

**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**

# **Immersive Telerobotic Modular Framework using stereoscopic HMD's**

**Filipe André Cachada Rodrigues**



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rui Pedro Amaral Rodrigues (PhD)

Co-Supervisor: Luis Paulo Reis (PhD)

January, 2015



# **Immersive Telerobotic Modular Framework using stereoscopic HMD's**

**Filipe André Cachada Rodrigues**

Mestrado Integrado em Engenharia Informática e Computação

January 15, 2015



# Abstract

Telepresence is the term used to describe the set of technologies that enable people to feel or appear as if they were present in a location which they are not physically in. Immersive telepresence is the next step and the objective is to make the operator feel like he is immersed in a remote location, using as many senses as possible and new technologies such as stereoscopic vision, panoramic vision, 3D audio and Head-Mounted Displays (HMDs).

Telerobotics is a subfield of telepresence and merge it with robotics, providing the operator with the ability to control a robot remotely. In the state-of-the-art solutions there is a lack of studies investing on immersive solutions, such as stereoscopic vision. Immersive telerobotics can also include more intuitive control capabilities such as haptic-based controls or movement and gestures that would feel more natural and translated more naturally into the system.

In this thesis we propose an alternative approach to common teleoperation methods such as some of the methods found, for instance, in search and rescue (SAR) robots.

We claim that immersive solutions increase depth perception and situational awareness, the operator in a telerobotics system will be less error-prone and have better performance both in terms of operation time and on successful identification, of particular objects, in remote environments.

Our main focus was to test the impact that immersive characteristics like stereoscopic vision and HMDs can bring to the control of robots from a distance (telepresence robots) and telerobotics systems. Besides that, and since this is a new and growing field, we were also aiming at a low-cost modular framework capable of being extended with hardware-based Android applications in slave side (robot side), providing the ability to use different robots in order to test different cases and aid researchers with an extensible platform.

The expansion of technologies in different areas, such as mobile (e.g. smartphones, tablets, arduino), low-cost immersive solutions like Oculus Rift DK2 and web-based technologies like WebRTC and WebGL turns possible the development of a real-time software solution.

A practical experiment has been performed where the majority of participants had improvements (80%) with stereo vision in a visual search task. A significant increase regarding distances

between objects were also observed. The results and feedback regarding the head tracking were also very positive.

**Keywords:** Stereoscopies, Telerobotics, WebRTC, WebGL, Immersive Telepresence, Robotics, Android



# Resumo

Telepresença é o termo utilizado para descrever o conjunto de tecnologias que proporcionam aos utilizadores a sensação de que se encontram num local onde não estão fisicamente. Telepresença imersiva é o próximo passo e o objetivo passa por proporcionar a sensação de que o utilizador se encontra completamente imerso num ambiente remoto, estimulando para isso o maior número possível de sentidos e utilizando novas tecnologias tais como: visão estereoscópica, visão panorâmica, áudio 3D e *Head-Mounted Displays* (HMDs).

Tele-robótica é um sub-campo da telepresença ligando esta à robótica, e que essencialmente consiste em proporcionar ao utilizador a possibilidade de controlar um robô de forma remota.

Nas soluções do estado da arte da tele-robótica existe uma falta de estudos a apostar em soluções de imersividade, tais como visão estereoscópica. A tele-robótica imersiva pode também incluir controlos mais intuitivos, tais como controladores de toque ou baseados em movimentos e gestos. Estes controlos são mais naturais e podem ser traduzidos de forma mais natural no sistema.

Neste documento propomos uma abordagem alternativa a métodos mais comuns encontrados na teleoperação de robôs, como, por exemplo, os que se encontram em robôs de busca e salvamento (SAR).

Pretendemos provar que soluções imersivas melhoram a perceção de profundidade e do ambiente em geral, e que o operador num sistema de tele-robótica imersiva estará menos propenso a erros e terá um melhor desempenho tanto em termos de eficácia como numa bem sucedida identificação de objectos de interesse num ambiente remoto.

O nosso principal foco foi testar o impacto que características imersivas, tais como visão estereoscópica e HMD's podem trazer para os robôs de telepresença e sistemas de tele-robótica. Além disso, e tendo em conta que este é um novo e crescente campo, também desenvolvemos uma *framework* modular e de baixo custo que possui a capacidade de ser estendida com diferentes robôs, com o fim de proporcionar aos investigadores uma plataforma com que podem testar diferentes casos de estudo.

A expansão de tecnologias em diferentes áreas, tais como a área móvel (e.g. *smartphones*, *tablets*, *arduino*), tecnologias de imersão de baixo custo como o Oculus Rift DK2 e tecnologias baseadas



na web, tais como WebRTC e WebGL, tornam possível o desenvolvimento de uma solução em tempo-real baseada em *software*.

Uma experiência foi realizada onde a maioria dos participantes obteve melhorias significativas (80%) com a visão estereoscópica em tarefas de busca visual. Também foi observado um aumento significativo em relação à identificação relativa de distâncias entre objectos. Os resultados e comentários em relação ao *head tracking* foram também bastante positivos.

**Palavras-Chave:** Estereoscopia, Tele-robótica, WebRTC, WebGL, Telepresença Imersiva, Robótica, Android



# Agradecimentos

Começo por agradecer ao Professor Rui Rodrigues e ao Professor Luís Paulo Reis, pela disponibilidade que sempre apresentaram e por toda a ajuda prestada desde o primeiro até ao último dia. O acompanhamento foi incansável, mesmo quando a agenda apertava.

A toda a minha família, que sempre esteve presente, nos bons, maus e feios momentos desta caminhada, prestando um apoio incondicional. E um agradecimento especial à minha mãe, sem ela não estaria “aqui”.

À Jessica, obrigado pelo carinho, apoio e paciência.

Aos amigos e colegas que me acompanharam ao longo deste percurso contribuindo também para o meu amadurecimento e enriquecimento pessoal.

Filipe Rodrigues



*The individual has always had to struggle to keep from being overwhelmed by the tribe. If you try it, you will be lonely often, and sometimes frightened. But no price is too high to pay for the privilege of owning yourself.*

Friedrich Nietzsche



# Contents

<b>1 Introduction .....</b>	<b>1</b>
1.1 Context .....	2
1.2 Project.....	3
1.3 Motivation and Objectives .....	4
1.4 Document Structure .....	5
<b>2 State of the Art .....</b>	<b>7</b>
2.1 Telepresence.....	7
2.2 Telerobotics.....	8
2.3 Real-Time Stereo Vision .....	10
2.4 First Person Robot Teleoperation .....	14
2.4.1 Human-Computer Interaction in Robotics .....	15
2.4.2 Movements and Gestures Controls .....	14
2.4.3 Robot Control: Classic Vs Fuzzy Approach .....	17
2.5 Mobile Platforms .....	17
2.5.1 Smartphones and Tablets.....	17
2.5.2 Android SDK .....	17
2.5.2 Android Compatible Integrated Circuit Boards .....	18
2.6 Android-Based Robotics.....	19
2.6.1 Lego Mindstorms EV3 .....	19
2.6.2 Comercial Telepresence Robots .....	20
2.7 Stream Through Android Devices .....	21
2.7.1 WebRTC.....	22
2.7.2 Libjingle: WebRTC in Native Android Application .....	24
2.8 Single-Page Application .....	24
2.8.1 WebGL.....	26
2.9 Conclusions .....	26

2.9.1 Hardware to be Used.....	27
2.9.2 Technologies and Software Modules to be Used .....	28
<b>3 TrekProbe - Immersive Telerobotic Modular Framework.....</b>	<b>30</b>
3.1 Requirements Specification .....	30
3.2 System Architecture .....	31
3.3 Communication Protocol from Master to Slave .....	32
3.3.1 Signaling .....	32
3.3.2 Robot Commands API.....	33
3.4 Master: Web Application.....	35
3.4.1 Requirements Specification .....	35
3.4.1.1 Technologies and Modules.....	37
3.4.1.2 Test WebRTC Capabilities with a videocall prototype .....	38
3.4.2 System Architecture and Modules .....	39
3.4.3 Stereo Vision System through WebGL .....	42
3.4.3.1 Stereo Image Rectification .....	42
3.4.3.2 Synchronizing Streams .....	44
3.4.4 Oculus Rift DK2 Integration .....	45
3.4.4.1 Head Tracking Algorithm .....	45
3.4.4.2 Positional Tracking Algorithm .....	46
3.5 Slave: Mobile Application .....	47
3.5.1 Requirements Specification .....	47
3.5.2 System Architecture .....	49
3.2.2.1 Application Modules .....	50
3.6 Conclusions.....	52
<b>4 Prototype Development.....</b>	<b>54</b>
4.1 Unity Simulator .....	54
4.2 Mindstorms EV3.....	55
4.2.1 Programming the Brick .....	55
<b>5 Experimental Procedure and Results .....</b>	<b>61</b>
5.1 Experimental Procedure .....	61
5.1.1 Prototype Equipment.....	62
5.1.1.1 General Controls .....	63
5.1.2 Scenario and Instructions.....	63
5.1.2.1 Types of Controls.....	64
5.1.2.1 Reconnaissance Course .....	64
5.1.3 Participants.....	67
5.1.4 Measured Results .....	68
5.1.4.1 Test A .....	68



5.1.4.2 Test B.....	68
5.1.5 Survey Presented .....	69
5.1.5.1 Survey Results .....	69
5.1.5.2 Comments and Testimonies .....	73
5.1.6 Experiment Conclusions .....	73
<b>6 Conclusion.....</b>	<b>76</b>
<b>References .....</b>	<b>78</b>

# List of Figures

Figure 2.1: Telepresence 3200 from Cisco System.	8
Figure 2.2: ‘iRobot 110 FirstLook’ – state of the art in recognition operations	9
Figure 2.3: Operation Environment (master) ‘iRobot 510 Packbot’	9
Figure 2.4: Modelling of stereo rig and pinhole model of cameras	11
Figure 2.5: Extrinsic parameters and relation of a point in world ( $P_w$ ) and camera ( $P_c$ ) coordinates	12
Figure 2.6: Intrinsic parameters. From camera to image plane and from image plane to pixel coordinates	12
Figure 2.7: Image rectification	13
Figure 2.8: Checkerboard pattern used in image rectification	13
Figure 2.9: Oculus Rift - Development Kit 2	14
Figure 2.10: Leap Motion	16
Figure 2.11: Oculus Rift DK2 Positional Tracking Camera	16
Figure 2.12: IOIO (left) and Arduino ADK Rev3 (right)	18
Figure 2.13: Romo (left) and Botiful (right)	21
Figure 2.14: WebRTC Application	23
Figure 3.1: Global System Architecture	31
Figure 3.2: Example of bytecode for robot control	32
Figure 3.3: JSON encoded Head Tracking command	35
Figure 3.4: Diagram Depicting Main Web Application Options	36
Figure 3.5: Web Application UI - Options	37
Figure 3.6: Web Application – Bidirectional Call	38
Figure 3.7: Architecture Overview	39
Figure 3.8: Main Modules of Web Application	41
Figure 3.9: Virtual Projection Cameras	43
Figure 3.10: Pseudo-code for Synchronization	44
Figure 3.11: Head Tracking Orientation	45
Figure 3.12: Positional Tracking pseudo-code	46
Figure 3.13: TrekProbe Slave – Sliding Menu	47
Figure 3.14: TrekProbe Slave – Streaming Options	48
Figure 3.15: TrekProbe Slave – Bluetooth Options	49
Figure 3.16: Architecture Overview of Slave Environment	49
Figure 3.17: Main Modules of Mobile Application	51
Figure 4.1: Unity Simulator	55

Figure 4.2: Direct Command Example	56
Figure 4.3: Modules Developed for Brick Control	57
Figure 4.4: Bytecode to move robot forward by rotating 90 degrees	58
Figure 4.5: Robot Development	58
Figure 5.1: EV3 Robot	62
Figure 5.2: Experimental Procedure – Operator Equipment	62
Figure 5.3: Reconnaissance course	64
Figure 5.4: Test A – Both layouts	65
Figure 5.5: Test B – Camouflaged Cube	66
Figure 5.6: Age distribution	67
Figure 5.7: Area of Training distribution	67



# List of Tables

Table 2.1: Robotic Platforms for research and education.....	19
Table 2.2: EV3 Communication Libraries Comparison .....	20
Table 2.3: Android media stream solutions.....	22
Table 2.4: Node.JS and Vert.x comparison.....	25
Table 3.1: Robot Control API commands .....	34
Table 5.1: Assignment of participants to layout setups .....	65
Table 5.2: Test A results.....	68
Table 5.3: Test B results .....	68
Table 5.4: Survey results .....	70



# Abbreviations and Symbols

API	Application Programming Interface
DOS	Disk Operating System
USA	United States of America
HMD	Head Mounted Display
OTG	On-The-Go
PAN	Personal Area Network
PC	Personal Computer
RC	Remote Controlled
SAR	Search and Rescue
SDK	Software Development Kit
UDK	Unreal Development Kit
UE4	Unreal Engine 4
USB	Universal Serial Bus
FPGA	Field Programmable Gate Array
FPS	Frames Per Second

## Chapter 1

# Introduction

The main objective in the research field of immersive telepresence is to provide the user with the ability to interact with the remote environment through movements and gestures, making the most to increase the sense of immersion on the remote environment or simulated system. When the fidelity of perception through these systems is equivalent to an *in-situ* observation from the user, the ultimate desired experience is the same as being in that location.

Telerobotics is a sub-field of telepresence that brings robotics to the telepresence, allowing the user to control robots at distance. There are a wide variety of situations where the human being would significantly gain from being remotely immersed in a location through a telerobotic system, for instance to provide safer work environments, to aid in search and rescue (SAR), patrolling and surveillance missions, space exploration and to visit points of interest remotely in a new concept of tourism.

In order to create a more immersive solution we need to stimulate the largest number of possible senses, being the most crucial the sense of sight. Humans use different cues to achieve 3D scene perception, such as accommodation, convergence, perspective, binocular disparity, motion parallax, and a lot more. Various stereoscopic vision techniques are used in order to enrich the immersion in many digital imaging areas, such as cinema, gaming and virtual reality. To provide stereo vision, and the perspective of depth (stereopsis) [StDisp09], two cameras, displaced horizontally one from another, are used to obtain two different views on a scene, in a similar way to the human binocular vision.



## Introduction

With this thesis we aim to test the impact that immersive characteristics, like stereoscopic vision, head tracking, positional tracking and Head Mounted Displays (HMD's), can bring to telepresence and telerobotics systems. Besides that, and since this is a new and growing field, we also aim at the creation of a modular platform. With this solution we want to provide the possibility of using the same platform, in any type of case study, by just extending the platform with different robots.

The project being developed is an immersive telerobotics platform that consists of a modular framework extensible with hardware-based Android applications. The platform will be connected to an HMD with a separate video source, in front of each eye, in order to achieve a stereoscopic effect. In terms of robot control, the user will be able to move the robot with head tracking and positional tracking, while immersed in the remote environment.

In the following sections, the context, definition of the problem, motivation and objectives will be presented.

### 1.1 Context

This document was produced as part of a MSc Thesis in Informatics and Computing Engineering, from the Faculty of Engineering of University of Porto. And aims to describe the work done throughout the dissertation as well as present tests and results obtained.

The term “Telepresence” dates from 1980, by the cognitive scientist Marvin Minsky [[TelePrs](#)]. However, as in many other areas of science and technology, the first references to the concept of telepresence appeared in science fiction. According to Marvin Minsky himself his first sight of a remotely operated mechanism originated in the prophetic novel by Robert A. Heinlein, “Waldo”, 1948. In his science fiction story, Heinlein mentioned a primitive telepresence solution based on a master-slave manipulation system.

