision free trajectory for the robot arm. This stack has the following components: inverse kinematics, collision checking, motion planning and trajectory filtering. Appendix B-2-5 describes the arm navigation stack.

UR5 driver: the driver enabled communication between the *arm navigation stack* and the physical robot arm.

Augmented reality node: the augmented reality node was responsible for visualizing the scene with the virtual robot arm to show the state and intentions of the robot arm.

B-2-2 Orange detection

The goal of the vision system was to detect and locate oranges on a table and present the data to the robot system so that the oranges could be picked up. The vision system consists of the parts shown in Figure B-6.

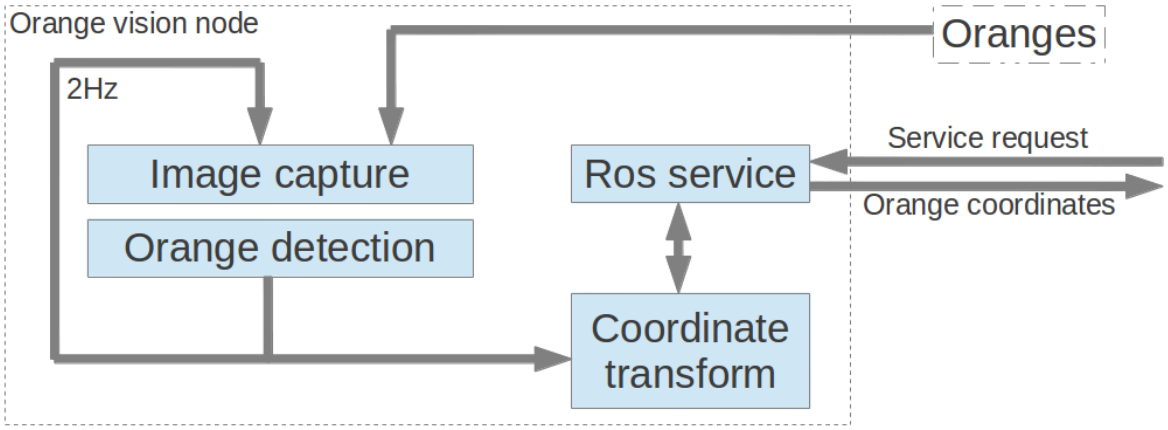
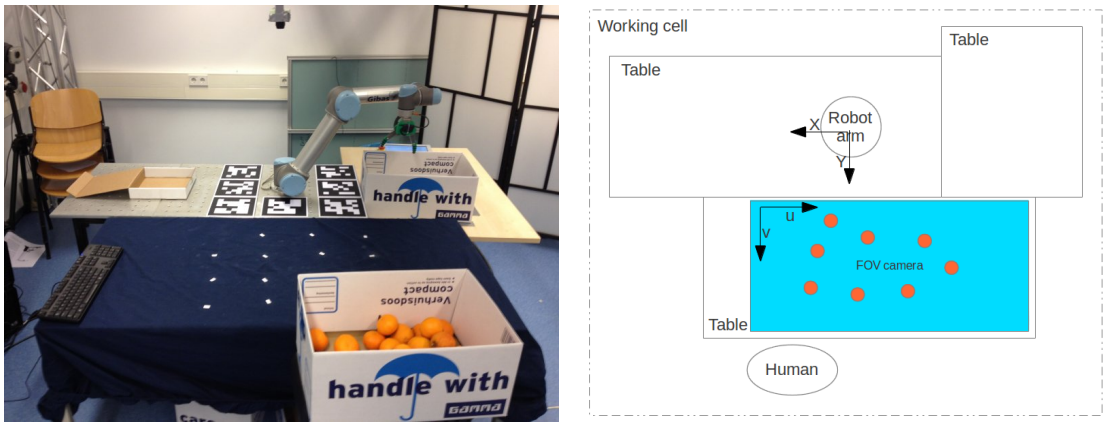


Figure B-6: Schematic overview of the orange detection software. The image taken by the camera were captured and stored for analysis. A Gaussian Mixture Model algorithm detected the oranges in the images. For the robot arm to be able to pick up the oranges, the coordinates needed to be converted from image frame coordinates (pixels) to real world coordinates (meters).

Image capture

To detect objects on the table the camera captured images to be analysed. The software used to capture the images was OpenCV [57]. The images were captured at a resolution of 1600x1200 pixels at 2 FPS. The FOV of the camera and coordinate frames are shown in Figure B-7.



(a) Overview of experimental setup.

(b) FOV and coordinates frames for orange detection.

Figure B-7: FOV and coordinate frames of the orange detection. Figure B-7(a) shows the FOV of the detection camera and Figure B-7(b) shows the image coordinate frames (u, v) and the robot coordinate frame (X, Y, Z).

Orange detection

To detect the oranges in the captured images, a detection algorithm was used. The detection algorithm used a Gaussian Mixture Model algorithm on the color images to detect the oranges. This algorithm was developed by Floris Gaisser and is implemented using OpenCV. The algorithm follows the steps of Algorithm 7.

Algorithm 7 Gaussian mixture model

```
repeat
    Capture image in BGR format
    Convert BGR image to Hue-Saturation-Value (HSV) image
    Split HSV into separate H, S and V images
    Create histogram of H image
    while not exceed iteration limit do
        Fit two Gaussians over histogram
    end while
    Segment H image using the two Gaussians
    Find contours in segmented image
    Fit ellipses on the contours
    Filter ellipses
    Sleep until 2 FPS
until forever
```

The images captured by the camera were analysed by a Gaussian Mixture Model algorithm. Figure B-8 shows the steps of the algorithm. The *input image* was stored to be analysed. The first step was to extract the *hue image*. The *hue image* was used, because there is a big difference between the hue value for blue (background) and orange (objects), therefore a clear distinction could be made. A *histogram* was made of the *hue image*, in the *histogram*

two peaks can be detected, one for each hue-value (blue and orange). The Gaussian Mixture Model algorithm iterates over the *histogram* to fit two Gaussian distributions over the peaks. These Gaussian distributions were used to segment the *input image* into a binary image, also called a *mask*. In the *mask* the background is represented a pure black and the oranges are pure white. After a binary morphology step a *filled masked* was used to fit ellipses to get the coordinates of the oranges.

