Improving Robots' Transparency with Mobile Augmented Reality

Rotsidis, Alexandros

Award date:
2019

Awarding institution:
University of Bath

Improving Robots' Transparency with Mobile Augmented Reality

Alexandros Rotsidis

Master of Science

The University of Bath

March 2018

Improving Robots' Transparency with Mobile Augmented Reality

submitted by

Alexandros Rotsidis

for the degree of Master of Science of the

University of Bath

March 2018

DECLARATION

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Master of Science in the Department of Computer Science. No portion of the work in this thesis has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signature of Author.....

Alexandros Rotsidis

Abstract

Robots are increasingly becoming more common and found in social, work and even healthcare environments. Robots' actions and reasoning are becoming more complex it is harder for humans to understand what the intentions or future actions of the robot are. Augmented reality is an emerging technology today, and during the last ten years it has seen a huge interest from a large number of industries and the academia. The rise of smartphones and the improvements in mobile computing performance has allowed Augmented Reality to slowly become more mobile and affordable. This research project proposes that mobile augmented reality can be used to create a real-time feed of the robot's AI and visualize it on a user's mobile device thus improving trust between both and user's perception about the robot, and radically increase transparency.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As robots are becoming more complex they also are assigned important tasks, such as in healthcare (surgeries) and becoming more integral for various scenarios in the industry. In the very near future more and more people without any background in STEM (Science, Technology, Engineering, and Mathematics)[1] or robotics will have to interact with robots. They should be able and have tools that will allow them to understand, judge and predict a robot's behaviour and intent. In this research project the term robot is used; it might refer to an autonomous agent, a self driving car, a robotic arm - anything that runs a form of AI. There will be many video (Youtube) links in this document, we encourage the reader to view them while reading.

1.1 Motivation

The motivation for this project is the hypothesis that AR (Augmented Reality) can greatly help in increasing transparency in robots, and reducing the gap in Human-Robot-Interaction (HRI) and collaborative scenarios, between a user and a robot. The latter will have a numerous beneficial outcomes if it is proven that AR (which is a promising upcoming technology) can help in achieving so. The future capabilities and opportunities that this offers are endless because of the reason that robots are becoming more and more an integral part of our lives. For example they can be seen present in education (Johal et al., 2018), industry (Quitter et al., 2017), healthcare (Riek, 2017) and there are also myriads of other places where they exist. Assigning more responsibilities

[1] https://en.wikipedia.org/wiki/Science,_technology,_engineering,_and_mathematics

or vital tasks to robots leads to greater damage output in case of an error. By providing an insight to the robot's code that is responsible for its decisions, we can increase utility output. We define a robot's AI in this thesis as the ability to perform the right action at the right time. Theodorou, Wortham and Bryson (2017) wrote *'We believe the implementation and usage of intelligent systems which are fundamentally transparent can help not only with debugging AI, but also with its public understanding, hopefully removing the potentially-frightening mystery around why that robot behaves like that".* There is recent work that has looked into AR and robots behaviour such as Walker et al. (2018), but overall this is still a new area as we will prove in Chapter 2. Personal curiosity to learn more about AR and its contribution to robotics provided further motivation. Last but not least Chatzopoulos et al. (2017) have written that there is no *killer-app* yet when it comes to AR, which shows off its true potential. There is room for a new killer-app in AR to be developed. Thus we decided to develop a such app.

1.2 Goal

The goal of this project will be to examine if AR can increase transparency in robots by developing an Android app that can efficiently track the R5 robot (i.e using video object tracking - VOT). One of the hypotheses is that AR can improve transparency in robots, thus increasing trust. The problem that we are trying to solve is that transparency is not always understandable for end users or robot designers and we believe that through this project AR can help in reducing the gap of ambiguity and uncertainty that can exist between a robot and a human being. By using certain existing tools that we will introduce in Chapter 3 we will allow the user of the app to view the robot's AI, (i.e current execution plans). By using our deliverable, the Android AR application, developed for this research. The user should be able to improve their understanding of the current robot's actions that he is observing, thus increasing his trust and perception for the robot. This will be achieved using an existing platform that was developed in Bath University, called the *R5 Robot* also used in previously published research (Wortham, Theodorou and Bryson, 2017a), (Wortham and Rogers, 2017) and *ABOD3* that was developed by Theodorou (2017). This research project aims to allow everyday users to answer questions such as *'Why is my robot doing this ?'.*

1.3 Layout of Thesis

Chapter 1 briefly presents the purpose, motivation and goal of this project. Chapter 2 gives a solid background on the current literature related to transparency in robots, AR and HRI (Human Robotic Interaction). Chapter 3 will explain the already existing tools and software that were used by previous research in order to develop this. The approaches that we considered for VOT are explained in Chapter 4. The main development of the app is explained in Chapter 5. Given the ample amount of work done this was split in two parts. The image processing and video tracking app development is described in Chapter 6. The field study, data collection carried out and discussing the end results of it are found in Chapter 7. The conclusion and future work is found in Chapter 8 and Chapter 9 respectively.

Chapter 2

Literature Review

This chapter will provide background knowledge regarding AR and transparency in robots. A literature review (previous work) of the current research follows, and some background knowledge prerequisites for the reader.

2.1 Transparency and Robots

The EPSRC Principles of Robotics have a definition when it comes to transparency and robots i.e: *'Robots are manufactured artefacts. They should not be designed in a deceptive way to exploit vulnerable users; instead their machine nature should be transparent'* (Boden et al., 2017). This means that robots or agents should minimise deception, and the lack of transparency could lead to exploitation (Wortham and Theodorou, 2017). There is a strong relationship between a robot's transparency, users' trust and understanding (Wortham, Theodorou and Bryson, 2017a). There has also been a considerable amount of the converse research in robots understanding humans (Lee and Makatchev, 2009),(Salem et al., 2015). It has been proven that transparency and trust plays a significant role in trusting a robot. *'Trust is only one of a number of critical elements essential to human-robot collaboration, but it continues to be a growing concern as robots advance in their functionality.'* (Hancock et al., 2011). Breazeal et al. (2005) have proven that in humans and robots collaborative scenarios if cues are well communicated that is if, there is enough transparency, the quality of teamwork increases. *'Increased robot transparency is associated with reduced assignment of credit or blame to the robot, and increased assignment to humans'* (Wortham, Theodorou and Bryson, 2017a).

'Humans must accept and trust a robot before effective interaction can occur' (Billings et al., 2012).

2.2 Mobile Augmented Reality

Augmented reality (AR) is the superimposition of computer generated graphical objects on a user's view of the real world executed in real time (Azuma, 1997). AR combines the real world with added digital visual information and allows real time interaction with them. The visual additions usually provide information that the user cannot detect without the help of AR (Azuma, 1997), in this research that would be the robot's A.I. According to He et al. (2017), *'AR is developed from VR'*, virtual reality. AR utilises visual information generated by a computer and mixes that with the real environment that the users observe in order to provide more information. Three dimensional viewing and manipulation is superior to traditional mouse and screen interactions, for 3D models as it was proven in Szalavári et al. (1998). This is one of the reasons we have turned into looking in AR. Applications of augmented reality can range from advertising, edutainment, education, engineering, medicine to industrial manufacturing (Imbert et al., 2013). In this chapter we will elaborate on the research that was carried out regarding augmented reality and HRI. Nowadays handheld mobile devices are used in our daily lives for a wide variety of applications. The majority of people today own a smartphone that is capable of running basic AR tasks. The improvement in mobile computing has allowed for the evolution of mobile augmented reality.

2.3 AR and HRI

The research related to AR today is vast. It is applied in new fields and industries constantly from shipbuilding (Blanco-Novoa et al., 2018), construction sites (Chu, Matthews and Love, 2018),(Zaher, Greenwood and Marzouk, 2018) to architecture (Steven et al., 1996). Redondo et al. (2013),Camba et al. (2016) have used Mobile augmented reality or Hand Held Augmented Reality (HHAR), for applications in education from elementary to higher levels. AR using HMD (Head Mounted Displays) has been very recently used with robots (Walker et al., 2018) to the task of improving collaboration.

Since augmented reality is a rapidly evolving field, research in AR and applying it in Human-Robot-Interaction scenarios has recently started to show up, Andersson et al. (2016) have proposed that AR Enhanced Human-Robot-Interaction can improve training, programming, maintenance and process monitoring, with focus on programming the robots. The question that they address in their paper is *'How can users program a robot to execute a task without the need to use a programming language?'*. Their work was mostly focused on providing a roadmap and a methodology for an AR application that will allow a user to program a robot, by presenting a prototype using mostly off-the-self software. As we will see further on, they also focused on industrial robots, not domestic ones or social robots. We have developed the AR app from square one which makes our project more novel.

Before we move more into depth it is worth mentioning that there is previous research of people trying to make robots more understandable. A good example that fits this is Baraka, Rosenthal and Veloso (2016), where they tried to increase human understanding of a mobile robot's state and actions using expressive lights. We found this very interesting as it can be seen as an alternative to using AR in the attempt of helping people understand robots. In contract to most research listed in this literature review, we have experimented on domestic-like robots (R5 robot). Baraka, Rosenthal and Veloso (2016) used CoBot[1], seen in figure 2-1. Their results showed that *'the presence of lights on a mobile robot can significantly help people understand the robot's state and actions'* (Baraka, Rosenthal and Veloso, 2016).

Research revolved around social robots and AR proposed semi-real robot agents. Subin, Hameed and Sudheer (2017) have designed a system using a Firebird-XII robot[2], as a moving base and by installing markers on it they where able to superimpose virtual characters in AR. The purpose of this paper was to lower the cost of socially interactive robots, but they have reduced at the same time the physical interaction capabilities of the robot since it is just a moving base in real life. Transparency is not explicit in Subin, Hameed and Sudheer (2017) but do mention that this idea can be used to create influence in children with disabilities. They also propose future work can be carried out to make the robot more interactive by receiving commands from the user. Sending commands to a robot, was set as future work for our project.

[1] http://www.cs.cmu.edu/ coral/projects/cobot/

[2] http://www.nex-robotics.com/products/fire-bid-xii.html

Figure 2-1: The Cobot.

There is research about using AR in a Human-Robot-Interaction environment, much of it is focused in industrial environments as seen in (Andersson et al., 2016). Michalos et al. (2016) have presented a tool that use AR to support operators in industrial environments in which humans and robot collaborate. They mostly focused on visualising the production steps to minimise errors and on very specific type of robots (arm robots used in manufacturing). The main goal of that research was to enhance users' safety by visualising the robot's motion and production related data, and also increase productivity in an industrial environment. They did not focus on the transparency aspect that this project is trying to explore. Unlikely that project, we focused mostly on domestic-looking type of robots (i.e R5 Robot). Michalos et al. (2016) are keen on researching further this by using AR glasses, that way the user can be more productive since they are heavily focused in increasing productivity in assembly lines using AR. They do mention aswell that future work can include more research about the robot's transparency, even having transparency levels. Michalos et al. (2016) used markers to achieve the tracking, we used marker-less AR, which adds to the novelty of our project.

Similarly Makris et al. (2016) propose an AR solution that would again

support operators in the assembly process (they focus in the industrial sector), providing the operator visual production data. One of their AR-based system task was to provide a *'safety feeling'* and acceptance, avoiding potentially hazardous situations. To be more exact the data they provide back to the user is: *'Assembly process information provision, robot workspace and trajectory visualization, audio/visual alerts and production data'* (Makris et al., 2016). Michalos et al. (2016) focused in increasing productivity output, but in Makris et al. (2016) user's safety was more of a concern. As it can be observed, *transparency* is crucial when it comes to human-robot-collaborative environments.

Walker et al. (2018) have explored how Augmented Reality can mediate collocated human-robot interactions by superimposing the robot's intents on the real environment. They have used HMD (Head Mounted Displays) in their research. We used mobile phones, a cheaper alternative. Walker et al. (2018) have stated *'We found that several of our AR designs significantly improved objective task efficiency over a base-line in which users only received physically-embodied orientation cues'* (Walker et al., 2018). Similarly findings from Wortham, Theodorou and Bryson (2017a), show that by visualising a robot's priorities in real-time can significantly improve human understanding of the machine's intelligence even for naive users. Walker et al. (2018) focus in collaborative environments between robots and humans. Also it is worth noting that the robots that they used for evaluating this were drones, they lacked anthropomorphic or zoomorphic features. The improvement identified in Walker et al. (2018) is the tracking of the robot i.e precise robot localization and navigation, is achieved by motion tracking. In this research project, tracking is achieved using video object tracking, more in Chapter 4.

Robots are present in many collaborative environment working along with humans and are becoming increasingly common in workplaces (Rudall, 2004). For many years robots and humans have been collaborating and as technology gets better people and robots work much more closely together (Kim and Hinds, 2006). There is always the risk of a machine malfunctioning or a misinterpretation of a message, or poor communication between a robot and a human. This can have devastating results especially in industrial situations or even everyday lives. Autonomous vehicles failing[3] and killing drivers. Alemzadeh et al. (2013) proved that 64.3% of computer-based medical surgery

[3]http://en.wikipedia.org/w/index.php?title=Tesla%20Autopilot&oldid=836349404

failures in clinical settings were because of software malfunction. This relates to robots[4]. When a robot malfunctions the surgeons have little control over it, resulting in unwanted consequences. We believe that allowing the surgeon having a better understanding of what the robot is doing, accidents can be reduced or even prevented. Breazeal et al. (2005) have found that transparency reduces conflicts and when errors happen during any task execution, recovery from it is still possible while minimising the allocation time for blaming. Robots are handling more complex tasks and more power is assigned to them. In collaborative environments a balance between the user and the robot needs to exist. Trusting the robot is a determinative component in this.

'Our aim is to not only use a MR headset for visualizing data, but to also integrate its sensor data, giving the user a more direct interface to the robot. This way, on the one hand the user can always be aware of the current robot status and ***intent.****'* (Renner et al., 2018). Real world robots' need to operate in dynamic and uncertain environments, and to react quickly as their environment changes is important (Wilkins et al., 1995). The most relevant questions around our problem have been identified in Wortham, Theodorou and Bryson (2017b), such as *'Why is the robot doing X behaviour'*?, which for the developer really means *'What code within the robot is executing to drive this behaviour?'*. They have also stated that observers and users might ask similar questions but slightly different; *'What is the robot trying to achieve by doing X behaviour? What is the purpose of this behaviour?'* (Wortham, Theodorou and Bryson, 2017b). We asked our participants similar questions too. Transparency also helps in facilitating traceability in case of a malfunction of a robot. This can lead to recreation of events and in controlled environments as they did in Wortham, Theodorou and Bryson (2017a). The novelty in Wortham, Theodorou and Bryson (2017a) is that instead of them focusing on humans-robots collaboration as we have seen in previous examples especially in industrial scenarios, they concentrate on unplanned robot encounters between humans. Users were asked to look at a video of a robot interacting with a researcher or look at the scene directly and come up with a theory of what the robot is doing and why. Their results have shown that providing a visual real-time transparency feed, using displays, of the robot's intelligence users were able to better understand the robot. Some participants' overestimated the robot's abilities (Wortham, Theodorou and Bryson, 2017a). By intelligence

[4]https://csl.illinois.edu/news/study-questions-safety-popular-robotic-surgical-device

here it is meant the decision mechanism of the robot. We aim to improve this in our project by using MAR (mobile augmented reality) and real-time transparency, which makes this project more novel.

Interesting research that we came across from Kim and Hinds (2006), where they conducted an observational study of a delivery robot in a hospital. To add more details it was a *Pyxis Mate* model, and its main functionalities were to deliver medication from the pharmacy units to nursing units in the hospital, navigating through hallways, call the elevator and ask for specific medications (Kim and Hinds, 2006). Their findings were quite intriguing as they finally suggested that when a robot is transparent and it explains to the user its actions in a understandable language people will help in accurately attribute credit or blame. As they have written: *'Therefore transparency should be considered when designing autonomous robots'* (Kim and Hinds, 2006).

Similarly Sanders et al. (2014) have investigated different types of data sent from the robot to the user, that included visual, audio and text how would it affect the users trust levels. The results were that the visual cues led to the most increased levels of trust between the robot and the human user. That is another reason why AR as chosen. There was also a drawback identified, that is the increase in trust came from the ability of the user to being able to constantly monitor the robot, and that might reduce productivity especially in industrial environment such as assembly lines.

In a user study that was carried out in Wortham, Theodorou and Bryson (2016) it was shown that there is a strong correlation between the participants' mental models of a robot, and the provision of additional transparency data. Wortham, Theodorou and Bryson (2016) used ABOD3, a software tool that we explain more in section 3.2. The necessary data was presented to the user through a screen. This is where we plan to replace the screen with MAR. One of the reasons is the location restrictions when using a screen. The robot could be anywhere in space, but the screen is stationary. The observer/designer could be standing behind the robot aswell, being able to move around the robot is a huge advantage.

'There is a significant correlation between the accuracy of the participants mental models of the robot, and the provision of the additional transparency data provided by ABOD3. We have shown that a real-time display of a robots decision making produces significantly better understanding of that robots intelligence, even though that understanding may still include wildly inaccu-

rate overestimation of the robots abilities' (Wortham, Theodorou and Bryson, 2016).

As we have seen there is a strong correlation between transparency, trust, productivity and we believe that MAR, although its contribution in the latter is still in early stages, can help in increasing transparency so that end users or even robot designers can benefit from it in term of increased productivity, better debugging, understanding why errors have happened. The increase in trust might lead in a better acceptance of robots from the general public. *'Unacceptable levels of anxiety, fear and mistrust will result in an emotional and cognitive response to reject robots'* (Wortham and Theodorou, 2017).

Chapter 3

Resources

In this section we elaborate more on previous work that was used as a starting point for our software system, but also explain a bit more about the theory behind it.

3.1 POSH and Instinct Planner

3.1.1 POSH

In this section we will very briefly explain that POSH [1] (Parallel-rooted, Ordered Slip-stack Hierarchical) plans are and the Instinct planner aswell. POSH dynamic plans are structures used for action selection. They are called dynamic plans because the next action depends on what the sensors of the robot are reading from the environment's current state, while working towards a predefined goal. The means by which a robot (agent to be more general) determines at any moment what to do next (strongly biologically inspired), by retrieving the action from an existing data structure. This leads to intelligent action selection. The leaves in a POSH plan can be primitives, *acts* or senses, and they are used along with aggregates that can be a *Drive collection*, *Competences* or *Action patterns*. A Drive collection can contain one or more *Drives*. An example of a drive can be *Recharge Robot*, that leads the robot back to its charging station. These are triggered by sensory input from the robots (by a releaser) and traverse down the tree where Competences, Action Patterns and Actions follow. The leaf nodes are real word actions, such as

[1] http://www.cs.bath.ac.uk/ jjb/web/posh.html

turn left, *flash light* (in the case of the R5 robot). It is important to note that each drive in a POSH plan has a priority that determines which Drive should execute; this can also lead to cancellation of a Drive and let another drive take over. POSH is part of *Behavior Oriented Design (BOD)*[2] which a methodology used to develop intelligent agents (in this scenario robots), but it can create intelligence in virtual characters aswell. It was developed by Bryson (2002). The purpose of BOD is to make the process intelligent agents as easy as possible. It is based on object oriented design and behaviour based AI. The main idea behind it is that living organism always sense the world around them, respond to outside stimuli and based on a hierarchical set of behaviours they act accordingly. Since it is impossible to explain BOD and POSH thoroughly here, more research can be found in Prof. Joanna Bryson's webpages[3][4].

3.1.2 Instinct Planner

The Instinct Planner[5] is software written in C++, by Wortham, Gaudl and Bryson (2016). Wortham, Gaudl and Bryson (2016) added some enchantments and extensions of POSH (Rohlfshagen and Bryson, 2010), (Gaudl and Bryson, 2014). That is it can load a POSH plan and execute it at variable number of cycles in at a specific frequency. It also features *Drive Execution Optimisation (DEO)* that avoids traversing the whole plan in order to execute an action. It was specifically developed and designed for low power chips and it is really small in size. This makes it ideal for robots built on the Arduino platform such as the R5 robot. Usually when a Drive is released from a robot's sensor for example, it will traverse the plan all the way down to the correct leaf which can be an Action, or the traversal just fails. The Instinct Planner loads up *Instinct plans*, one of the main reasons is their smaller memory footprint compared to the POSH plans. The Instinct planner is the first tool based on Bryson's work that allows applications of POSH plans on real time platforms such as the Arduino on the R5 robot.

[2] http://www.cs.bath.ac.uk/ jjb/web/bod.html
[3] http://www.cs.bath.ac.uk/ jjb/
[4] https://en.wikipedia.org/wiki/Joanna_Bryson
[5] http://www.robwortham.com/instinct-planner/

3.1.3 Instinct Server

This is a crucial component that was the main starting point for our system, as explained further in Chapter 5. After the robot was successfully connected to this server the user could type in robot commands in the server's terminal and communicate with it. Simple commands such as `STOP`, that stops the motors and `RATE`, that set the rate (i.e how fast) the plan is executed, could be sent and feedback from the R5 robot would show up back on the server's terminal. It is a 2 way communication. As the plan executes on the robot callbacks send data back to the Instinct server via WiFi, over a TCP/IP stream. Using the Instinct Server a plan can be sent to the R5 robot, and also logs all the Instinct Planner runtime trace in text files. The source code is found in a Github repository[6].

3.2 ABOD3

ABOD3, developed in Java and the JavaFX GUI-framework, is a graphical visualisation, real-time development and debugging tool for BOD agents. A screen-shot of ABOD3 can be found in figure 3-1. *The simple UI and customisation allows the editor to be employed not only as a developers tool, but also to present transparency related information to the end users, helping them to develop more accurate mental models of the agent* (Theodorou, 2017). ABOD3 was tested and used in Wortham, Theodorou and Bryson (2016) as well as Wortham, Theodorou and Bryson (2017a). It is meant to be used a developer's tool but aswell to present transparency related information to the end users, in order to help them develop more accurate mental model of the robot.

Theodorou (2017) and Wortham, Theodorou and Bryson (2017a) stated that the transparency level that ABOD3 provided helped the users to understand the behaviour of the robot, in that case the R5 robot. An extension of this by using AR is what our project will focus on. Reason why we plan to use ABOD3 is that it provides built-in visualisation as shown, editing, and debugging support for any BOD compliant robot, including POSH and Instinct plans. It also allows for expansion as it can be expanded to work with BOD derivatives. It is written in Java 8, in a modular/expendable manner, which is the latest stable version as of now and the code is free available on

[6] https://github.com/rwortham/Instinct-Server

Github[7]. ABOD3 is currently integrated with a TCP/IP server that is uses to communicate with the R5 Robot. There is future work regarding ABOD3, that is releasing a beta version of the editor to inexperienced AI developers and gather feedback on how ABOD3 and its debugging capabilities helped them understand, develop, and tune intelligent agents (Theodorou, 2017).