Small ground robots, such as the PackBot and TALON, have been widely used by warfighters to hunt for terrorists and perform all types of reconnaissance duties [[RT12](#)]. Search and rescue robots have been used since the September 11 attacks at World Trade Center, in the aftermath of disasters (natural and otherwise) and building collapses. While they have definite benefits, such as being able to get into spaces that human or canine rescuers might find too difficult or dangerous to reach, they are currently expensive and complicated to use. Dr. Julie Adams, a professor at Vanderbilt who studies human-robot interactions, says that using robots in search and rescue usually requires the presence of maybe four human experts to one robot, making the cost of using robots relatively high compared to the benefit they provide.

## Introduction

Immersive solutions have been kept away from these tele-operated robots because of technology limitations and cost. However in recent years there has been a great evolution of mobile platforms (e.g., smartphones, tablets, Arduino, etc ...) and mobile robotics. We are also currently witnessing a growing explosion that makes immersive technologies, such as HMDs (Head Mounted Displays), touchless and gesture interaction technologies, affordable. All these facts put together make it possible to develop immersive telepresence solutions in robotics, emerging as a new area called Immersive Telerobotics. These cost-effective hardware solutions associated with the advances in video and audio streaming, provided by WebRTC technologies, achieved mostly by Google, and 3D/2D rendering and image processing provided by Web Graphics Library (WebGL) make the development of a real-time software solution possible. Currently there is a lack of real-time software methods in stereoscopy, but the arrival of WebRTC technologies can be a game changer.

As a conclusion, not only we believe that with a more immersive interface the need for special formed operators will fade, but the integration of immersive modules in these robots will also benefit the operator, with a more intuitive perception of the remote environment, making the operator less error prone induced by a wrong perception and interaction with the teleoperation of the robot.

The development of this project aims to test and evaluate the impact that stereoscopy with head tracking can have in tele-operated robots and, since this is a new field that certainly will grow in the time to come, we are aiming at a modular platform to help researchers in the areas of immersive telepresence and telerobotics to test their case studies, saving time and money that would go to creating a specific platform, and benefiting from the time saved to focus on the core issues of their study.

## 1.2 Project

The project described here concerns a low / medium cost platform for immersive telerobotics which first aims to be used in a set of ambient controlled tests, in order to evaluate the impact that stereoscopy would have in tele-operated robots.

A second goal is to make the platform available to researchers in immersive telepresence and telerobotics field, developing a modular and extensible framework with hardware-based software in the slave side (i.e. remote robot) that would meet the inherent requirements of the case study that the researcher aims for. From these requirements comes the motivation to develop this project.

## Introduction

Our platform will consist of three major modules:

- A module that supports a head mounted display and head tracking in the operator environment
- Stream of stereoscopic vision through Android with software synchronization
- And a module that enables the operator to control the robot with positional tracking

Our aim with these modules is to provide the operator with a robust solution in terms of immersiveness in the remote environment.

The framework can be extended with any robot. If the researcher chooses an Android compatible robot there is no need to be an expert in electronics. However, if the researcher decides to choose a more low-cost approach, the framework can also be extended using an integrated circuit board (e.g. Android IOIO, Arduino MEGA ADK) which can easily be attached to any robot or even to a remotely controlled vehicle.

### 1.3 Motivation and Objectives

State of the art studies in telepresence/telerobotics lack a focus on evaluating the impact that immersive elements could have in telerobotics.

As proved in [RT12], where a limited system with stereoscopic vision was used, 13 of 18 users said that they would prefer an immersive telepresence condition. Reasons provided included its overall ease of use and, in particular, ease of visual search and target localization. Overall comments included descriptions of telepresence as “intuitive”, “easy” and “second nature”.

In a general way state of the art telerobotics have not enjoyed of the recent developments in human-computer interaction and the evolution of immersive technologies.

A recent study [BSK12b] points out that controllers with touch interface are a poor alternative to immersiveness, at least when the operator lacks experience.

In the telerobotics field the research with other methods of interaction, like controls for positional tracking or movements and gestures, is near non-existence. Following this line of thinking, and assuming the lack of conclusive tests proving the effectiveness of control technologies by positional tracking and/or movements and gestures, we also got motivation to develop, with a secondary focus, a positional tracker module in the project.

As will be discussed in Chapter 2, the benefits of immersiveness in telepresence and telerobotic systems begin to be undeniable.

## Introduction

Researchers increasingly will need ways to test if immersion really is a good bet in the various contexts that Telerobotics covers. Currently researchers end up wasting too much time and money building specific solutions each time they plan to test a specific case. And often the solutions fall short of expectations at the technical level.

In short, this project aims to address some shortcomings identified in the Telerobotics field. Namely:

- Lack of immersive elements
- Absence of a modular and adaptable platform that can be used in various case studies with different robots
- Poor use of recent evolution in mobile, HMDs, human-computer interaction and real-time streaming technologies

And the main objectives identified to develop in the course of this thesis:

- **Scientific Component:** Test and evaluate the impact that stereoscopic vision can introduce in tele-operated robots
- **Practical Component:** Provide a framework and tool to develop rapid prototypes based on the requirements of the case study
  - A module that supports an head mounted display and head tracking in the operator environment
  - Stream of stereoscopic vision through Android with software synchronization
  - And a module that enables the operator to control the robot with positional tracking

## 1.4 Document Structure

Apart from the introduction, this document contains 4 more chapters.

In Chapter 2, we describe the state of the art and related work is presented. In Chapter 3 the entire development of the platform is detailed. Chapter 4 presents the prototype developed as a case study demonstration of the platform's potential, and also is a demonstration of how all the components of this project interact. In Chapter 5 we show the experimental procedure and results obtained. Finally, in Chapter 6 the conclusions and some possible future work are presented.



## Chapter 2

# State of the Art

State of the art in the various areas touched by this Dissertation are presented in this Section. Related work is also presented in order to show what is being made and the major problems faced. We start with the state of the art in telepresence and telerobotics, then a survey and review of the evolution of technologies that allow this dissertation are presented. Finally we review the problems that researchers in this field are currently facing.

### 2.1 Telepresence

State of the art in telepresence essentially consists of static solutions that pass through webcams and monitors, for image transmission, and keyboards / joysticks for data entry and remote control. There is no use of immersive solutions, this meaning that current commercial solutions try to bring the user to the remote environment without worrying about getting the remote environment to the user. In Figure 2.1 we can see one of the top comercial solutions of telepresence - TelePresence 3200 Cisco System<sup>1</sup>.

---

<sup>1</sup> <http://www.cisco.com/c/en/us/products/collaboration-endpoints/telepresence-system-3200-series/index.html>



Figure 2.1: Telepresence 3200 from Cisco Systems.

The main objective of Telepresence 3200 is to provide a video conferencing solution and undoubtedly this is achieved. However we believe that telepresence can take a big leap with the recent development of mobile platforms and technologies, as stated in [\[RMT13\]](#).

We also claim that if we add immersive components to the Mobile Robotic Telepresence (MRP), telepresence can “explode” and grow from the niche in which it commercially lies now - that is the video-conference niche. Some examples of immersive components are: stereo or 3D sound, stereoscopic vision through HMDs, head tracking and touchless controls.

The bandwidth required in this solution can be as high as 20.4 Mbps, with a 1080p resolution running at 30 frames per second (FPS) and a variable bit rate (VBR). The average latency lies around 150ms, with a 10ms jitter and packet loss below 0.05%.

## 2.2 Telerobotics

Telerobotics is quite different from telepresence in concept. However, the technological limitations found in both areas are quite similar, as we can check in academic projects such as [\[PlatRb13\]](#) and [\[RMT13\]](#), in more simple and commercial ones [\[AtkRb13\]](#) or even in the top commercial products like iRobot robots [\[iRobot\]](#).

In Figure 2.2 we can see the state of the art solution in search and rescue environments, recognition and patrolling telerobotics, from iRobot.

## State of the Art



Figure 2.2: iRobot 110 FirstLook – state of the art in recognition operations.

Used in military and search and rescue (SAR) environments, the iRobot 110 is a lightweight robot that provides immediate perception of remote environments, distance observation and allows to search environments and hazardous materials keeping the operator's safe.

The control console (master) is also quite complex, requiring an operator to have prior training to operate these devices. In Figure 2.3 you can see a control panel of such systems, in this case from Packbot robot.



Figure 2.3: Operation console (master) of iRobot 510 Packbot.

As seen in Figure 2.3, the operator interface has a small screen and low resolution. In this system the reaction time of operator is increased and the overall interaction with the remote environment is subject to extreme noise.

In a recent study [RT12] carried out by the research laboratory of the US Army, in which a telerobotics immersive solution was tested, they came to the conclusion that in reconnaissance



operations an operator can develop its tasks more quickly using a system with immersive technologies. The reaction time of operator can decrease as much as about 47% when compared with a conventional solution.

Head tracking and stereoscopic vision through the HMD were the immersive capabilities implemented in this study. Regarding the technical specifications it was a somewhat limited system, with a optical resolution of 320 by 240 pixels, estimated latency of 100-150ms and a HMD with an estimated weight of 950g.

In conclusion, we believe that in SAR missions the operator environmental perception can increase with an immersive telerobotics system, along with a more intuitive and user-friendly control module. The operator response time will be reduced, increasing the efficiency and reducing the time required to take decisions, while reducing the noise and barriers inherent of such environments and conditions.

Besides, state of the art still has a large gap in what concerns using stereoscopy, solutions or even research with movements and gestures controllers are also near zero.

### **2.3 Real-Time Stereo Vision**

Various stereoscopic vision techniques are used in order to enrich the immersion in many digital imaging areas, such as cinema, gaming and virtual reality. To provide stereo vision, and the perspective of depth (stereopsis) [[StDisp09](#)], two cameras, displaced horizontally one from another, are used to obtain two different views on a scene, in a similar way to the human binocular vision. Humans use different cues to achieve 3D scene perception, such as accommodation, convergence, perspective, binocular disparity, motion parallax, and a lot more.

In a best case scenario the two image sensors, used in a stereo vision system, need to be perfectly aligned along a horizontal or vertical line that passes through the key points of both images. Cameras are liable to lens distortion, which will introduce convexity or concavity to the image projection. Different cameras will introduce different distortions on the images acquired. The process called stereo pair rectification [[EpiLR96](#)] [[ImgRect](#)] is adopted to remap distorted projection into an undistorted plane. In Figure 2.4.a we can see the arrangement of image planes

and Figure 2.4.b illustrates the pinhole model [CamCal04] of two cameras to show how the projection of a real world object is formed in left and right images.

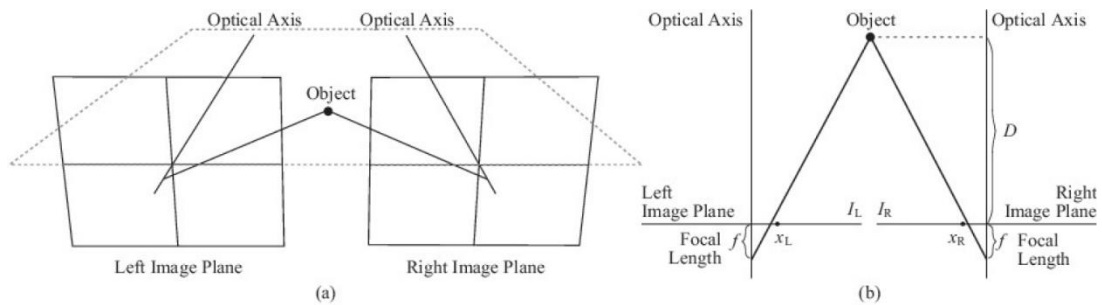


Figure 2.4: Modelling of stereo rig and pinhole model of cameras [CamCal04]

The algorithm already established in stereo vision systems have the following main modules; Calibration, Rectification, Stereo Correspondence and Triangulation.

The last two modules are used to acquire a disparity map and depth perception, but since we are not dealing with autonomous robots they are not relevant in the context of this work, however the remaining ones are essential.

Calibration is a procedure usually done offline and aimed at finding the intrinsic and extrinsic parameters of cameras.

Extrinsic parameters are the ones that define the location and orientation of the camera reference frame, with respect to a known world reference frame. The intrinsic parameters are the ones necessary to link the pixel coordinates of an image point, with the corresponding coordinates in the camera reference frame.

In Figure 2.5 we can see how to use the extrinsic camera parameters to find the relation between the coordinates of point  $P$  in world ( $P_w$ ) and camera ( $P_c$ ) coordinates.

State of the Art

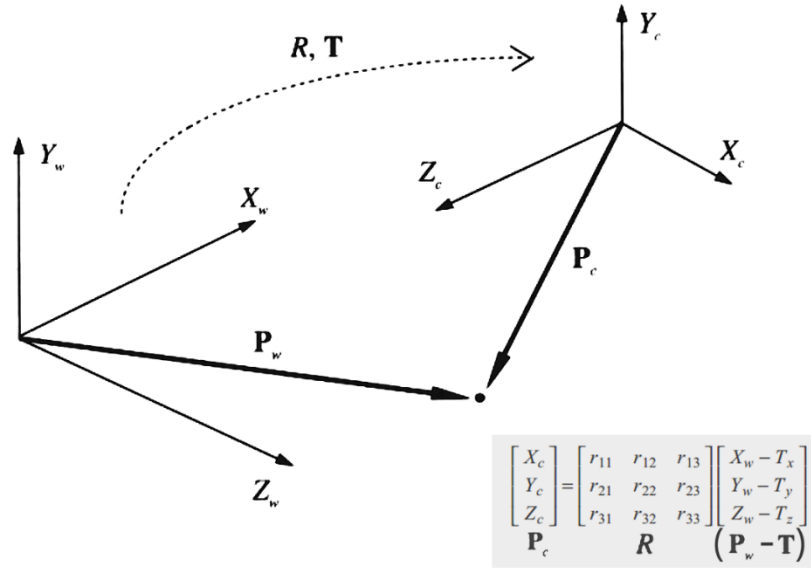
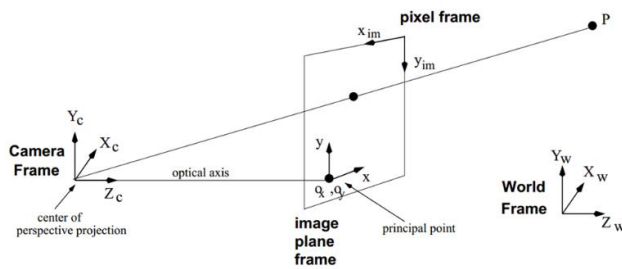


Figure 2.5: Extrinsic parameters and relation of a point in world ( $P_w$ ) and camera ( $P_c$ ) coordinates.

Intrinsic parameters are the ones that characterize the optical and geometric characteristics of the camera:

- The perspective projection (focal length  $f$ )
- Transformation between image plane coordinates and pixel coordinates
- Geometric distortion introduced by the optics

In Figure 2.6.a we have the perspective projection equations to find the image plane coordinates from camera coordinates. In Figure 2.6.b we can see how to get the pixel coordinates from image plane coordinates.



$$x = -(x_{im} - o_x)s_x \quad \text{or} \quad x_{im} = -x/s_x + o_x$$

$$y = -(y_{im} - o_y)s_y \quad \text{or} \quad y_{im} = -y/s_y + o_y$$

(a)

$$x = f \frac{X_c}{Z_c} = f \frac{R_1^T (P_w - T)}{R_3^T (P_w - T)},$$

$$y = f \frac{Y_c}{Z_c} = f \frac{R_2^T (P_w - T)}{R_3^T (P_w - T)}$$

(b)

Figure 2.6: Intrinsic parameters. From camera to image plane (a) and from image plane to pixel coordinates (b).

Image Rectification is a procedure that uses the information from calibration in order to remove lens distortion and align the epipolar lines. The image below, Figure 2.7, shows a basic setup with two cameras taking the image of same scene and will be used to explain the process of aligning the epipolar lines.