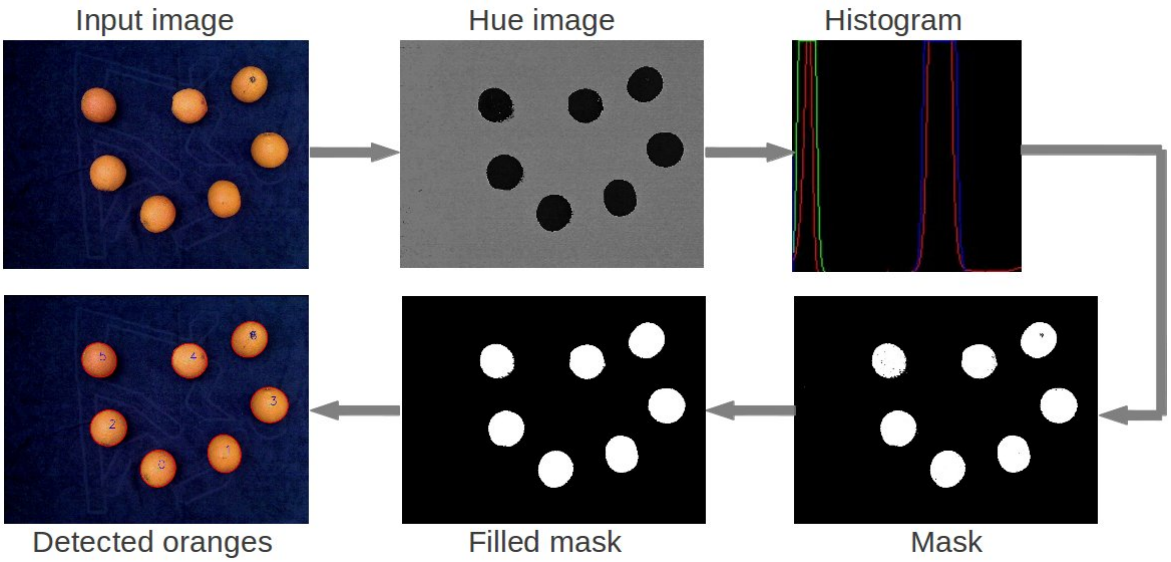


Figure B-8: Overview of Gaussian Mixture Model detection algorithm. From the *input image* the *hue image* was extracted. A *histogram* was calculated from the *hue image* and two Gaussian distributions were fit over the *histogram*. Segmentation was done using the two Gaussian distributions to get a *mask*. Ellipses were fit over the *filled mask* to get the coordinates of the *detected oranges*.

Coordinate Transformation

The detected oranges have coordinates in the image coordinate frame (u, v). These coordinates needed to be transformed to world coordinates (X, Y, Z). See Figure B-7 for coordinate system definitions. The transformation uses the pin-hole camera model [69, 70].

$$X_c = \frac{(u - c_x)}{f_x} z_c \tag{B-1}$$

$$Y_c = \frac{(v - c_y)}{f_y} z_c \tag{B-2}$$

$$Z_c = 0 \tag{B-3}$$

In this transformation (c_x, c_y) is the principle point of the image and f_x and f_y and the focal lengths. z_c is the height of the camera above the table, which is a constant $0.9m$.

B-2-3 System controller node

With the coordinates of the oranges known, the robot needed to be activate to pick them up. The *system controller node* was responsible for coordinating which orange to pick up, when to pick it up and where to put it. Figure B-9 shows an overview of the control node.

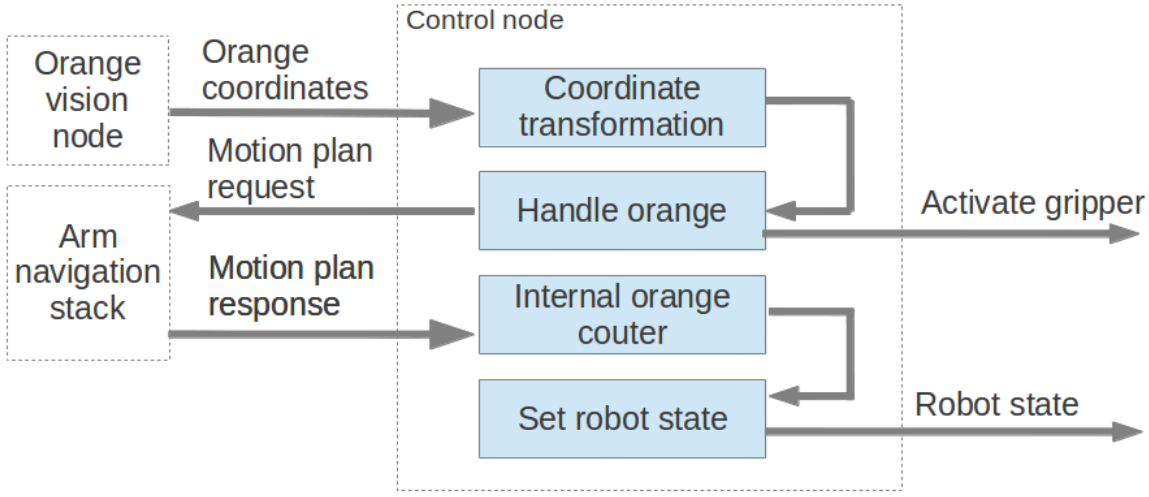


Figure B-9: Overview of the control node. The *system controller node* requested the coordinates of the oranges, handled the oranges by generation motion plans (executed by the *arm navigation stack*), operated the gripper and set the *internal state* of the robot arm.

Algorithm 8 Control node algorithm

```
Move robot arm to initial position
Power on gripper
Open gripper
repeat
    Get orange coordinates
    if Detected orange = 0 then
        Set robot internal state to waiting for oranges
    end if
    Set robot internal state to moving
    Move robot arm to (X_r, Y_r, (Z_r + 0.16))
    Move robot arm to (X_r, Y_r, Z_r)
    Close gripper
    Move robot arm to (X_r, Y_r, (Z_r + 0.16))
    Move robot arm to drop position
    Increment orange counter
    if Number of oranges handled = 1, 2 or 3 then
        Set robot internal state to waiting for user
    end if
until forever
```

Algorithm 8 shows the algorithm used in the *system controller node*. The first step was

to request the coordinates of the oranges from the *orange vision node*. If no oranges were detected the *internal state* of the robot was set to *waiting for oranges*. From these coordinates a random orange was chosen to be picked up. The next step in picking up the oranges was to transform the coordinates of the detected oranges from the camera coordinate frame (X_c, Y_c, Z_c) to the robot coordinate frame (X_r, Y_r, Z_r). The coordinates of the oranges were provided in meters, so adding the offset (ΔX and ΔY) between the camera and the base of the robot arm was enough to transform the coordinates:

$$X_r = X_c + \Delta X \tag{B-4}$$
$$Y_r = Y_c - \Delta_Y \tag{B-5}$$
$$Z_r = Z_c \tag{B-6}$$

With the oranges defined in the robot coordinate frame, a motion plan request was generated. A motion plan request is predefined list of parameters needed by the *arm navigation stack* to generate a motion plan. The position of the orange is the goal of the robot and needed to be defined as an position and orientation of the end-effector. The position (P_r) and orientation (f_r) of the end-effector needed to be defined in the following format:

$$P_r = \begin{bmatrix} X_r & Y_r & Z_r \end{bmatrix}^T \tag{B-7}$$
$$f_r = \begin{bmatrix} x & y & z & w \end{bmatrix}^T \tag{B-8}$$

ROS requires the orientation to be in quaternions. When picking up the oranges the orientation of the end-effector was fixed facing down, so that picking the oranges up was easier. Once the motion plan request was generated, it was send to the *arm navigation stack* for execution (Appendix B-2-5). When the *arm navigation stack* gives a positive response, meaning that the motion plan is correctly executed, the gripper was operated and the orange was handled and put in the box. An internal counter counts the number of oranges handled. If the number equals a preset number, the *internal state* of the robot changes to *waiting for user* and the robot stops until the user tells it to continue. This confirmation comes from the augmented reality node.