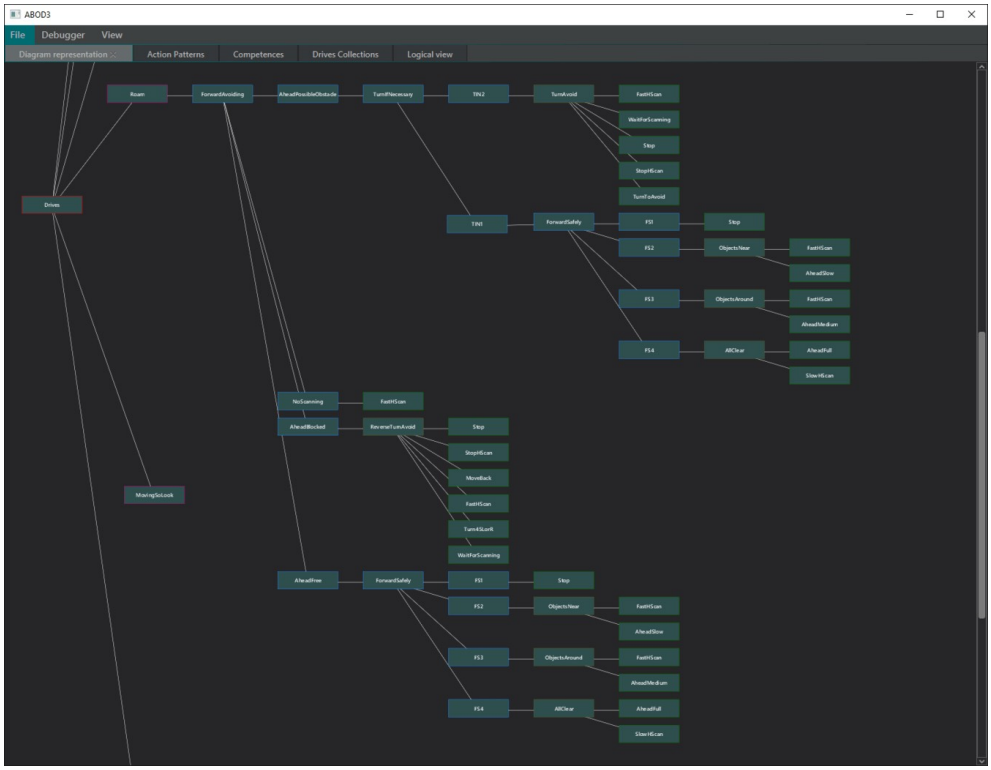


Figure 3-1: UI of ABOD3

3.3 R5 Robot and Google Pixel Phone

3.3.1 R5 Robot

As first presented by Wortham, Gaudl and Bryson (2016), R5 (can be seen in figure 3-2) is a low cost maker robot, based on the Arduino micro-controller[8]. R5's main sensors are active infrared distance sensors at each corner and proprioceptive sensors for odometry and drive motor current. It has a head with

[7]https://github.com/RecklessCoding/ABOD3
[8]https://en.wikipedia.org/wiki/Arduino

two degrees of freedom, designed for scanning the environment. Mounted on the head is a passive infrared (PIR) sensor to assist in the detection of humans, and an ultrasonic range finder with a range of five metres. It also has a multicoloured LED headlights that can be used for communicating to humans around it. It can also vocalise textual sentences and is equipped with a speech synthesis module and a small loudspeaker. In noisy environments, a bluetooth audio module allows wireless headphones or other remote audio devices to receive the vocalisation output. It also has a real-time clock, (RTC) allowing the robot to maintain accurate date and time, a wifi module for communication and an electronically erasable programmable read only memory (EEPROM) to store the robots configuration parameters. The robot software is written as a set of C++ libraries.

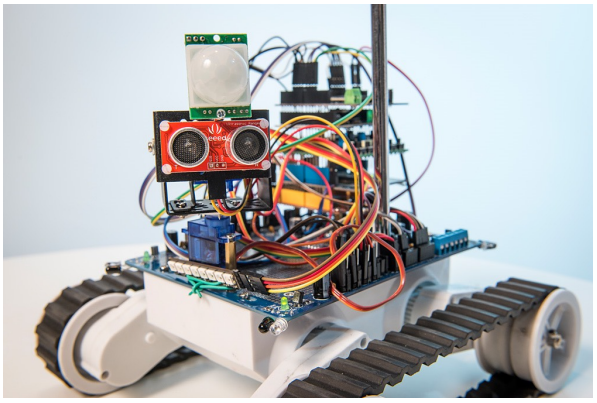


Figure 3-2: The R5 Robot

The robot's instinct plan execution speed is usually set a value from 1 to 10. The normal value is 8. In later chapters it was set to 1 for development reasons, then back to 8. The lower the value the slower the robots' reactions are. If set to 1 it would crash into objects and then try to avoid them. If set to anything above 8 it worked fine. Also more data is sent from it, to the Instinct server when set to a higher number.

3.3.2 Google Pixel Phone

A Google Pixel phone provided by Mr Ken Cameron was used as a test and development platform. Property of the university. The device at this time run the latest Android version 8.1[9]. This device was used along with Android

[9]https://en.wikipedia.org/wiki/Android_Oreo

Studio[10] where it was used to write the necessary code for the app.

[10]https://en.wikipedia.org/wiki/Android_Studio

Chapter 4

VOT Approaches

In this chapter we will explain the *Video Object Tracking (VOT)* methods that we have encountered through our literature review and development. *'Real-time object tracking is the critical task in many computer vision applications such as surveillance perceptual user interfaces,* ***augmented reality*** *...'* (Comaniciu, Ramesh and Meer, 2003). Video Object Tracking is the process of registering a region of interest (ROI) in a frame and then attempt to track that ROI in the following frames in the video stream. This has numerous challenges such as dealing with the ROI's object rotation, change of colour in some cases.

4.1 Challenges

The challenge in this project was to track the R5 Robot, and superimpose (by using Augmented Reality) the ABOD3 real-time execution of an Instinct plan. In this chapter we will brielfy explain the algorithms behind the software (i.e the mobile app). In Chapter 6 we consider these algorithms and their software and explain why each was rejected.

4.2 Augmented Reality Libraries

In the process of trying to identify a suitable AR library to use for our Android app, we identified that none of the available ones was suitable. The main reason is that we had concluded after the initial research proposal that visual

object tracking (i.e video tracking[1]) was more suitable for our scenario. This is because of the R5 robot's nature, it is a constantly moving rover robot. Also the constant movement of the user (means a moving camera in space) and the phone was taken into consideration. *ARCore*[2], does seems to provide some basic tracking but not sufficient. The most promising alternative was *Vuforia* [3], but there is a license that need to be bought in order to use the full potential of the library. The free version of Vuforia includes an unremovable watermark and has limited functionality. Google states for *ARCore*: *'ARCore cannot track a moving image, but it can resume tracking that image after it stops moving.'*[4]. This was not ideal for the project, were the robot has the freedom to move randomly (in variable speed) on a surface.

4.3 QR Marker Tracking

This was was the first approach we looked in. QR codes were considered first. The plan was to register a marker and then track it. This was discarded after a short period of time because of the robot's nature. It was inconvenient to add a 2D rectangular marker on top of the robot large enough to be tracked up to 6 meters. In order visualise the inefficiency of this solution for our particular scenario, figures 6-1 and 6-2 from Chapter 6 shows the R5 Robot with a number of test QR markers that we had printed.

4.4 Circle Hough Transform

The Circle Hough Transform[5] is a specialization of Hough Transform a survey of its derivations can be found in Mukhopadhyay and Chaudhuri (2015). The Hough Transform was introduced in 1962 (Hough 1962)[6] and first used to find lines in images a decade later by Duda and Hart (1972). A Hough transform uses a parameter space or a feature space to achieve its goal of detecting lines,

[1] https://en.wikipedia.org/wiki/Video_tracking
[2] https://developers.google.com/ar/discover/
[3] https://en.wikipedia.org/wiki/Vuforia_Augmented_Reality_SDK
[4] https://developers.google.com/ar/develop/c/augmented-images/
[5] https://en.wikipedia.org/wiki/Circle_Hough_Transform
[6] https://en.wikipedia.org/wiki/Hough_transform

in this case circles. If we consider a line equation,

$$y = mx + b \tag{4.1}$$

and the same equation expressed for a parameter space (for a point (x_1, y_1) on the line):

$$b = -x_1 m + y_1 \tag{4.2}$$

then if we visualize this in figures 4-1 and 4-2 (provided by Hoff (2018)):

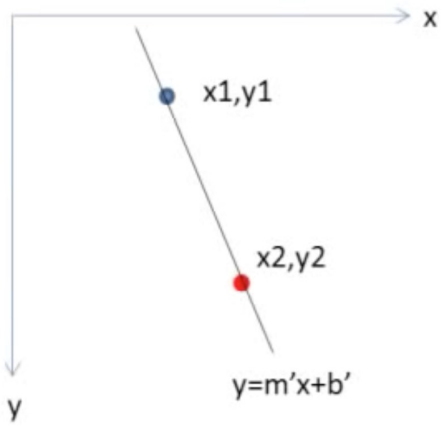


Figure 4-1: X and Y space

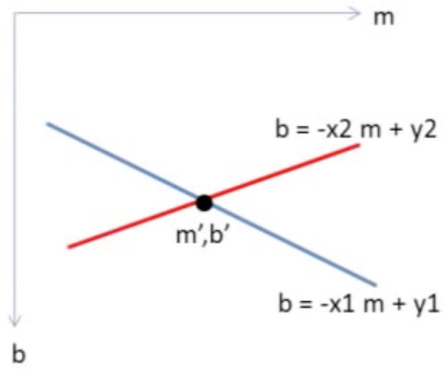


Figure 4-2: M and B space (Note the lines that pass from the intersection point, are formed by the points $(x_1, y_1), (x_2, y_2)$)

Every point on the first space represent a line in the parameter space. A Hough transform also needs a $2D$ accumulator (array) which uses a coordinate system (m, b), with limits m_{min}/m_{max} and b_{min}/b_{max} both user set values (initially at zero). Each time a pixel is found to satisfy $b = -xm + y$ the value in the cell at (m, b) is increased by one (see fig. 4-3). The peaks are chosen from the accumulator and those represent lines that pass through edge points.

As mentioned earlier there are derivations of the Hough transform and of them is the *Circle Hough Transform*, that detects circles. The reason the *Circle Hough Transform* was chosen is because the goal was to track a sphere (Chapter 6 contains more details). A sphere projection's shape from a 3D space to a 2D space is a circle no matter where the camera is in the 3D

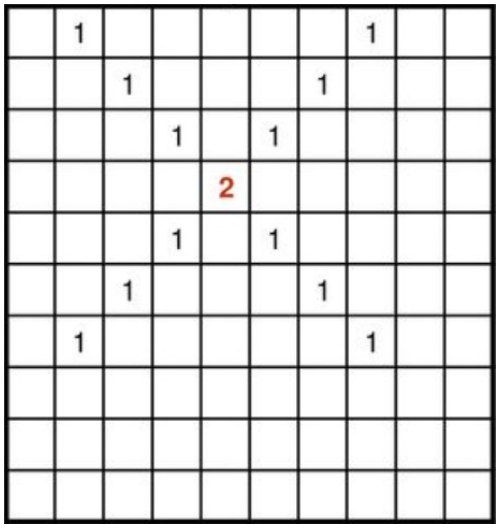


Figure 4-3: An example of a 2D accumulator. Value 2 is a peak.

space. That makes it ideal for tracking as the user holding the phone moves and the robot moves as well. A circle can be described by three parameters the centre (a, b) and a radius r, as seen here $circle_x = a + R * \cos\theta$ and $circle_y = b + R * \sin\theta$ (i.e parametric form), where θ takes values from 0 to 360 then a circle is generated with radius r. There are three parameters here so the accumulator will be three-dimensional. If the radius is known then it will be two-dimensional. Please note that if r is not known then generally different values in a range are tried. That increases processing demands along with the increased sized of the accumulator. In section 6.1 we show that these had major impact on performance.

In order to make the Circle Hough Transform work in OpenCV the original frames needs to go through some preprocessing. The preprocessing generally includes edge detection, noise filtering (such Gaussian filter), image thresholding (so the result is a black and white image with circles), then finally the Circle Hough Transform is applied and centres of circles along with their radius are returned. The techniques used for preprocessing are described below.

4.4.1 Color Thresholding

Color thresholding is a simple technique that given a lower limit and an upper limit (i.e light shade of green and dark shade of green) will return a black and white image. The white pixels will represent the pixel that fall into that range

and the rest will be just black. This is also called colour based segmentation.

4.4.2 Eroding and Dilating

Basic image morphological oparations. Given a binary image *erosion* will remove elements smaller then a defined element (most of the times a squared template). *Dilation* is exactly the opposite as it will increase the size of small areas and connect those small areas if the in-between are is smaller than the structuring element. This is better explained by visualizing it, in figures 4-4 and 4-5, provided by[7].

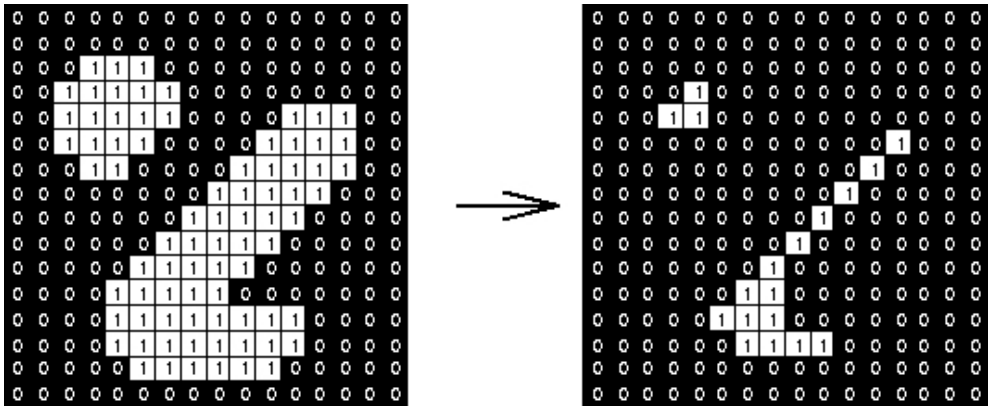


Figure 4-4: An example of erosion given a structuring element 3 by 3

.

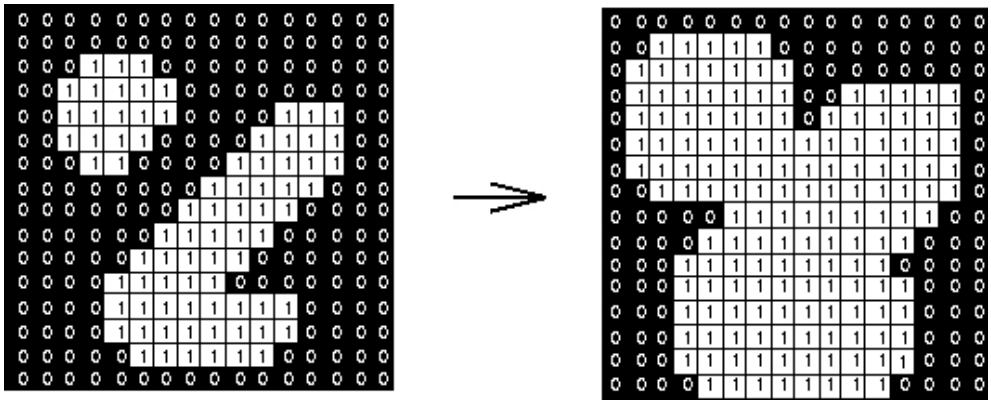


Figure 4-5: Gaps are filled using dilation and a structuring element 3 by 3

.

[7]www.cs.princeton.edu/ pshilane/class/mosaic

This operations were really useful, as stated in Chapter 6, in the removal of unwanted noise in incoming camera frames. The figures in Chapter 6 show the results of applying erosion and dilation and compare the previous figures with noise and the result that has significant noise.

4.4.3 Gaussian Blur

A Gaussian blur is a form of image and template convolution that is convolving an image with a 2D Gaussian function with an image. In simple English kernel convolution is the method of taking a small grid of numbers (can be 3 by 3, 9 by 9 .. $2N + 1$ by $2N + 1$), and in turn we pass that over an image and transforming it based on what those numbers are. We can blur, sharpen/unsharpen the image, apply edge detection and more depending on the values in that kernel. The values in a Gaussian template are generated by the following equation:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.3}$$

And a resulting template can look like this for a 5 by 5 template can look like this:

$$G(x, y) = \begin{bmatrix} 0.0121 & 0.0261 & 0.0337 & 0.0261 & 0.0121 \\ 0.0261 & 0.0561 & 0.0724 & 0.0561 & 0.0261 \\ 0.0337 & 0.0724 & 0.0935 & 0.0724 & 0.0337 \\ 0.0261 & 0.0561 & 0.0724 & 0.0561 & 0.0261 \\ 0.0121 & 0.0261 & 0.0337 & 0.0261 & 0.0121 \end{bmatrix} \tag{4.4}$$

An illustration of the template convolution operation is seen in fig. 4-6.

A Gaussian blur was applied on the result of the erosion and dilation in order to smooth out the circles, and improve the Circle Hough Transform detection. By smoothing extra noise was removed but not completely, more in Chapter 6.

4.5 Circulant Matrices Tracker

This approach is based on previous research from Henriques et al. (2012). The main idea here is that a ROI is chosen and only a sub-window of the frame

[8] https://developer.apple.com

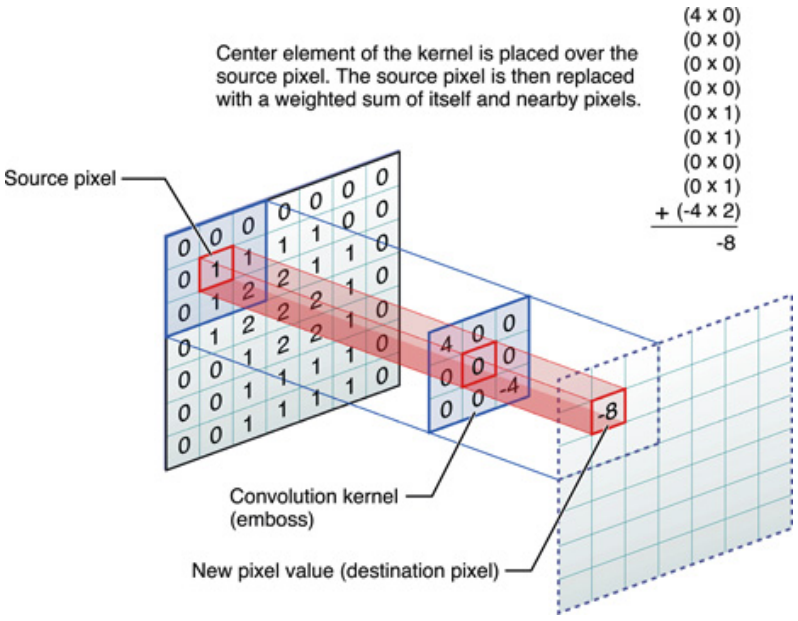


Figure 4-6: Kernel convolution visualilsed [8]

is scanned for that ROI. In this case the ROI is a rectangle defined by the user from the frame that contains the robot. This is called *dense sampling*. In Henriques et al. (2012) they propose solutions that run in speeds $\mathcal{O}(n^2 \log n)$ for n by n images. The tracker in this solution follows this pipeline. Once a ROI is defined, a window double the size is cropped from the input image at the ROI's position and then later on the estimated target's position. Then for each subframe in that subwindow a formula is evaluated and the subframe that returns the highest value is the subframe or new ROI that contains the target, in this case the robot. More details can be found in Henriques et al. (2012). This tracker was chosen to be the final tracker in the application for its simplicity and increased performance compared to the others explained in this chapter. Even though it does not handle rotation of the object being tracked or occlusion, these did not seem to be an issue. The R5's rotation did not cause any issues. This is because of the robot's lack of colour diversity. More technical details in Chapter 6.

4.6 Mean Shift Object Tracking

This approach was the only one that could generate RGB frames in the beginning of the app development. It was chosen because of its speed, but the stability of the tracking was not stable enough compared to the other methods. More about how we overcame this problem in Chapter 6. Since it was used for the majority of the development phase (before switching to TLD, section 4.7 and Circulant Matrices Tracker section 4.5) we will briefly explain it. This method is based on previous research from Comaniciu, Ramesh and Meer (2003) and Comaniciu, Ramesh and Meer (2000), which can deal with partial occlusions, clutter, and target's scale changes in size. The idea remains the same, given a target in the first frame we try to detect it and track it in all the next frames. Mean shift is an algorithm that iteratively shifts a data point to the average of data points in its neighbourhood. Figure 4-7 shows the basic idea behind the Mean Shift algorithm. The mean of points in an area for interest is calculated and then this process is repeated. The final target of the process would be the blue dot in fig. 4-7.

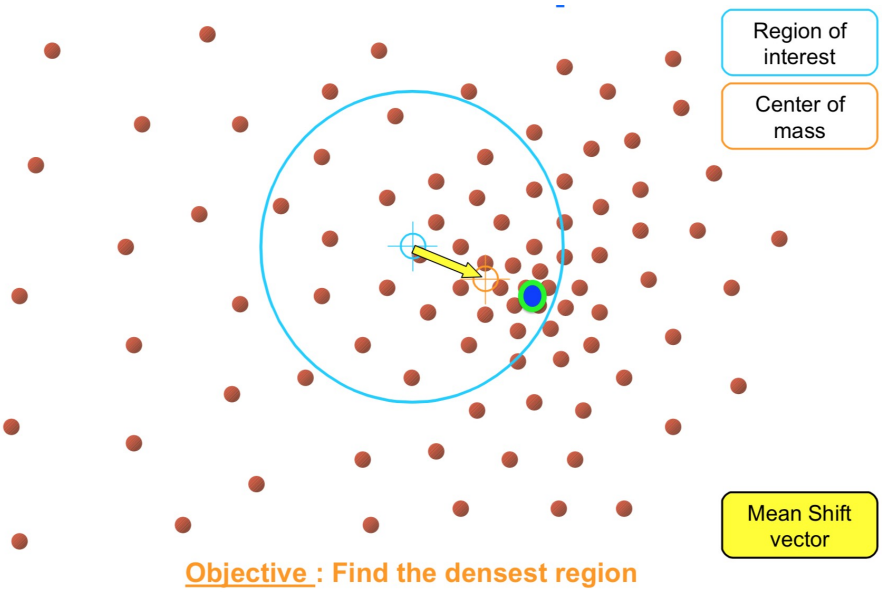


Figure 4-7: Intuitive Description of the Mean Shift algorithm [9]

That is at every iteration $\boldsymbol{x} \leftarrow m(\boldsymbol{x})$ where $m(\boldsymbol{x})$ is the mean of the data samples, and $\boldsymbol{x}$ is merely a sample. The algorithm stops when $m(\boldsymbol{x}) = \boldsymbol{x}$.

[9] http://crcv.ucf.edu/people/faculty/shah.php

Taking into consideration a 10 by 10 grayscale image (100 pixels) that each pixel takes values from 0 to 255, then a histogram of that image can be created by having 255 "bins", and each bin contains the number of those 100 pixels that have that value. The histogram then can be represented as a distribution. This can be seen in fig. 4-8.

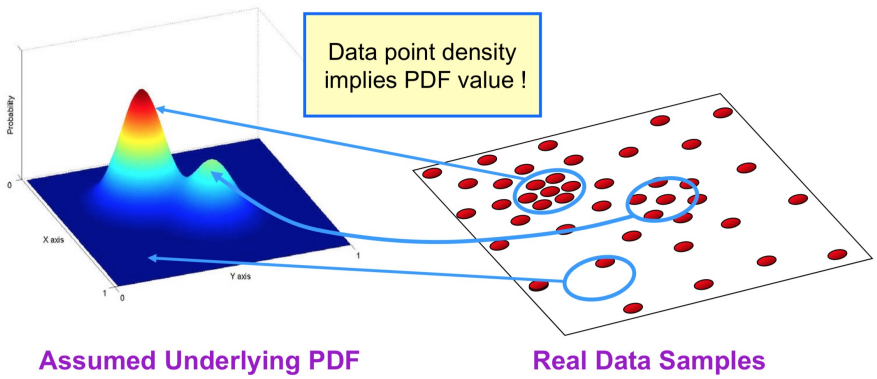


Figure 4-8: A PDF representation of an image [10]

By choosing an appropriate kernel *Uniform, Gaussian or Epanechnikov*, the mean shift algorithm can be used to follow the gradient ascent and reach a peak in the *PDF*[11], which is the central idea. Given RGB images a colour distribution is used, and the dissimilarity between the selected ROI (i.e the robot) and the target candidates is measured by the Bhattacharyya coefficient[12]. More details in Comaniciu, Ramesh and Meer (2000).