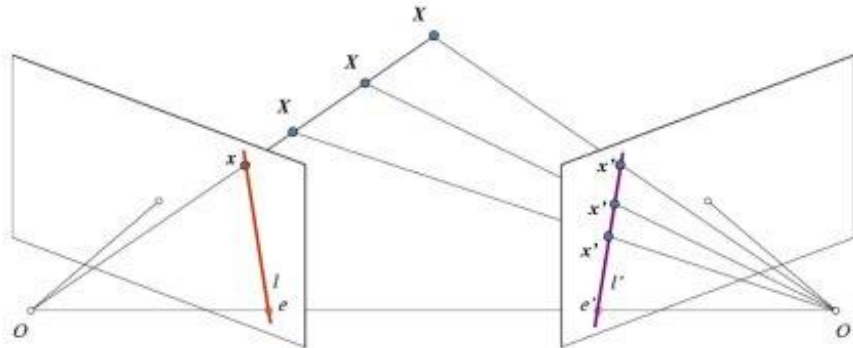


Figure 2.7: Image rectification. Taken from [epipGeom]

In Figure 2.7  $O$  and  $O'$  are the camera centers. From the setup given above, you can see that projection of right camera  $O'$  is seen on the left image at the point  $e$ . It is called the epipole. Epipole is the point of intersection of line through camera centers and the image planes. Similarly  $e'$  is the epipole of the right camera. All the epipolar lines pass through its epipole. Furthermore, the epipolar lines are parallel to the line  $O-O'$  between the centers of projection, and can in practice be aligned with the horizontal axes of the two images. In our project we align the epipolar lines in order to avoid vertical desynchronization, and a checkerboard pattern is used in this process, like the one in Figure 2.8.

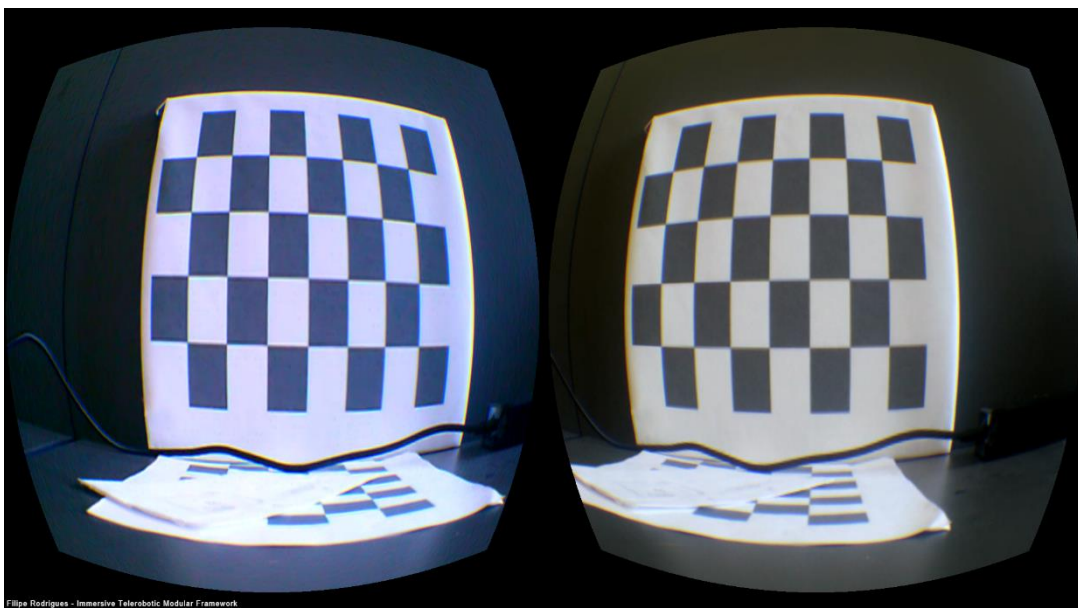


Figure 2.8: Checkerboard pattern used in image rectification

## State of the Art

Current state-of-the-art stereo vision systems are heavily dependent on hardware using integrated circuits, such as field programmable gate arrays (FPGA) to achieve real-time stereo vision. In order to achieve this real-time processing, many FPGA-based systems have been proposed to date, from relatively simple ones [LCStVi07] to more complex solutions [FPGA06] [FPGA10]. In these solutions the reported video stream lag lies around 100-200ms.

The major flaw with any hardware implementation, besides the cost, lies in the stereo synchronization and overall image control.

A software implementation would allow to have a better control over the streams while sacrificing the response time. However, as we will see in Section 2.7, new software solutions in the field of web-based communication are arriving.

## 2.4 First-Person Robot Teleoperation

With the advent of new solutions in the field of binocular HMDs, such as Oculus Rift or Sony HMZ-T2, these technologies had a significant price decrease, from prices above 2,500 \$ to much more affordable values, below 400\$. Besides the affordable prices, they continue to introduce technological innovations raising the state of the art.

In Figure 2.9 we can see the version 2 of the Oculus Rift Development Kit.



Figure 2.9: Oculus Rift - Development Kit 2

Oculus Rift Development Kit 2 is priced at \$ 350 and has top technical specifications, among them a 960 by 1080 resolution (per eye) with a refresh rate of 75Hz, positional tracking and an open source<sup>2</sup> SDK which includes source media for the reference game engines of today, such as UDK, UE4 and Unity4.

---

<sup>2</sup> Software for which the original source code is made freely available and may be redistributed and modified.

### 2.4.1 Human-Computer Interaction in Robotics

Controls such as those used in Packbot are complex, requiring a huge amount of training by the operator and a great ability to focus what may interfere with overall performance.

In order to bring these systems to a larger share of audience, and make them more intuitive, the scientific community has tried to find alternatives, now focusing on two methodologies in concrete, namely:

- Haptic-based controls
- Movements and gestures controls

Haptic-based controls, or touch controls, have been widely tested using mobile platforms like smartphones and tablets.

In [BSK12b] an example where an alternative interface using Android tablets was tested, and compared, with a more conventional approach including keyboards and gamepads. Despite the order lead the researchers in [BSK12b] to think that the touch interface would provide a greater sense of immersiveness, because we are using platforms to which users are already familiar, precisely the opposite has been showed by the end result of these tests. But the explanation is simple: with the gamepad operators can connect the movement of the robot to the physical touch of analog directional-pad (or joystick), besides that the lack of tactile feedback in touch interfaces has been identified as a major problem.

### 2.4.2 Movements and Gestures Controls

Currently we can find, in the field of touchless control, some affordable technologies that allows us to develop far more intuitive interfaces based on movements and gestures.

A new way of tracking movements and gestures is introduced with the Leap Motion <sup>3</sup>controller, with sub-millimeter accuracy. Contrasting with standard multi-touch solutions, this sensor is designed to be used in interactions with realistic 3D and stereo systems, especially regarding the selection of objects arranged stereoscopically. In terms of precision and accuracy Leap Motion has considerable gains against its main rival currently on the market (i.e. Microsoft Kinect<sup>4</sup>).

In Figure 2.5 we can see the Leap Motion.

---

<sup>3</sup> <https://www.leapmotion.com/>

<sup>4</sup> <http://www.microsoft.com/en-us/kinectforwindows/>



Figure 2.10: Leap Motion

As recent studies show in [ARLM13] and [ARLM14], when Leap Motion is tested in a real situation of real robot control in a dynamic environment, it can maintain an average accuracy of 0.7mm. This is a relatively high value when compared with the documentation, which claims an accuracy of 0.01mm, however, still above the accuracy of Kinect that is of 2mm at 1m distance from Kinect and 2.5cm at 3m distance [kinect12].

Oculus Rift DK2 also features positional tracking via an infra-red camera, seen in Figure 2.11. The camera works best at 1.5 meters from the Rift, and the tracking is lost if the “face” of the rift goes out of the frame of the camera, but depending on the type and accuracy of positional tracking required it can also be a solution.



Figure 2.11: Oculus Rift DK2 Positional Tracking Camera

Since our main focus is to study the impact of stereoscopic vision, the Oculus Rift DK2 camera can be a good solution if we decide to implement a more “simple” positional tracking module.

### **2.4.3 Robot Control: Classic Vs Fuzzy Approach**

We tried to understand the best way to control a robot and its actuators, and, as stated in [FuzzL10], the classical approach can have good results and give excellent control when dealing with a physical actuator where freedom of movement is not limited (i.e. a thumb). However, in head tracking the movement of the head is more physically limited, also in positional tracking the movement of the body is equally limited while sitting. And these limitations are non-linear due to the skeletal and muscular structure of the body.

When dealing with body restrictions a fuzzy system that takes into account the anatomy of the body and its physical limitations will feel more natural and intuitive.

Passing for a brief explanation of the concepts, in classical logic a simple proposition P is a statement contained within a universe of elements, X, that can be identified as being a collection of elements in X that are strictly true or strictly false. A fuzzy logic proposition, Q, is a statement involving some concept without clearly defined boundaries.

## **2.5 Mobile Platforms**

In recent years there has been a considerable development of mobile and mobile robotic platforms. Allied to this mobile development an increasing interest in developing immersive solutions in the telerobotics field gains momentum, and a new area starts to germinate - Immersive Telerobotics.

### **2.5.1 Smartphones and Tablets**

Smartphones and tablets are common devices with an amazing processing power. Not only this power has increased in last years but it continues to grow at an incredible rate. If we couple this processing capacity with modern operating systems and the useful extras (i.e. a huge number of available sensors, good battery life and compact size) these platforms are an excellent choice for mobile robotics, serving as onboard computers (robot brains).

### **2.5.2 Android SDK**

The Android software development kit (SDK) includes a comprehensive set of development tools [AndTools]. These tools include a debugger, libraries, a handset emulator based on QEMU<sup>5</sup>, documentation, sample code and tutorials. Currently supported development platforms include Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows XP or later.

---

<sup>5</sup> [http://wiki.qemu.org/Main\\_Page](http://wiki.qemu.org/Main_Page)



The official integrated development environment (IDE) is Android Studio<sup>6</sup>.

### 2.5.3 Android Compatible Integrated Circuit Boards

The growing interest in having smartphones to interact with peripheral devices such as motors, servos and sensors, led to the development of electronic boards that can easily be purchased at a low cost. These cards are a bridge of communication between Android and external devices. The top two currently available boards are IOIO, which costs around €25, and the Arduino ADK Rev3 with a price around €40. In Figure 2.6 we can see both solutions.

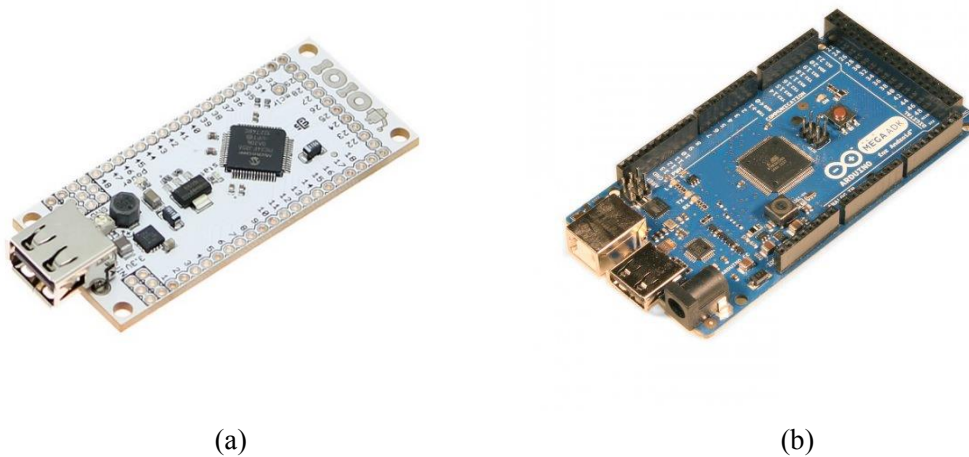


Figure 2.12: IOIO (a) and Arduino ADK Rev3 (b)

A significant number of projects carried out by enthusiasts, teachers and students, using these electronic boards, are now available online. We are talking about projects involving remote controlled vehicles (RC Vehicles) or robots controlled via Android IOIO or Arduino boards. It's easy to find some interesting projects, like, for example, the researchers controlling a RC sailing boats using an Android smartphone via IOIO board.

---

<sup>6</sup> <http://developer.android.com/tools/studio/index.html>

## 2.6 Android-Based Robotics

As stated in Section 2.5 a growing interest in mobile robotics has grown, more specifically on an Android based one.

Currently we have at our disposal a large number of robotic platforms, for education and research that enable the creation of robots without electronics knowledge.

In Table 2.1 we can see a list of some platforms, ordered by price.

Robotic Platform	Price
<b>iRobot Create</b>	95€
<b>Thymio II</b>	140€
<b>VEX Robotics (VEX IQ; VEX)</b>	185€; 300€
<b>Lego Mindstorms EV3</b>	250€
<b>Robotis (Bioloid; DARwin-OP)</b>	250€; 8,500€
<b>TETRIX</b>	280€
<b>Surveyor (SRV-1)</b>	370€
<b>K-Team Corporation (K-Junior; Kilobot; Khepera; Koala)</b>	600€; 850€; 2,400€; 6,200€
<b>Adept MobileRobots (AmigoBot; Pioneer DX; Pioneer AT)</b>	1250€; 3,000€; 4,800€
<b>Scout (Dr Robot)</b>	6,500€
<b>Aldebaran Robotics (NAO)</b>	11,500€

Table 2.1: Robotic Platforms for research and education

These platforms are available in kits and can be purchased by teachers, students and researchers willing to program behaviors. All these platforms already have compatibility with Android OS. Robotic platforms like Lego Mindstorms EV3, Thymio II, iRobot Create, TETRIX, Bioloid or VEX have a cost affordable enough to be used in robotics research.

Developed at the University of Oklahoma, an example of a low-cost platform that confirms the previous statements can be seen in [[PlatRb13](#)] and is a project that joins iRobot Create with Android devices through ROS<sup>7</sup>.

### 2.6.1 Lego Mindstorms EV3

Lego Mindstorms EV3 is the third generation robot in LEGO's LEGO Mindstorms robotics line. It is the successor to the second generation Lego Mindstorms NXT 2.0 robot.

We decided to go with the Lego Mindstorms EV3 because is easy to use and flexible and such will allow a rapid prototyping what will help to test various approaches in a limited period of time.

---

<sup>7</sup> ROS: Robot Operating System

In terms of programming, the native EV3 Software is not the best solution in market, since it relies on a visual programming approach. We tested the two best alternatives in market to connect Android devices with the EV3 and the results can be seen in the Table 2.2.

	<b>Wifi</b>	<b>Bluetooth</b>	<b>USB</b>
<b>LeJOS</b>	Yes. With all the API functionality	Yes. But with limitations	No.
<b>EV3JLib</b>	Yes. In autonomous and direct mode	Autonomous and Direct mode, but only Bluetooth PAN	Yes. Only in autonomous mode

Table 2.2: EV3 Communication Libraries Comparison

But in our platform we will need Bluetooth and Wifi running at same time in the Android device, this is explained in detail in Section 3, EV3JLib was discarded because Bluetooth PAN uses Wifi to create the personal area.

LeJOS enables the connection between Android and EV3 via Bluetooth with some API limitations, but the major drawback is that it requires an application that simulates the NXT<sup>8</sup> environment in the EV3, and in our tests the application was somewhat unstable.

Another option in LeJOS is to simulate the LCP<sup>9</sup> from NXT, allowing direct control of the EV3 (without any application running on the brick), but this feature is in an early stage of life with some instability and only a few part of the API is ported.

Finally we have the option of creating a direct connection via Bluetooth and send direct commands in bytecode to the EV3 brick.

## 2.6.2 Comercial Telepresence Robots

Over the last year it was possible to see a new wave of commercial products to grow in the area of Telepresence and Telerobotics. As mentioned in [AtkRb13]:

*“A handful of innovative high-tech startups have emerged to create a new market: remote telepresence robots.”*

Romo, on the left side of Figure 2.13, is a small robot that uses an iPhone as board computer and can be controlled with another iPhone via Wi-Fi. And Botiful, on the right side of Figure 2.13, is a similar solution to Romo, but with Android OS as a target.