B-2-4 Gripper node

The gripper node operates the gripper mounted as an end-effector on the robot arm (Appendix B-1-3). It does this by opening a socket connection to the robot arm and executing a command. The commands were used to open of close the digital ports to which the gripper is connected.

B-2-5 Arm navigation stack

The oranges needed to be pick up and to be able to do that, a trajectory for the robot arm needed to be calculated. This is done in ROS using the *arm navigation stack*. This stack is a collection of nodes that together calculate a collision free trajectory for the robot arm. Figure B-10 shows an overview of the arm navigation stack. To get it to work properly it had

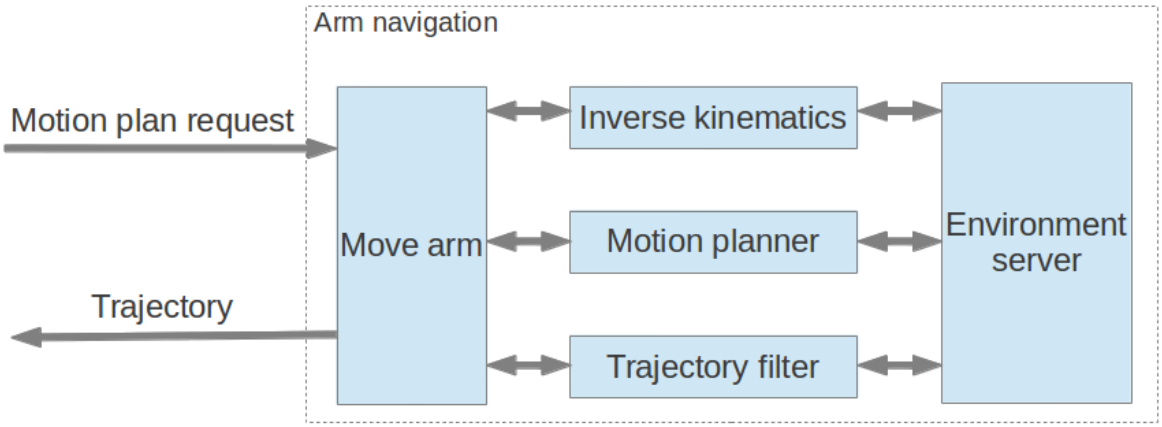


Figure B-10: Overview of the *arm navigation stack* in ROS. A motion plan request was send to the *move arm* node. This node individually calls the *inverse kinematics* node, the *motion planner* and the *trajectory filter* to calculate is valid motion plan.

to be customized and tweaked, because the *arm navigation stack* is generic and is designed to work on every robot arm (with some modifications).

The *arm navigation stack* consists of the following parts:

Move arm: this node is the central connection point of the *arm navigation stack*. It receives the motion plan request and calls the other nodes to calculate the trajectory. The first step was to check the *inverse kinematics node* to be sure the goal pose could be reached and that the robot was not in collision with the environment or itself. For the next step the motion planner was called which tries to find a trajectory that allowed the robot to move from the initial position to the goal position without collisions. The motion planner output was a sparse trajectory, meaning the trajectory was described using a small number of waypoints, therefore the trajectory needed to be filtered. The *trajectory filter node* calculated a smooth and collision free trajectory based on the output of the motion planner by using interpolation.

Inverse kinematics: the *inverse kinematics node* calculated if the requested pose could be reached and that it could be reached without collisions. These checks were done based on a kinematic model of the robot arm and a model representing objects in the environment. Appendix B-2-5 describes the inverse kinematics in more detail.

Environment server: the environment server was responsive for keeping track of objects in the environment. These objects were used to check for collisions with the robot. Appendix B-2-5 explains how the environment server works.

Motion planning: the motion planner was used to find a collision free trajectory that satisfies the constraints based on the kinematic model and the environment. Appendix B-2-5 explains in more detail the important parts of the motion planner.

Trajectory filtering: the trajectory filter calculated a smooth and collision free trajectory based on the sparse output of the motion planner. Appendix B-2-5 explains why filtering was needed and how the filtering was done.

Inverse kinematics

ROS uses the Unified Robot Description Format (URDF) to encode and extract the kinematics of the robot arm. URDF is an XML format to describe the physical structure of a robot. The basic structure of an URDF file consists of links and joint with dimensions and properties (length, width, position, rotation axis, rotation speed). These links and joints are setup in a tree structure with parents and children. This way the relation between the links and joints can be defined. The URDF file is loaded into the ROS parameter server so that is accessible to various nodes. Figure B-11 shows a visualization of the URDF model of the UR5 using RVIZ (visualization tool used in ROS).

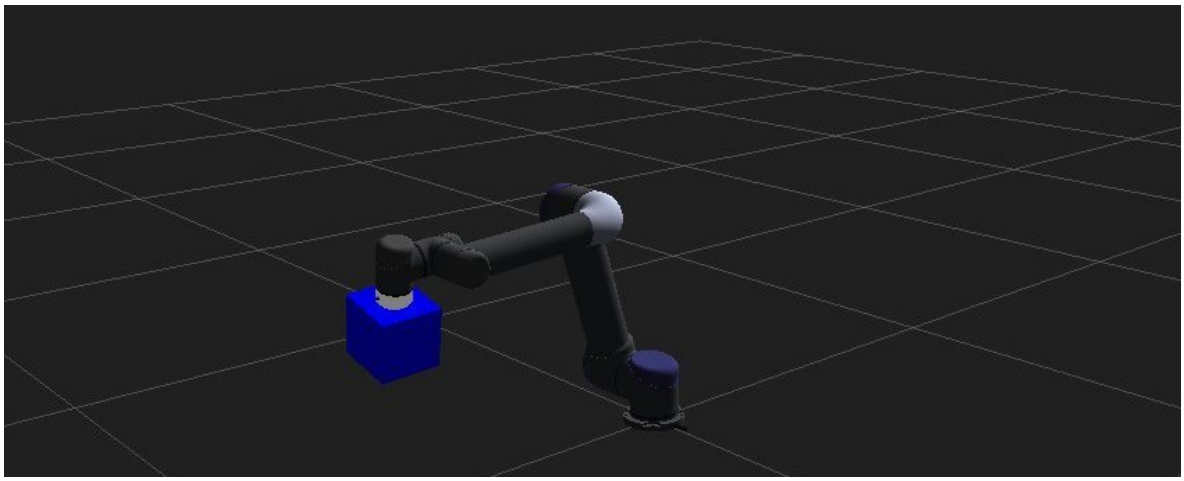


Figure B-11: ROS visualization of the UR5 URDF model. This figure shows a virtual representation of the Universal Robots UR5 in RVIZ. All joints and links a shown with their mesh for a more detailed representation. The gripper is not part of the standard robot arm and was modeled by a blue cube.