4.7 Tracking-Learning-Detection

In previous sections we briefly explained some methods that would track an object in continuous frames, that is for example, video or camera frames. In this section we explain a very similar approach from Kalal, Mikolajczyk and Matas (2012) that uses a tracker, a detector and learner, also called TLD. The goal of this is *long-term tracking* of an object. By long-term tracking it is also meant that the tracker should recover from occlusion that is if the tracked object disappears from the camera feed, for example passed behind another object, then the *detector* part should be able to recover and continue

[10] http://crcv.ucf.edu/people/faculty/shah.php
[11] https://en.wikipedia.org/wiki/Probability_density_function
[12] https://en.wikipedia.org/wiki/Bhattacharyya_distance

tracking the object (which the previous methods do not). Usually detectors require an offline training phase. In this method the detector is trained online that means based on the incoming camera frames. The main setup here is having a tracker that provides training data to the detector. A high level of the algorithm is:

- **Initialization**
  - Train a classifier from a single example (detector)
- **For every frame**
  - Evaluate the classifier
  - Estimate errors (feedback)
  - Update classifier

Then using a method called *P-N Learning* the detector's output is evaluated at each frame, by two experts as Kalal, Mikolajczyk and Matas (2012) call them P-expert and *N-expert*. P-expert identifies only false negatives, and *N-expert* identifies only false positives. The input to the algorithm is a ROI (region of interest), that the user selects that is the robot. As the *P-N Learning* method identifies positives samples it creates a dataset that the detector (classifier) is trained on in order to improve detecting performance. Figure 4-9 from Kalal, Mikolajczyk and Matas (2012) shows a basic workflow diagram of the TLD algorithm, and fig. 4-10 visualises the learning process.

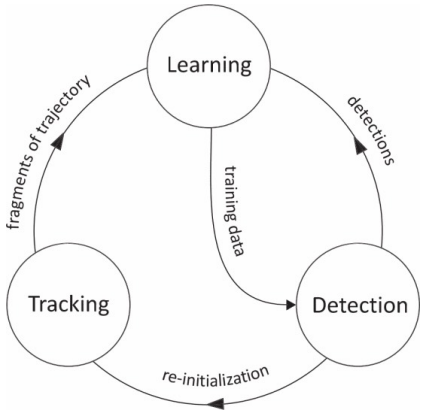


Figure 4-9: TLD Workflow visualized

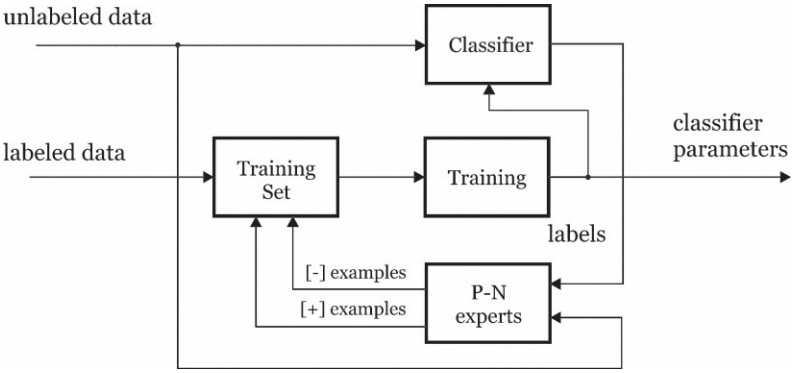


Figure 4-10: The block diagram of the learning method (P-N).

This is a form of on-line training and task is to simplify and speed-up training. In fig. 4-11 you can see the difference between offline and online training (Kalal, Mikolajczyk and Matas, 2012).

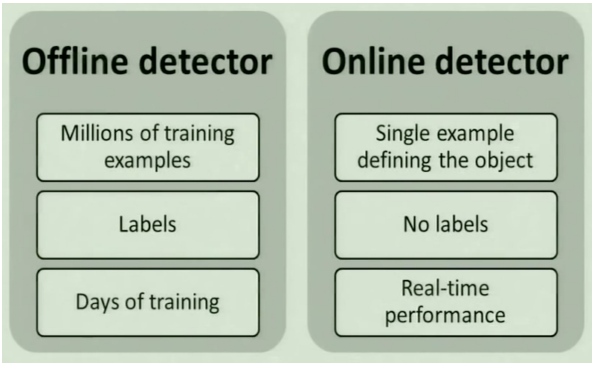


Figure 4-11: Offline vs Online Training

Fig. 4-11 from Kalal, Mikolajczyk and Matas (2012) explains better why offline training was not suitable for this project. One of the main reasons was a missing dataset of training samples. One other major difference in offline training is the feature engineering that happens in order to reduce classification errors. In online training as mentioned in Kalal, Mikolajczyk and Matas (2012) it is accepted that errors will happen and use those errors to improve the detector. The classifier used in the paper by Kalal, Mikolajczyk and Matas (2012) is a simple KNN[13] with $k = 1$.

[13] http://www.saedsayad.com/k_nearest_neighbors.htm

4.8 Object Detection using ML (TensorFlow)

Another approach that was considered but never implemented was to use ML (Machine Learning) to create a tracking system by using detection, also called tracking-by-detection. This required some offline training. The examples that we came across did require a dataset of the object that was going to be tracked. *Tensorflow*[14] was consider on Android but given the unfamiliarity with the framework this idea was discarded. Creating a dataset of hundreds of images of the R5 robot, training and choosing a good enough ML model was infeasible given the available time before the field study in Chapter 7. This was left for future work described in Chapter 9.

[14] https://www.tensorflow.org/mobile/android_build

Chapter 5

Software Development

This chapter describes the technical development of the server and client communication, and a brief description of the AR design. More details about the image processing and tracking are found in Chapter 6. The design and implementation stages are executed iteratively similarly to SCRUM[1]. The app development was split into two main parts. First goal was to achieve a communication between the mobile device and the Instinct server. The second part was to achieve a reliable tracking method of the robot in order to load the robot's plan. Given the size of the latter we have devoted a separate Chapter 6. The majority of the code was written in Java[2] and some parts in C++.

5.1 Requirements

The very high levels of requirements, for the mobile app were:

1. Visualise the robot's remote data about its plan in AR.
2. Continuously tracks the robot.

The software requirements specification is stated to set up the functional and non-functional goals of the system. The requirements were iteratively reviewed in weekly meetings with Mr Ken Cameron, Mr Andreas Theodorou, and Dr Rob Wortham, following SCRUM. SCRUM is an agile framework that is frequently used in software development. It is adaptive and iterative which

[1] https://en.wikipedia.org/wiki/Scrum_(software_development)
[2] https://en.wikipedia.org/wiki/Java_(programming_language)

allowed incremental builds of the app. At each meeting current work was evaluated and suggestions made. The overall system was gradually refined and design decisions were justified. For a full description please see its Wikipedia page[3]. The *MoSCoW*[4] method was used to compile the requirements, as we have more experience with it. The mobile application requirements were identified as the following:

Functional Requirements

1. Overall performance must be between minimum 20 fps and 30 fps.
2. Tracking of the robots must be fast enough to perform as above.
3. The app must be able to communicate with a remote server.
4. The app must be able to receive a stream of incoming strings from that server.
5. The app must be able to load an Instinct plan exactly in the same way that ABOD3 loads the plan.
6. The app must be able to handle the amount of the server's incoming stream data.
7. Tracking of the robot in the camera video stream must work as the user is moving the device in a 3D space.
8. Tracking must work up to 6 meters away from the robot. This was because of the setup in the field study.

Non-Functional Requirements

1. Must be an Android Application (since we had an Android phone).
2. The text on the screen must be visible enough for users to read.
3. Tracking should be stable enough to last more than three minutes for experiment purposes.

[3]https://en.wikipedia.org/wiki/Scrum_(software_development)
[4]https://en.wikipedia.org/wiki/MoSCoW_method

5.2 Design

In this section we will describe the overall design that our implementation was based on, regarding the network communication and the augmented reality.

5.2.1 Server Extraction and Setup

The communication between all agents, the robot and the server was achieved on the TCP/IP[5] protocol. A router was used to create a private Wifi network, called *InstinctWifi* (see fig. 5-3) on which the Instinct server, R5 Robot, and mobile devices would connect to. The Instinct Server run on a laptop during the field study. The first main task of the project was to extract the TCP/IP server (Instinct Server) from ABOD3 and use it in its original form for the main server that my application would connect to. Details regarding the structure of the Instinct Server can be found in section 3.1. This was crucial as the robot needed a server to send the plan executions to. There is already an implementation of the Instinct Server[6].

That specific implementation did not have support for clients wishing to read the incoming data from the robot. Diagram in fig. 5-1 shows how the original Instinct Server behaved. The server was designed to spawn a new thread every-time a robot connected to it. That thread would be responsible for sending data to the robot, such as the *command file* and the robot's *plan*. The desired setup that we eventually implemented is shown in fig. 5-2. The main two approaches for establishing communication between the elements (server - robot - mobile device) were the following:

Approaches:

1. Ping the server, from the mobile client, to retrieve robot's data at extremely high frequencies.

2. Send all the incoming data from the robot to the server, and then to the mobile client.

For the majority of the project development period we were using option number one. The reason is that the amount of data that was coming from

[5] https://en.wikipedia.org/wiki/Internet_protocol_suite
[6] http://www.robwortham.com/instinct-planner/

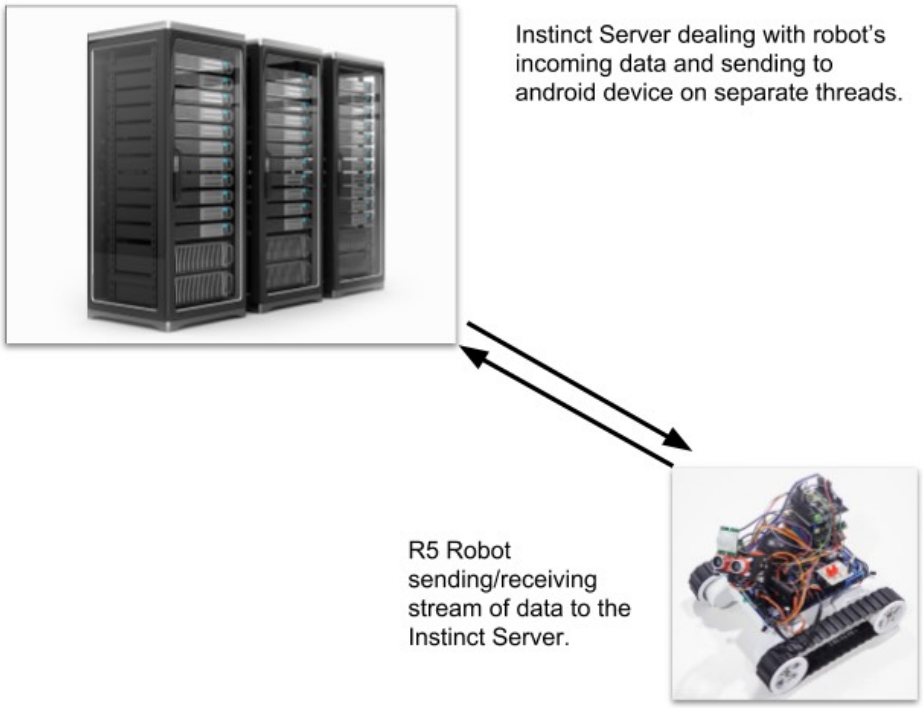


Figure 5-1: A simple visualization of how the initial version of Instinct Server communicated with the R5 Robot

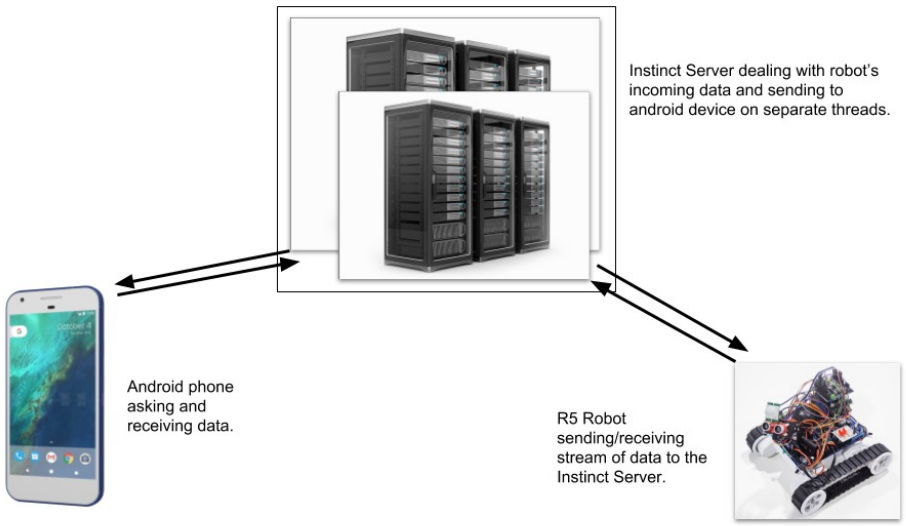


Figure 5-2: Instinct Server setup after adding support for mobile clients

the robot at that time was enormous in size. Streaming all that back to the mobile device seemed like an unnecessary overhead. At that point we did not even have a working tracker, so performance was our priority. As we explain later this was a inefficient approach and we ended up using the second option.

5.2.2 Augmented Reality Design

The second and major design part of this project was to develop an android app that would use augmented reality to visualize the robot's AI execution plan. This would allow the user to **tap** on elements on the screen and view a more detailed subtree of action for the selected action. Given the amount of work that was carried out to achieve the tracking of the robot, everything has been described in a separate Chapter 6. The final diagram of the system's design is given in fig. 5-3.

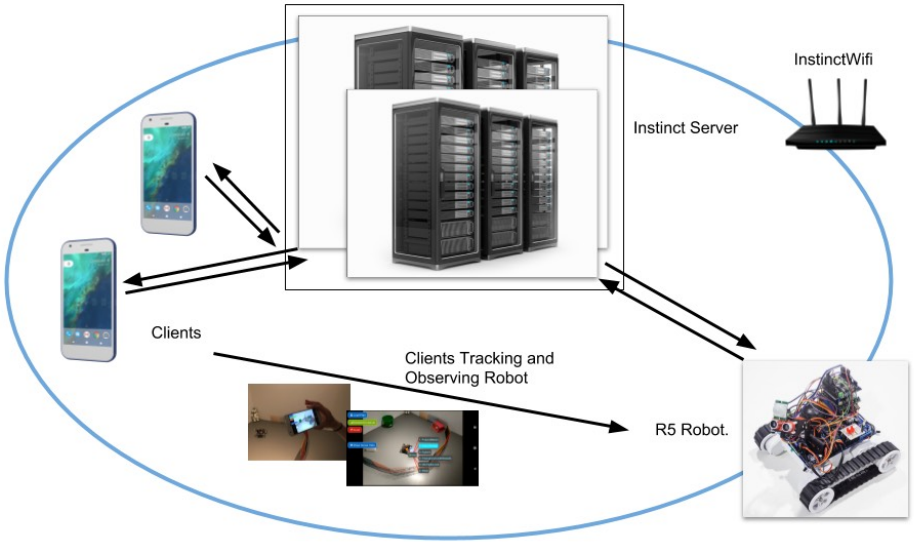


Figure 5-3: The overall research project's app diagram

5.3 Server and Client Connectivity Implementation

This section will focus on the implementation software details of the Android application and the addition/changes made to the Instinct Server in order to achieve a three-way communication as shown in fig. 5-2.

5.3.1 Instinct Server Modifications

In order to add support for multiple mobile clients connecting to the server we had to modify the existing code so messages could be shared between two threads. The first approach was to write a second server that would read the robot's incoming data from the first server and send it back to any mobile clients connected to it. The second server called `ThreadedEnquiryServer`, would accept any mobile requests and create a new thread to handle each one of them. It will then subsequently send data back to the mobile device(s).

For this task a *volatile*[7] Java variable, named `ROBOT_INCOMING_STRING` was used. This was wrapped in a Java class, `RobotStreamData`. An instance of this class was created on the boot up of the server and passed in the constructors of both servers. As the robot's data was received, this variable was updated with the latest robot's data string only. The client (app) was querying the server every 50ms or so. This was the very first version of our application capable of retrieving data from the Instinct server.

We did discover soon that the this approach did not yield the best results as we where missing messages. Thread 1 (robot's incoming data) would update the variable too fast and thread 2 would pick up changes but miss some messages. Then the client would query the server every 50ms and messages were lost.

We then moved to using a `ConcurrentLinkedDeque`[8] data structure to store the messages from the robot in, and then retrieve them when the mobile client queried about them. A ConcurrentLinkedDeque is an appropriate choice when many threads share access to a common collection. It uses a LIFO[9] (Last-In-First-Out) stack underneath. This lead to timing issues as the messages received on the mobile device were delayed and outdated (the `ConcurrentLinkedDeque` was filling up too fast and messages were not picked up fast enough). In order to avoid any timing issues no intermediate data storing medium or data structure was used. The final version used in the field study, immediately send the strings to the client (as long as there was a connected client) as they were received from the robot. The server would keep an *ArrayList*[10] of connected clients and every time a message was received

[7] http://tutorials.jenkov.com/java-concurrency/volatile.html
[8] https://developer.android.com/reference/java/util/concurrent/ConcurrentLinkedDeque
[9] https://en.wikipedia.org/wiki/Stack_(abstract_data_type)
[10] https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

from the robots, it would update his clients. Please note that even though the server was modified to support multiple clients given we had one device for development it was never tested, with multiple devices. Given the simplicity of the implementation, there is a low risk of failure.

5.3.2 Android Network Client

The first step was to establish a connection from the Android app to the Instinct Server. This was as simple as just sending a string to the Instinct Server and acknowledging its successful delivery by checking what the server had received from the android application. This was first achieved by using an `AsyncTask`[11]. The final implementation used a background service or `Executor`. Various approaches to this were implemented, but all needed a background process in order to execute the network calls given how Android is built. As mentioned in section 5.3.1 throughout the development of this app the mobile client would query the server at very short intervals (i.e 50 - 100 ms) and retrieve data. All the code related to network calls is found in the file `NetworkTask.java` in the project's source code. In Chapter 8 we give the repository to the source code.

AsyncTasks and Threads

The alpha version of the mobile application used an AsyncTask. We quickly found that this caused memory leaks. AsyncTasks tasks in Android are not suited for long running tasks. The reason that an AsyncTask was chosen is the simplicity of implementation and convenience when it comes to updating the UI, more in later section. Further on a generic Java thread replaced the AsyncTask that would execute the queries in the background. Part of the code that was used in the alpha version is seen in Appendix B.1. The reason the thread approach was discarded is because it was using a `while loop`, which consumed quite a lot of unnecessary CPU cycles, even though the code in the `while loop` only executed every 50 to 100ms.

[11] https://developer.android.com/reference/android/os/AsyncTask

Scheduled Executor Service

For extremely long running background tasks Google suggests using services[12] in Android. For this project a `ScheduledExecutorService` seemed ideal. Code sample in 5.1.

Code Snippet 5.1: A ScheduledExecutorService, 'serverPingerScheduler'

```
final Runnable pinger = new Runnable() {
  @Override
  public void run() {
    try {

      response = input.readLine();
      Message message = new Message();
      message.what = SERVER_RESPONSE;
      message.obj = response;
      handler.sendMessage(message);

    } catch (IOException e) {
      e.printStackTrace();
    }
    }
  };

serverPingerScheduler.scheduleAtFixedRate(pinger,
                                50, 50, TimeUnit.
                                   MILLISECONDS);
```

The variable `input` was an instance of a `BufferedReader` and it was overfilling with incoming data and the 50ms `pinger` was not keeping up, so it was returning delayed messages every-time it was querying the server. This approach was discarded aswell. The final version that was used in the field study followed a much more simplistic approach. Instead of querying the server every n ms we read any incoming strings from the server and updated the UI. The reason this was chosen is because after switching from OpenCV to BoofCV (both libraries explained in Chapter 6) we had a huge increase in performance. This allowed the app to comfortably deal with the huge incoming amount of data.

[12] https://developer.android.com/guide/components/services

Updating the UI - Handler

Since the UI in the Android platform runs on a separate thread, a mechanism is required to communicate with it. Network tasks such as server polling and I/O can not run on the UI thread [13]. Our current app architecture consisted of a thread that was querying the server at every set interval time. An AsyncTask can update the UI with the method `onProgressUpdate(String... values)` that the class offers, but AsyncTask was removed for reason mentioned previously. The most optimal method to update the UI according to official documentation was to use a `Handler`[14]. Figure 5-4 demonstration the basic functionality of a Handler in Android. Using a Handler made it possible to display the server's incoming data on screen and update nodes in the AR environment as explained in section 5.4.2.

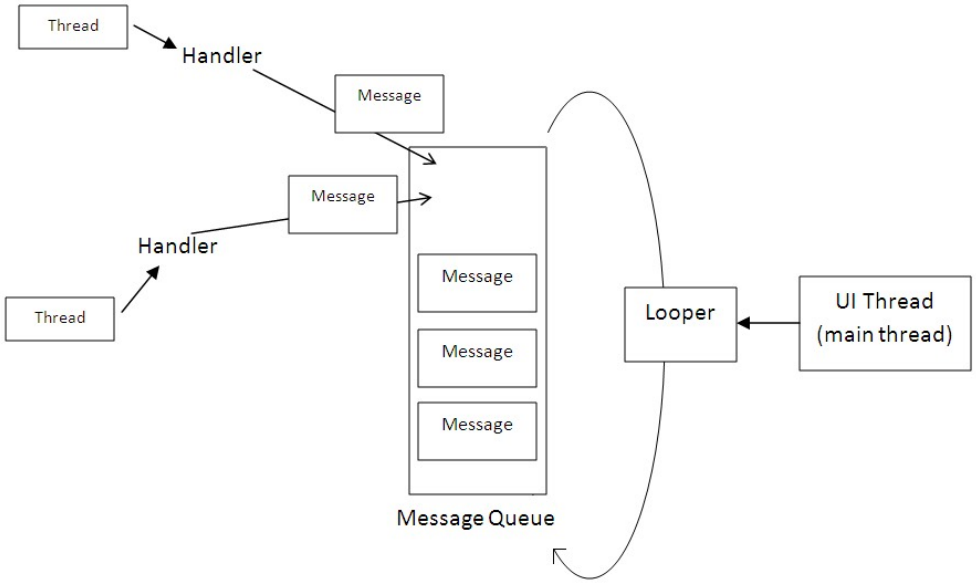


Figure 5-4: Android Handler and UI Communication[15]

'A Handler allows you to send and process Message and Runnable objects associated with a thread's MessageQueue. Each Handler instance is associated with a single thread and that thread's message queue. When you create a new Handler, it is bound to the thread/message queue of the thread that is creating

[13] https://developer.android.com/training/multiple-threads/communicate-ui
[14] https://developer.android.com/reference/android/os/Handler
[15] https://medium.com/@manishgiri/android-handler-tutorial-ccda6994f01c

it'[16].

This approach was used because of the nature of our project, messages represented the stream of strings from the server. The Handler was created on the UI thread and messages were sent to it from the Network thread (that was started from the UI thread). The Network thread was responsible for querying the server. The Looper from fig. 5-4 is responsible for adding messages to the MessageQueue and also sending the messages to the Handler. Code for the final Handler class used in the demo version for the field study is found in code snippet 5.2.

Code Snippet 5.2: Handler used in Demo Version

```
private void createGeneralHandler() {
  generalHandler = new Handler(Looper.getMainLooper()){
    @Override
    public void handleMessage(Message msg){
      switch (msg.what){
        case SERVER_RESPONSE:
          if(serverTextView.getVisibility() == View.
              VISIBLE) {
            serverTextView.append("\n" + msg.obj);
          }
          updateARElementsVisuals(msg);
          break;
        default:
          super.handleMessage(msg);
      }
    }
  };
}
```

5.4 ABOD3 Port to Android

The main component that needed to be used from ABOD3 was the plan loader. TreeView[17] library was initially considered to use for loading the plan. *Dia-Plan3.inst* was used for most developement, and *Plan6.inst* was used for the demo runs in the field study. The reason we used *Plan6.inst* for the demo

[16] https://en.proft.me/2017/04/15/understanding-handler-android/
[17] https://github.com/Team-Blox/TreeView

runs is because it was the most stable plan to use as it was used in previous research (Wortham, Theodorou and Bryson, 2017a). *TreeView* quickly seemed inflexible and quite structured to fit the needs of the project. Adding nodes in the recommended guide was cumbersome and time consuming. The nodes had to be positioned using specific (x,y) coordinates that *TreeView* does not allow, as the current implementation is.

5.4.1 Plan Loader

In order to render the plan on the phone's screen, the Instinct plan had to be loaded in the exactly same way as it was loaded in ABOD3. Then in later Chapter 6 we explain how we augment the environment with the robot's plan. Since ABOD3 was written in Java and Android is Java compatible, the *Gluon*[18] framework was considered. This was because it allows cross platform development using Java. We discarded it because it requires a licence. Instead we moved and rewrote code from ABOD3 to the Android application and stored the plans locally on the phone. The application loads the plan upon starting up. In order to load the plan in the augmented reality environment an *anchor point (i.e a pixel on the screen)* had to be defined for it. That means that a pixel will act as the centre of the plan and that ideally will be the robot's ROI center position in realtime.