<sup>8</sup> NXT Mindstorms is the antecessor of EV3, released in 2006

<sup>9</sup> The Lego Communication Protocol from Mindstorms



Figure 2.13: Romo (left) and Botiful (right)

Such solutions are devices to be used for conferencing, and an interesting market, however not modular enough or appropriate for the purposes of education or research.

## 2.7 Stream Through Android Devices

We started with the following question: Can we achieve real time stereoscopic vision, in order to develop immersive telepresence, with an inexpensive software solution through some widely available hardware?

We made a survey of the state of the art in real-time video stream, with the aim to test the best solutions developing small prototypes in order to compare and select the most suitable one for this project. The criteria used in the selection of technologies was the quality of video and audio streaming, seeking to reduce the maximum latency. Different technologies were analyzed in order to choose the best for the project context and the most suitable one will be selected and described more deeply.

The alternatives compared for video transmission between Android and PC were:

- **IP Camera:** Turn the Android device into an IP Camera that will stream to the operator computer.
- **Java Server:** Native Java server on operator computer, using Java Servlets and Apache Tomcat.

- **WebRTC:** Develop a Webapp and use the new WebRTC<sup>10</sup> technology, which consists in an API framework for real-time communication in the Web.

	<b>IP Camera</b>	<b>Java Server</b>	<b>WebRTC</b>
<b>Advantages</b>	- Easy to develop - Latency 200-250ms @ 720p	- Java Application - Server running on operator computer	- Real time communication - Peer-to-peer connection - Low latency (below 50ms)
<b>Disadvantages</b>	- Too heavy to run in Android application	- Latency >= 1000ms @ 720p	- Still in an early stage, lack documentation - Limited OpenCV support

Table 2.3: Android media stream solutions

WebRTC libraries, behind a web application, was the chosen solution.

The WebRTC libraries are optimized for real-time communications and, despite being a new technology, already shown to be in the right path. Note that behind this project we find Google, Mozilla and Opera.

We were able to achieve resolutions of 1080p running at a stable framerate of 30FPS with an estimated delay of around 40-50ms in the best circumstances under a 2.5Mbps available bandwidth. With WebRTC the FPS stability depends on packet loss.

The platform developed in this thesis also aims at a user that is accustomed to the internet browser, and the tendency in the next years is to increasingly bring software to the cloud. With this solution the application is also multi-platform.

### 2.7.1 WebRTC

WebRTC is an API definition drafted by the World Wide Web Consortium (W3C) to enable end-to-end browser communication without using any plug-in. This browser-to-browser connection can be comprised of an audio stream, video stream and/or data channel. WebRTC uses SRTP<sup>11</sup> for media transmission and ICE<sup>12</sup> for traversal through NAT's and firewalls.

<sup>10</sup> <http://www.webrtc.org/>

<sup>11</sup> Secure Real-Time Transport Protocol

<sup>12</sup> Internet Communications Engine

## State of the Art

Some features, among others found in [WRTC14], stated in [ReRR14]:

- Provides APIs and access rules for end-user devices such as microphones, cameras etc.
- An end-to-end security architecture and protocol is given. It uses SRTP.
- NAT transversal techniques for peer connectivity are implemented.
- Signaling mechanisms for setting up, updating and tearing down the sessions.
- Support for different media types is given.
- Media transport requirements.
- Quality of Service, congestion control and reliability requirements for the session over the Best-Effort Internet is provided.
- Identity architecture and mechanisms for peer identification are provided.
- Codec for audio and video compression.
- HTML and JavaScript APIs for use by application developers are provided.

The bit-rate is variable and depends on the current available bandwidth. As stated in [ReRR14], this option is the best for real-time applications because the loss of a small percentage of packets is tolerable. In Figure 2.14 we can see how a WebRTC application works.

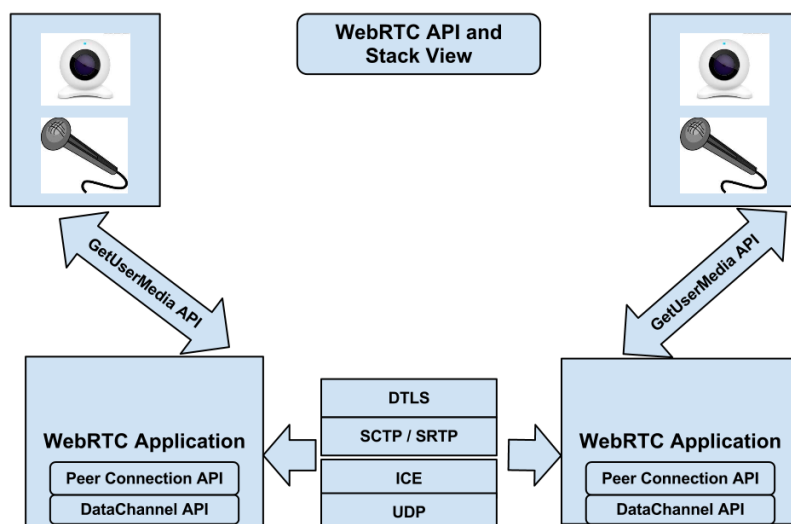


Figure 2.14: WebRTC Application<sup>13</sup>

<sup>13</sup> Figure taken from <http://blogs.cisco.com/openatcisco/webrtc-bringing-real-time-communications-to-the-web-natively/>

The most useful APIs for this project are `getUserMedia`, `PeerConnection`, `RTCDataChannel` and `RTCStats`.

- **getUserMedia API:** defines requirements for a Web application to access end-users media sources such as camera and microphone
- **RTCPeerConnection API:** specifies SDP-based session description APIs and the state machine to session setup, update and tear-down between the peers.
- **PeerConnectionStats API:** will allow to get peer connection statistics *i.e.*, bandwidth usage, packets lost, stream resolution, framerate, current delay in stream, local/remote ip addresses and ports, type of connection, etc...
- **RTCDataChannel API:** will enable peer-to-peer exchange of arbitrary data, with low latency and high throughput.

Next we can see a video stream bitrate to bandwidth ratio, over WebRTC, with VP8<sup>14</sup> codec:

- 1080p at 30 FPS causes 2.5+ Mbps bandwidth usage
- 720p at 30 FPS causes 1.0~2.0 Mbps bandwidth usage
- 360p at 30 FPS causes 0.5~1.0 Mbps bandwidth usage
- 180p at 30 FPS causes 0.1~0.5 Mbps bandwidth usage

A full overview of WebRTC can be found in [[WebRTC](#)].

## 2.7.2 Libjingle: WebRTC in Native Android Application

For the native Android application connection to the server, Google's libjingle<sup>15</sup> library will be used, which contains a set of components to interoperate with Google Talk's peer-to-peer voice and video chat. Libjingle contains the API of WebRTC and, besides that, it has the stacks of Extensible Messaging and Presence Protocol (XMPP) and Session Traversal Utilities (STUN) implementation that will be used in the signaling process (process explained in Section 3.3.1).

## 2.8 Single-Page Application

Since we were looking for a non-blocking Single-Page Application (SPA), for intensive real-time streaming across distributed devices, we take into consideration two platforms, namely: Node.js<sup>16</sup> and Vert.x<sup>17</sup>. To take a decision we consider the engine in which the platforms are running, overall documentation, community support and package management. In Table 2.4 we can see the comparison between these two platforms.

---

<sup>14</sup> VP8 is a video compression format owned by Google

<sup>15</sup> <https://code.google.com/p/libjingle/>

<sup>16</sup> <http://nodejs.org/>

<sup>17</sup> <http://vertx.io/>

## State of the Art

	<b>Node.js</b>	<b>Vert.x</b>
<b>Source Language</b>	C++	Java
<b>Operating System</b>	OS X, Linux, Solaris, FreeBSD, OpenBSD, Microsoft Windows (older versions require Cygwin), webOS	Cross-platform
<b>Platform</b>	V8 JavaScript Engine	Java Virtual Machine (JVM)
<b>Language</b>	Javascript	Java, JavaScript, Groovy, Ruby, Python, Scala, Clojure and Ceylon
<b>Package Management</b>	Node Package Management (NPM) (highly mature)	Still maturing
<b>Use Cases and Examples</b>	Highly mature	Very young community

Table 2.4: Node.JS and Vert.x comparison

Since Node.js is a platform built on Chrome’s V8 JavaScript<sup>18</sup> runtime, developed by Google, and we are aiming at real-time streaming using WebRTC APIs, also developed by Google, Node.js seemed the first obvious choice. Besides that, the highly matured community and the highly matured NPM, were decisive. As stated in [Node14]:

*“When discussing Node.js, one thing that definitely should not be omitted is built-in support for package management using the NPM tool that comes by default with every Node.js installation. The idea of NPM modules is quite similar to that of Ruby Gems: a set of publicly available, reusable components, available through easy installation via an online repository, with version and dependency management.”*

Some important modules for this project would be:

---

<sup>18</sup> <https://code.google.com/p/v8/>



- **Express.js:** A Sinatra<sup>19</sup>-inspired web development framework for Node.js, and the standard for the majority of Node.js applications out there today. Using a robust set of features, developers can create single, multi-page, and hybrid web applications.
- **Socket.io:** Server-side component of the two most common websockets components out there today.

A full list of packaged modules can be found on the NPM website<sup>20</sup>, or accessed using the NPM CLI tool that automatically gets installed with Node.js.

Since we also knew that the need to use OpenCV could be significant, a survey to find some valid options was made, which led to the next ones:

- Node-OpenCV<sup>21</sup> - OpenCV module for Node.js (missing some features).
- OpenCV<sup>22</sup> for Google Chrome.
- OpenCVjs<sup>23</sup> – A javascript implementation of OpenCV.

## 2.8.1 WebGL

WebGL (Web Graphics Library) is a JavaScript API for rendering interactive 3D graphics and 2D graphics within any compatible web browser without the use of plug-ins. It is integrated completely into all the web standards of the browser allowing GPU accelerated usage of physics and image processing and effects as part of the web page canvas.

We will use WebGL projection and rendering to create both virtual cameras that will be fed via WebRTC media stream from Android devices.

The transformation needed to implement the barrel distortion [Distort14] required by the HMD, because of pincushion distortion [Distort14] created by the lenses, will be also implemented in WebGL.

## 2.9 Conclusions

We can conclude that recent developments in Mobile Platforms, Immersive Technologies and Mobile Robotics turns possible the development of an immersive telepresence solution in the field of telerobotics, emerging as a new field – Immersive Telerobotics. One survey was carried out and aimed to find which technologies, in terms of hardware, should be used to develop immersive

---

<sup>19</sup> <http://www.sinatrarb.com/>

<sup>20</sup> <https://npmjs.org/>

<sup>21</sup> <https://github.com/peterbraden/node-opencv>

<sup>22</sup> <http://opencv.org/opencv-ported-to-google-chrome-nacl-and-pnacl.html>

<sup>23</sup> <https://github.com/sakiyamaK/OpenCVjs>

modules. Being these modules in the fields of stereoscopic vision, head tracking and movements and gestures controls. We now present the conclusions.

### **2.9.1 Hardware to be Used**

For the stereoscopic vision and head tracking modules the Oculus Rift it is the best option available for a low / medium cost system. Besides having an open-source SDK and out-of-the-box support for the most recognized game engines of the market, still possess technical specifications that reach the state of the art solutions.

For a possible person-computer interaction module to be developed in the future, and taking into account what has already been confirmed by previous studies, stating that a touch-based module using the most recognized platforms is not the best option, and also taking into account previous studies [[ARLM13](#)] and [[ARLM14](#)], we conclude that for the development of a control module by movements and gestures with the precision that a system like this would need, the Leap Motion platform will be the best option.

But since this is not our main scientific focus, the “Oculus Rift DK2 Positional Tracking Camera” can be a good solution to implement a simpler module with a zoom-in and zoom-out “feeling”, giving the operator the possibility of using its body in order to have a finer control on the actuators of the robot.

In the remote environment, the slave, we concluded that in robotics field there are two paths to be followed. The first path is most suitable for those who have few, if any, knowledge of electronics and focus on the use of a robotic kit to build the robot. The other option, which can also be cheaper, is to acquire an open source electronic controller board, like the Arduino ADK Rev3 or IOIO and use any RC vehicle.

We decided to take the first path in our case study, and we are using the Lego Mindstorms EV3 kit as a proof of concept.

As onboard computer on the robot, and for stereoscopic vision streaming, two medium range smartphones will be used.

## 2.9.2 Technologies and Software Modules to be Used

In terms of software to be used throughout the development of this thesis, we now present the conclusions organized in terms of web (Master) and mobile (Slave) applications.

Master:

- **Node.js:** as cross-platform runtime environment
- **Express.js:** web application framework
- **Socket.IO:** for signalling and communication protocol
- **WebRTC APIs:** for audio and video streams and stereoscopic synchronization
- **WebGL:** for image display, processing and synchronization

Slave:

- **Android SDK:** to develop native application and camera access
- **Google's Libjingle:** for WebRTC and ICE (STUN) implementation
- **Socket.IO-client.java:** for communication protocol



## Chapter 3

# TrekProbe - Immersive Telerobotic Modular Framework

In this Chapter we will start presenting the identified requisites, proposed architecture, and then the development and implementation of the platform, which is comprised of two major components the master and slave, web application and mobile, respectively.

### 3.1 Requirements Specification

In this section we have a specification of requirements that define which features we want in the global platform.

We want to develop a turnkey solution in which the user can test a telerobotic system with easy to find and cost effective devices.

Our main focus, in the platform development, is to develop a low cost software solution to achieve real time stereoscopic vision through Android devices. We want our platform to be modular and capable of being extended, in order to help researchers to study as many case studies as possible. We can think of a black box solution that receives input from the operator and sends the data to robot, receiving input from robot and sending back to the operator.

The main requirements identified for the global platform are:

- Real-time stereoscopic vision through Android devices

- A modular platform
- A black box software solution that can be used with any kind of robot
- Control of a robotic head with head flexion, extension and axial rotation
- Robot control with body movement and classic Keyboard/Gamepad input

### 3.2 System Architecture

With the defined requirements a global architecture, which we can see in Figure 3.1, was designed.

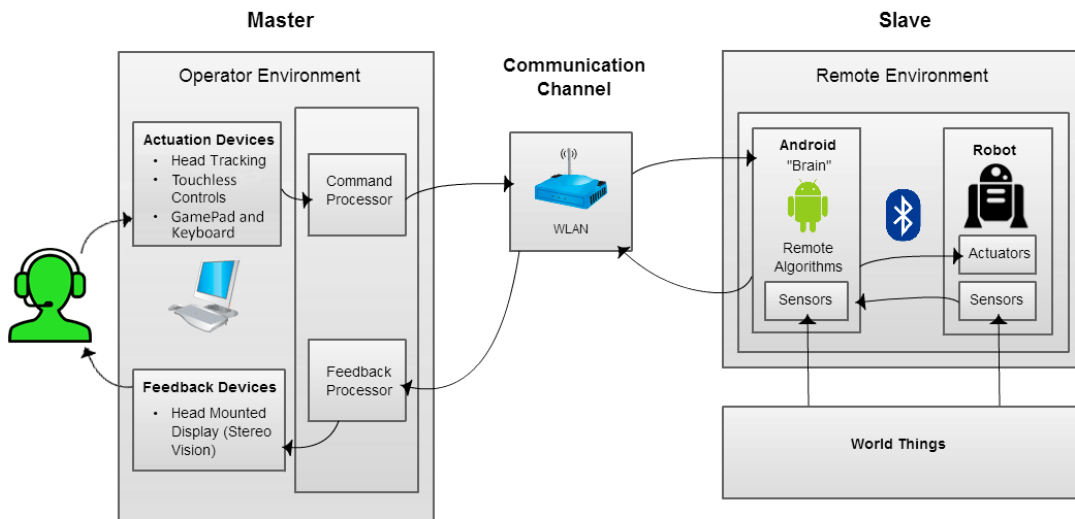


Figure 3.1: Global System Architecture

The operator is interacting with the system through the actuation devices, all the data generated in the operator environment, through gamepad, positional and head tracking, is then sent to the robot by our communication protocol. The operator's senses are stimulated with stereo images feeding an HMD and audio through real time streaming. The sense of immersion is also enhanced via the positional and head tracking that mimics the operator's movements in the remote environment.