The *move arm* node translated the motion plan request to a request for the *inverse kinematics* node. With the end-effector position (P_r), end-effector orientation (f_r) and dimensions of the robot known, the required angle of the joints were calculated. This was done using an inverse kinematics algorithm. ROS uses a numerical solver, which can only find good solutions when the end state is close to the starting state, but allows solutions to be calculated for all available robots without special prior knowledge about the robot. An alternative to the standard inverse kinematic algorithm was found to be the IKfast v61 inverse kinematics solver. This is an analytical solver from the Open Robotics Automation Virtual Environment (OpenRAVE) motion planning software [71] and was specially designed for the Universal Robots UR5 using the URDF file. Not only does this solver find more solutions, it is also faster. The solver needed to be fast, because not only were inverse kinematic calculations used to find the joint angles of the end position, but also for every point in the trajectory. So this algorithm is executed hundreds of times during the planning of one motion. The IKfast algorithm finds inverse kinematic solutions in $< 10\ ms$.

Environment server

Collision checking was done to check if the robot was not in collision with itself or the environment. ROS uses the *environment server* to simulate the environment with its obstacles. Every time an inverse kinematic solution was requested, it was checked against the *environment server*. Figure B-12 shows a visualisation of the *environment server* using RVIZ with the obstacles:

Universal Robots UR5: the robot arm is shown with its mesh. All joints and links are represented in the *environment server* to enable self-collision checking and collision checking with the obstacles in the environment.

Gripper: the gripper was modeled by a cube. This cube represents the dimensions of the gripper when it is open.

Tables: the table at which the robot is mounted, the *shared working area* table and the table with the *output* box were also put in the environment server. The were modeled as large flat planes.

Output box: the *output* box was modeled as a cube with dimensions of the real box.

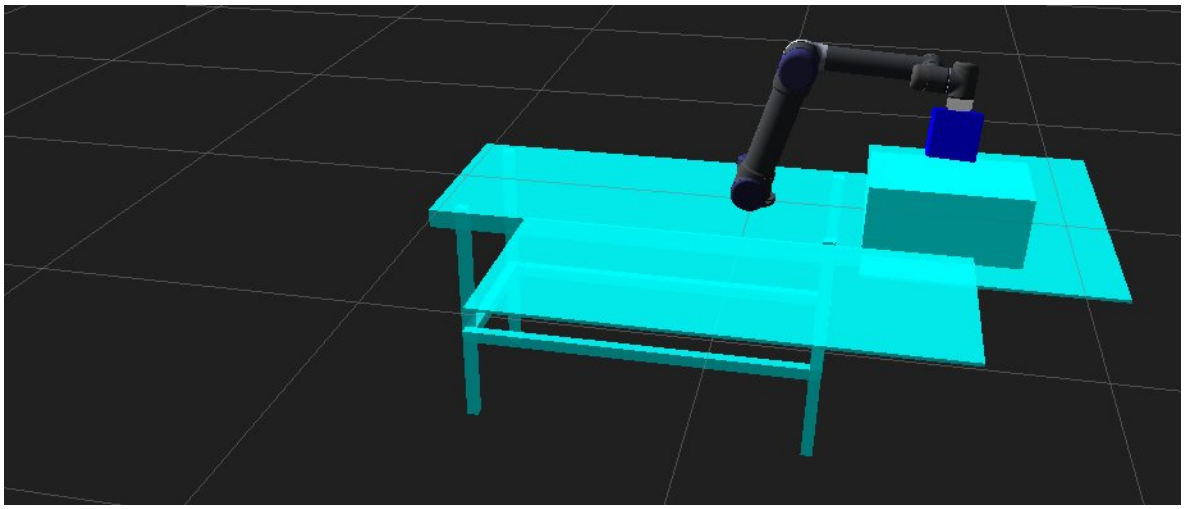


Figure B-12: ROS visualization of planning environment using RVIZ. The direct environment was modeled in ROS to be used for collision checking. The URDF model was used for self-collision checking and also for collisions checking with obstacles in the environment (light blue).

Motion planner

To go from an end-effector pose (P_r and f_r) to a trajectory, motion planning was needed. The motion planner used the end-effector position and orientation, environment constraints and trajectory constraints to calculate a motion plan. In ROS there are various motion planners which can be categorized into sample based, search based and stochastic trajectory optimization motion planner. In ROS the sample based motion planners are found in Open Motion Planning Library (OMPL) [72]. The OMPL has the following sample based planners: Rapidly-exploring Random Trees (RRT) [73], Probabilistic Roadmap (PRM) [74, 75, 76],

Expansive Space Trees (EST) [77] and Kinodynamic Motion Planning by Interior-Exterior Cell Exploration (KPIECE) [78]. The search based planners are implemented in the Search Based Planner Library (SBPL). The planners are: ARA* [79] (anytime version of A* [80]), Anytime D* [81] (anytime version of D* [82]) and R* (randomized version of A*). The stochastic trajectory optimization motion planners are: Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [83] and Stochastic Trajectory Optimization for Motion Planning (STOMP) [84].

By far the most used motion planners are the KPIECE algorithm from OMPL and the CHOMP algorithm. KPIECE was the go-to motion planner, but is gradually being replaced by CHOMP. The advantage of CHOMP is that the generated trajectories are smooth, which means that filtering is not needed. Also CHOMP allows for orientation constraints during motion, this enables the gripper to be oriented down during the whole motion. Unfortunately the version of CHOMP supplied in ROS is not robust enough for continues operation, the planner is slower than KPIECE and has a lower success rate. That is why, for robustness reasons, the KPIECE algorithm from the OMPL was used as the motion planner.

Trajectory filter

The motion planner generated a set of waypoints which the robot arm has to follow the reach the goal. The *trajectory filter* is the final step in the motion plan generation phase and validates the generated motion plan. It made sure that the generated motion plan does not make the robot arm violate the speed or acceleration constraints and not collide with the environment or itself. It also interpolated the generated motion plan, because the set of waypoint generated by the motion planner was very sparse and can't be directly executed on the robot.