First Loader Implementation

The first approach was similar to the original ABOD3 plan loader. The way the current plan loader `InstPlanReader.java` works in ABOD3; it returns a list of *Drives* that each Drive contains a reference to a list of *Actions* or *Competences*, which all of these make up a POSH plan as explained in section 3.1. A Drive can be hierarchically composed by a number of Competences, Action Patterns and Actions (Wortham and Rogers, 2017). The lists are represented as Java `ArrayLists`. Figure 5-5 shows *Plan 4* loaded in ABOD3. The same plan is shown loaded in augmented reality, see figure 5-7.

This was the very first version that had the ability to load any plan in an AR environment. It would only display the root *Drives* and its children. In fig. 5-6 you can see the draft sketch before implementing the actual plan in augmented reality. The goal was to position the elements based on the

[18] https://gluonhq.com/products/mobile/

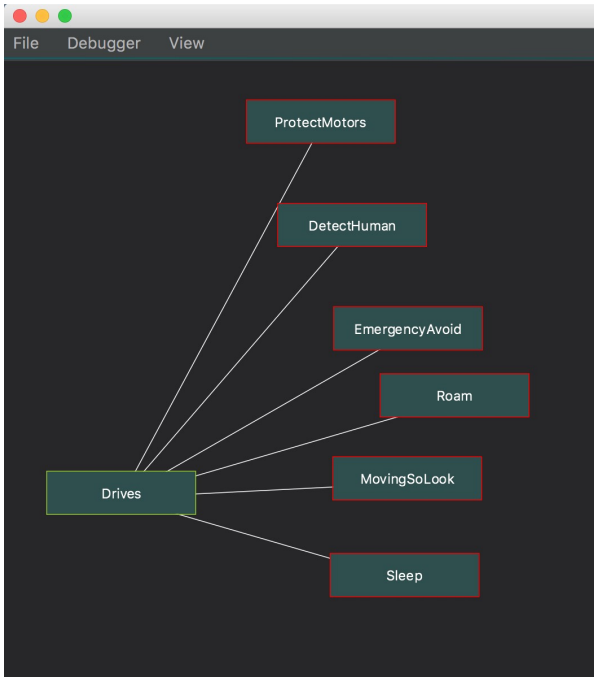


Figure 5-5: Instinct Plan 4 loaded in ABOD3 - showing only drives.

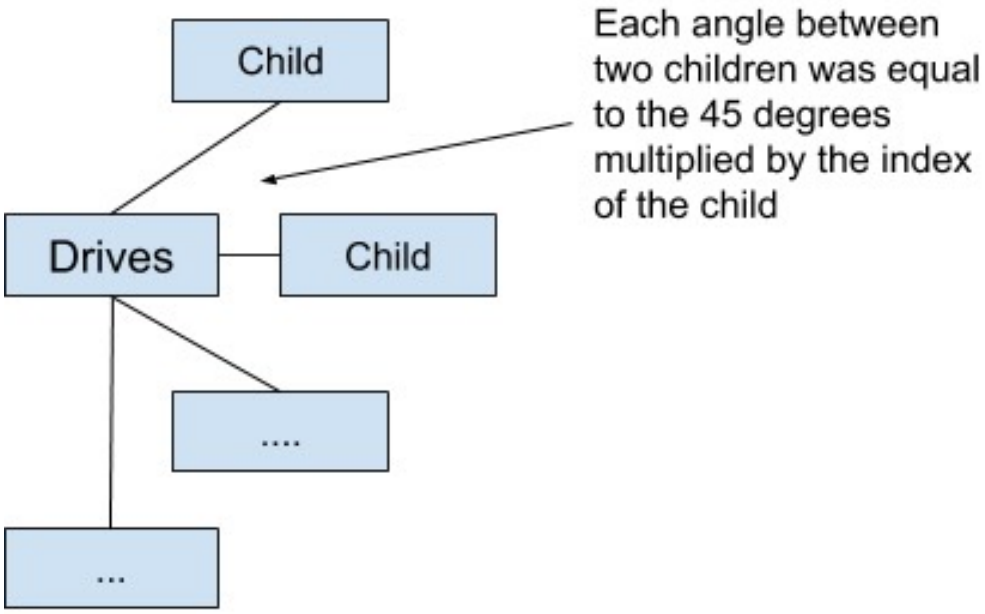


Figure 5-6: Draft sketch of the first version of the plan design.

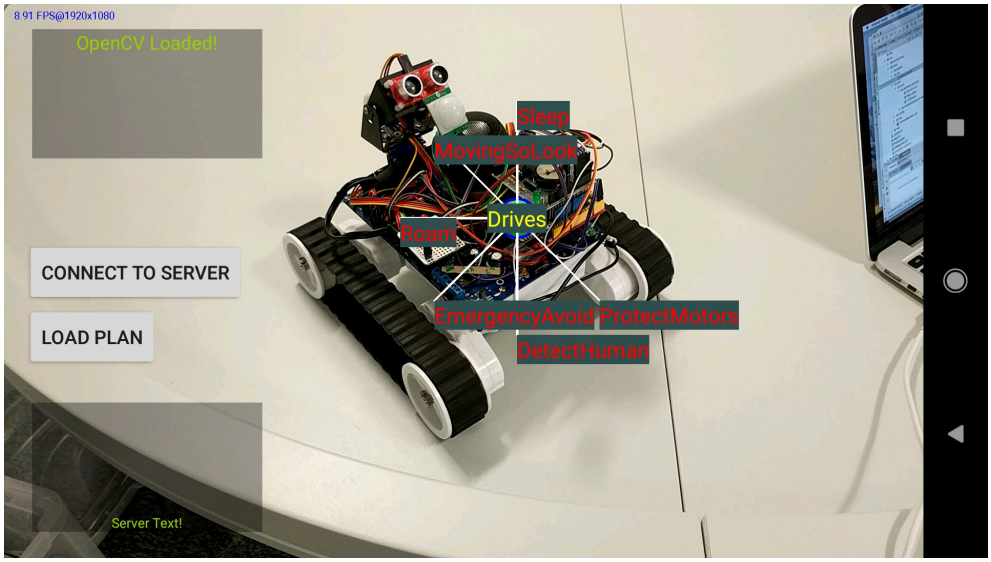


Figure 5-7: Loading Plan 4 in AR for the first time. This version still uses the green ball as the tracker.

child's index. The formula used seen in equation 5.1 and 5.2 for each x and y coordinates.

$$x = circleCenter_x + radius * 300 * \cos(\pi/4 * k) \tag{5.1}$$

$$y = circleCenter_y + radius * 300 * \sin(\pi/4 * k) \tag{5.2}$$

This are polar coordinates[19], with a center that dynamically changes at each frame as a new circle with center $(circleCenter_x, circleCenter_y)$ is detected. We add and multiply the radius by 300, for the simple reason that those values gave the best positioning on screen as seen in fig. 5-7. Value of k was simply the index of each child. This approach was not sufficient for plans with more than five drive elements as seen in fig. 5-9. This led to elements not having enough free room and even with adjusting the values in equations 5.1 and 5.2 we did not achieve an optimal solution that would work for most plans. We took this first implementation a step further and added the ability for the user to click on a *Drive* node and display one level further in the tree that could be an *Action* or a *Competence*.
A video showing this: https://www.youtube.com/watch?v=mPhZv1W861s

[19] https://en.wikipedia.org/wiki/Polar_coordinate_system

And fig. 5-8 also illustrates this. The green text on the bottom left show the raw data received from the Instinct server.

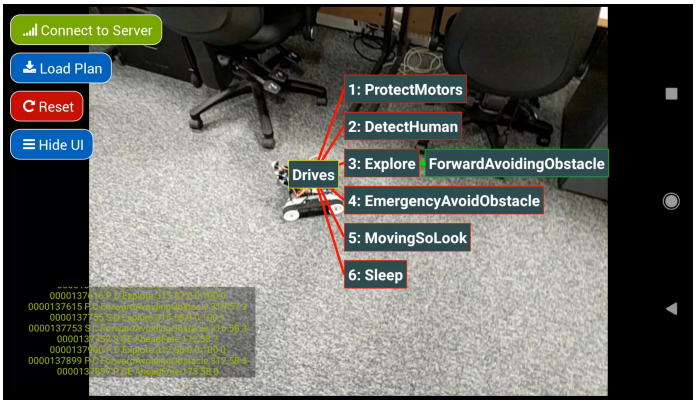


Figure 5-8: Allowing the user to click on a Drive and show its child.

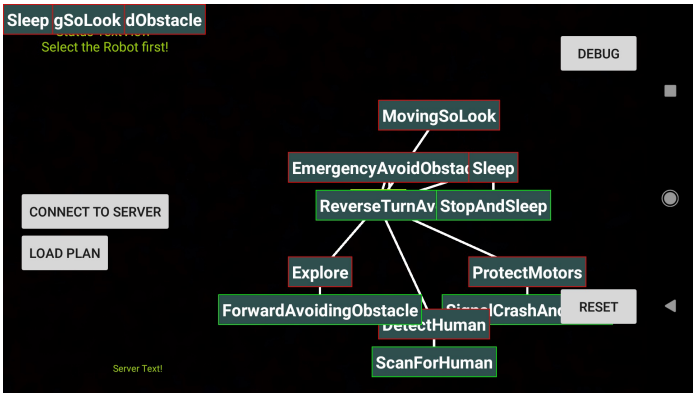


Figure 5-9: Loading a larger plan in AR . Elements are overlapping and hiding other elements.

Dragging plan nodes

A temporary approach that was implemented and tested in order to deal with the large number of nodes in certain plans, especially the overlapping of nodes as seen in fig. 5-9, was to allow the user to drag the nodes. As seen in:

https://www.youtube.com/watch?v=qQJnVcP85PY
https://www.youtube.com/watch?v=bpf9n5tuPt0

Plan Loader: N-ary Tree Implementation and Cyclic Navigation

As mentioned earlier in this chapter the desktop version of ABOD3 returned a plan in a form of a list of *Drives* that each child was a list of POSH elements. This was changed in using a proper n-ary tree[20]. That means that a tree with n numbers of nodes for each node was produced from mobile app. The code for this is found in file `UIPlanTree.java`. Initially the tree would only hold information about the root, and its children, that is the *Drives*. As requirements changed from Prof. Joanna Bryson and Mr Andreas Theodorou, more elements had to be shown on screen. Next step was to add *grandchildren* in the tree structure. This led to problem such s running out of room on the phone's screen to display elements. The main difference in this project compared to ABOD3, is that ABOD3 was designed to run on desktop machines but this app had to run on small/restricted in size mobile phone screens. Eventually the tree was designed to hold all of the plan's nodes.

After discussions with Mr Ken Cameron and testing, it was best to implement a *cyclic* type of navigation to solve this problem, of showing plan elements on the screen. The logic behind this, was to initially display the plan's root *Drives* and then its first level children. The functionality to *'navigate'* through the n-ary tree was implemented, that means that the user could **tap** on a node, that node would become the current root, and it children would appear. The user could navigate back by clicking on the node's parent that was kept visible on screen. This was better than having a *back button* distraction the user from looking at the plan. This process is better understood in the video:

https://www.youtube.com/watch?v=Gqqng1uh3Ys

This video shows the latest version of the app that was used in the field study. Figures 5-10 and 5-11, also demonstrate this.

The code that was written regarding this *cyclic* type of navigation is found in `UIPlanTreeNodeTouchListener.java` and `UIPlanTree.java`.

5.4.2 Flashing the Plan elements

Each incoming piece of data (string) from the Instinct server was parsed to identify to which plan element on screen it corresponds to, by name. It was

[20] https://en.wikipedia.org/wiki/K-ary_tree

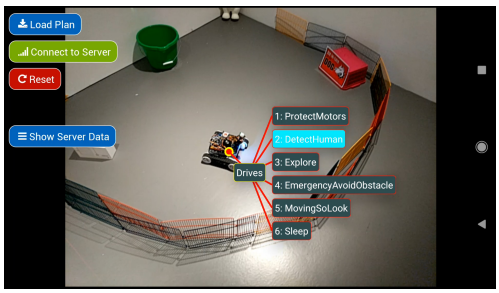


Figure 5-10: Showing the initial plan as soon as the user loads up the plan.

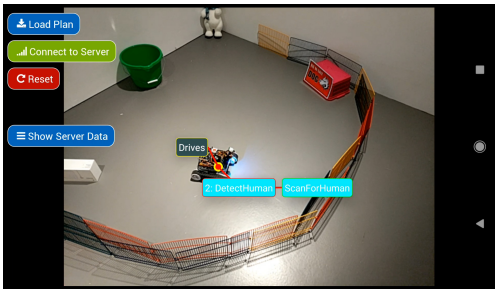


Figure 5-11: By clicking the node *DetectHuman* then that becomes the root and the node *Drives* still remains as a *back button* option.

then possible to light up the corresponding node in AR by just finding the node that had the same name. An example string that was received by the Instinct server has the form of:

```
0000001150 E D Explore 2 0 0 0 240 1
```

The substring *'Explore'* is what the application needs in order to update its UI. The on-screen plan elements are simple Android `TextView` objects. The background colour of them can be easily changed by calling built-in methods. The first approach was to set the background colour of the node to blue when an incoming string from the server was identified to match a node that was visible on the screen. If not then the background would be set to dark green that is the default colour. This was applied by calling a recursive function on the tree's root node that would traverse all nodes. Code shown in 5.3.

Code Snippet 5.3: Recursive node colouring code

```
public void setNodeBackgroundColor(String
    planElementName, Node<ARPlanElement> node) {

  if(node.getData().getName().equals(planElementName)){
    node.getData().setBackgroundColor(Color.parseColor("
        #0000ff"));
  }else{
    node.getData().setBackgroundColor(Color.parseColor("
        #2f4f4f"));
  }

  node.getChildren().forEach(it ->
      setNodeBackgroundColor(planElementName, it));
}
```

A video showing the resulting end:

https://www.youtube.com/watch?v=mPhZv1W861s

Performance was not an issue regarding this approach. It seemed to work fine as long as the robot's execution plan rate was set to the minimum value that is 1. As explained in section 3.3 this was not efficient. Setting the rate to the recommended value 8 increased the incoming data by a tremendous amount and this broke the flashing effect. This proved to be inadequate as the end result did not resemble to a flashing effect as we increased the plan execution rate of the robot.

Another approach followed was to mimic ABOD3 behaviour of how it was updating its nodes. Each node has a starting value of a *frequency* that would increase as matching elements would come in from the server. The higher the frequency the faster the flashing, up to a limit. At the same time the frequency will drop, so if no matching elements were received, for some time, the nodes will eventually 'die out' and stop flashing. Note that ABOD3 used JavaFX[21] that is only available for desktop platforms, and that effect was achieved by using built-in methods of JavaFX such as `runLater(Runnable runnable)`. In Android we attempted to achieve the same by creating a single Thread

[21] https://en.wikipedia.org/wiki/JavaFX

for each node on screen that would be responsible of decreasing or increasing the *flashing frequency* of its node, between some limits. This again proved to be very resource consuming. For example, if a plan had six or more drive elements that meant that six or more threads would have been created and started and that had a major impact on performance.

The last approach that was included in the final version of the app was the following. We implemented a `ScheduledExecutorService`, called `backgroundPingerScheduler` to run every 500ms that would set all the current visible nodes to their default background colour. The massive incoming amount of data would come from the server and flash the nodes too fast, but the `backgroundPingerScheduler` would make sure that there were set back to their default colour. This would solve our problem that nodes would flash in such a high frequency and not have a flash effect as in ABOD3.

5.5 Final version of Android app

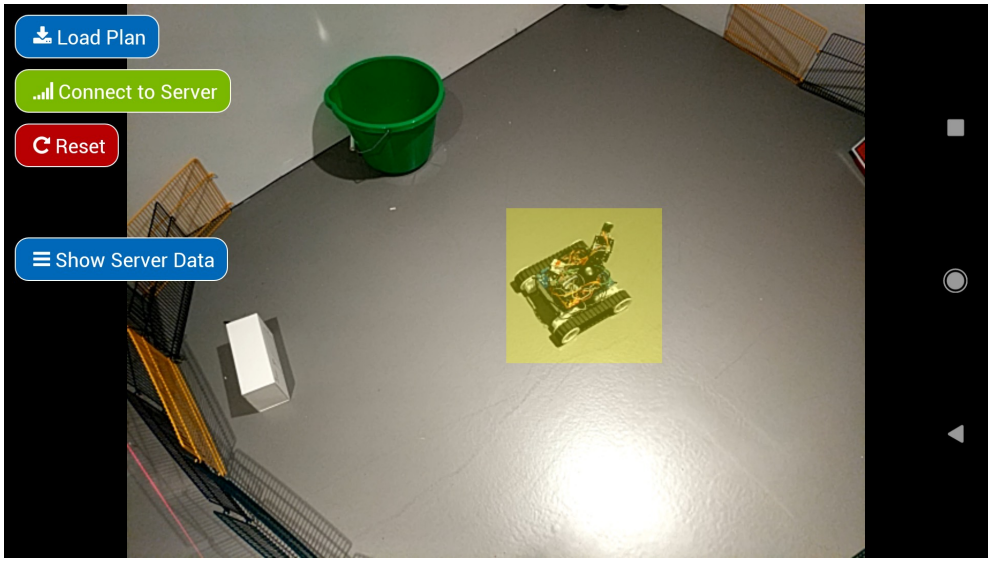


Figure 5-12: User dragging an area that contains the robot to be tracked (yellow).

Here we describe the basic functionalities and the UI of the app. Figures 5-12, 5-13, 5-14, show a basic workflow of using the app during the field study. User selects the robot first, then releases. At this point the app is able to track the robot. The plan is then loaded by clicking *Load Plan* and then establishing

a connection to the server by clicking *Connect to Server*. The nodes on the plan then start lighting up as soon as the robot's realtime stream of data from the Instinct server are received. The button *Show Server Data* when clicked displays a small text box in the lower left corner that displays the server's data, mainly used for debugging purposes.

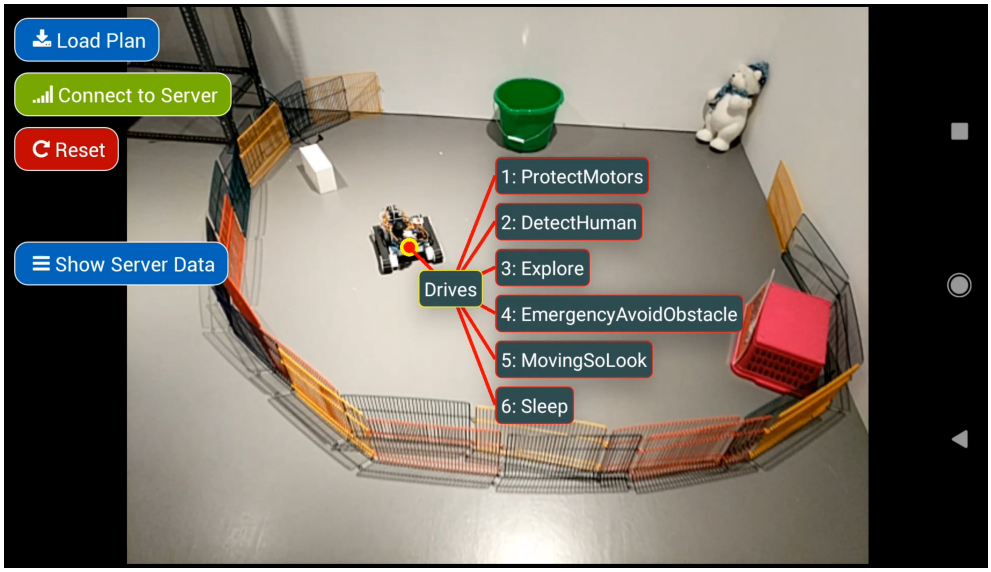


Figure 5-13: As soon as the user clicks *Load Plan*, plan shows up.

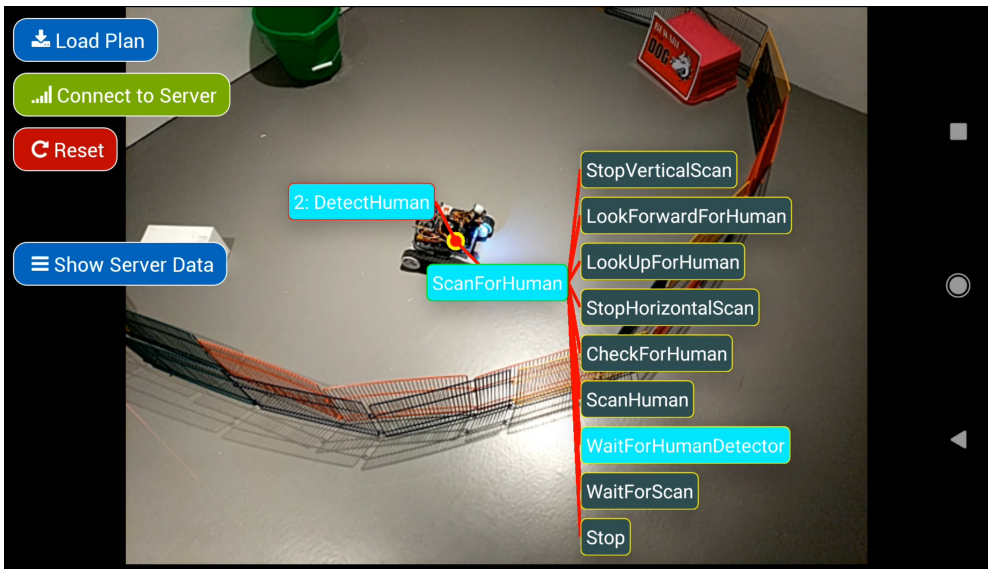


Figure 5-14: User clicking through plan elements, and nodes flashing.

Button *Reset* rests the app to its initial stage, as it was started and kills any background networking services. How we achive the tracking of the robot and the approaches followed are explained in Chapter 6.

Chapter 6

AR Tracking Development

In this chapter we will explain the implementation to the algorithms proposed in Chapter 4 and a description of the technical approaches and solutions using *OpenCV*[1] and *BoofCV*[2]. The main goal from this part of this project was to be able to track the robot in real time and define a point (*anchor point*) which the plan would use as a centre for its local coordinate system. It is worth mentioning that we experimented with QR codes using OpenCV's ORB features detector `ORB::create()` to detect a QR marker (on the robot) in the camera frame. Even though we used the Android NDK[3] which is C++ code, performance was slow. Another approach we tried was augemented images with *ARCore*, but discarded aswell as mentioned in section 4.2. Please see in figures 6-1 and 6-2. It was obvious that adding a marker on a moving robot that had to be tracked from a distance of maximum six meters would not work.

6.1 Phase 1 - OpenCV

The decision to use OpenCV was made because none of the available libraries that were examined during the literature review can achieve VOT, video object tracking, in an efficient and robust way. Also OpenCV is a well tested and mature software library, and its *core* functionality has been ported to run on Android perfectly as it would on a desktop machine. Note that it may be that the core functionality was ported but *extra* modules useful for this project

[1] https://opencv.org
[2] https://boofcv.org/
[3] https://developer.android.com/ndk/

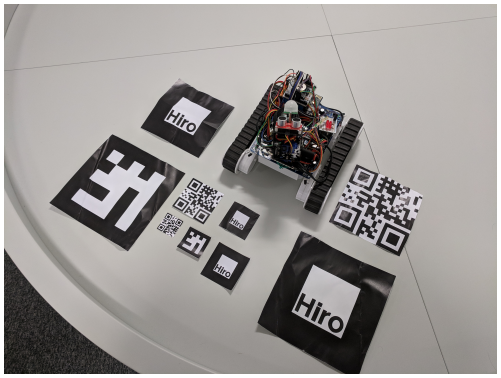


Figure 6-1: Potential Markers 1.