The main objective here was to achieve real time stereoscopic vision through everyday hardware and software synchronization. The second objective is to keep the platform modular and with a software framework design approach, the aim of this second objective is to provide a solution in which the researcher can use any kind of robot depending on the case study needs.

In the researcher perspective it would be easy to extend the platform, just needing to override the Robot Communication module in the Android application, developing the proper bytecode to communicate, with the desired robot, via Bluetooth.

As an example, we can see in Figure 3.2 the bytecode written in the prototype developed with this framework, throughout this thesis, that was aiming at test the stereoscopic vision impact in teleoperation of robots. This prototype, and all the experimental procedure, is described in more detail in Chapter 4.

```

public void startEngine( byte port, byte speed) {
    byte[] cmd = new byte[15];

    cmd[0] = (byte)0x00;
    cmd[1] = (byte)0x00;
    cmd[2] = (byte)0x00;
    cmd[3] = (byte)0x00;
    cmd[4] = (byte)0x80;
    cmd[5] = (byte)0x00;
    cmd[6] = (byte)0x00;
    cmd[7] = (byte)0xA4;
    cmd[8] = (byte)0x00;
    cmd[9] = port;
    cmd[10] = (byte)0x81;
    cmd[11] = speed;
    cmd[12] = (byte)0xA6;
    cmd[13] = (byte)0x00;
    cmd[14] = port;

    bluetoothEV3Service.write( cmd );
}

```

Figure 3.2: Example of bytecode for robot control

The bytecode written, in Figure 3.2, was developed while extending this platform with the selected robotic platform (see the detailed experiment in Chapter 4). As we can see this function starts a desired actuator with a specific motor speed, sending the command via the Bluetooth communication module. Extending the framework to another robot might be as simple as rewriting these functions.

### 3.3 Communication Protocol from Master to Slave

In this section we describe the communication protocol between Master and Slave. The media stream is done via WebRTC and while it enables a peer to peer connection between the Android application and Web application browser, as stated in section 2.7.1, it still needs servers to deal with signaling, network address translators (NATs) and firewalls.

#### 3.3.1 Signaling

Signaling is the process of coordinating communication. For a WebRTC application to establish a connection, as stated in [H5Rocks], its clients need to exchange the following information:

- Session control messages used to open or close communication

- Error messages
- Media metadata such as codecs and codec settings, bandwidth and media types
- Key data, used to establish secure connections
- Network data, such as a host’s IP address and port as seen by the outside world

This signaling process needs a way for clients to exchange messages. This mechanism is not implemented by the WebRTC APIs: we need to build it ourselves.

In our project a WebSockets solution is used to establish the signalling process, through Socket.IO library, on Node.js.

After signalling, in order to deal with Network Address Translation (NAT) and firewalls, the Interactive Connectivity Establishment (ICE) framework is used. We used Google’s STUN servers with the RTCPeerConnection to test the implementation. But note that in our project the objective was to use the platform in a wireless local area network (WLAN) and, since we are behind NAT, there’s no need to use STUN.

### 3.3.2 Robot Commands API

In this Section we present the commands developed during the project and that are currently available in order to control a robot. The API was developed with JavaScript Object Notation (JSON) encoding.

We have three types of commands, namely:

- Robot Movement
- Head Tracking
- Positional Tracking

In Table 3.1 we can see the major commands currently available in this platform for robot control.

	<b>Command</b>	<b>Parameters</b>	<b>Description</b>
<b>Robot Movement</b>	<i>{ stop }</i>		Stop all the actuators used in the locomotion of the robot
	<i>{ front    back }</i>	<ul style="list-style-type: none"> <li>• <i>speed</i></li> </ul>	Move the robot forward or backward  Parameter <i>speed</i> accepts [0, 100] or -1 for a default preset speed



	{ <i>left</i>    <i>right</i> }	<ul style="list-style-type: none"> <li>• <i>speed</i></li> </ul>	Robot turns left or right Parameter <i>speed</i> accepts [0, 100] or -1 for a default preset speed
Head Tracking	{ <i>liftHead</i>    <i>lowerHead</i> }	<ul style="list-style-type: none"> <li>• <i>speed</i></li> <li>• <i>degree</i></li> </ul>	Lift or lower the robotic head Parameter <i>speed</i> accepts [0, 100] or -1 for a default preset speed Parameter <i>degree</i> from [0°, 180°] <sup>24</sup>
	{ <i>leftHead</i>    <i>rightHead</i> }	<ul style="list-style-type: none"> <li>• <i>speed</i></li> <li>• <i>degree</i></li> </ul>	An axial rotation to left or right, in the robotic head Parameter <i>speed</i> accepts [0, 100] or -1 for a default preset speed Parameter <i>degree</i> from [0°, 180°] <sup>24</sup>
	{ <i>liftHeadRect</i>    <i>lowerHeadRect</i>    <i>leftHeadRect</i>    <i>rightHeadRect</i> }		Control the robotic head in the lift/lower or left/right movements Acts as a calibration command, to be used as a manual rectification when the robotic head gets out of sync with the HMD
Positional Tracking	{ <i>forwardWS</i>    <i>backWS</i>    <i>leftWS</i>    <i>rightWS</i> }	<ul style="list-style-type: none"> <li>• <i>speed</i></li> </ul>	Robot moves without stop. Keep the robot moving while the operator's body is in the same position. Changing speed according to the movement of the body <sup>25</sup> Parameter <i>speed</i> from [0, 100]

Table 3.1: Robot Control API commands

In Figure 3.3 we can see a practical example of a JSON formatted message. In this case we have a Head Tracking type message making the robot lift its robotic head by 3 degrees.

<sup>24</sup> The degree conversion, from head position to the platform, will be explained in Section 3.4.2.

<sup>25</sup> Entire Positional Tracking algorithm is explained in Section 3.4.3

```

{
  "actuatorProtocol": {
    "streamID": "123456789",
    "type": "headTracking",
    "payload": {
      "msg": "liftHead",
      "speed": 100,
    }
    "degree": 3
  }
}

```

Figure 3.3: JSON encoded Head Tracking command

### 3.4 Master: Web Application

In the current Section we present all the details concerning the development of the Web Application. We were aiming at creating an independent web-based user interface (UI) over a platform independent system. A web-based UI allows the user/operator to quickly start using the robot without plugins or installations. As stated in Section 2.8, the platform selected was Node.js.

We will start pointing the main features and requirements of the platform, progressing to the system architecture and main modules, finally explaining the hardware integration and the major algorithms developed.

#### 3.4.1 Requirements Specification

We wanted a telerobotic platform with immersive modules like: HMD with stereoscopy, head tracking and positional tracking. But we also wanted some more “classic” modules of interaction like HMD with mono vision and keyboard/gamepad controls.

We are also aiming at a real-time streaming with software synchronization, thus it will require a manual image rectification to adjust the different focal lenses distances and also a manual calibration of the epipolar lines in order to avoid a bigger conflict between the accommodation and vergence of the eye<sup>26</sup>.

Therefore, the following are the main requirements that have been identified for the solution:

<sup>26</sup> The image rectification process is explained in detail in Section 3.4.4.

- Manually-assisted Image Rectification
- Automatic Stereoscopic Synchronization
- Telerobotic platform with:
  - Classic interface (no HMD)
  - Non-stereoscopic vision with HMD
  - Stereoscopic vision with HMD
- Head tracking to control the robotic head
- Positional tracking to have more intuitive control of robot movement

Regarding the main options available in terms of interaction in our telerobotic platform, the diagram in Figure 3.4 was developed.

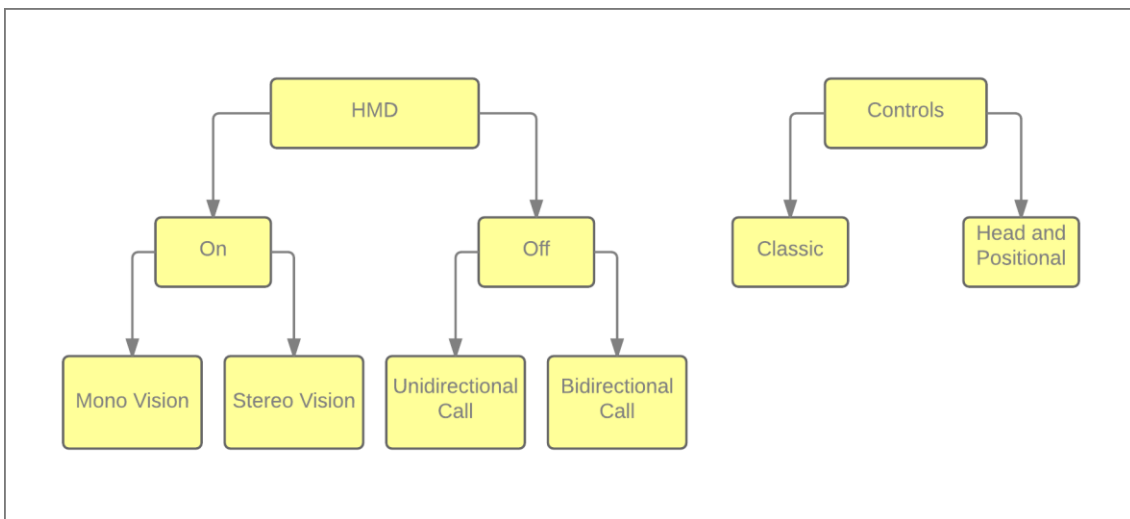


Figure 3.4: Diagram Depicting Main Web Application Options

When the operator is using the HMD, he has the option of mono vision which consists in feeding the HMD with the video stream of one Android device, or the stereo vision consisting in the feed of the HMD by two sources, through the video stream of two Android devices.

When the HMD is off we developed an extra feature that allows the user to perform a “bidirectional call” that consists not only in receiving audio and video from the remote location, but also send audio and video to the robot through the peer-to-peer connection. In Section 3.4.1.2 we briefly explain this extra feature.

The user is able to go with only a “classic” approach, meaning that it can control the robot and robotic head with the keyboard or gamepad. But on the other hand, the user can complement these controls with head tracking and positional tracking modules: the head tracking to control the robotic head and positional tracking for a finer control of the robot movement (both of these

options are explained in depth in Section 3.4.4). We can see in Figure 3.5 the UI developed that incorporates all this options.

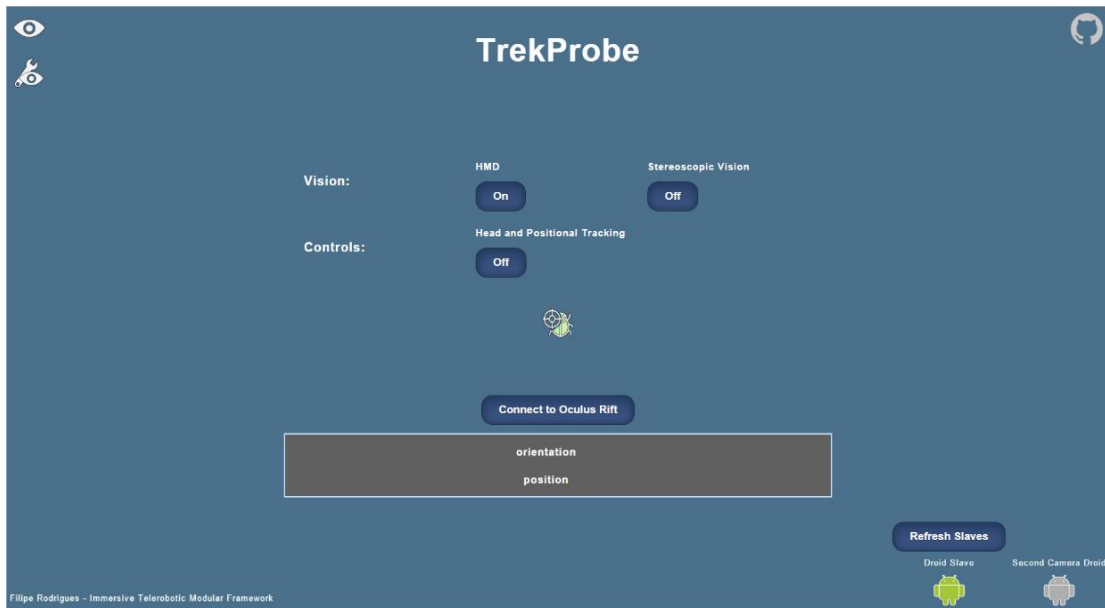


Figure 3.5: Web Application UI - Options

### 3.4.1.1 Technologies and Modules

In this Section we present the technologies used in this Web Application.

We already stated this in Section 2.9.2, however, as a brief summary, the runtime environment is Node.js, the web framework is Express.js, Socket.io is used for communication protocol and signalling, Three.JS<sup>27</sup> for image display, rectification and synchronization and finally for media stream we are using WebRTC APIs, namely:

- ***getUserMedia*** to access local media sources, like camera and microphone.
- ***RTCPeerConnection*** that represents the WebRTC connection and handles all the media streaming between two peers.
- ***PeerConnectionStats*** that allows to get all kind of data and statistics from the stream, and is used in one of the synchronization algorithms developed, this algorithm being explained in more detail in Section 3.4.3.2.

### 3.4.1.2 Test WebRTC Capabilities with a videocall prototype

We will use WebRTC to stream video between two devices. And since we were dealing with a new technology, as a proof of concept, we implemented a videocall-like prototype, which implements a telepresence feature similar to the ones seen in Section 2.6.2.

<sup>27</sup> Three.JS is a WebGL API. For more information: <http://threejs.org/>

## TrekProbe – Immersive Telerobotic Modular Framework

In Figure 3.6 we can see this feature running in the platform developed.

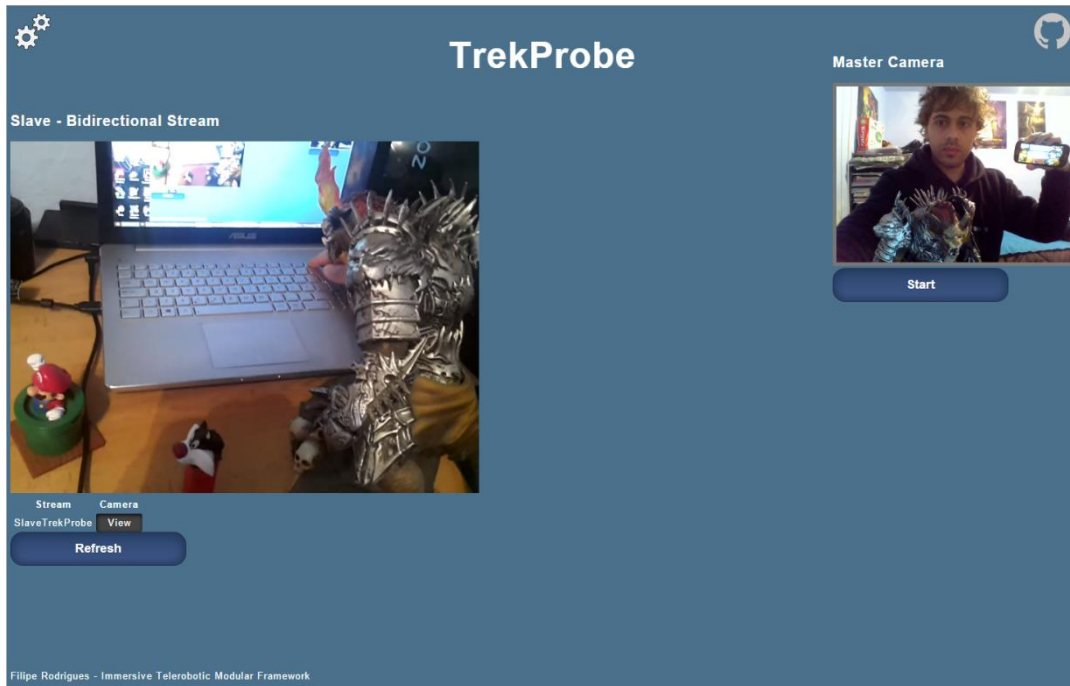


Figure 3.6: Web Application – Bidirectional Call

In Figure 3.6, that shows the bidirectional feature running, we can see on the left side of the image the Slave stream, which is coming from the Android device. On the right side we have the Master Camera that shows our own camera, in this case the laptop one. The Android device is also receiving audio and video coming from PC microphone and camera, and reproducing both via the device speaker and screen.