The standard interpolation method in the *arm navigation stack* uses cubic splines to interpolate the trajectory. It was found that when the generated motion plan has only 2 or 3 waypoints (short motions), the interpolation algorithm would take about 3 seconds to calculate a smooth trajectory. When a parabolic interpolation algorithm was used, the calculation time was reduced to about 0.1 seconds. After filtering the trajectory is smooth and does not violate the constraints (environmental and dynamic) and can be executed on the robot arm.

B-2-6 UR5 driver

To communicate the calculated trajectory to the robot arm, a driver was needed. The *UR5 driver* is a node that transforms the generated trajectory into movement of the robot. Figure B-13 shows an overview of the driver.

The driver checked the calculated trajectory for invalid data points and interpolates the data points again for smooth execution. It then sent the joint angles and velocities to the robot and receives the current joint angles and velocities from the robot at a frequency of $100Hz$. The communication to the robot is by UTP.

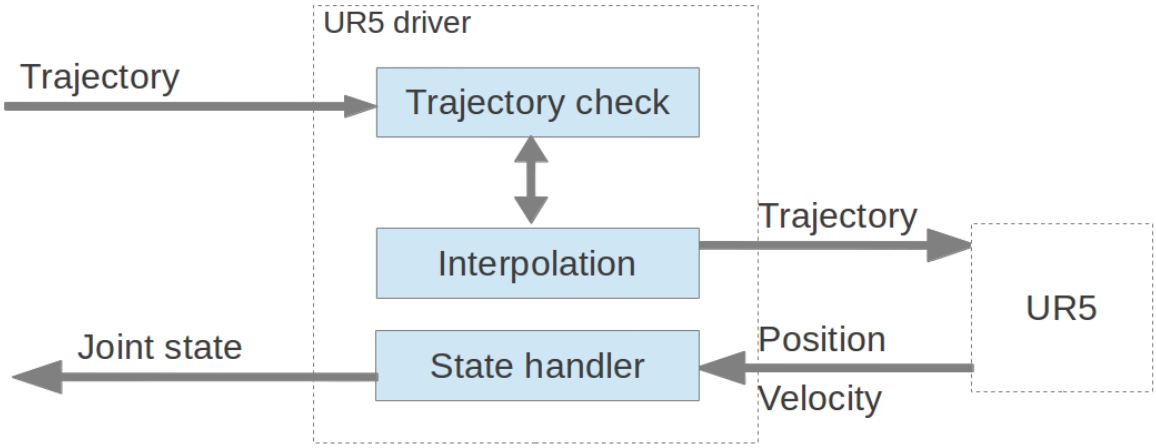


Figure B-13: Overview of the *UR5 driver*. The driver makes sure that the calculated trajectory is executed on the robot arm. It also receives the current position and velocity from the robot arm.

Appendix C

Camera calibration

C-1 Calibration algorithm

To be able to accurately detect objects using this camera, the camera needed to be calibrated [85]. The most used camera model is the pinhole camera model and needs the following parameters to represent the camera:

- Camera matrix
- Distortion matrix

The camera matrix represents the relationship between the real world points (X, Y, Z) and the points in the image frame (u, v, w).

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_x & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = A \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{C-1}$$

The relation between real world coordinates and image coordinate is defined as a function of the focal lengths (f_x and f_y) and the principle point (c_x, c_y) of the camera. Because consumer grade cameras have a lot of distortion it needs to be corrected. There are two types of distortion, radial (fish-eye effect) and tangential (lens not parallel to image plane). ARtoolKitPlus does not use tangential distortion and is not taken into account. The distortion matrix is:

$$D = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{C-2}$$

The equations to correct for distortion:

$$\tilde{u} = u \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \tag{C-3}$$

$$\tilde{v} = v \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \tag{C-4}$$

The unknown variables measured by the calibration algorithm are: f_x, f_y, c_x, c_y, k_1, k_2, k_3, p_1 and p_2. The calibration was done using the ROS camera calibration package. To calibrate, a lot of images were taken of a checker board. The checker board needed to be captured from a lot of different angles and position to properly calculate the variables. The camera calibration package showed a GUI that indicated when enough images are captured in x and y direction, when enough different sizes were captured and when enough different angles were captured. The calibration results of the cameras are shown in Appendix C-2. Because the camera was calibration at a resolution of 640x480 pixels and ARToolKitPlus used images with a resolution of 1600x1200 pixels, the camera matrix needed to be scaled with the ratio of the two resolutions.

C-2 Calibration results

Section C-2-1 shows the calibration results for the orange vision camera, Section C-2-2 and Section C-2-3 show the results for the left and right camera of Marty. All camera were calibration at a resolution of 640x480 pixels using the ROS camera calibration package.

C-2-1 Logitech C905 for the orange vision system

$$A = \begin{bmatrix} 568.19 & 0 & 319.50 \\ 0 & 568.19 & 239.50 \\ 0 & 0 & 1 \end{bmatrix} \tag{C-5}$$

$$D = \begin{bmatrix} 0.009826 & -0.022168 & 0 & 0 & -0.17508 \end{bmatrix} \tag{C-6}$$

C-2-2 Logitech C905 for Marty's left eye

$$A = \begin{bmatrix} 539.45 & 0 & 311.65 \\ 0 & 542.72 & 226.05 \\ 0 & 0 & 1 \end{bmatrix} \tag{C-7}$$

$$D = \begin{bmatrix} 0.05988 & -0.13146 & -0.00199 & -0.00601 & 0 \end{bmatrix} \tag{C-8}$$

C-2-3 Logitech C905 for Marty's right eye

$$A = \begin{bmatrix} 538.60 & 0 & 303.16 \\ 0 & 541.30 & 228.99 \\ 0 & 0 & 1 \end{bmatrix} \tag{C-9}$$

$$D = \begin{bmatrix} 0.06320 & -0.14467 & -0.00223 & -0.00530 & 0 \end{bmatrix} \tag{C-10}$$

Appendix D

Instructions for test subjects

You will be conducting an experiment to evaluate the human-robot interaction. During the experiment you will be recorded on video. After the experiment you will be asked to fill out a questionnaire.

D-1 Goal

Work together with the robot arm to fill the boxes with oranges.

D-2 Procedure state projection

The experiment will be done twice: once without virtual information in the head-mounted display and once with virtual information in the head-mounted display. The goal is to fill boxes oranges and sticker them.