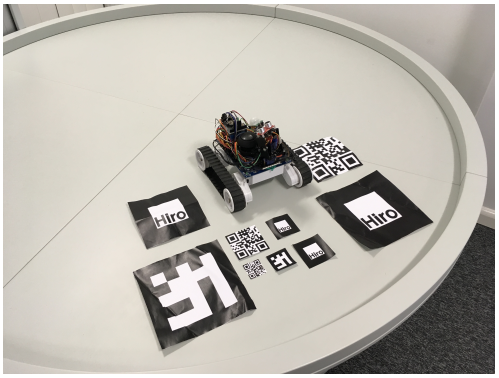


Figure 6-2: Potential Markers 2.

have not. More in section 6.1.5. We purchased green *Play-Doh*[4] to use for the green ball for tracking.

6.1.1 Hough Transform

As explained in section 4.4 this approach was chosen as the first main one. Due to the method's simplicity and rapid prototyping a very basic application that could connect to the server and fetch robot's data was created. Figure 6-3 shows a low resolution screen-shot of that alpha version. A green circle is detected and then using its center as an anchor point required UI element is drawn. It is worth mentioning that before circle detection, we experimented with `Imgproc.findContours()` that retrieves contours in an image. The largest contour would be assumed to be the *Play-Doh* green ball on the robot. This quickly failed, as the largest contours was most of the times not the green ball.

Video: https://youtu.be/ktFKGVT11ns. In the video you can clearly see the text box with the incoming stream data of the robot's actions. The reason that the screenshot and video are on low resolution is because of the recording software on the phone. We later on achieve much higher resolution pictures and videos.

6.1.2 Color Thresholding

The simplest and fastest method to come up with a working prototype was to use a green sphere attached on the robot, and track that by colour thresh-

[4]https://en.wikipedia.org/wiki/Play-Doh

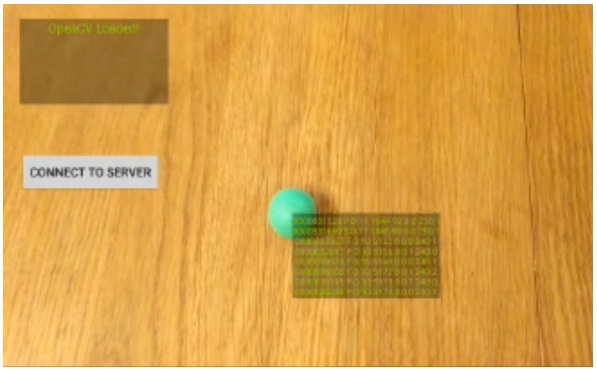


Figure 6-3: Basic app that communicates with server and tracks sphere.

olding the incoming frames. This was achieved by using OpenCV's functions `Imgproc.cvtColor()` and `Core.inRange()`, and two lower and and upper green colour bounds defined as in code snippet 6.1.

Code Snippet 6.1: Upper and Lower threshold bounds

```
lower = new Scalar(29, 86, 6);
upper = new Scalar(64, 255, 255);
```

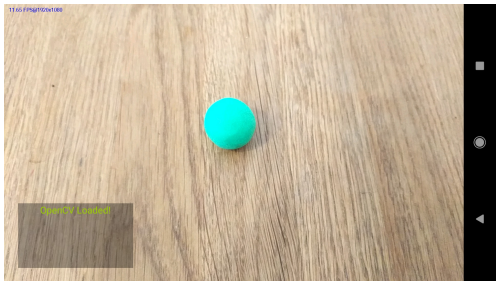


Figure 6-4: Original camera frame.

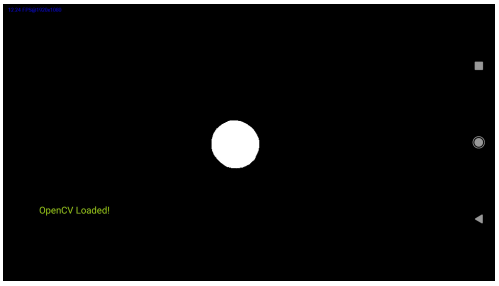


Figure 6-5: Results of colour thresholding.

Both green values in *RGB* format; anything in between would show up as white and the rest black. The results are demonstrated in the fig. 6-4, 6-5 and 6-6. The aim is visible in fig. 6-6, where the detected center is visible in red; that was used as the anchor point for the robot's plan. The code for this is seen in Appendix B.3. You will notice that at one point we have used a Gaussian blur. The reason we used a Gaussian Blur is explained in section 6.1.4. Testing this straightforward implementation of the circle Hough transform gave unsatisfying results as seen in figures 6-7, 6-8.

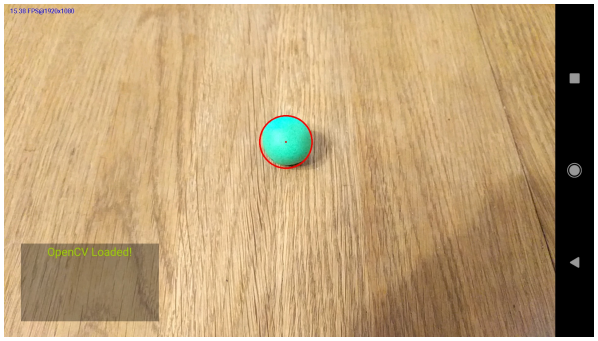


Figure 6-6: Final Frame with detected circle (red) drawn.

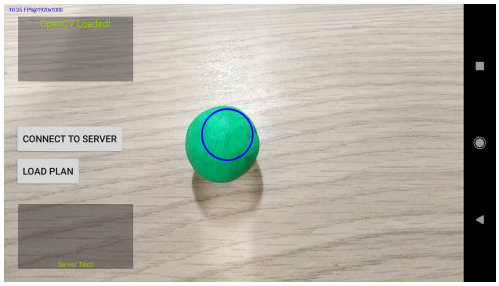


Figure 6-7: Short distances would give inaccurate results.

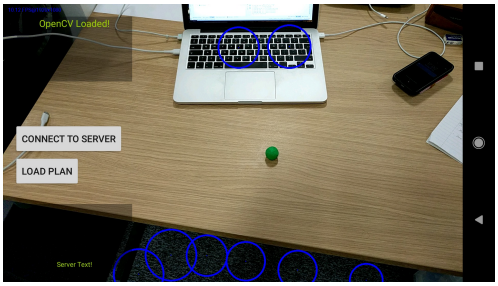


Figure 6-8: Moving just 1-2 meters away would breaks the tracking (blue circles are noise).

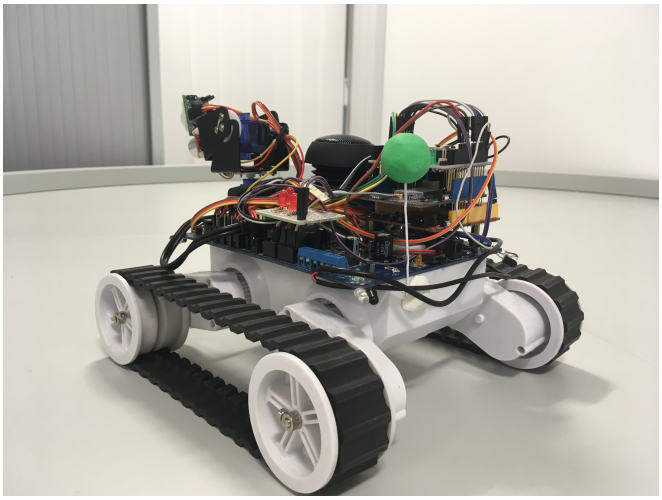


Figure 6-9: Installing the green ball on the R5 Robot.

Figure 6-9 shows the ball as we installed it on the R5 Robot. In order to find out what caused the problem of detecting/multiple circles as seen in figure

6-8, different stages of the image processing pipeline were compared with the end result. The problem was found in the result of the colour thresholding function `Core.inRange(...)`. In figures 6-10, 6-11 there are multiple pixels that were set to white after the colour threshold operation. This caused the circle detection to return not only the target green sphere but loads of unrelated circles. This was because some of the white pixels returned defined a circle. Similarly figures 6-12 and 6-13 show the same but with the ball installed on the robot. It seems that the round speaker on the robot was also picked up as a circle, even though it is black. Figure 6-14 show the current tracking method working with an alpha version of the plan loaded. A video can be watched of running one of the early app versions using the circle Hough transform. This was filmed in the lab were the lighting conditions were much better than outside in the field study: https://youtu.be/r-j2McVHFbI.

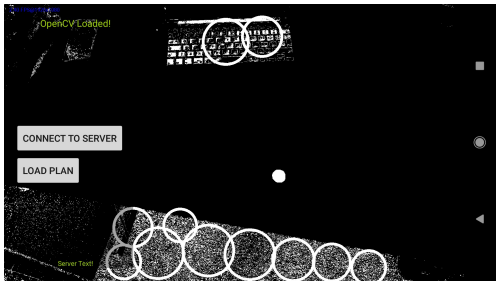


Figure 6-10: Detecting multiple extra circles on the colour thresholded frame

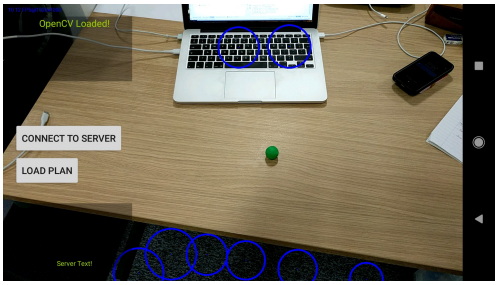


Figure 6-11: Detecting multiple extra circles on the colour frame

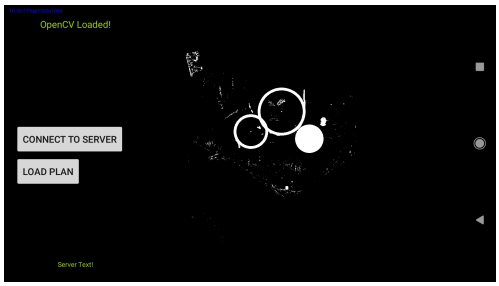


Figure 6-12: Detecting unwanted circles.

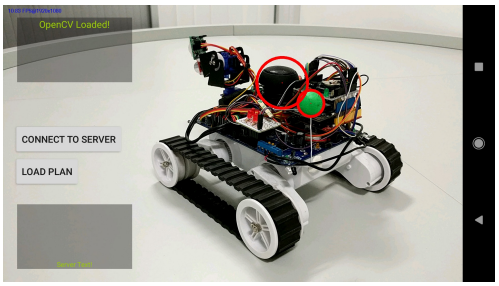


Figure 6-13: Detecting unwanted circles shown on colour frame.

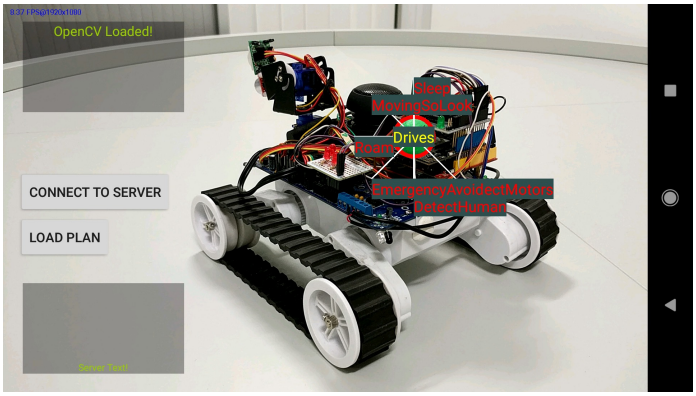


Figure 6-14: Loading an alpha version of the plan using the first implementation of the Circle Hough Transform (Running at 8 fps).

6.1.3 Eroding and Dilating

In addition to the Gaussian problem another approach was to use eroding and dilating. We explain in section 4.4.2 these two morphological transformations. OpenCV for Android provides methods `Imgproc.erode(...)` and `Imgproc.dilate(...)` respectively for each transformation. Results were quite improved than section 6.1.2, from just colour thresholding.

Figure 6-15: Previous results using colour thresholding and Gaussian blurring - Thresholded image (No Erosion/Dilation).

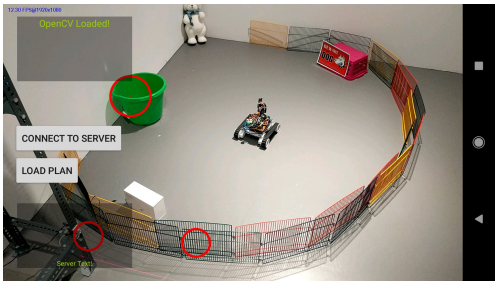


Figure 6-16: Previous results using colour thresholding and Gaussian blurring - Color image (No Erosion/Dilation).

There is a significant difference between eroding and dilating and not. You can notice the circles in fig. 6-16 are now gone in fig. 6-18. The difference in improvement is much more noticeable as-well in fig. 6-19 versus fig. 6-20. Where there is a big circle in fig. 6-19 but in fig. 6-20, it is gone. The noise in fig. 6-15 (lower left corner) disappears in fig. 6-17.

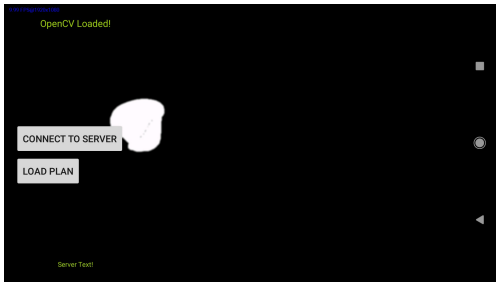


Figure 6-17: Eroding and Dilating previous results using colour thresholding and Gaussian blurring - Thresholded image.

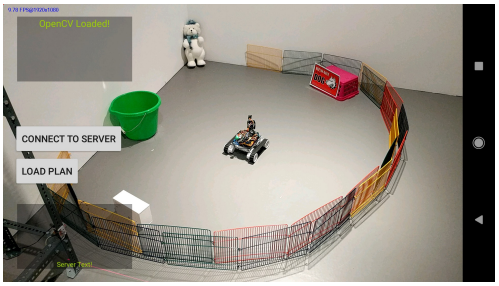


Figure 6-18: Eroding and Dilating previous results using colour thresholding and Gaussian blurring - Color image.

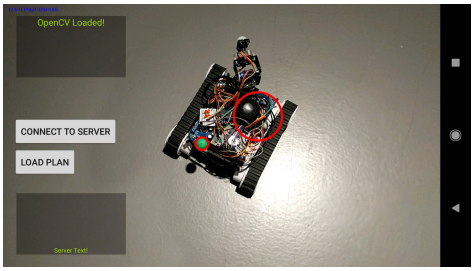


Figure 6-19: Previous results using colour thresholding and Gaussian blurring - Shot from a closer distance (No Erosion/Dilation).

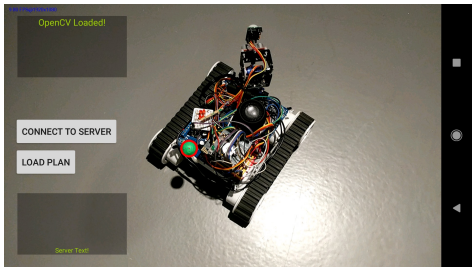


Figure 6-20: Eroding and Dilating previous results using colour thresholding and Gaussian blurring - Shot from a close distance.

6.1.4 Gaussian Blur

Even though a Gaussian blur was applied on the colour thresholded frame (binary image) after the erosion and dilation process, the problems from section 6.1.2 occurred. We used a Gaussian 11 by 11 pixels template and a Gaussian kernel standard deviation in X direction of 3 and a Gaussian kernel standard deviation in Y direction of 3 as seen in Appendix B.3. After experimenting with those parameters we found that those specific values gave the best results, in term of performance and quality. The sphere was detected in every frame with great success. The problem still remained, there was too much noise that the current implementation could not handle. At this point the *fps* (i.e frames per second) number was quite low returning on average 8-12 fps which was far from the 20-30 fps required.

6.1.5 Performance Problems

OpenCV for Android (core + extra modules) has not been fully ported to Android. The core functionality has been; but not the extra modules of it[5]. The reason is that developers of OpenCV decided to keep non-free features (such as SIFT and SURF) outside the core version of OpenCV. There was an attempt to compile them on our platform with no success as we came across compilation errors that we could not solve. The current implementation of tracking a green ball running at a resolution of 1920 by 1080 pixels ran at a rate of 8 to 12 frames per second, which was far from our 20 to 30 fps. Another attempt was to use `JavaCamera2View` instead of `JavaCameraView`, which are both responsible for connecting the hardware camera to OpenCV on Android, and the first promises better performance[6]. Because `JavaCamera2View` is quite new and has a number of bugs; it was not compatible with the Google Pixel phone and the frames returned were full of random noise.

Gaussian Blur using C++ (Android NDK)

On attempt to solve the performance issue with OpenCV was to refer to Android NDK. This allows the developer to write C++ code on the Java-based platform of Android. Code in 6.2 shows the C++ code that we attempted to run. This increased the frames per second by 2-4 on average totalling in 12 frames per second on average. This again was not enough to satisfy the initial requirements.

[5] https://github.com/opencv/opencv_contrib
[6] https://github.com/opencv/opencv/issues/11229

Code Snippet 6.2: NDK C++ Gaussian

```
JNIEXPORT void JNICALL
  Java_com_alexbath_abod3ar_MainActivity_gaussianBlur(
    JNIEnv *env,
    jobject,
    jlong sourceAddress,
    jlong targetAddress) {
      Mat &source = *(Mat *) sourceAddress;
      Mat &target = *(Mat *) targetAddress;

      GaussianBlur(source, target, Size(7, 7), 3, 3);
      }
}
```

6.1.6 Alternative solutions to OpenCV

At this point in the development stage the problems that accumulated from using OpenCV required to many tweaks that it was impossible to create a sustainable robot-tracking solution for the upcoming field study. Lowering the resolution from 1920 by 1080 to one with the same aspect ration such as 1600 by 900 would give higher fps numbers and looking at Renderscript[7] for performance increases that would still not solve the tracking related problems such as user's distance from the robot and stability of tracking. The circle was not always detected at each frame, that resulted in the plan showing up in the default coordinates $(x = 0, y = 0)$ on the screen. In fig. 6-17 the sphere disappears after eroding and dilating. This approach was not ideal for large distances between the user and the robot. Adjusting parameters such as the minimum/maximum radius of the circle to be detected took ages and the result was not accurate. The increased size of the OpenCV Hough Transform three dimensional accumulator, crippled the mobile phone performance. How having a variable radius value and three dimensional accumulator affects performance and memory consumption was explained in section 4.4.

Last but not least adding an object on a robot is not suitable for industrial or domestic scenarios. This led us to explore alternative solutions, read about

[7] https://developer.android.com/guide/topics/renderscript/compute

VOT, video object tracking. OpenCV does offer solutions for this such as *KCF tracker* or *GOTURN* implementation, but they could not have been used as mentioned in section 6.1.5 (extra modules). This lead us to *BoofCV*.

6.2 Phase 2 - BoofCV

BoofCV is an open source Java library written by Peter Abeles[8] for real-time computer vision and robotics applications. Written from the ground up focusing on high performance. It includes, optimized low-level image processing operations, camera calibration, feature detection/tracking, structure-from-motion, and recognition. BoofCV has an Apache 2.0 license for both academic and commercial use.

6.2.1 BoofCV Integration

The reason why *BoofCV* was selected is because it is written in Java, and that makes it compatible with the Android platform. This helped us integrate our current application with BoofCV, which was a major task. The reason is that most of the code had to be deleted as it was not relevant anymore, and very basic functionality, such opening a live camera preview feed, has to be re-written from scratch. The first version was running on a resolution of 1920 by 1080 which was sufficient for development purposes (running at 15+ fps). This was switched to 640 by 480 just before the field study as performance matched the initial requirements. Even though the resolution was low, on the mobile device it was not very obvious.

6.2.2 Tracking in BoofCV

BoofCV contains several general purpose tracking methods, and according to its documentation[9]:

1. Circulant (Local Tracker)
    - Simple and robust, but can not recover tracks

[8] https://boofcv.org/

[9] https://www.codeproject.com/Articles/797144/Object-Tracking-on-Android-and-Desktop

- Custom improvements in BoofCV where it has constant runtime independent of region size

2. Track-Learning-Detect (TLD)
   - Only long term tracking algorithm in BoofCV
   - More computationally expensive and can be finicky
3. Sparse-Flow
   - Only tracker in BoofCV which can estimate rotations
   - Is brittle and works best on planar objects
4. Mean-Shift Histogram
   - Matches the histogram of a local neighbourhood
   - Can be configured to crudely estimate scale
5. Mean-Shift Likelihood
   - Extremely fast but only works well when a single colour dominates

BoofCV creates a processor according to which tracker the developer chooses. Then the camera frames are passed in the appropriate processor and the output is what the user sees on the mobile device's screen. By trying all of them *TLD* and *Circulant* were rejected because they could not produce RGB frames as output. *Sparse-Flow* was buggy and it was performing terribly as soon as the phone (i.e user) would move significantly. The remaining options were *Mean-Shift Likelihood*, which did not work that well as the robot had minor parts of multiple colours (even though there was not much colour diversity). The working options were *Mean-Shift Histogram* and its option to estimate scale. The latter was too slow, so it was opted to go for the simple *Mean-Shift Histogram* (without estimation of scale). This was sufficient for the development phase. After meeting with Mr Ken Cameron it was noticed that the plan would "shake" (i.e randomly move in a small range in the x and y axes) this was due to the tracking not being stable enough.

A video showing the very first app version integrated with BoofCV, using the first tracking method *Circulant* (as you can see greyscale frames).

https://youtu.be/mE4XftFZ8dk

6.2.3 Positioning an Instinct Plan using BoofCV

The user is asked to draw an area on the screen of the object that will represent a ROI, region of interest. Then that ROI would be tracked as a rectangular area. This happens on each incoming frame (around 30 fps). The center of that rectangular is calculated and similarly as in section 4.4, were a sphere's center was used, now the center of this rectangular is used to place the root of the plan. At this point we did not consider occlusion or the ROI being obscured. It is worth mentioning that *'The tracking algorithms below are referred to as general purpose because they algorithms make few assumptions about the environment. For example, they don't assume the camera is stationary.'*[10]. This is ideal as we did not expect a user to look at the robot and be completely stationary while holding the phone. Videos in this section demonstrate how BoofCV works. How the user is selecting the ROI and after releasing the center (yellow dot) is calculated for each frame and that point is used for the actor point of the plan. An example can be seen in the video:

https://www.youtube.com/watch?v=HyBqev57k58

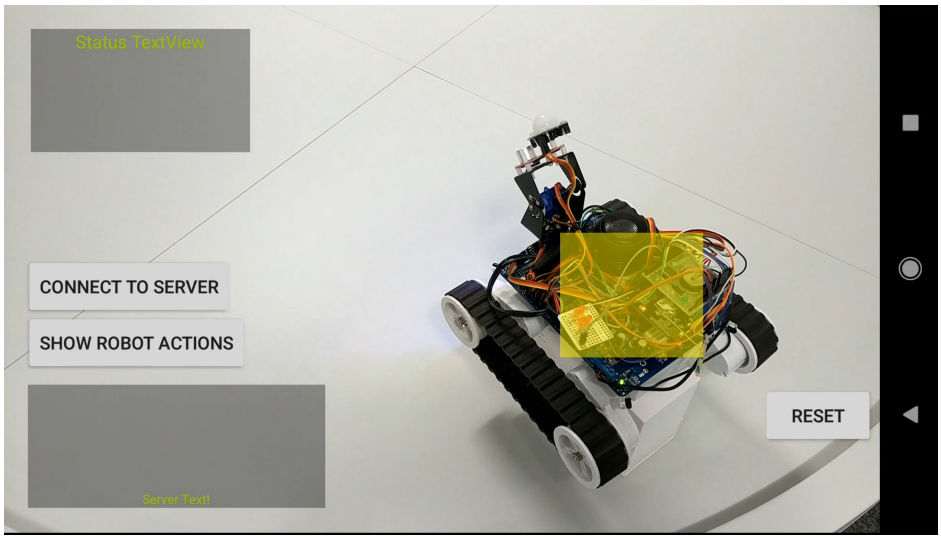


Figure 6-21: User selecting the ROI.