The Slave stream has a resolution of 1280x768 running at 30fps, this and other options can be changed in the mobile app, and this will be explained in detail in Section 3.5. The Master stream has a resolution of 640x480.

### 3.4.2 System Architecture and Modules

With the defined requirements we developed the architecture presented in Figure 3.7.

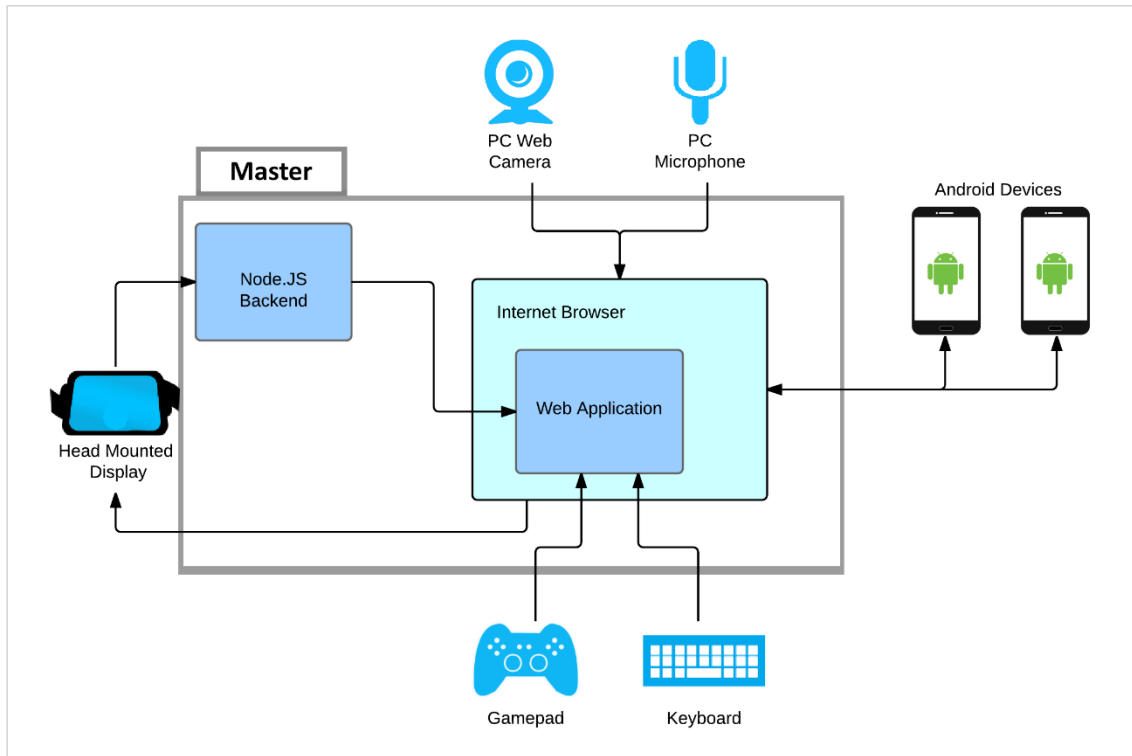


Figure 3.7: Architecture Overview

The HMD provides Node.JS with head and positional tracking data. These data are then redirected to the Web Application through websockets. Finally, after processing the data in accordance with the developed specification, our Communication Protocol is used in order to send commands to the primary Android device, which is the device controlling the robot.

The web application uses Signaling in order to establish a direct peer-to-peer connection between browser and Android device. Android device uses a specific library to establish this connection, which is explained in detail in Section 3.5.

In the web application, after the image rectification and stereoscopic synchronization, we have a real time display of the remote environment through the projection of virtual cameras, process explained in Section 3.4.3.

Event Listeners are used to get input from keyboard and gamepad.

Finally, the Web Application uses *getUserMedia()* API, invoked in browser (Mozilla Firefox or Chrome), to get a *MediaStream* which allows the web application to access local camera and microphone.

Proceeding to a description of the main modules of the system's source code, where a class diagram can be visible in Figure 3.8.

- **app.js:** this module is the Node.js server. It connects to the *routes* package essentially to handle the active streams requests. The *serverSocketHandler* is a module that allows our server to handle new requests and incoming connections from clients, and also is here that the server is polling the Oculus Rift.
- **routes:** this package deals with the routing between server and web application, essentially keeping the streams indexed.
- **app\_modules:** this package contains two main modules, *streams* is used to represent an active list of streams, while *serverSocketHandler* handles all the websockets communication and Oculus Rift polling with a delay of 150ms.
- **app\_scripts:**
  - *TrekProbeViewModel* is a module that encapsulates all the functionality the system provides to the user.
  - *ClientManager* represents the web application, it handles the head tracking and positional tracking input using the communication protocol, is also responsible for handling exchanged data between clients.
  - *OculusRender* is the module responsible for WebGL, it does the image rectification and stereoscopic synchronization, displaying the HMD view in a WebGL canvas.

# TrekProbe – Immersive Telerobotic Modular Framework

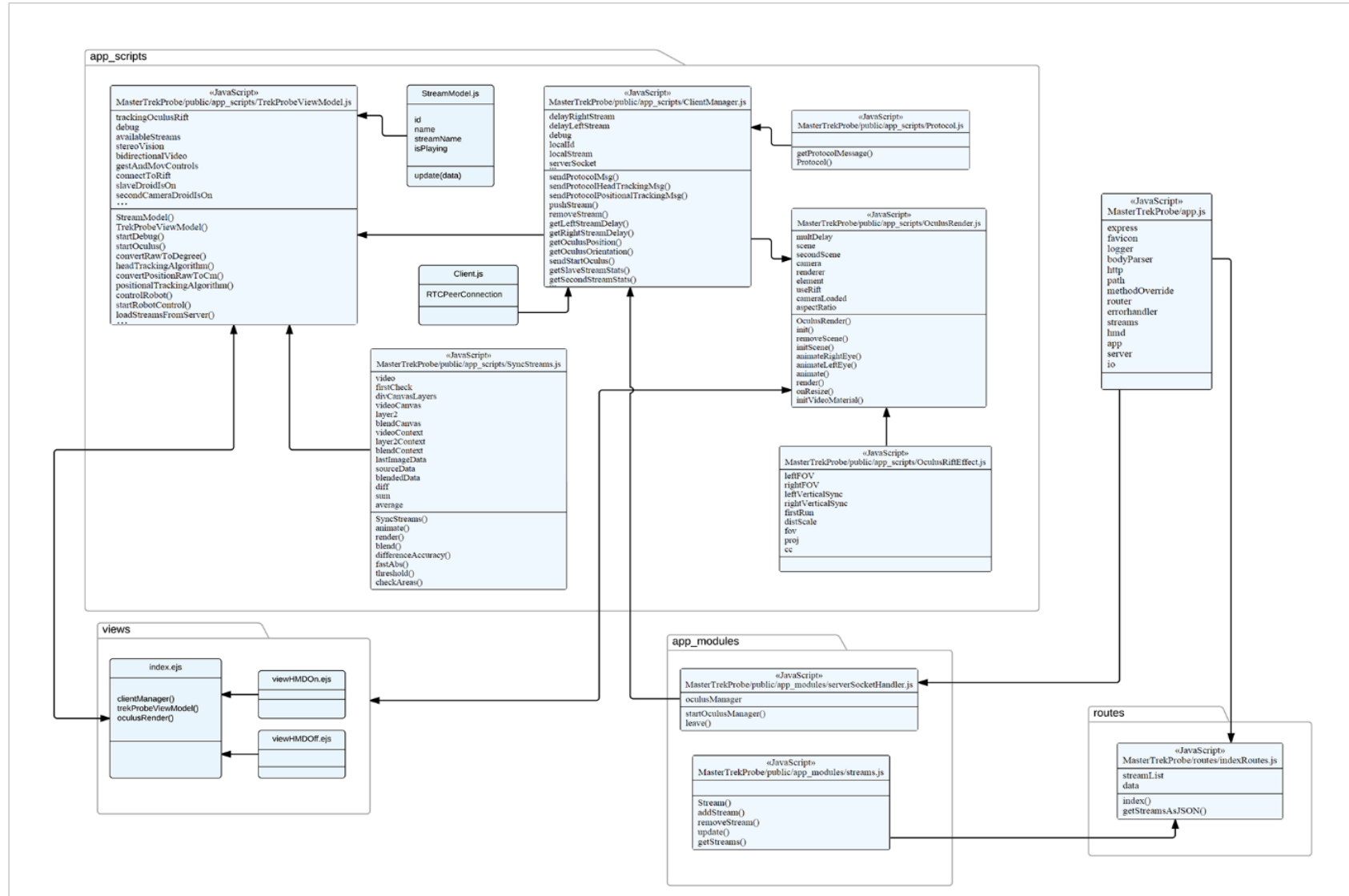


Figure 3.8: Main Modules of Web Application



### 3.4.3 Stereo Vision System through WebGL

As previously stated we aim at a software solution through WebRTC technology with WebGL projection and rendering. For a true real-time application, latency is a critical factor.

Maintaining a stable frame rate is also crucial in order to avoid motion sickness. There is a lot of effort put by the research community into the real-time processing of 3D stereo vision and we know that the rate of 30 frames per second (FPS) is the standard, desirable value, for the human eye. With WebRTC the connection is peer-to-peer, meaning that the frame rate stability depends on packet loss, or by another words we are mainly dependent on bandwidth to achieve really acceptable results.

Three.js library is used to render two virtual cameras, each one simulating left and right eye, using perspective cameras and WebGL renderer capabilities. The transformation needed to implement the barrel distortion required by Oculus Rift, because of pincushion distortion created by the Oculus lenses, is also implemented in Three.js.

#### 3.4.3.1 Stereo Image Rectification

As stated in Section 2.3, image rectification is a process that aims at removing vertical misalignment, aligning the epipolar lines and distortion due to different focal length distances.

Different Android devices have different camera specifications - like different focal length distances or apertures.

In real time stereo vision the focal length is a very important factor, both images need to be aligned by the epipolar lines in order to have a vertical synchronization and avoid a bigger conflict between the accommodation and vergence, which can lead to an increase of two errors: loss of accommodation resulting in a blurred image or loss of fusion resulting in double vision (or both) [[VisualF09](#)].

The image rectification process can be considered as a reprojection of the 3D world into our virtual cameras. With the concept of virtual cameras we have a few advantages, for instance both rectified images are independent from the real projection system, this means that the processing software does not need to know anything about the real cameras and can be adapted to the projection of virtual cameras.

Regarding the epipolar lines rectification, we used a manual calibration that can be used by the operator at anytime. The concept is simple and is explained with the aid of Figure 3.9.

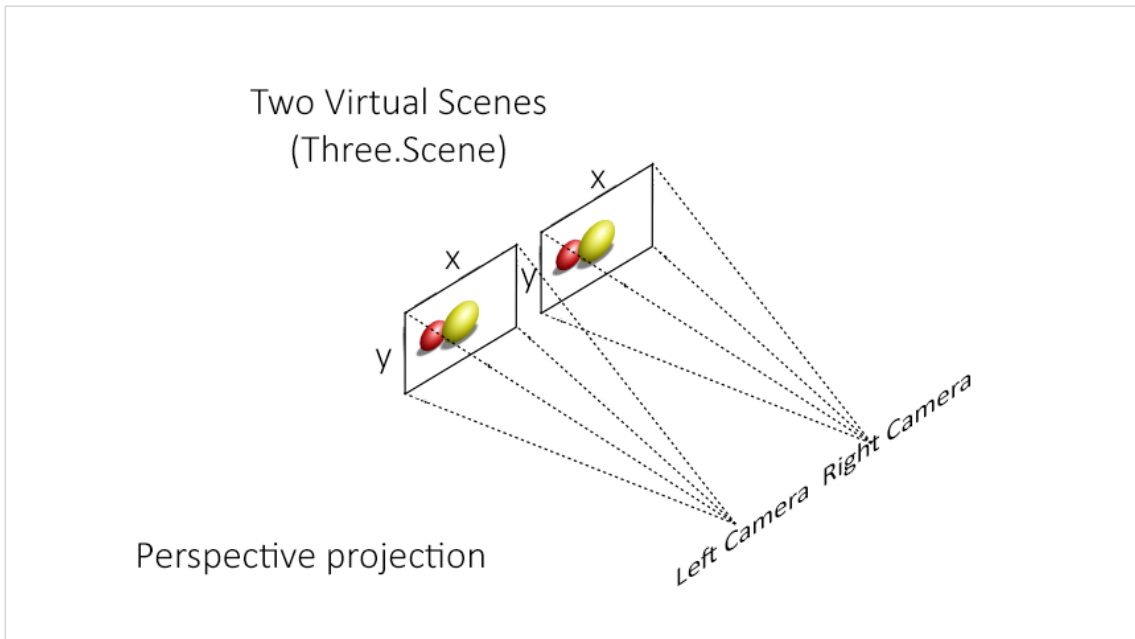


Figure 3.9: Virtual Projection Cameras

We have two projection cameras, `THREE.PerspectiveCamera`, and two scenes, instances of `THREE.Scene`, each scene has an instance of `THREE.PlaneGeometry` with 960x1080 resolution (same of each eye in the Oculus Rift DK2) that receives an object *movieScreen*, which is an instance of `THREE.Mesh` with a canvas texture applied into it. This texture is the video stream for each eye.

We provide the user with option of changing the y axis of each virtual scene, this makes it possible to rectify the vertical misalignment, aligning the epipolar lines with the help of a physical checkerboard pattern, like the one shown in Section 2.3.

Another problem with stereo vision arises due to the different focal length distances when dealing with different devices. In order to acquire the stereo image calibration we implemented a method that, once more using a chess pattern, allows the operator to manually calibrate the virtual cameras (`THREE.PerspectiveCamera`) frustum vertical field of view (vFOV) and rectify the differences induced by different focal length distances. With this feature the user can manually calibrate the virtual projection in order to compensate different specifications of real cameras.

### 3.4.3.2 Synchronizing Streams

The WebRTC PeerConnection API is used in both streams to receive video while audio stream comes from one of them.

The major desynchronization factor in this platform would be due to WLAN instability.

The Google statistics API (*PeerConnectionStats*) is used in order to get the delay information, from both streams. After acquiring the delay from the streams a synchronizing algorithm is implemented, this algorithm consists in acquiring the difference between both delay values and then, if the absolute difference is  $\geq 0$ , the faster stream is delayed by that very same value. In Figure 3.10 we can see this implementation in pseudo code form.

```

for each eye{
    delay_val = (other_stream_val - current_stream_val)
    if(delay_val > 0){
        delay current stream by delay_val
    }
}

```

Figure 3.10: Pseudo-code for Synchronization

With this technique we are able to counter network latency in an efficient manner.