Follow these steps:

- Once the system has started you will pick up the oranges one by one from the box
- Put a sticker on the orange you picked up.
- Put the orange on the grid on the table.
- When there are about four oranges on the table, the robot will detect them and start handling them.
- Just continue to sticker oranges.
- Continue to sticker the oranges until the robot arm stops moving. This simulates that the box is full and that an action by you is needed.
- When the box is full (1, 2 or 3 oranges in a box), hit space bar. The robot will then continue.
- Your task is to fill the grid as fast as possible and to minimize robot stopping time.
- Virtual information shown in the HMD

  - The virtual robot arm will indicate its internal state by projecting directly over the real robot arm.
  - A red arm indicates that the robot arm is moving and not finished.
  - A green flashing arm indicates that the robot arm is not moving and finished with filling the box. It requires action from you.
  - A blue arm indicates that no oranges are detected.

D-3 Procedure goal projection

The experiment will be done twice: once without virtual information in the head-mounted display and once with virtual information in the head-mounted display. The goal is to fill boxes oranges and sticker them.

Follow these steps:

- Once the system has started you will pick up the oranges one by one from the box
- Put a sticker on the orange you picked up.
- Put the orange on the grid on the table.
- When there are about four oranges on the table the robot will detect them and start handling them.
- Just continue to sticker oranges.
- The robot arm will move to the left or right once it picks up an orange.
- To simulate quality checks by the robot, it will randomly put an orange in the box on the left.
- Once you detect that the robot is moving to the left, hit space bar.
- The robot arm always continues moving.
- Your task is to fill the grid as fast as possible and react to the robot when it is moving to the left as fast as possible.
- Virtual information
  - The virtual robot arm will indicate the goal by projecting the goal position of the robot arm.

Appendix E

Questionnaires

E-1 Personal Info

Name:
Gender:
Age:
E-mail:
Left or right handed:
Previous experience with augmented reality:
Glasses:

E-2 Data

Date:
Starting time:
Ending time:
Experiment 1 starting condition: Without projection - With projection
Experiment 2 starting condition: Without projection - With projection

Test	Without projection	With projection
State		
Goal		

Table E-1: Measurements.

Appendix F

Code

F-1 Statistical analysis

The statistical analysis was done using Matlab R2013b.

```
clear all
close all
clc

% Paired t-test state projection reaction time
state_wo = [3.352;3.684;2.240;2.876;1.774;2.328;1.624;3.266;1.836;1.952];
state_w =  [2.336;1.308;2.142;1.312;1.258;1.318;1.136;2.152;1.704;1.554];
mean_s_wo = mean(state_wo);
mean_s_w = mean(state_w);
diff_state = state_wo - state_w;

[h,p,ci,stats] = ttest(state_wo, state_w);

figure;
G = {'w/o virtual information', 'w/ virtual information'};
boxplot([state_wo, state_w], G);
ylabel('Reaction time [s]')

% Paired t-test goal projection reaction time
goal_wo =  [1.104;1.824;1.364;2.840;1.554;1.486;1.568;1.984;1.414;1.608];
goal_w =   [0.934;1.222;1.298;2.066;1.148;1.420;1.526;1.612;1.368;1.574];
mean_g_wo = mean(goal_wo);
mean_g_w = mean(goal_w);
diff_goal = goal_wo - goal_w;

[h,p,ci,stats] = ttest(goal_wo, goal_w);

figure;
boxplot([goal_wo, goal_w], G);
```

```
30 ylabel('Reaction time [s]')
31
32 % Paired t-test state projection oranges
33 state_o_wo = [9.873;NaN
       ;9.189;9.767;6.456;7.636;11.803;11.029;7.059;11.215];
34 state_o_w  = [7.133;NaN
       ;9.936;13.846;8.276;7.286;13.091;12.000;11.500;10.500];
35 mean_state_o_wo = mean(state_o_wo);
36 mean_state_o_w = mean(state_o_w);
37 diff_state_o = mean_state_o_wo - mean_state_o_w;
38
39 [h,p,ci,stats] = ttest(state_o_wo, state_o_w);
40
41 figure;
42 boxplot([state_o_wo, state_o_w], G);
43 ylabel('Orange handling rate [orange/min]')
44
45 % Paired t-test goal projection oranges
46 goal_o_wo = [13.269;NaN
       ;11.205;7.246;9.580;9.351;12.712;13.171;7.021;10.345];
47 goal_o_w  = [10.949;NaN;11.215;8.182;NaN
       ;7.297;13.091;13.483;7.651;9.048];
48 mean_goal_o_wo = mean(goal_o_wo);
49 mean_goal_o_w = mean(goal_o_w);
50 diff_goal_o = mean_goal_o_wo - mean_goal_o_w;
51
52 [h,p,ci,stats] = ttest(goal_o_wo, goal_o_w);
53
54 figure;
55 boxplot([goal_o_wo, goal_o_w], G);
56 ylabel('Orange handling rate [orange/min]')
57
58 % Plots learning curve
59 x = [1,2,3,4];
60 one  = [7.133;9.873;10.949;13.269];
61 two = [11.029;12.000;13.483;13.171];
62 three = [11.803;13.091;13.091;12.712];
63 four = [7.246;8.182;9.767;13.846];
64 five = [7.636;7.286;7.297;9.351];
65 six = [9.189;9.936;11.205;11.215];
66 seven = [9.048;10.345;11.215;10.500];
67 eight = [7.059;10.500;9.048;10.345];
68 nine = [6.456;8.276;9.580;9.580];
69
70 orange_total = [one';two';three';four';five';six';seven';eight';nine'];
71
72 figure;
73 G_orange = {'1st experiment', '2nd experiment', '3rd experiment', '4th
       experiment'};
74 boxplot(orange_total, G_orange)
75 ylabel('Orange handling rate [orange/min]')
```

Bibliography

[1] EN953, "Safety of machinery - Guards - General requirements for the design and construction of fixed and movable guards," 1997.

[2] ISO10218, "Robots for industrial environments - Safety requirements - Part 1: Robot," 2006.

[3] S. Haddadin, A. Albu-schaffer, and G. Hirzinger, "Safety Evaluation of Physical Human-Robot Interaction via Crash-Testing," in *In Robotics: Science and Systems Conf. (RSS2007)*, pp. 217–224, 2007.

[4] "SMErobot." `http://www.smerobot.org`, aug. 2013.

[5] M. Endsley, "Toward a Theory of Situation Awareness in Dynamic Systems: Situation Awareness," *Human factors*, vol. 37, no. 1, pp. 32–64, 1995.

[6] M. Goodrich and D. Olsen, "Seven Principles of Efficient Human Robot Interaction," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, vol. 4, pp. 3942–3948, oct. 2003.

[7] M. Goodrich and A. Schultz, "Human-Robot Interaction: a Survey," *Foundations and Trends in Human-Computer Interaction*, vol. 1, pp. 203–275, jan. 2007.

[8] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger, "Collision Detection and Reaction: A Contribution to Safe Physical Human-Robot Interaction," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 3356–3363, 2008.