Note that this was running at Full HD that is 1920 by 1080 but after experimenting with resolutions of the similar aspect ratio we found that 640

[10]https://www.codeproject.com/Articles/797144/Object-Tracking-on-Android-and-Desktop

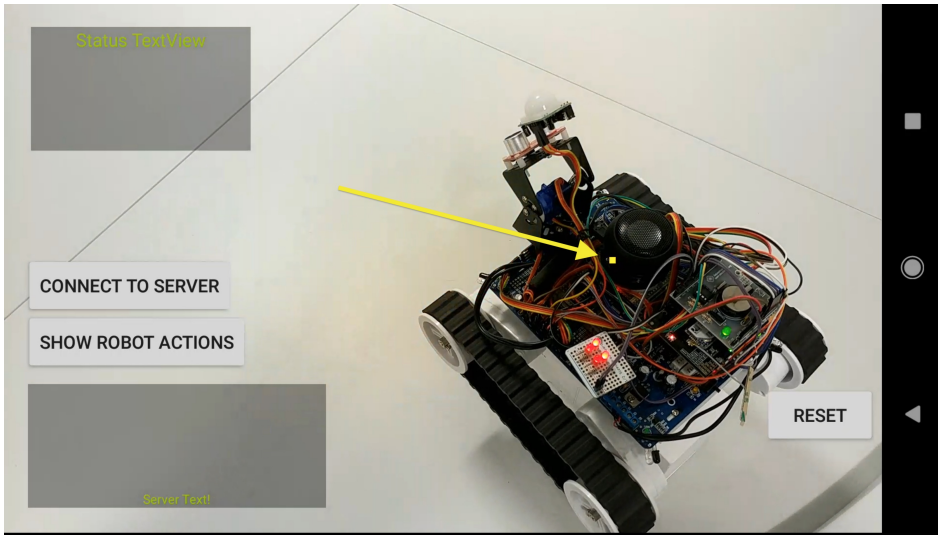


Figure 6-22: After the user has released, the center is calculated and shown.

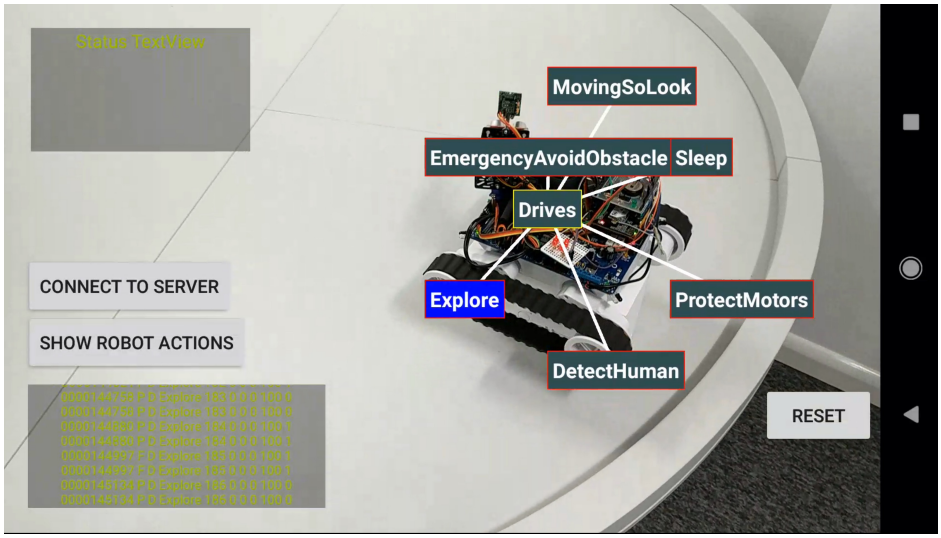


Figure 6-23: When loading the plan, the yellow center is used to "anchor" the root of the plan.

by 480 was ideal. BoofCV as of now does not offer a realtime fps counter as OpenCV does. The difference was noticeable and much smoother with using the latter. Figures 6-21, 6-22, and 6-23 show how the main workflow using BoofCV is executed.

6.2.4 Matrix transformation for lower resolution

In order to make the lower resolution work as seen in fig. 6-26 since the camera (video/image at 640 by 480 pixels) live stream was smaller than the drawing area (canvas/view at 1920 by 1080) there needs to be a transformation of a pixel from the camera feed to the drawing area. BoofCV uses a matrix called `imageToView` to achieve that conversion (i.e of a pixel in video frame to view frame). The inverse of that matrix, `viewToImage`, achieves a conversion from a view frame (canvas) to the camera feed, coordinates system. Figures 6-24 and 6-25, show how the matrix transformations were used to achieve a working version of the app at 640 by 480. The reason that matrix transformation are used here is also because it deals with rotations without rotating the whole Android view. Rotating a camera preview in Android is a very cumbersome process.

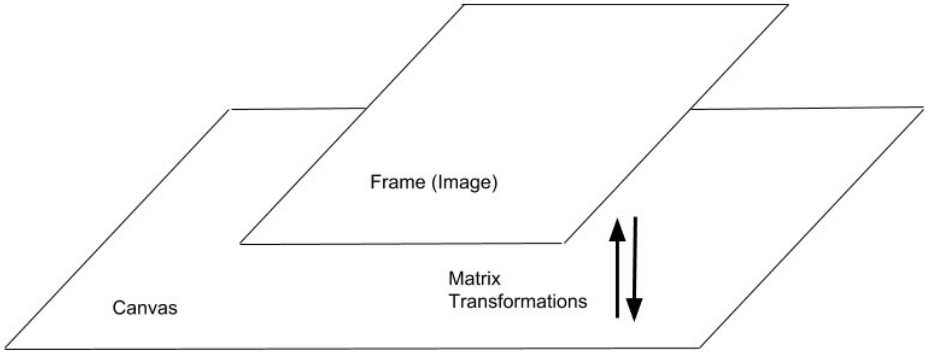


Figure 6-24: A diagram showing the matrix transformations as needed.

In order to get the plan anchor to the center (the red and yellow) dot in fig. 6-26, the center's coordinates had to be transformed to the canvas coordinate system. The code is:

```
Point2D_F64 viewCenter = getViewCenter(location, imageToView);
```

The `imageToView` is the matrix as explained before, and location is a data structure of the type *Quadrilateral_F64*[11] that all it does it holds the four coordinates that make up the ROI. The resulting variable `viewCenter` was

[11] http://georegression.org/javadoc/georegression/struct/shapes/Quadrilateral_F64.html

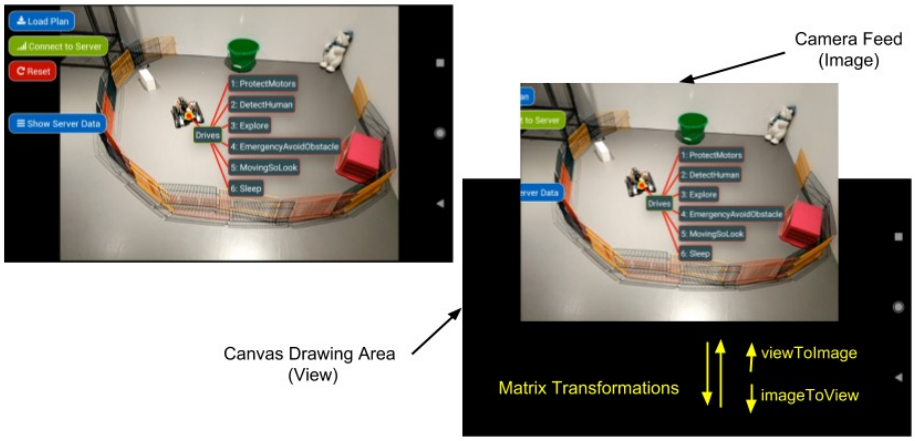


Figure 6-25: A more detailed sketch showing the matrix transformations as needed, this time including a sample camera frame and the canvas (black area).

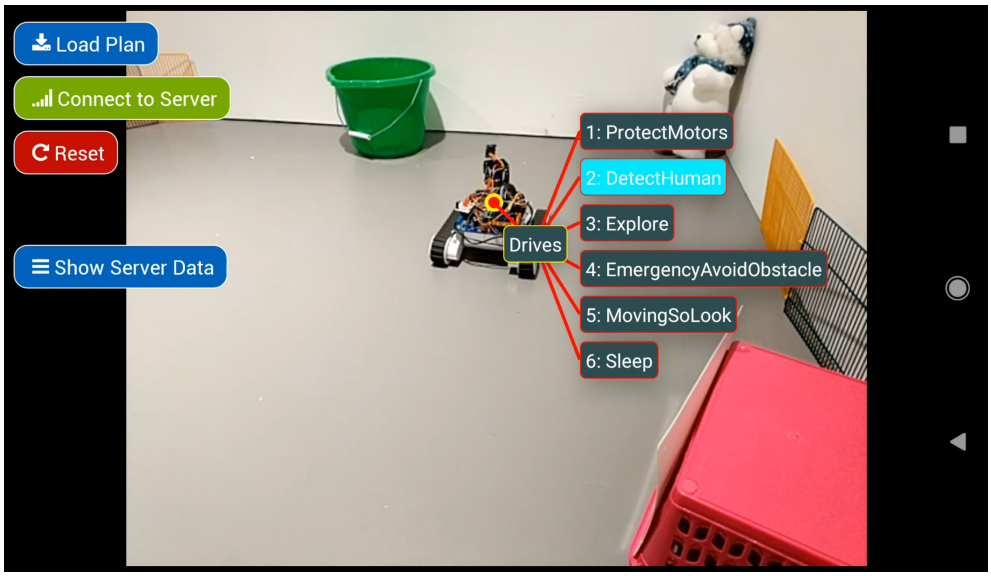


Figure 6-26: Running at 640 by 480 pixels.

used to draw the plan's tree structure on the screen in AR. The code for this is contained in file `ObjectTrackerActivity.java`.

In fig. 6-26 you can see clearly that the camera feed is smaller than the canvas area. The quality is still more than adequate and the performance was fluid. An example is found in this video, https://youtu.be/Gqqng1uh3Ys.

6.2.5 Modifying the BoofCV Circulant Tracker and TLD

Circulant Tracker and *Track-Learning-Detect (TLD)* both methods did not render in RGB as explained earlier. This was happening because *BoofCV* set the incoming's frame from the camera hardware to the same colour type the current processor supports. In this case *Circulant Tracker* and *Track-Learning-Detect (TLD)* processor's by default their colour type was grayscale. The solution to this problem was to replace code in 6.3 with code in 6.4.

Code Snippet 6.3: Default code

```
setImageType(processor.getImageType(),
                    processor.getColorFormat());
```

Code Snippet 6.4: Setting the incoming frame to RGB

```
setImageType(new ImageType(ImageType.Family.PLANAR,
                                        ImageDataType.U8,3),
                                        ColorFormat.RGB)
```

The processor would still fail at this point because it was given an RBG image. A simple averaging of the RGB frame before passing it to the processor worked fine as seen in 6.5.

Code Snippet 6.5: Grayscaling the incoming frame before processing

```
GrayU8 grayU8Image = new GrayU8(image.getWidth(),
                                        image.getHeight());

ConvertImage.average((Planar) image, grayU8Image);
processor.process(grayU8Image);
```

The RGB frame was kept to render back on the screen. In this scenario the only information that was required was the coordinates of a the center of the ROI, and not anything based on colour. The code for this is found in `Camera2Activity.java`.

6.2.6 Drawing on same canvas problem

In the first version of the app that used BoofCV the drawing was happening on the same surface that the camera frames were shown. This caused a problem that was only noticed while using the Track-Learning-Detect (TLD) method.

This is because the ROI was changing too fast, because of the flashing of the nodes. The drawing was happening on the same surface the algorithm from Track-Learning-Detect (TLD) was using and was failing because of this. The area ROI, that the algorithm was trying to detect was changing faster than it could learn it. A solution to this problem was to have two surfaces, one for the camera stream and one for the drawing of the tree. This way when the nodes of the tree were flashing it would not affect any camera frames and any of the trackers. The two views used was a `FrameLayout`[12] for the camera stream and a `ConstraintLayout`[13] used the layout on top to render the plan's tree. Code for this is found in the file `activity_camera.xml`.

6.2.7 Field Study App Tracker

As mentioned earlier *Mean-Shift Histogram* was used for most of the development phase. It was soon noticed that it was not tracking the ROI (robot), with enough stability. The new *anchor* coordinates' difference from the previous frame were quite substantial, and the tree of the plan would spring and bounce in nearby locations so the text was harder for the user to read. After taking into consideration occlusion, such as the user might accidentally put his hand in front of the phone's camera *Track-Learning-Detect (TLD)* was chosen. This was the starting app version for the first two days of the field study. It was proven that the *Track-Learning-Detect (TLD)* tracker was very high in memory consumption, and it would slow down the tracking to similar results as in section 6.1 (with OpenCV) were fps would fall down to 4-5 fps. Since after the first two days no issues related to occlusion occurred (i.e users blocking the view of the robot with their hands, etc.) this was no longer a requirement. The fallback was to use *Circulant Tracker*, due to its performance not degrading over time, and stability of the tracking, making the text stable enough for users to read. The main problem was temperatures of the mobile device. Using a third party app[14] the temperatures recorded on average were from 46 to 49 degrees of Celsius. The final frame drawing code using *BoofCV* can be seen in Appendix B.2.

[12] https://developer.android.com/reference/android/widget/FrameLayout
[13] https://developer.android.com/training/constraint-layout/
[14] https://play.google.com/store/apps/details?id=com.glgjing.stark

Chapter 7

Field Study

7.1 Introduction

Figure 7-1: The Fantastical Multimedia Pop-up Project

7.1.1 The Fantastical Multimedia Pop-up Project

The Fantastical Multimedia Pop-up Project was a project run from the 20th of July to the 24th of August 2018 at the Edge; at the University of Bath. A number of researchers were presenting their work. More details can be found in the web-link Edge Arts website. That is where we ran our experiment.

7.2 Setup

The setup consisted of a small perimeter (fences) and three, a bear, a green bucket and a sign were placed randomly inside the fences, as distactrions. Figures 7-2 and 7-4 illustrate the initial and final setups.

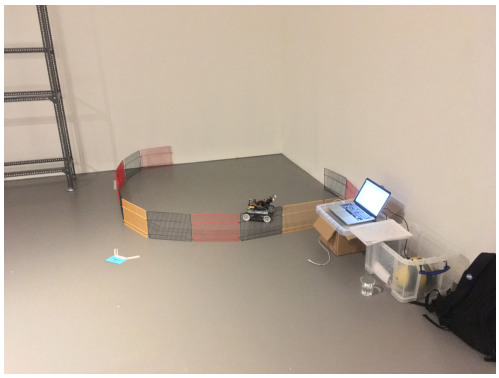


Figure 7-2: Robot area, before the room was fully set up.

Figure 7-3: Initial stages of setup.

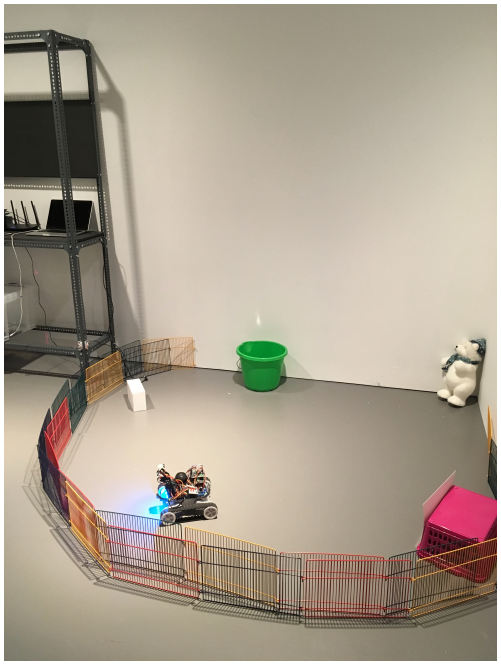


Figure 7-4: The experiment are after the setup was completed.

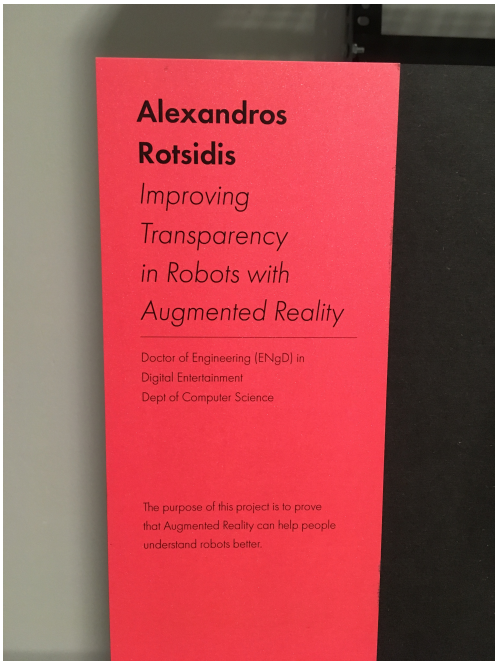


Figure 7-5: Project label.

The purpose of this experiment was to investigate if there would be any

change between users observing the robot using the app and user not using the app. We define these changes by using questionnaires based on the Godspeed questions explained more in section 7.3. The participants were asked to observe the robot by either looking at it without using the app or using the app for at least three minutes, and try to guess what the robot is doing, understand its objectives (if any) and try to construct a mental model. The user was able to use the app in anyway they wanted, and tap on any element on the screen to expand it. A video showing what a participant observed on the phone, while using the app can be found here: https://youtu.be/Gqqng1uh3Ys.

An **independent group design** was decided for this experiment; that means each participants was either allocated to group one or group two. The alternative was to have a *repeated measures design* having each participant be in both groups. That is observe the robot first then use the app to observe the robot. The reason we chose an **independent group design** was time restrictions (take less time for a participants to complete the experiment), and that also avoids the problem of fatigue which can cause distractions and boredom thus affecting results. Also participants' answers to the second part of the experiment might be influenced/biased by the answers they have given in the first part (Cozby, 2003).

To clarify, in the video the current user was **tapping** on the plan elements on the screen in order to learn more about the current action. The robot did not have any specific goals. It was running *Instinct Plan 6* and it mostly roam around, would stop if there was an obstacle in front of it. It could detect heat sources such as a human hand, so if a participant put his/her hand in front of the robot it should stop and turn, changing direction. Then when participants were done they were given a number of questionnaires as explain in section 7.3. Fig. 7-6 show a participant interacting with the app. This lays out the basic structure of our experiment; two experimental conditions and different participants for each group.

7.3 Questionnaires

Three questionnaires (plus a consent form) were handed to participants in total. The first was a basic demographics questionnaire the second was the most important that we used to produce most of the results in sections 7.4 and 7.5. The latter was based on the Godspeed questionnaire series Bartneck

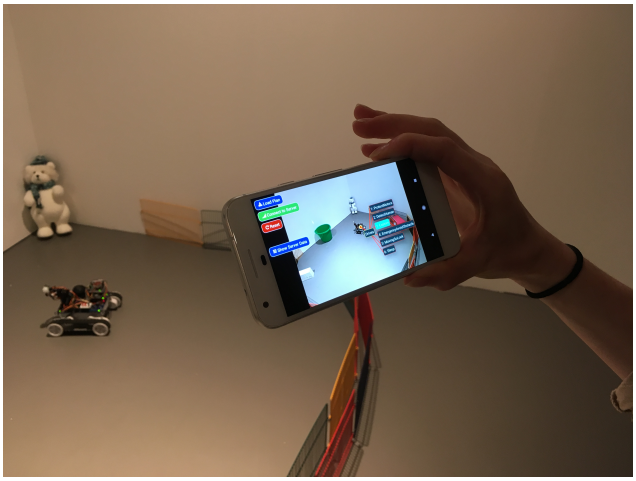


Figure 7-6: A participant using the app. The participant would tap on any element to get to the next level in the plan's tree.

et al. (2009) that are the standard questions used in research regarding Human Robot Interaction (HRI) projects, also used in similar research (Salem et al., 2015). Bartneck et al. (2009) used a Likert scale of 1 to 5, and they measure the users' perception of robots, which can help infer if there is a development of trust in users towards the robot. Questions are grouped in *Anthropomorphism*, *Animacy*, *Likeability*, *Perceived Intelligence*, *Perceived Safety*. The last questionnaire contained questions regarding the app, such as how good the tracking of the robot, if the text was readable etc. All of the questionnaires are found in the Appendix A.

7.4 Data Collection

Data was entered in SPSS [1] after the data collection was completed. The tests explained in section 7.5 were done in SPSS. The demographics information of each group of participants is shown in Table 7.1. The total number of participants were 45 ($N = 45$). Both groups were similar in terms of age and demographics information. The main difference in this sample set is that most participants compared to previous research (Wortham, Theodorou and Bryson, 2017a), did not have a STEM background.

[1] https://en.wikipedia.org/wiki/SPSS

Demographics	Group 1	Group 2
Total Participants	23	22
Highest Frequency Group	36-45	36-45
Gender Male	10	9
Gender Female	12	13
Gender Agender	1	0
Work with computers regularly (Yes) ?	20	21
Are you a software developer (Yes) ?	5	1
Do you have a background in STEM (No) ?	18	21

Table 7.1: Demographics of the 45 participants.

7.5 Statistical Analysis

The test used to get results from the Godspeed questions (Bartneck et al., 2009) was of type *independent samples t-test* or *unpaired t-test*. These type of tests are used when there are two experimental conditions and for each one, a group of participants is assigned. The groups can not share participants, they need to be different (Field, 2013). In addition a *t-test* is appropriate when means from both groups differ Cozby (2003). The *t-test* is one type of inferential statistics. The approach of using a *t-test* was also because it was used on previous related research from Wortham and Rogers (2017) and Wortham, Theodorou and Bryson (2017a). Hypothesis testing was used in combination with *t-tests* as explained in Forshaw (2007). It was concluded that based of Field (2013) and Forshaw (2007) this was the most optimal approach to analyse the results form the data collection, regarding the Godspeed questions. Wortham and Rogers (2017) used a similar approach aswell to previous research. For questions that had a binary answer such as *'Is the Robot Thinking? Yes/No'* we used simple frequencies charts, or clustered bar charts. This made it easier to understand results. For elaborate analysis for certain questions such as *'Would you trust a robot like this in your home ?'* we used a binomial test for each group.

7.6 Main Findings

The primary results obtained from the experiments are outlined in this section. The most explanatory way was to use figures. The groups were named **Group 1** for the group that did not use the AR app, and **Group 2** for the group that

did use the app. In figures, left cluster of bars is Group 1 and the right, Group 2. Blue represents **Yes**, and Green **No**. Figures 7-7, 7-8, 7-9, 7-10 illustrate some of the results from the Godspeed questionnaire. The y axis represents the number or participants and the x axis the groups.

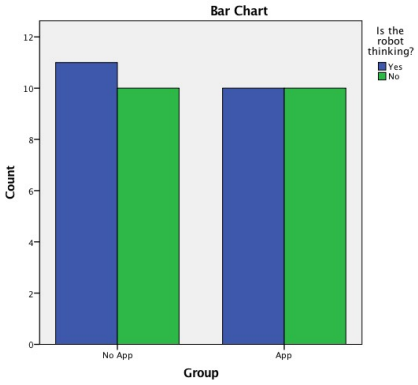


Figure 7-7: Is the robot thinking ?

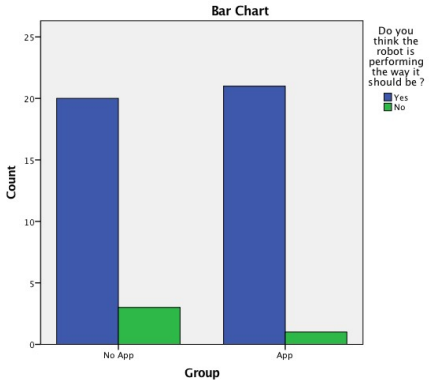


Figure 7-8: Do you think the robot is performing the way it should be ?

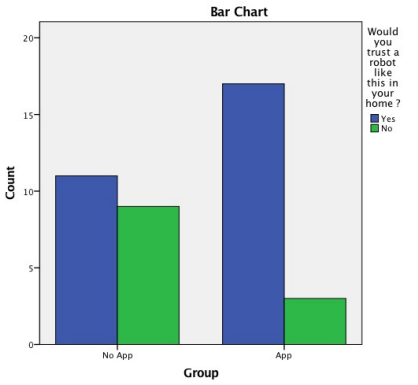


Figure 7-9: Would you trust a robot like this in your home ?