However, we can experience another type of desynchronization, introduced by hardware. Since the platform will allow Android devices of any range, some latency may be introduced when using more modest devices, which can cause discrepancy due to hardware limitations. To counter this problem we developed a motion recognition synchronization. This solution is used one time in order to establish the hardware introduced lag.

The motion detection sample in [\[motionJS\]](#) was adapted in order to output a timestamp when the user slides the finger in a defined area of the camera frame. This works as a movie slate, marking the timestamp in each stream.

This system only performs recognition on the canvas, which means that the recognition is done after the stream has passed through the network and before being drawn on the screen, but already lying on the server. If we remove the delay between streams, the remaining is the hardware introduced lag that can be used to synchronize streams before drawing on screen.

### 3.4.4 Oculus Rift DK2 Integration

For hardware access and to obtain positional and head tracking data a javascript wrapper, based on node-ovrSdk<sup>28</sup> (at time of writing was only supporting v0.3.2 of the SDK), was developed resorting to a Dynamic-Link Library (DLL) compiled from v0.4.2 source code, written in C language, of the SDK.

#### 3.4.4.1 Head Tracking Algorithm

When the module responsible for handling the head tracking logic is updated by the server (that is polling the HMD), the raw data received about orientation of the head is on the fly converted to an angle value (degrees), with linear interpolation. This conversion is explained with the aid of Figure 3.11.

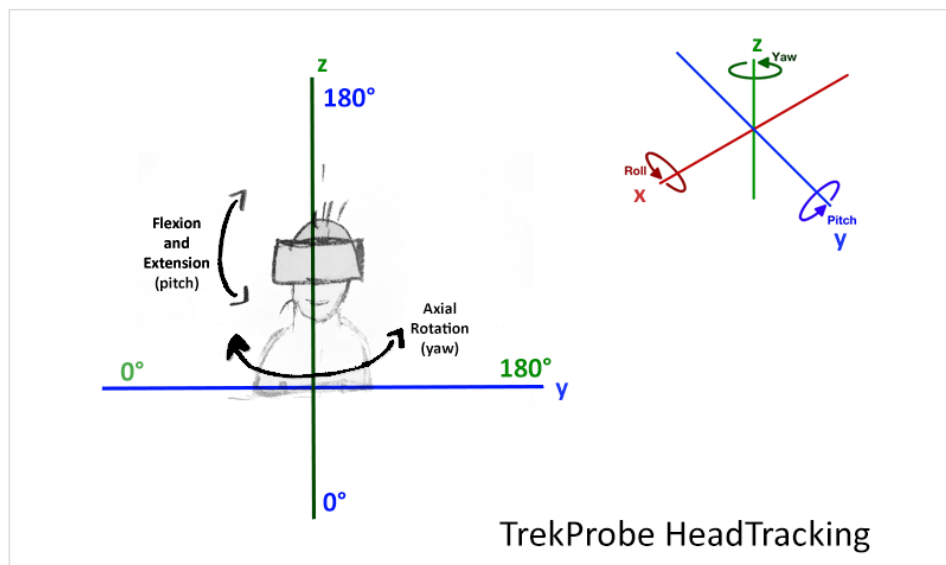


Figure 3.11: Head Tracking Orientation

In the axial rotation movement (around yaw):

- With user looking at screen, the value converted by our algorithm would be 90°.
- User looking left, maximum value will be 180° when the face is perpendicular to roll axis.
- User right, minimum value will be 0° when the face is perpendicular to roll axis.

In flexion and extension movement (around pitch):

- With user looking at screen, the value converted by our algorithm would be 90°.
- User looking up, maximum value will be 180° when the face is perpendicular to roll.
- User looking down, minimum value will be 0° when the face is perpendicular to roll.

<sup>28</sup> <https://github.com/wwwtvro/node-ovrSdk>

To avoid noise we use a threshold of 1 degree in the logic that checks if the head changed orientation.

### 3.4.4.2 Positional Tracking Algorithm

Positional tracking is used for a more intuitive control of the robot moving actuators, for example can be used for a “zoom-in” experience when the gamepad control is too blunt.

We implemented a simple fuzzy logic system with the following criteria:

- When the body is straight the output in both axis should be minimal.
- When the body is leaned left or right, the output in horizontal axes should reflect the direction of the body.
- When the body is leaned forward or backward, the vertical axes should reflect the forward or backward direction.

Raw data received about position of the body is on the fly converted to a distance value in centimeters (cm), with the mathematical rule of three.

The algorithm starts by getting the initial position of the body  $\{DEFAULT\_X, DEFAULT\_Z\}$ , then fuzzy logic is used to obtain a speed related to the position of the operator. In Figure 3.12.a the pseudo code for this algorithm is presented, in Figure 3.12.b we can see the area of effect in a graphical representation.

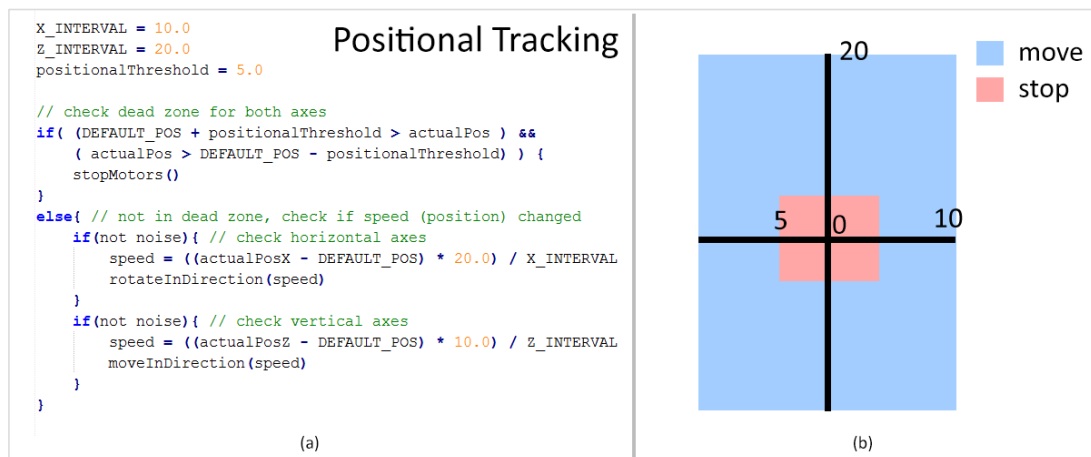


Figure 3.12: Positional Tracking pseudo-code

As we can see, the operator has a degree of freedom of 10cm for each side, and 20cm to lean forward and backward, however, as we can see in the pseudo-code, the maximum speed achieved in vertical axes (moving forward and backward) is 10% of the actuator capabilities, while maximum speed achieved in horizontal axes (rotation) is 20%. These values can be changed easily, but for the prototype being developed in a later phase, explained in Chapter 4, these values

seem a good default ones. Finally, we have a “dead motor” zone in the center, marked in red in Figure 3.12 (b), that is a square with 10cm in each side. To avoid noise a threshold of 1 degree is used in the logic that checks if the position of the body changed.

### 3.5 Slave: Mobile Application

In the current Section we present all the details concerning the development of the mobile application. This is a native application that extends the platform, and it was developed modular enough so that anyone wanting to reuse the code could focus on their needs.

The Android SDK was used for development, Google’s Libjingle is used for WebRTC and ICE (STUN) implementation and Socket.IO-client.java is used in communication protocol with the master.

We will start pointing the main features and requirements of the platform, progressing to the system architecture presentation with an UML diagram and, finally, ending with the description of the major modules.

#### 3.5.1 Requirements Specification

The application was developed with a minimum SDK requirement of 14 (Android 4.0 APIs), in order to have a more dynamic UI we opted for the Fragment<sup>29</sup> API. Since we wanted the architecture as modular as possible, we decided that the UI should reflect this approach and have three main Fragments, namely the Server, Stream and Robot configuration. Figure 3.13 shows the Sliding Menu with these options.

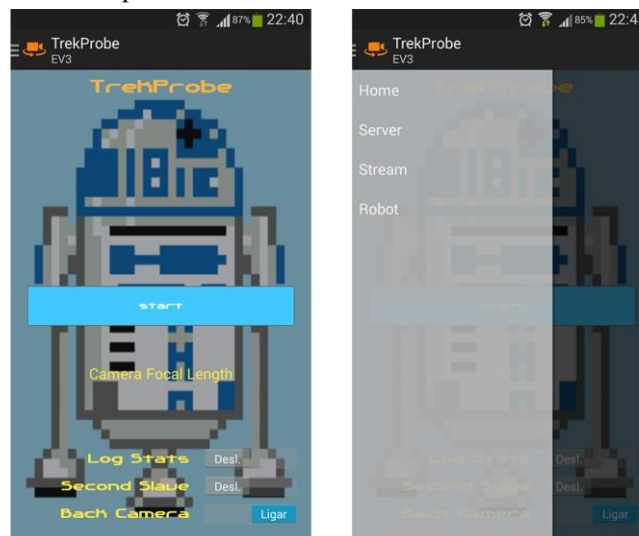


Figure 3.13: TrekProbe Slave – Sliding Menu

<sup>29</sup> <http://developer.android.com/guide/components/fragments.html>

Regarding the stream options we decided that the more important details to implement in the application would be:

- Maximum and Minimum Resolution
- Maximum and Minimum Framerate
- Aspect Ratio
- Turn the audio stream on/off
- Turn the local stream on/off
- Show remote stream in fullscreen or corner window

In Figure 3.14 we can see stream options in the UI developed.

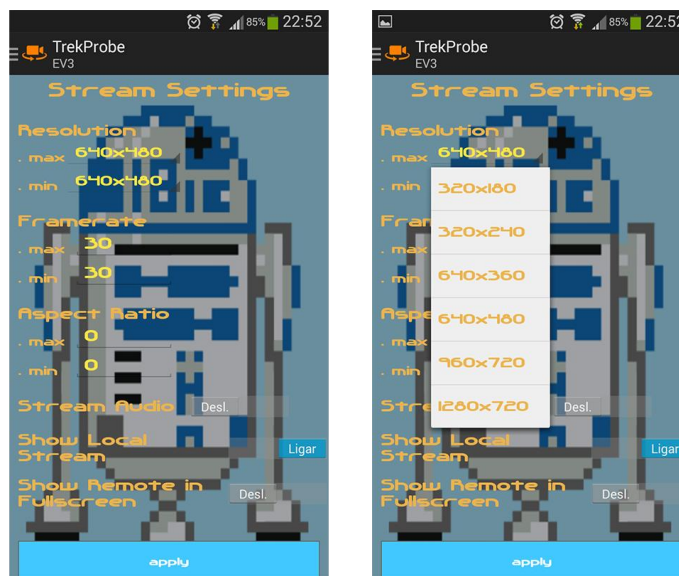


Figure 3.14: TrekProbe Slave – Streaming Options

It was also important to have one application for both devices, although they have different responsibilities, as it facilitates the deployment.

Another important requirement is to be able to choose which camera we want to use on device, since for a telepresence robot the front camera is more suitable, but for the stereoscopic immersive telerobotic option the back camera will achieve better results.

Finally, the robot configuration Fragment needs the Android system Bluetooth's turned on, and so the application checks Bluetooth status, warning the user if it is off, and then allows the user to connect to previous paired Bluetooth robots, this behaviour is shown in Figure 3.15.

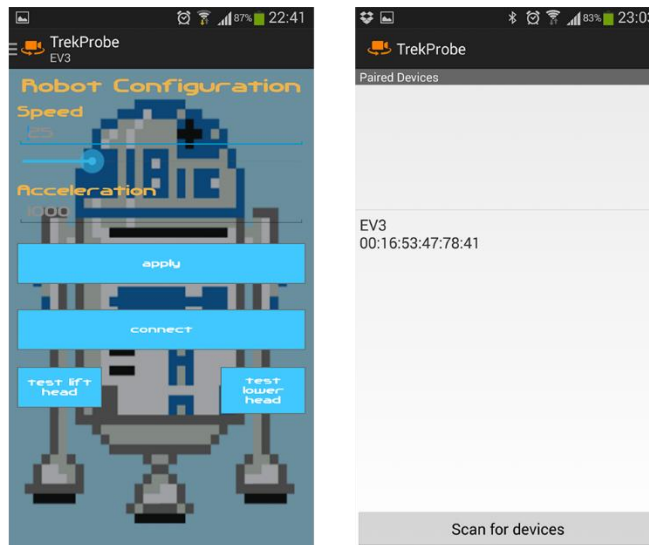


Figure 3.15: TrekProbe Slave – Bluetooth Options

### 3.5.2 System Architecture

With all the requirements defined the architecture presented in Figure 3.16 was designed.

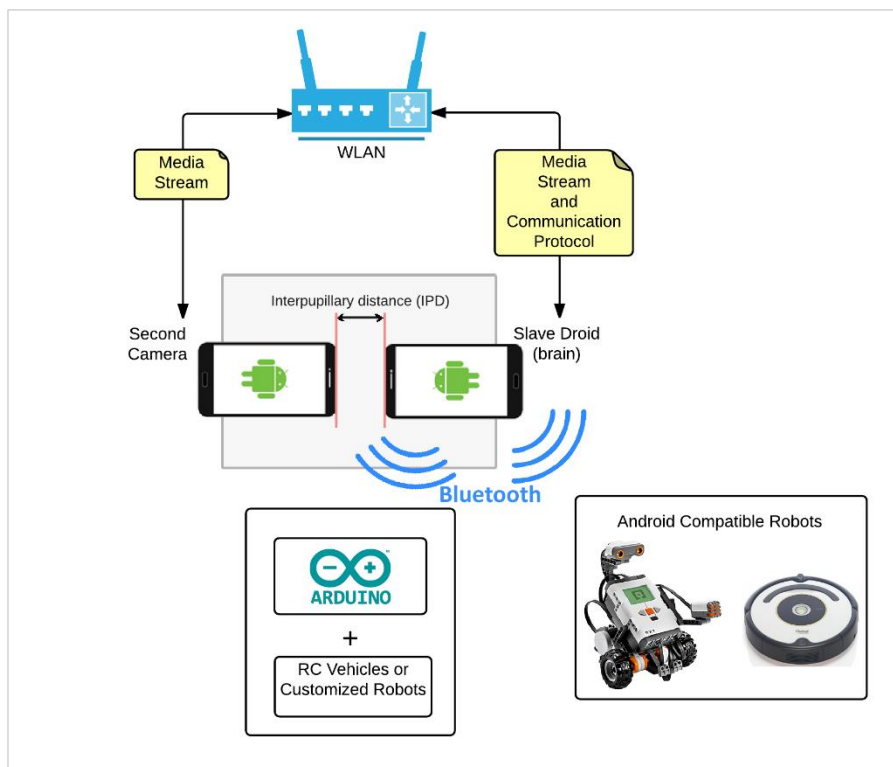


Figure 3.16: Architecture Overview of Slave Environment



This diagram represented in Figure 3.14 depicts a stereoscopic solution, both Android device are communicating with the web application via peer-to-peer and exchanging media data, however, the Slave Droid (on the right) also uses the Communication Protocol via websockets to receive the robot commands.

Slave Droid is the real “brain” of the robot, since it will communicate with any kind of hardware via Bluetooth.

### 3.5.2.1 Application Modules

Here we present the main modules from the mobile application and a diagram of the global architecture can be seen in Figure 3.17.

The main modules are:

- **com.trekprobe.connection:** this package handles all the communication with the server.
- **com.trekprobe.connection.robot:** this is the package that needs to be extended or refactored in order to establish a Bluetooth connection with the desired robotic hardware. This package was used in our prototype and is explained in detail in Chapter 4.
- **com.trekprobe.models:** this package serves to accommodate data models.
- **com.trekprobe.utilities:** several utilities used in the application are here.
- **com.trekprobe.variables:** reserved for the major variables used by the application.
- **com.trekprobe.view:** package reserved for the Activities developed for this application.
- **com.trekprobe.fragments:** here we have the Fragments developed.