[9] T. Ende, S. Haddadin, S. Parusel, T. Wusthoff, M. Hassenzahl, and A. Albu-Schaffer, "A Human-centered Approach to Robot Gesture Based Communication within Collaborative Working Processes," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 3367–3374, 2011.

[10] K. Kobayashi and S. Yamada, "Motion Overlap for a Mobile Robot to Express its Mind," *Advances in Human-Robot Interaction*, vol. 11, no. 8, pp. 964–971, 2007.

[11] Y. Wakita, S. Hirai, and T. Hori, "Knowledge Projection on Robot Task Environment," in *Robot and Human Communication, 1997. RO-MAN '97. Proceedings., 6th IEEE International Workshop on*, pp. 136–141, oct. 1997.

[12] Y. Wakita, S. Hirai, T. Suehiro, T. Hori, and K. Fujiwara, "Information Sharing via Projection Function for Coexistence of Robot and Human," *Autonomous Robots*, vol. 10, pp. 267–277, 2001.

[13] A. M. Zanchettin, L. Bascetta, and P. Rocco, "Acceptability of Robotic Manipulators in Shared Working Environments through Human-like Redundancy Resolution," *Applied Ergonomics*, apr 2013.

[14] T. Matsumaru, "Mobile Robot with Preliminary-announcement and Indication of Scheduled Route and Occupied Area using Projector," in *Mobile Robots Motion Planning, New Challenges*, ch. 18, pp. 361–380, Published online, jul. 2008.

[15] T. Matsumaru, "Experimental Examination in Simulated Interactive Situation between People and Mobile Robot with Preliminary-Announcement and Indication Function of Upcoming Operation," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 3487–3494, may 2008.

[16] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A Survey of Socially Interactive Robots," *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 143–166, 2003.

[17] C. L. Sidner and C. Lee, "Robots as Laboratory Hosts," *interactions*, vol. 12, pp. 24–26, Mar. 2005.

[18] K. Kobayashi and S. Yamada, "Informing a User of Robot's Mind by Motion," in *The third International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2005), SS4B-3*, 2005.

[19] K. Kobayashi, Y. Kitamura, and S. Yamada, "Action Sloping as a Way for Users to Notice a Robot's Function," in *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, pp. 445–450, 2007.

[20] P. Milgram, S. Zhai, D. Drascic, and J. Grodski, "Applications of Augmented Reality for Human-Robot Communication," in *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 3, pp. 1467–1472 vol.3, 1993.

[21] A. Tang, C. Owen, F. Biocca, and W. Mou, "Comparative Effectiveness of Augmented Reality in Object Assembly," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, (New York, NY, USA), pp. 73–80, ACM, 2003.

[22] C. Bethel and R. Murphy, "Review of Human Studies Methods in HRI and Recommendations," *International Journal of Social Robotics*, vol. 2, pp. 347–359, Dec. 2010.

[23] A. Whiten, *Natural Theories of Mind*. Oxford: Blackwell, 1991 1991.

[24] S. Baron-Cohen, *Mindblindness: An Essay on Autism and Theory of Mind*. The MIT Press, 1997.

[25] J. Flavell, "Theory-of-mind development: Retrospect and prospect," *Merrill-Palmer Quarterly*, vol. 50, no. 3, pp. 274–290, 2004.

[26] D. Premack and G. Woodruff, "Does the Chimpanzee have a Theory of Mind?," *Behavioral and Brain Sciences*, vol. 1, pp. 515–526, 12 1978.

[27] J. Scholtz, B. Antonishek, and J. Young, "Evaluation of a Human-Robot Interface: Development of a Situational Awareness Methodology," in *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pp. 9 pp.–, 2004.

[28] E. Adam, "Fighter Cockpits of the Future," in *Digital Avionics Systems Conference, 1993. 12th DASC., AIAA/IEEE*, pp. 318–323, 1993.

[29] "Wikipedia, the free encyclopedia." `http://en.wikipedia.org/wiki/Situation_awareness`, aug. 2013.

[30] M. R. Endsley and D. J. Garland, *Situation Awareness Analysis and Measurement*. Lawrence Erlbaum Associates, 2000.

[31] T. Fong, C. Thorpe, and C. Baur, "Collaboration, Dialogue, and Human-Robot Interaction," in *In 10th International Symposium of Robotics Research*, Springer-Verlag, 2002.

[32] T. W. Fong, C. Thorpe, and C. Baur, "Robot, Asker of Questions," *Robotics and Autonomous Systems*, 2003.

[33] T. Matsumaru, H. Endo, and T. Ito, "Examination by Software Simulation on Preliminary-Announcement and Display of Mobile Robot's Following Action by Lamp or Blowouts," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 1, pp. 362–367, sept. 2003.

[34] T. Matsumaru, "Mobile Robot with Preliminary-announcement and Display Function of Forthcoming Motion using Projection Equipment," in *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, pp. 443–450, sept. 2006.

[35] T. Matsumaru, T. Kusada, and K. Iwase, "Mobile Robot with Preliminary-Announcement Function of Forthcoming Motion using Light-ray," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 1516–1523, oct. 2006.

[36] T. Matsumaru, "Mobile Robot with Preliminary-announcement and Indication Function of Forthcoming Operation using Flat-panel Display," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1774–1781, apr. 2007.

[37] Y. Wakita, S. Hirai, and T. Suehiro, "Request from Robot to Person in Information Sharing," in *Robot and Human Interaction, 1999. RO-MAN '99. 8th IEEE International Workshop on*, pp. 231–236, 1999.

[38] S. Green, M. Billinghurst, X. Chen, and G. Chase, "Human-Robot Collaboration: A Literature Review and Augmented Reality Approach in Design," *International Journal of Advanced Robotic Systems*, pp. 1–18, 2007.

[39] R. Bischoff and A. Kazi, "Perspectives on Augmented Reality Based Human-Robot Interaction with Industrial Robots," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 4, pp. 3226–3231 vol.4, 2004.

[40] J. L. Drury, J. Richer, and N. Rackliffe, "Comparing Situation Awareness for Two Unmanned Aerial Vehicle Human Interface Approaches," in *In Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR*, 2006.

[41] P. Milgram and F. Kishino, "A Taxonomy of Mixed Reality Visual Displays," *IEICE Transactions on Information Systems*, vol. E77-D, dec. 1994.

[42] "Wikipedia, the free encyclopedia." `http://en.wikipedia.org/wiki/Mixed_reality`, jun. 2013.

[43] D. Van Krevelen and R. Poelman, "A Survey of Augmented Reality Technologies, Applications and Limitations," *The International Journal of Virtual Reality*, vol. 9, pp. 1–20, June 2010.

[44] R. T. Azuma, "A Survey of Augmented Reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, pp. 355–385, 1997.