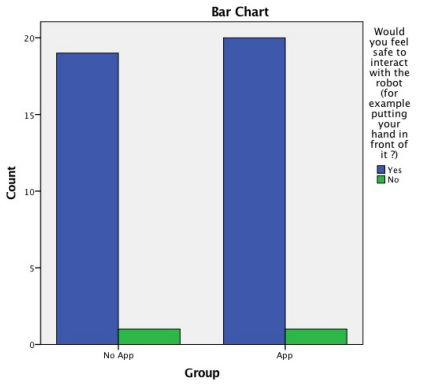


Figure 7-10: Would you feel safe to interact with the robot (for example putting your hand in front of it ?)

Figures 7-7, and 7-8, indicate that the perception of the participants about the robot did not change between groups. People that used the app reported the same with the people that did not use the app. In both groups participants felt safe to interact with the robot. Note the significant difference in fig. 7-9, showing that the extra transparency feed provided by the AR app **did**

increase trust for participants in Group 2 (using the app). Binomial testing[2] was carried out for each group, on the hypothesis that no difference in answers would occur in each group for the question *'Would you trust a robot like this in your home ?'*. The findings are found below in tables 7.3 and 7.2. In table 7.3 a *p-value* of less than 0.05 was calculated. This binomial test indicated that the proportion of participants that answered *"Yes"* of 0.85 was much higher than the expected 0.5 with a *p-value* of than 0.003. A significant result was defined as a *t-test* result that *p-value* of significance level is anything equal or less to 0.05 (Cozby, 2003).

Answer	N	Observ. Prop.	Test Prop.	p-value
Yes	11	0.55	0.50	0.824
No	9	0.45		

Table 7.2: Binomial test results for Group 1.

Answer	N	Observ. Prop.	Test Prop.	p-value
Yes	17	0.85	0.50	**0.003**
No	3	0.15		

Table 7.3: Binomial test results for Group 2.

The results for the Godspeed questions are found in Table 7.4. All were Likert[3] type of questions, with a scale from 1 to 5. **Bold** fonts indicates results significant to at least $p = 0.05$ or less. We concluded and rejected/accepted the corresponding null hypotheses based on the difference of means from both groups ($N = 45$), Group 1 ($n = 23$), Group 2 ($n = 22$) and the *p-value*. Participants were also asked to enter a value for the following emotional states, also part of the Godspeed questions as seen in Table 7.5.

By examining these two tables, 7.4 and 7.5, we can see that there were three significant results from Table 7.4. For the Godspeed questions **Dead - Alive**, **Stagnant - Lively** and **Unfriendly - Friendly**, i.e with *p-value* less than 0.05.

[2]https://en.wikipedia.org/wiki/Binomial_test
[3]https://en.wikipedia.org/wiki/Likert_scale

Question	Groups	Mean	Std. Dev.	p-value
Fake - Natural	No App	2.39	1.033	0.638
	App	2.55	1.143	
Machinelike - Humanlike	No App	1.87	1.014	0.97
	App	1.41	0.796	
Unconscious - Conscious	No App	2.26	1.096	0.487
	App	2.50	1.185	
Inconscient - Conscient	No App	2.61	1.196	0.743
	App	2.50	1.012	
(Moving) Rigidly - Elegantly	No App	2.09	1.041	0.225
	App	2.45	0.963	
Dead - Alive	No App	2.39	0.988	**0.01**
	App	3.27	1.202	
Stagnant - Lively	No App	3.30	0.926	**0.02**
	App	4.14	0.710	
Mechanical - Organic	No App	1.91	1.276	0.158
	App	1.45	0.800	
Artificial - Lifelike	No App	1.96	1.065	0.995
	App	1.95	1.214	
Inert - Interactive	No App	3.26	1.176	0.211
	App	3.68	1.041	
Apathetic - Responsive	No App	3.35	0.982	0.368
	App	3.64	1.136	
Dislike - Like	No App	3.57	0.728	0.435
	App	3.77	1.020	
Unfriendly - Friendly	No App	3.17	1.029	**0.041**
	App	3.77	0.869	
Unpleasant - Pleasant	No App	3.43	0.788	0.232
	App	3.77	1.066	
Awful - Nice	No App	3.61	0.656	0.494
	App	3.77	0.922	
Incompetent - Competent	No App	3.13	0.815	0.171
	App	3.55	1.143	
Ignorant - Knowledgeable	No App	2.70	1.063	0.699
	App	2.81	0.873	
Irresponsible - Responsible	No App	2.65	1.027	0.579
	App	2.81	0.814	
Unintelligent - Intelligent	No App	3.17	0.937	0.922
	App	3.14	1.153	
Foolish - Sensible	No App	3.43	0.728	0.980
	App	3.43	0.926	

Table 7.4: Godspeed Questions T-test Results

Question	Groups	Mean	Std. Dev.	p-value
Anxious - Relaxed	No App	4.15	0.933	0.308
	App	3.81	1.167	
Agitated - Calm	No App	4.10	0.852	0.863
	App	4.05	1.071	
Quiscent - Surprised	No App	2.45	0.945	0.203
	App	2.86	1.062	
Sad - Happy	No App	3.55	0.686	0.172
	App	3.86	0.727	
Bored - Interested	No App	3.80	0.834	0.110
	App	4.19	0.680	

Table 7.5: Godspeed Questions T-test Results (Participant Emotional State)

7.6.1 Hypotheses

The null hypothesis corresponds to the common belief about the parameter in question. It is interpreted as no change in the value of the parameter. The alternative hypothesis corresponds to a new claim which we wish to prove. It is interpreted as a change in the value of the parameter. The outcome of a test of significance (i.e if *p-value* is less than 0.05 (Cozby, 2003)) is the decision whether to reject or not the null hypothesis. We only list the significant questions, whose *p-value* was less that 0.05, i.e significant, from table 7.4.

Dead - Alive

Among all the participants ($N = 45$) that took part in the field study, there was a statistically significant difference between the two groups; one using the app and other not using the app while observing the robot for around three minutes. Group 1 ($M = 2.39, SD = 0.988$) and Group 2 ($M = 3.27, SD = 1.202$), $t(43) = -2.692$, $p \leq .05$. Therefore, the null hypothesis that there is no difference in perceiving if the robot is *dead* or *alive* between groups teams 1 and 2 is rejected. Group 2 perceived the robot to be more *alive*.

Stagnant - Lively

Among all the participants ($N = 45$) that took part in the field study, there was a statistically significant difference between the two groups; one using the app and other not using the app while observing the robot for around three

minutes. Group 1 ($M = 3.30, SD = 0.926$) and Group 2 ($M = 4.14, SD = 0.710$), $t(43) = -3.371$, $p \leq .05$. Therefore, the null hypothesis that there is no difference in perceiving if the robot is *stagnant* or *lively* between groups teams 1 and 2 is rejected. Group 2 perceived the robot to be more *lively*.

Unfriendly - Friendly

Among all the participants ($N = 45$) that took part in the field study, there was a statistically significant difference between the two groups; one using the app and other not using the app while observing the robot for around three minutes. Group 1 ($M = 3.17, SD = 1.029$) and Group 2 ($M = 3.77, SD = 0.869$), $t(43) = -2.104$, $p \leq .05$. Therefore, the null hypothesis that there is no difference in perceiving if the robot is *unfriendly* or *friendly* between groups teams 1 and 2 is rejected. Group 2 perceived the robot to be slightly more *friendly*.

7.6.2 Users Verbal Feedback

Participants were asked to answer free text questions in addition to the Godspeed questions. In this section we list some of their responses. The questions are listed below divided in both groups.

In your own words, what do you think the robot is doing?

Group 1: Some of the answers were, *'Trying to build a memory of the distance between itself and the objects to judge its own location in space'*, *Processing Data'*, *'Random'*, *'I think the robot is actively looking for something specific. At some points he believes he has found it (flashes a light) but then continues on to look'*, *'He is looking for something'*. More responses were *'Taking pictures of the objects'*, *'Occasionally taking pictures'*, *'Analyzing data'*. Note that some people here referred to the robot as **he**.

Group 2: Some of the answer from participants *'Exploring*, *'Imitating commands, responding to stimuli'*, *'The robot is registering programmed behaviours and connecting it to it surroundings'*, *'Exploring its surroundings and trying to detect humans'*, *'The robot likes to scan for obstacles, humans and find new paths to follow it can understand animals and obstacles'*. The last participants

was referring to the bear. *'The movement looks random I would say it is using sensors to avoid the obstacles'*, *'Roaming detecting objects and movement through sensors'*. Some noticeable feedback after using the app, was *'It looks less anthropomorphic with the app'*.

Some of the responses from the Group 2 were very accurate compared to the responses from Group 1 and taking into consideration the robot's true behaviour as discussed in 7.2. It seems that Group 2 developed a more accurate mental model of the robot.

7.7 Discussion

From the above results in section 7.6 and 7.6.2 it seems that users that used the app, and had a live feed of the robot actions seemed to have associated this with human thoughts. The increase in trust led to assignment of human attributes such as liveliness and friendliness as shown from the results. Also having that live feed, helped remove any privacy and security concerns. This seems to have contributed in perceiving the robot as more **alive**, **lively** and **friendly**. It is worth noting here that whereas in previous research most participants had a STEM degree Wortham, Theodorou and Bryson (2017a), in our experiment they did not. This proves that using the AR app, it can increase trust and improve perception for a person even if that person does not have a STEM background. When it came to the emotional responses we did not find anything significant from the results of the t-tests, but from verbal feedback such as *'It looks less anthropomorphic with the app'* it seems that this confirms predictions from previous research Wortham, Theodorou and Bryson (2017a) such as *'We had expected that if ABOD3 resulted in increased transparency, that there would be a corresponding reduction in the use of anthropomorphic cognitive descriptions'*. From the verbal feedback from both groups it was clear that the participants from group two showed a clearer and better understanding of the robot's tasks and intelligence. This resulted in an improved mental model of the robot. There could also infer the robot's tasks with more accuracy than group one.

7.7.1 App Feedback

In this section the users' feedback about the app is visualised using pie charts. This only concerns participants that belonged to group two. Pie charts are used to show users' responses. The following questions used a *Likert scale* from 1 to 5 and some a binary answer, i.e *Yes/No*.

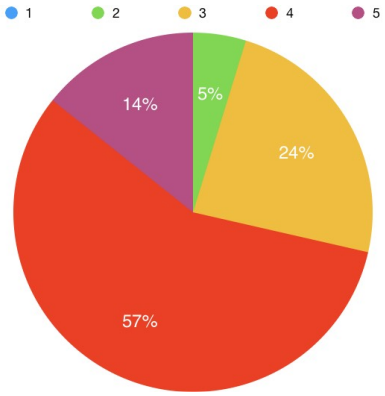


Figure 7-11: How would you rate the mobile app ?

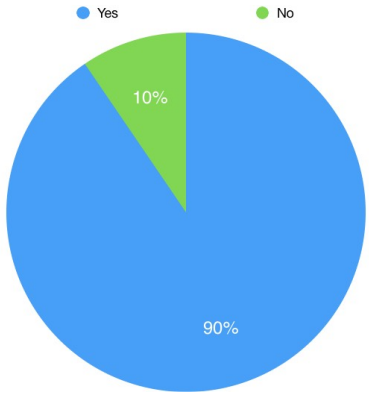


Figure 7-12: Was the text on the screen clear and stable enough to read (Yes/No)?

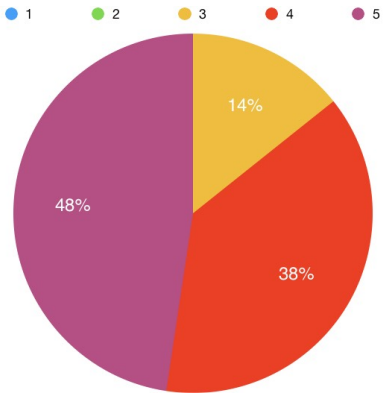


Figure 7-13: How easy was to understand the robots current instructions ?

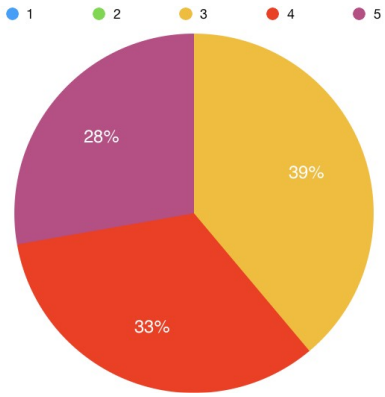


Figure 7-14: How good was the tracking of the robot ?

From the data we can infer that the UI was very successful at delivering the plan of the robot on the screen, 90% of users thought the text was stable enough and clear to read. This shows how stable the tracking of the robot was. Another indication was the answer to the question *'How good was the*

tracking of the robot ?'. The results again showed that most users chose a value of 3 and higher on the Likert scale; 39% chose 3/5, 33% chose 4/5, and 28% chose 5/5 which means the tracking according to them was excellent. Overall the feedback was more than positive. Users found the app easy to use, and friendly. No complains were collected regarding the UI of the app, except some regarding the text size mostly from the elderly demographic. 54% of the participants chose a 4/5 rating of the app, which is a high percentage and 14% chose 5/5, that means the graded the app as excellent.

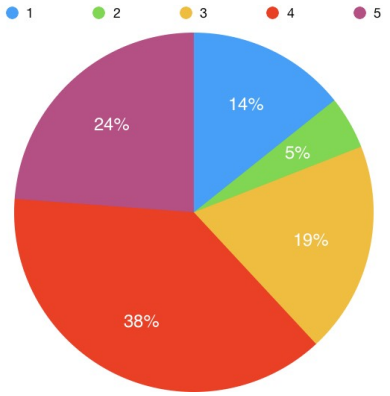


Figure 7-15: How likely are you to use this app in a human-robot collaborative work environment?

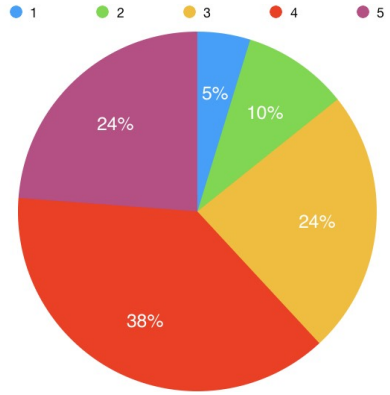


Figure 7-16: How likely are you to use this app in a human-robot collaborative domestic environment?

To support this figures 7-17, 7-15 and 7-16, show that a high percentage of participants would likely use the app in a domestic and work environment.

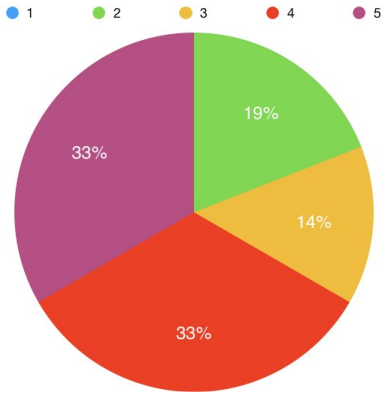


Figure 7-17: You encounter a robot in a hotel-lobby. How likely are you to use this app ?

7.7.2 User Answers - How can we improve the app ?

Participants that took part in the second group (using the app) were asked to answer the question *'How can we improve the app ?'* in the questionnaire related to the app feedback found in the Appendix. Some of their answers were, *'Bigger cleaner text'*, *'Not very sure'*, *'More aesthetically pleasing'*, *'Control the robot with the app'*, *'Humanising the terms'*. The last, referred to the Instinct plan elements names. More answers were *'No specific thoughts. The app was easy to use'*, *'I don't think you could the app was fully functional'*.

Chapter 8

Conclusion

As far as we are aware this is the first attempt that uses mobile augmented reality and focuses solely in increasing transparency in robots and users' trust. Previous research relied on screen and audio output or non real-time transparency. The novelty in this project lies in the use of augmented reality in contrast to using conventional ways such as screens or speakers. There are several assets of augmented reality that makes it a promising platform for both industrial and domestic robots. These include the affordability of AR enabled devices, its availability on multiple platforms such as mobile phones/tablets, the rapidly increasing progress in mobile processors and cameras; and the convenience of not requiring headsets or other paraphernalia unlike its competitor; virtual reality.

The code for this project could not be included in this dissertation because of its size. The code is available on Github[1] in a public repository. Instructions on how to run the app are provided in a *README* file on the repository's webpage. The mobile app has been named after *ABOD3* and *AR - Augmented Reality*, **ABOD3AR**.

https://github.com/alexs7/ABOD3AR

The source code of our modified version of the Instinct Server can be found in Github aswell:

https://github.com/alexs7/Instinct-Server/tree/develop

The code has also been attached in USB flash drives on the printed versions

[1] https://en.wikipedia.org/wiki/GitHub

of this thesis. We encourage the reader to view the Github repository though. A thorough literature survey has been conducted in the area of augmented reality and how it can be used with robotics and transparency in robots. This project's main contribution was proving that mobile augmented reality can be used in combination with previous research regarding transparency in robots and provide similar results and even better. Participants in our sample (group two; used the app) from the experiment did show an increase in trust and perception for the *R5 Robot* and concluded that it was more lively/friendly. In the verbal feedback they answered with a more accurate description of the robot's tasks compared to group one (i.e did not use the app).

In the near future, robots will take part in people's daily activities or collaborate with them in the workplace, can use similar applications based on **ABOD3AR** to understand what a robot's decision mechanisms are currently executing and give them an insight into them. This is a good starting point. This could be extended to drones, self driving cars, delivery robots anything that works based on a form of AI. The same principle could be transferred on AR-capable glasses such as *Magic Leap One*[2], or *Vuzix Blade AR*[3], or even headsets such as the *Microsoft Hololens*[4]. **ABOD3AR** received very positive feedback from the field study which is an indication that people will use a similar app if it is well designed.

[2]https://www.magicleap.com/magic-leap-one
[3]https://www.vuzix.com/products/blade-smart-glasses
[4]https://en.wikipedia.org/wiki/Microsoft_HoloLens

Chapter 9

Future Work

This research has many areas that are open to improvement. The robot used for this experiment was the *R5 Robot*. The University of Bath owns two *Pepper*[1] robots (humanoid-form), seen in fig. 9-1. The same application could be modified to use with the Pepper robots, leading to more research questions to answer. Tracking of the robot currently requires the user to manually select an area of ROI which contains the robot. Future versions of ABOD3AR would skip this part and replace it with a machine learning approach. This will enable the app the detect and recognize the robot by a number of features such as colour, shape and be able to retrieve its model and plan of execution from a database of robots. A simpler alternative to that will be to install a QR code on the robot, not used for the tracking, but used to scan the robot and retrieve its AI plan. Then a connection to an appropriate server would be established and data will be shown on the mobile device.

ABOD3 as explained in section 3.2 is a real-time debugging tool for BOD Agents. *ABOD3* capabilities were not fully used in this research, as not editing of the plan was incorporated in the app. The participants could not edit the Instinct plan of the robot, for example add more *Drives* or *Actions*. This was due to time limitations (of the experiment) and also because it was not the main focus of this research project. Future versions of ABOD3AR will allow the user to edit the robot's plan on the mobile device in AR as one would on the desktop version. The exact flashing ability of ABOD3 was not replicated in ABOD3AR due to Java API limitations as mentioned in Chapter 5. To replicate it, is set as future work.

[1] https://en.wikipedia.org/wiki/Pepper_(robot)

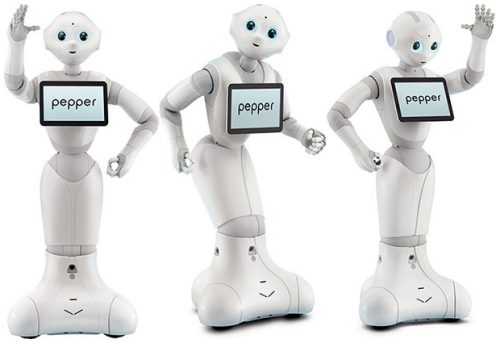


Figure 9-1: The Pepper Robot

Different experiments can be designed with the R5 Robot, for example a different plan can be loaded with a more complicated end task. For conducting more efficient experiments and get more concrete results the sample size should be increased, and variability between participants backgrounds and age should be taken into consideration. In addition to that users should be encouraged to interact with the robot more. In the case of the Pepper robots this leads to a myriad of experiments types. The environment of the experiment could also vary. It would be interesting to locate the experiment in an industrial environment and use production robots such as robotic arms in a automotive manufacturing factory. In turn, this influences the demographics. Longer-term goals would be to replace the mobile phone view. Remote interaction could be used instead but still with a smartphone connected to a remote camera (for example surveillance camera in a warehouse). Then new challenges arise for more efficient tracking, same research challenge arises if the robot is a drone or a self-driving car.

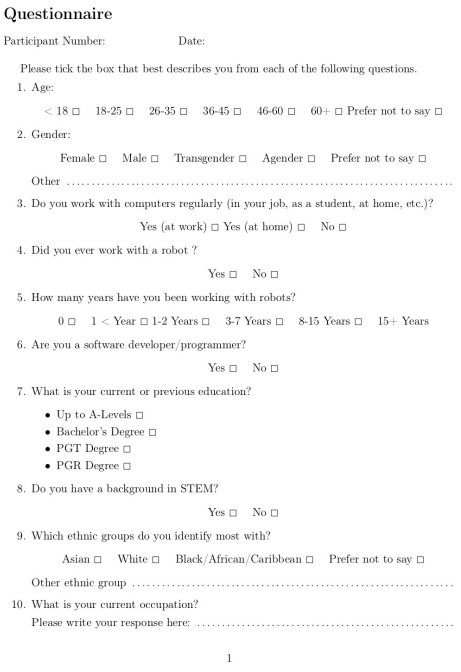
Appendices

Appendix A

Questionnaires

This section contains the questionnaires that were handed to participants during the field study (or experiment). These were all stapled together and given to the participant.

Figure A-1: The demographics questionnaire that was used in the field study



Questionnaire

Participant Number: Date:

Please tick the box that best describes you from each of the following questions.

1. Age:

   < 18 □ 18-25 □ 26-35 □ 36-45 □ 46-60 □ 60+ □ Prefer not to say □

2. Gender:

   Female □ Male □ Transgender □ Agender □ Prefer not to say □

   Other .....

3. Do you work with computers regularly (in your job, as a student, at home, etc.)?

   Yes (at work) □ Yes (at home) □ No □

4. Did you ever work with a robot ?

   Yes □ No □

5. How many years have you been working with robots?

   0 □ 1 < Year □ 1-2 Years □ 3-7 Years □ 8-15 Years □ 15+ Years

6. Are you a software developer/programmer?

   Yes □ No □

7. What is your current or previous education?

   - Up to A-Levels □
   - Bachelor's Degree □
   - PGT Degree □
   - PGR Degree □

8. Do you have a background in STEM?

   Yes □ No □

9. Which ethnic groups do you identify most with?

   Asian □ White □ Black/African/Caribbean □ Prefer not to say □

   Other ethnic group .....

10. What is your current occupation?

    Please write your response here: .....

1

Figure A-2: The consent form that was used in the field study (page 1/2)

CONSENT FORM UNIVERSITY OF BATH

Participant Number: Date:

Title of Research:

Understanding robots through observation.

Researcher:

Alexandros Rotsidis - ar2056@bath.ac.uk

Information about this project:

My name is Alexandros Rotsidis and I am running this experiment as part of my MSc in Digital Entertainment at the University of Bath, and would like to invite you to take part. My supervisor is Mr Ken Cameron. Before deciding whether or not you would like to take part, we would ask you to read this information sheet which describes why we are doing this research project, and what it would involve for you should you choose to participate. Your participation is completely voluntary. You are **free to withdraw** from it at any time. In this case any data you have provided up till that point will be deleted.

Purpose of this research:

Investigate people's understanding of robots through observation.

Confidentiality:

All records from this study will be kept confidential. Your responses will be kept private, and we will not include any information that will make it possible to identify you. The **anonymous** results of the project will be included in my master's thesis. You may request to have a copy of the thesis.

Study Procedures:

You are invited to examine the robot for at least 3 minutes and then asked to fill a questionnaire.

Who can take part?

You cannot take part if you have epilepsy or for some reason you cannot use a mobile device with a touchscreen.