[45] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent Advances in Augmented Reality," *IEEE Computer Graphics and Applications*, vol. 21, pp. 34–47, nov. 2001.

[46] N. Tanke, "Forward Projection of Robot Intentions using Augmented Reality for Small and Medium-sized Enterprises," in *MSc literature study*, 2013.

[47] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic, "Augmented Reality Technologies, Systems and Applications," *Multimedia Tools and Applications*, vol. 51, pp. 341–377, jan. 2011.

[48] G. Papagiannakis, G. Singh, and N. Magnenat-Thalmann, "A Survey of Mobile and Wire-

less Technologies for Augmented Reality Systems," *Comput. Animat. Virtual Worlds*, vol. 19, pp. 3–22, feb. 2008.

[49] F. Zhou, H. Duh, and M. Billinghurst, "Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR," in *Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on*, pp. 193–202, sept. 2008.

[50] R. Azuma, "Tracking Requirements for Augmented Reality," *Commun. ACM*, vol. 36, pp. 50–51, July 1993.

[51] H. Kato and M. Billinghurst, "Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System," in *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, IWAR '99, pp. 85–94, 1999.

[52] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose Tracking from Nnatural Features on Mobile Phones," in *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pp. 125–134, 2008.

[53] M. Fiala, "ARTag, a Fiducial Marker System Using Digital Techniques," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '05, pp. 590–596, 2005.

[54] E. Olson, "AprilTag: A Robust and Flexible Visual Fiducial System," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3400–3407, IEEE, May 2011.

[55] D. Olsen and M. Goodrich, "Metrics for Evaluating Human-Robot Interactions," in *Proc. NIST Performance Metrics for Intelligent Systems Workshop*, 2003.

[56] B. Donmez, P. Pina, and M. Cummings, "Evaluation Criteria for Human-Automation Performance Metrics," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, pp. 77–82, 2008.

[57] Willow Garage, *Open Source Computer Vision Library.* Available: http://opencv.willowgarage.com/, 2013.

[58] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an Open-Source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[59] D. Kirsh, "The Intelligent use of Space," *Artificial Intelligence*, vol. 73, no. 1–2, pp. 31 – 68, 1995.

[60] F. Biocca, A. Tang, D. Lamas, J. Gregg, R. Brady, and P. Gai, "How do Users Organize Virtual Tools Around Their Body in Immersive Virtual and Augmented Environment?: An Exploratory Study of Egocentric Spatial Mapping of Virtual Tools in the Mobile Infosphere," tech. rep., Citeseer, 2001.

[61] J. Gage, *Color and Culture: Practice and Meaning from Antiquity to Abstraction.* University of California Press, 1999.

[62] P. K. Kaiser, "Physiological Response to Color: A Critical Review," *Color Research and Application*, vol. 9, no. 1, pp. 29–36, 1984.

[63] F. Faul, E. Erdfelder, A. Buchner, and A.-G. Lang, "Statistical Power Analyses using G*Power 3.1: Tests for Correlation and Regression Analyses," *Behavior Research Methods*, vol. 41, no. 4, pp. 1149–1160, 2009.

[64] M. S. Banks, J. R. Read, R. S. Allison, and S. J. Watt, "Stereoscopy and the Human Visual System," *SMPTE Motion Imaging (also appears in SMPTE International Conference on Stereoscopic 3D for Media and Entertainment Conference proceedings)*, p. in

press, 2012.

[65] H. Fang, S. Ong, and A. Nee, "Interactive Robot Trajectory Planning and Simulation using Augmented Reality," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 227–237, 2012.

[66] M. Fiala, "Comparing ARTag and ARToolkit Plus Fiducial Marker Systems," in *Haptic Audio Visual Environments and their Applications, 2005. IEEE International Workshop on*, oct. 2005.

[67] G. Schweighofer and A. Pinz, "Robust Pose Estimation from a Planar Target," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, pp. 2024–2030, dec 2006.

[68] M. Segal and K. Akeley, "The Design of the OpenGL Graphics Interface," tech. rep., Silicon Graphics Computer Systems, 1994.

[69] J. Heikkila and O. Silven, "Calibration Procedure for Short Focal Length Off-the-shelf CCD Cameras," in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, vol. 1, pp. 166–170, aug. 1996.

[70] J. Heikkila and O. Silven, "A Four-step Camera Calibration Procedure with Implicit Image Correction," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pp. 1106–1112, jun. 1997.

[71] R. Diankov and J. Kuffner, "OpenRAVE: A Planning Architecture for Autonomous Robotics," Tech. Rep. CMU-RI-TR-08-34, Robotics Institute, Pittsburgh, PA, July 2008.

[72] "Open Motion Planning Library." `http://ompl.kavrakilab.org/`, jul. 2012.

[73] J. Kuffner, J.J. and S. LaValle, "RRT-connect: An Efficient Approach to Single-Query Path Planning," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 2, pp. 995–1001, 2000.

[74] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, pp. 566–580, aug. 1996.

[75] R. Bohlin and L. Kavraki, "Path Planning using Lazy PRM," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 1, pp. 521–528, 2000.

[76] G. Sánchez and J.-C. Latombe, "A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking," in *Robotics Research* (R. Jarvis and A. Zelinsky, eds.), vol. 6 of *Springer Tracts in Advanced Robotics*, pp. 403–417, Springer Berlin Heidelberg, 2003.

[77] D. Hsu, J.-C. Latombe, and R. Motwani, "Path Planning in Expansive Configuration Spaces," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 3, pp. 2719–2726, apr. 1997.

[78] I. Sucan and L. Kavraki, "Kinodynamic Motion Planning by Interior-Exterior Cell Exploration," in *Algorithmic Foundation of Robotics VIII* (G. Chirikjian, H. Choset, M. Morales, and T. Murphey, eds.), vol. 57 of *Springer Tracts in Advanced Robotics*, pp. 449–464, Springer Berlin Heidelberg, 2009.

[79] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with Provable Bounds on Sub-Optimality," in *In Aadvances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference (NIPS-03*, 2003.

[80] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, pp. 100–107, jul. 1968.

[81] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime Dynamic A*: An Anytime, Replanning Algorithm," in *In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS*, 2005.

[82] A. Stentz, "Optimal and Efficient Path Planning for Partially-Known Environments," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 3310–3317, may 1994.

[83] N. Ratliff, M. Zucker, J. Bagnell, and S. Srinivasa, "CHOMP: Gradient Optimization Techniques for Efficient Motion Planning," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 489–494, may 2009.

[84] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic Trajectory Optimization for Motion Planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4569 –4574, may 2011.

[85] Z. Zhang, "Flexible Camera Calibration by Viewing a Plane from Unknown Orientations," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, pp. 666–673 vol.1, 1999.