Figure A-3: The consent form that was used in the field study (page 2/2)

Please fill in with your initials:

1. I confirm that I have read and understand the consent form dated for the above project. I understand that my participation is voluntary. I agree to take part in the project.
   .....

Participant's Signature: .....

Date: .....

Figure A-4: The Likert scale questionnaire given after the expirement (page 1/2) in the field study

Questionnaire

Participant Number: Date:

1. Is the robot thinking ?

   Yes □ No □

2. In your own words, what do you think the robot is doing?

   .....
   .....
   .....
   .....

3. Please rate your impression of the robot on these scales:

Fake	1	2	3	4	5	Natural
Machinelike	1	2	3	4	5	Humanlike
Unconsious	1	2	3	4	5	Conscious
Inconscient	1	2	3	4	5	Conscient
Moving rigidly	1	2	3	4	5	Moving elegantly
Dead	1	2	3	4	5	Alive
Stagnant	1	2	3	4	5	Lively
Mechanical	1	2	3	4	5	Organic
Artificial	1	2	3	4	5	Lifelike
Inert	1	2	3	4	5	Interactive
Apathetic	1	2	3	4	5	Responsive
Dislike	1	2	3	4	5	Like
Unfriendly	1	2	3	4	5	Friendly
Unpleasant	1	2	3	4	5	Pleasant
Awful	1	2	3	4	5	Nice
Incompetent	1	2	3	4	5	Competent
Ignorant	1	2	3	4	5	Knowledgeable
Irresponsible	1	2	3	4	5	Responsible
Unintelligent	1	2	3	4	5	Intelligent
Foolish	1	2	3	4	5	Sensible

4. Do you think the robot is performing the way it should be ?

   Yes □ No □

Figure A-5: The Likert scale questionnaire given after the expirement (page 2/2) in the field study

5. Would you trust a robot like this in your home ?

   Yes □ No □

6. Please rate your emotional state on these scales:

Anxious	1	2	3	4	5	Relaxed
Agitated	1	2	3	4	5	Calm
Quiescent (Neutral)	1	2	3	4	5	Surprised
Sad	1	2	3	4	5	Happy
Bored	1	2	3	4	5	Interested

7. Would you feel safe to interact with the robot (for example putting your hand in front of it ?)

   Yes □ No □

8. The way the robot moved made me uncomfortable

   Yes □ No □

Thank you for filling out this questionnaire! Please notify the researcher that you have finished.

Figure A-6: The app feedback questionnaire given after the expirement in the field study

Questionnaire

Participant Number: Date:

Please answer the following questions, only if you have used the app.

1. How would you rate the mobile app?

   Terrible 1 2 3 4 5 Excellent

2. Was the text on the screen clear and stable enough to read ?

   Yes □ No □

3. How easy was to understand the robot's current instructions ?

   Very Hard 1 2 3 4 5 Very Easy

4. How likely are you to use this app in a human-robot collaborative domestic environment?

   Very Unlikely 1 2 3 4 5 Very Likely

5. How likely are you to use this app in a human-robot collaborative work environment?

   Very Unlikely 1 2 3 4 5 Very Likely

6. You encounter a robot in a hotel-lobby. How likely are you to use this app?

   Very Unlikely 1 2 3 4 5 Very Likely

7. How good was the tracking of the robot ?

   Terrible 1 2 3 4 5 Excellent

8. How could we improve the app?

   .....
   .....
   .....

Thank you for filling out this questionnaire! Please notify the experimenter that you have finished.

1

Appendix B

Code

B.1 Generic Java Network Thread

Code Snippet B.1: Code for basic networking Thread

```
networkEnquirerThread = new Thread(new Runnable() {
  @Override
  public void run() {
  ...
  try {
    socket = new Socket("192.168.178.21", 3001);
    out = new PrintWriter(socket.getOutputStream(), true
        );
    br = new BufferedReader(new InputStreamReader(socket
        .getInputStream()));
    while(true){
    try {
      Thread.sleep(150);
    }...
    response = br.readLine();
    ...
    message.obj = response;
    networkEnquirerResponseHandler.sendMessage(message);
    ...
```

B.2 Final Frame Drawing Code

Code Snippet B.2: Code for basic frame drawing (part 1/3)

```
@Override
public void onDraw(Canvas canvas, Matrix imageToView) {
  canvas.concat(imageToView);
  if( mode == 1 ) {
    Point2D_F64 a = new Point2D_F64();
    Point2D_F64 b = new Point2D_F64();

    applyToPoint(viewToImage, click0.x, click0.y, a);
    applyToPoint(viewToImage, click1.x, click1.y, b);

    double x0 = Math.min(a.x,b.x);
    double x1 = Math.max(a.x,b.x);
    double y0 = Math.min(a.y,b.y);
    double y1 = Math.max(a.y,b.y);

    canvas.drawRect((int) x0, (int) y0, (int) x1, (int)
        y1, paintSelected);
  } else if( mode == 2 ) {
    if (!imageToView.invert(viewToImage)) {
      return;
    }
    applyToPoint(viewToImage,click0.x, click0.y,
        location.a);
    applyToPoint(viewToImage,click1.x, click1.y,
        location.c);

    // make sure the user selected a valid region
    makeInBounds(location.a);
    makeInBounds(location.c);
    ...
```

Code Snippet B.3: Code for basic frame drawing (part 2/3)

```
    ...
    if( movedSignificantly(location.a,location.c) ) {
      // use the selected region and start the tracker
      location.b.set(location.c.x, location.a.y);
      location.d.set( location.a.x, location.c.y );

      visible = true;
      mode = 3;
    } else {
      runOnUiThread(() -> Toast.makeText(
          ObjectTrackerActivity.this,
      "Drag a larger region", Toast.LENGTH_SHORT).show()
          );
      mode = 0;
    }
  }
  if( mode >= 2 ) {
    if( visible ) {
      if(uiPlanTree != null){

        Point2D_F64 imageCenter = getImageCenter(
            location);
        //view = canvas
        Point2D_F64 viewCenter = getViewCenter(location,
            imageToView);

        int startingXPoint = (- uiPlanTree.
            getFocusedNode().getData().getView().getWidth
            () / 2) + 80 ;
        int startingYPoint = (- uiPlanTree.
            getFocusedNode().getData().getView().
            getHeight() / 2) + 80 ;
      ...
```

Code Snippet B.4: Code for basic frame drawing (part 3/3)

```
        ...
        canvas.drawCircle((float) imageCenter.x, (float)
            imageCenter.y, 8, yellowPaint);
        canvas.drawCircle((float) imageCenter.x, (float)
            imageCenter.y, 5, redPaint);

        uiPlanTree.setUpTree(startingXPoint,
            startingYPoint,viewCenter);

        if(uiPlanTree.getFocusedNode().getParent() !=
            null){
          drawTreeUIElementsConnectors(uiPlanTree.
              getFocusedNode().getParent(),canvas,
              viewToImage,imageCenter);
        }else{
          drawTreeUIElementsConnectorsInitState(
              uiPlanTree.getRoot(),canvas,viewToImage,
              imageCenter);
        }
      }
    } else {
      canvas.drawText("x",width/2,height/2, textPaint);
    }
  }
}
```

B.3 Hough Transform Code

Code Snippet B.5: Code Snippet for basic circle Hough transform

```
public Mat onCameraFrame(CameraBridgeViewBase.
    CvCameraViewFrame inputFrame) {

frame = inputFrame.rgba();
Imgproc.cvtColor(frame,frameHSV,Imgproc.
    COLOR_BGR2HSV);
Core.inRange(frameHSV,lower,upper,thresh);
Imgproc.GaussianBlur(thresh, mat4, new Size(11,
    11), 3, 3);
Imgproc.HoughCircles(mat4, circles,
Imgproc.CV_HOUGH_GRADIENT,2.0,
mat4.rows() / 8, iCannyUpperThreshold,
iAccumulator, iMinRadius, iMaxRadius);

for (int x = 0; x < circles.cols(); x++){

double vCircle[]=circles.get(0,x);
Point center = new Point(Math.round(vCircle[0]),
Math.round(vCircle[1]));
int radius = (int)Math.round(vCircle[2]);
// draw the circle center
Imgproc.circle(frame, center, 3,new Scalar
    (255,0,0), -1, 8, 0 );
// draw the circle outline
Imgproc.circle( frame, center, radius, new
    Scalar(255,0,0), 3, 8, 0 );
}
return frame;
}
```

Bibliography

Alemzadeh, H., Iyer, R.K., Kalbarczyk, Z. and Raman, J., 2013. Analysis of safety-critical computer failures in medical devices. *IEEE Security Privacy*, 11(4), pp.14–26. Available from: http://doi.org/10.1109/MSP.2013.49.

Andersson, N., Argyrou, A., Nagele, F., Ubis, F., Campos, U.E., Zarate, M.O. de and Wilterdink, R., 2016. Ar-enhanced human-robot-interaction - methodologies, algorithms, tools. *Procedia CIRP*, 44, pp.193 – 198. 6th CIRP Conference on Assembly Technologies and Systems (CATS). Available from: http://doi.org/"https://doi.org/10.1016/j.procir.2016.03.022".

Azuma, R.T., 1997. A survey of augmented reality. *Presence: Teleoper. Virtual Environ.*, 6(4), pp.355–385. Available from: http://doi.org/10.1162/pres.1997.6.4.355.

Baraka, K., Rosenthal, S. and Veloso, M., 2016. Enhancing human understanding of a mobile robot's state and actions using expressive lights. *2016 25th ieee international symposium on robot and human interactive communication (ro-man)*. pp.652–657. Available from: http://doi.org/10.1109/ROMAN.2016.7745187.

Bartneck, C., Kulić, D., Croft, E. and Zoghbi, S., 2009. Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots. *International Journal of Social Robotics*, 1(1), pp.71–81. Available from: http://doi.org/10.1007/s12369-008-0001-3.

Billings, D.R., Schaefer, K.E., Chen, J.Y. and Hancock, P.A., 2012. Human-robot interaction: Developing trust in robots. *Proceedings of the seventh annual acm/ieee international conference on human-robot interaction*. New

York, NY, USA: ACM, HRI '12, pp.109–110. Available from: http://doi.org/10.1145/2157689.2157709.

Blanco-Novoa, O., Fernndez-Carams, T.M., Fraga-Lamas, P. and Vilar-Montesinos, M.A., 2018. A practical evaluation of commercial industrial augmented reality systems in an industry 4.0 shipyard. *IEEE Access*, 6, pp.8201–8218. Available from: http://doi.org/10.1109/ACCESS.2018.2802699.

Boden, M., Bryson, J., Caldwell, D., Dautenhahn, K., Edwards, L., Kember, S., Newman, P., Parry, V., Pegman, G., Rodden, T., Sorrell, T., Wallis, M., Whitby, B. and Winfield, A., 2017. Principles of robotics: regulating robots in the real world. *Connection Science*, 29(2), pp.124–129. https://doi.org/10.1080/09540091.2016.1271400, Available from: http://doi.org/10.1080/09540091.2016.1271400.

Breazeal, C., Kidd, C.D., Thomaz, A.L., Hoffman, G. and Berlin, M., 2005. Effects of nonverbal communication on efficiency and robustness in human-robot teamwork. *2005 ieee/rsj international conference on intelligent robots and systems*. pp.708–713. Available from: http://doi.org/10.1109/IROS.2005.1545011.

Bryson, J.J., 2002. The behavior-oriented design of modular agent intelligence. In: R. Kowalszyk, J.P. Muller, H. Tianfield and R. Unland, eds. *Agent technologies, infrastructures, tools, and applications for e-services.vol. 2592.*, Springer, Lecture Notes in Computer Science, pp.61–76. ID number: ISI:000182992800007. Available from: http://doi.org/10.1007/3-540-36559-1_7.

Camba, J.D., Leon, A.B.D., Torre, J. de la, Saorn, J.L. and Contero, M., 2016. Application of low-cost 3d scanning technologies to the development of educational augmented reality content. *2016 ieee frontiers in education conference (fie)*. pp.1–6. Available from: http://doi.org/10.1109/FIE.2016.7757673.

Chatzopoulos, D., Bermejo, C., Huang, Z. and Hui, P., 2017. Mobile augmented reality survey: From where we are to where we go. *IEEE Access*, 5, pp.6917–6950. Available from: http://doi.org/10.1109/ACCESS.2017.2698164.

Chu, M., Matthews, J. and Love, P.E., 2018. Integrating mobile building information modelling and augmented reality systems: An experimental study. *Automation in Construction*, 85, pp.305 – 316. Available from: http://doi.org/https://doi.org/10.1016/j.autcon.2017.10.032.

Comaniciu, D., Ramesh, V. and Meer, P., 2000. Real-time tracking of non-rigid objects using mean shift. *Proceedings ieee conference on computer vision and pattern recognition. cvpr 2000 (cat. no.pr00662)*. vol. 2, pp.142–149 vol.2. Available from: http://doi.org/10.1109/CVPR.2000.854761.

Comaniciu, D., Ramesh, V. and Meer, P., 2003. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5), pp.564–577. Available from: http://doi.org/10.1109/TPAMI.2003.1195991.

Cozby, P.C., 2003. *Methods in behavioral research*. McGrawHill.

Duda, R.O. and Hart, P.E., 1972. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1), pp.11–15. Available from: http://doi.org/10.1145/361237.361242.

Field, A., 2013. *Discovering statistics using ibm spss statistics*. 4th ed. Sage Publications Ltd.

Forshaw, M., 2007. *Easy statistics in psychology: a bps guide*, A BPS Blackwell book. The British Psychological Society/BPS Blackwell. Available from: https://books.google.co.uk/books?id=bOucAAAAMAAJ.

Gaudl, S.E. and Bryson, J.J., 2014. Extended ramp goal module: Low-cost behaviour arbitration for real-time controllers based on biological models of dopamine cells. *2014 ieee conference on computational intelligence and games*. pp.1–8. Available from: http://doi.org/10.1109/CIG.2014.6932887.

Hancock, P.A., Billings, D.R., Schaefer, K.E., Chen, J.Y.C., Visser, E.J. de and Parasuraman, R., 2011. A meta-analysis of factors affecting trust in human-robot interaction. *Human Factors*, 53(5), pp.517–527. PMID: 22046724. https://doi.org/10.1177/0018720811417254, Available from: http://doi.org/10.1177/0018720811417254.

He, J., Han, P., Liu, H., Men, S., Ju, L., Zhen, P. and Wang, T., 2017. The research and application of the augmented reality technology. *2017 ieee 2nd information technology, networking, electronic and automation control conference (itnec)*. pp.496–501. Available from: `http://doi.org/10.1109/ITNEC.2017.8284781`.

Henriques, J.a.F., Caseiro, R., Martins, P. and Batista, J., 2012. Exploiting the circulant structure of tracking-by-detection with kernels. *Proceedings of the 12th european conference on computer vision - volume part iv*. Berlin, Heidelberg: Springer-Verlag, ECCV'12, pp.702–715. Available from: `http://doi.org/10.1007/978-3-642-33765-9_50`.

Hoff, W., 2018. William hoff. Available from: `https://cs.mines.edu/project/hoff-william/` [Accessed 14/08/2018].

Imbert, N., Vignat, F., Kaewrat, C. and Boonbrahm, P., 2013. Adding physical properties to 3d models in augmented reality for realistic interactions experiments. *Procedia Computer Science*, 25, pp.364 – 369. 2013 International Conference on Virtual and Augmented Reality in Education. Available from: `http://doi.org/"https://doi.org/10.1016/j.procs.2013.11.044"`.

Johal, W., Kennedy, J., Charisi, V., Park, H.W., Castellano, G. and Dillenbourg, P., 2018. Robots for learning - r4l: Inclusive learning. *Companion of the 2018 acm/ieee international conference on human-robot interaction*. New York, NY, USA: ACM, HRI '18, pp.397–398. Available from: `http://doi.org/10.1145/3173386.3173567`.

Kalal, Z., Mikolajczyk, K. and Matas, J., 2012. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7), pp.1409–1422. Available from: `http://doi.org/10.1109/TPAMI.2011.239`.

Kim, T. and Hinds, P., 2006. Who should i blame? effects of autonomy and transparency on attributions in human-robot interaction. *Roman 2006 - the 15th ieee international symposium on robot and human interactive communication*. pp.80–85. Available from: `http://doi.org/10.1109/ROMAN.2006.314398`.

Lee, M.K. and Makatchev, M., 2009. How do people talk with a robot?: An analysis of human-robot dialogues in the real world. *Chi '09 extended*

abstracts on human factors in computing systems. New York, NY, USA: ACM, CHI EA '09, pp.3769–3774. Available from: http://doi.org/10.1145/1520340.1520569.

Makris, S., Karagiannis, P., Koukas, S. and Matthaiakis, A.S., 2016. Augmented reality system for operator support in humanrobot collaborative assembly. *CIRP Annals*, 65(1), pp.61 – 64. Available from: http://doi.org/https://doi.org/10.1016/j.cirp.2016.04.038.

Michalos, G., Karagiannis, P., Makris, S., Tokcalar, O. and Chryssolouris, G., 2016. Augmented reality (ar) applications for supporting human-robot interactive cooperation. *Procedia CIRP*, 41, pp.370 – 375. Research and Innovation in Manufacturing: Key Enabling Technologies for the Factories of the Future - Proceedings of the 48th CIRP Conference on Manufacturing Systems. Available from: http://doi.org/https://doi.org/10.1016/j.procir.2015.12.005.

Mukhopadhyay, P. and Chaudhuri, B.B., 2015. A survey of hough transform. *Pattern Recogn.*, 48(3), pp.993–1010. Available from: http://doi.org/10.1016/j.patcog.2014.08.027.

Quitter, T., Mostafa, A., Norman, D., Miede, A., Sharlin, E. and Finn, P., 2017. Humanoid robot instructors for industrial assembly tasks. *Proceedings of the 5th international conference on human agent interaction*. New York, NY, USA: ACM, HAI '17, pp.295–304. Available from: http://doi.org/10.1145/3125739.3125760.

Redondo, E., Fonseca, D., Snchez, A. and Navarro, I., 2013. New strategies using handheld augmented reality and mobile learning-teaching methodologies, in architecture and building engineering degrees. *Procedia Computer Science*, 25, pp.52 – 61. 2013 International Conference on Virtual and Augmented Reality in Education. Available from: http://doi.org/https://doi.org/10.1016/j.procs.2013.11.007.

Renner, P., Lier, F., Friese, F., Pfeiffer, T. and Wachsmuth, S., 2018. Facilitating hri by mixed reality techniques. *Companion of the 2018 acm/ieee international conference on human-robot interaction*. New York, NY, USA: ACM, HRI '18, pp.215–216. Available from: http://doi.org/10.1145/3173386.3177032.

Riek, L.D., 2017. Healthcare robotics. *Commun. ACM*, 60(11), pp.68–78. Available from: `http://doi.org/10.1145/3127874`.

Rohlfshagen, P. and Bryson, J.J., 2010. Flexible latching: A biologically-inspired mechanism for improving the management of homeostatic goals. *Cognitive Computation*, 2(3), pp.230–241. Available from: `http://doi.org/"10.1007/s12559-010-9057-0"`.

Rudall, B.H., 2004. World robotics 2003, united nations publication-coauthored by international federation of robotics, new york, geneva, 2003, xiv + 370 pp., isbn: 92-1-101059-4, issn: 11020-1076 (price on application). *Robotica*, 22(5), pp.589–590. Available from: `http://doi.org/10.1017/S0263574704250949`.

Salem, M., Lakatos, G., Amirabdollahian, F. and Dautenhahn, K., 2015. Would you trust a (faulty) robot?: Effects of error, task type and personality on human-robot cooperation and trust. *Proceedings of the tenth annual acm/ieee international conference on human-robot interaction*. New York, NY, USA: ACM, HRI '15, pp.141–148. Available from: `http://doi.org/10.1145/2696454.2696497`.

Sanders, T.L., Wixon, T., Schafer, K.E., Chen, J.Y.C. and Hancock, P.A., 2014. The influence of modality and transparency on trust in human-robot interaction. *2014 ieee international inter-disciplinary conference on cognitive methods in situation awareness and decision support (cogsima)*. pp.156–159. Available from: `http://doi.org/10.1109/CogSIMA.2014.6816556`.

Steven, A.W., Feiner, S., Macintyre, B., Massie, W. and Krueger, T., 1996. Augmented reality in architectural construction, inspection, and renovation.

Subin, E.K., Hameed, A. and Sudheer, A.P., 2017. Android based augmented reality as a social interface for low cost social robots. *Proceedings of the advances in robotics*. New York, NY, USA: ACM, AIR '17, pp.43:1–43:4. Available from: `http://doi.org/10.1145/3132446.3134907`.

Szalavári, Z., Schmalstieg, D., Fuhrmann, A. and Gervautz, M., 1998. "studierstube": An environment for collaboration in augmented reality. *Virtual Reality*, 3(1), pp.37–48. Available from: `http://doi.org/10.1007/BF01409796`.

Theodorou, A., 2017. Abod3: A graphical visualization and real-time debugging tool for bod agents. *CEUR Workshop Proceedings*, 1855, pp.60–61. Proceedings of the EUCognition Meeting (European Society for Cognitive Systems) "Cognitive Robot Architectures", Vienna, Austria, December 8-9, 2016.

Theodorou, A., Wortham, R.H. and Bryson, J.J., 2017. Designing and implementing transparency for real time inspection of autonomous robots. *Connection Science*, 29(3), pp.230–241. `https://doi.org/10.1080/09540091.2017.1310182`, Available from: `http://doi.org/10.1080/09540091.2017.1310182`.

Walker, M., Hedayati, H., Lee, J. and Szafir, D., 2018. Communicating robot motion intent with augmented reality. *Proceedings of the 2018 acm/ieee international conference on human-robot interaction*. New York, NY, USA: ACM, HRI '18, pp.316–324. Available from: `http://doi.org/10.1145/3171221.3171253`.

Wilkins, D.E., MYERS, K.L., LOWRANCE, J.D. and WESLEY, L.P., 1995. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental & Theoretical Artificial Intelligence*, 7(1), pp.121–152. `https://doi.org/10.1080/09528139508953802`, Available from: `http://doi.org/10.1080/09528139508953802`.

Wortham, R. and Rogers, V., 2017. The muttering robot: Improving robot transparency though vocalisation of reactive plan execution. 26th IEEE International Symposium on Robot and Human Interactive Communication (Ro-Man) Workshop on Agent Transparency for Human-Autonomy Teaming Effectiveness, RO-MAN2017 Workshop on Agent Transparency for Human-Autonomy Teaming Effectiveness ; Conference date: 01-09-2017 Through 01-09-2017. Available from: `https://sites.google.com/view/roman17transparencywkp`.

Wortham, R., Theodorou, A. and Bryson, J., 2017a. Improving robot transparency: real-time visualisation of robot ai substantially improves understanding in naive observers. IEEE RO-MAN 2017 : 26th IEEE International Symposium on Robot and Human Interactive Communication ; Conference date: 28-08-2017 Through 01-09-2017. Available from: `http://www.ro-man2017.org/site/`.

Wortham, R.H., Gaudl, S. and Bryson, J.J., 2016. Instinct: A biologically inspired reactive planner for embedded environments. *Proceedings of the international conference on automated planning and scheduling (icaps)*. Available from: `http://opus.bath.ac.uk/50295/`.

Wortham, R.H. and Theodorou, A., 2017. Robot transparency, trust and utility. *Connection Science*, 29(3), pp.242–248. `https://doi.org/10.1080/09540091.2017.1313816`, Available from: `http://doi.org/10.1080/09540091.2017.1313816`.

Wortham, R.H., Theodorou, A. and Bryson, J.J., 2016. What does the robot think? transparency as a fundamental design requirement for intelligent systems. *Proceedings of the ijcai workshop on ethics for artificial intelligence*. Available from: `http://opus.bath.ac.uk/50294/`.

Wortham, R.H., Theodorou, A. and Bryson, J.J., 2017b. Robot transparency:improving understanding of intelligent behaviour for designers and users. *Taros 2017*. Available from: `http://opus.bath.ac.uk/55794/`.

Zaher, M., Greenwood, D. and Marzouk, M., 2018. Mobile augmented reality applications for construction projects. *Construction Innovation*, 18(2), pp.152–166. `https://doi.org/10.1108/CI-02-2017-0013`, Available from: `http://doi.org/10.1108/CI-02-2017-0013`.