# TrekProbe – Immersive Telerobotic Modular Framework

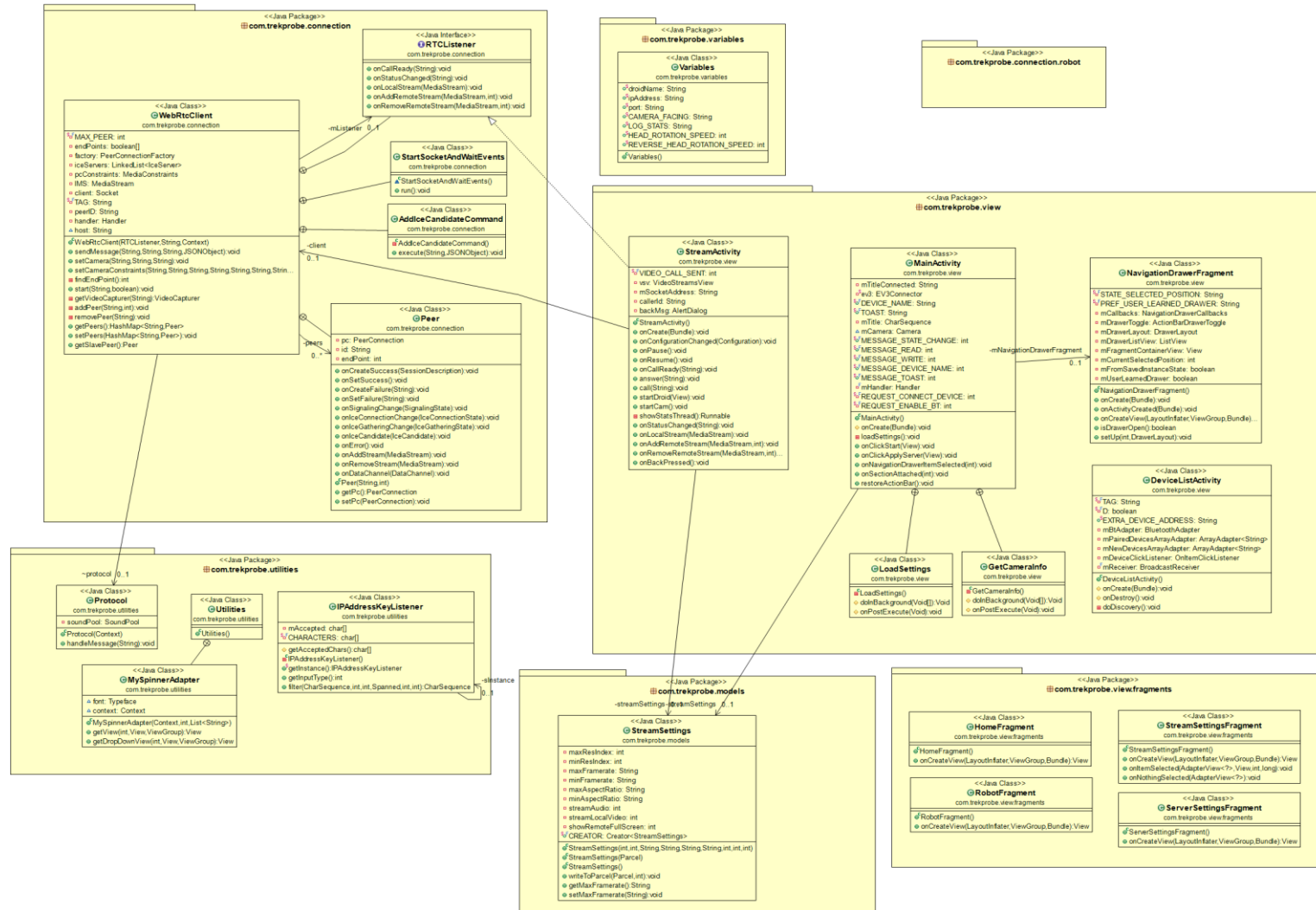


Figure 3.17: Main Modules of Mobile Application

### 3.6 Conclusions

The web application was developed as a software framework, the server runtime used was Node.js and it was developed with the web framework Express.js. We also used the module Socket.IO in order to implement the communication protocol between the server and web application and the web and mobile applications, and the Knockout.js module that provides a complementary, high-level way to link the data model to a UI. The Three.js library is used to create the virtual perspective cameras, representing each eye, image processing and stereo synchronization. An altered version of the OculusRiftEffect example is used in order to render the scene in 3D stereo with the lens distortion required by the Oculus Rift DK2 v0.4.2. WebRTC APIs are used to stream the media content and acquire stream statistics in order to implement stereo synchronization algorithms. A javascript wrapper based on node-ovrSdk (only supporting v0.3.2 at time of writing) was developed, with the proper compiled .dll from Oculus Rift DK2 source code version 0.4.2, in order to obtain the head tracking and positional tracking data.

A keyboard input system was also developed in order to allow a more classic control via keyboard and/or gamepad.

The mobile application extends the web framework and was developed with the Android SDK. Libjingle library, supported by Google, was used for media stream and signalling. Socket.IO-client.java library was used to develop the communication protocol with the web application. A module that allows to communicate with the robot chosen for the prototype, via direct connection through Bluetooth, was also developed, and will be addressed in Chapter 4.

This platform architecture was developed in a way that allow a researcher to reuse it, for different types of robots and/or RC vehicles, by just developing a new Bluetooth communication module.



## Chapter 4

# Prototype Development

With the aim of testing the impact that stereoscopic vision can have in telerobotic systems, and at same time test the platform in terms of usability and extensibility, a prototype was developed. This prototype was developed for the Experimental Procedure described in detail in Chapter 5. This phase was about the validation of the developed theoretical concepts and the implementation of all the modules required to integrate the controls for head tracking and positional tracking in the robot.

### 4.1 Unity Simulator

Due to delays in the delivery of the robotic kit, a simulator was created in Unity. The objective was to test the developed theoretical solutions, namely the control of the cameras for head tracking in a direct relationship with a small threshold and, also, the positional tracking controls of the robot body using fuzzy logic. In Figure 4.1 we can see the Unity simulator running.

## Prototype Development

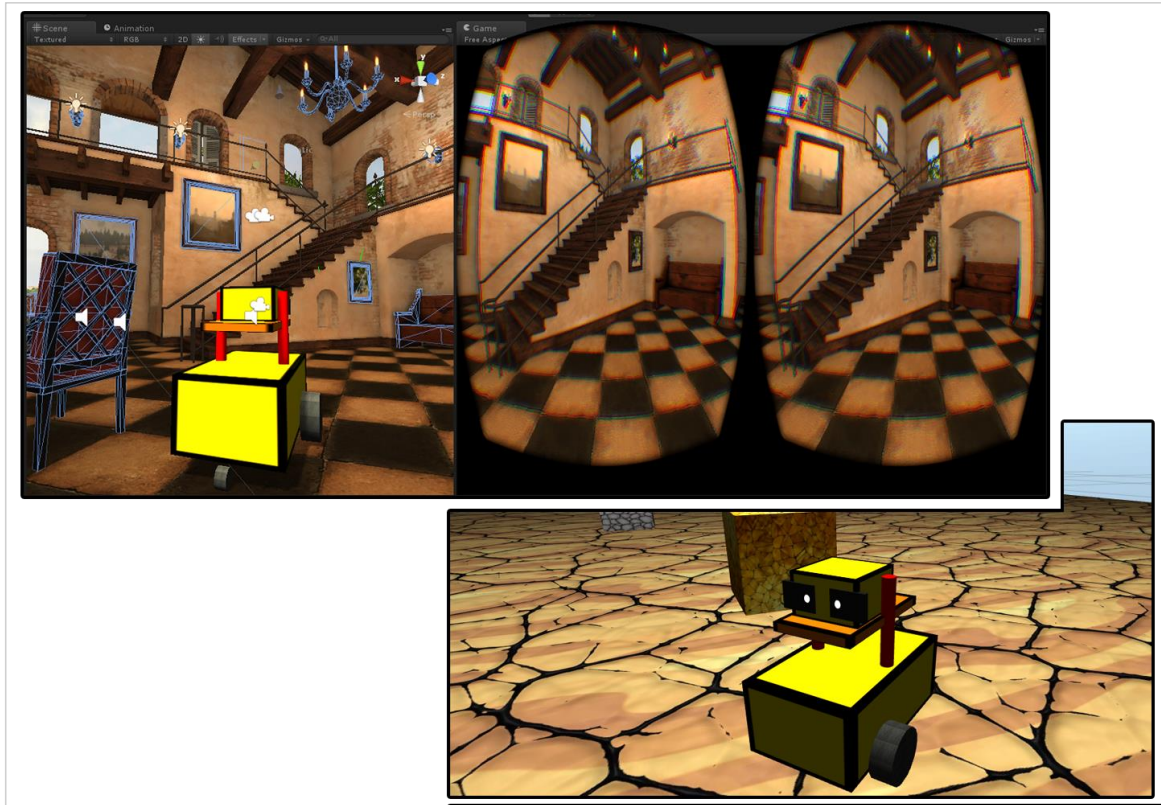


Figure 4.1: Unity Simulator

The initial design of the robot was also made in the simulator developed.

In this phase we also included in the platform as a whole (master and slave) the necessary modules to control the robot with positional and head tracking.

## 4.2 Mindstorms EV3

Since we were going to use 4 motors on EV3, the maximum amount allowed, we started looking for the motors capabilities:


- EV3 Large Engine Power: 160-170 RPM (approximately  $960 \sim 1020^\circ / s$ )
- EV3 Medium Motor Power: 240-250 RPM (approximately  $1440 \sim 1500^\circ / s$ )

### 4.2.1 Programming the Brick

As already stated in Section 2.6.1 the only library available, at time of writing, for controlling the EV3 via Bluetooth was LeJOS, however, the API is in an early stage of development. After a few tests we decided to discard due to instability issues.

## Prototype Development

The official “EV3 Communication Developer Kit” [LegoProg] was read for the purpose of understand the EV3 protocol, in order to develop all the bytecode instructions that we would need to control the robot actuators. We are controlling the EV3 brick via direct commands, an example of these commands is in Figure 4.2.



### 4.2.2 Start motor B & C forward at power 50 for 3 rotation and braking at destination

This example uses the special OUTPUT\_STEP\_SPEED motor command. This command sets the speed (setpoint) for the motors in the motor-list. The command includes a ramp-up and ramp-down portion. Especially the ramp-down is usefull for getting a more precise final destination. The motor brakes when the 3 rotations (3 \* 360 degrees) are finished.

```
opOUTPUT_STEP_SPEED,LC0(LAYER_0),LC0(MOTOR_A + MOTOR_B),LC1(SPEED_50),LC0(0),LC2(900),
LC2(180),LC0(BRAKE)
```

opOUTPUT_STEP_SPEED	Opcode
LC0(0)	Layer 0 – encoded as single byte constant
LC0(MOTOR_A + MOTOR_B)	Motor B & C (motor list) encoded as single byte constant
LC1(SPEED_50)	Speed 50% encoded as one constant byte to follow
LC0(0)	No STEP1 i.e. full speed from beginning – encoded as single byte constant.
LC2(900)	STEP2 for 2.5 rotation (900 degrees) – encodes as two bytes to follow.
LC2(180)	STEP3 for 0.5 rotation (180 degrees) for better precision at destination – encoded as two bytes to follow.
LC0(BRAKE)	Brake (1) – encoded as single byte constant.

Bytes sent to the brick:

```
1200xxx800000AE000681320082840382B40001
Bbbbmmmtthhhcccccccccccccccccccccccc
```

bbbb = bytes in message 21 excl. packet length bytes  
mmmm = message counter  
tt = type of command - Direct command no reply  
hhh = header – variable alloc?  
cc/CC = byte codes.

<sup>h</sup>hhh = 10 least significant bits are number of globals, 6 most significal bits are locals

Figure 4.2: Direct Command Example<sup>30</sup>

<sup>30</sup> Figure taken from [LegoProg]

## Prototype Development

In Figure 4.3 the specific modules developed for hardware access are shown.

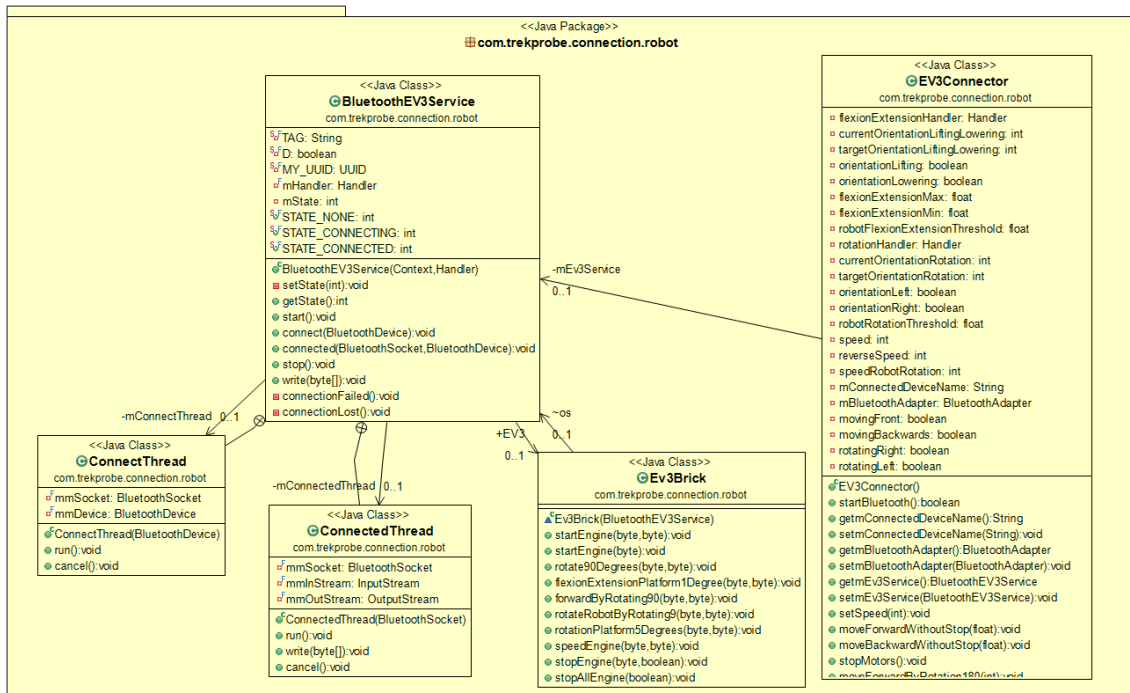


Figure 4.3: Modules Developed for Brick Control

Some of the main bytecodes developed are encapsulated in the following functions:

- *forwardByRotation90(byte port, byte speed)* – this function moves the robot front by rotating both actuators 90 degrees, in the first 30 degrees gaining power and in the last 30 losing it.
- *rotateRobotByRotating9(byte port, byte speed)* – this function rotates the robot 9 degrees, while gaining power in the initial 3 degrees and losing in the last 3 degrees.
- *moveForwardWithoutStop(byte port, byte speed)* – this function keeps the actuators running forward, if invoked again while still running, just update the speed value in the actuators. This is used with positional tracking fuzzy logic.
- *liftPlatform(byte port, byte degrees, byte speed)* – this function lift the robotic head by x degrees and with y actuator speed.

As an example we can observe, in Figure 4.4, how the functionality described for function *forwardByRotation90(byte port, byte speed)* is implemented with bytecode.



## Prototype Development

```
public void forwardByRotating90(byte port, byte speed) {
    byte[] cmd = new byte[20];
    cmd[0] = (byte)0x12;
    cmd[1] = (byte)0x00;
    cmd[2] = (byte)0x00;
    cmd[3] = (byte)0x00;
    cmd[4] = (byte)0x80;
    cmd[5] = (byte)0x00;
    cmd[6] = (byte)0x00;
    cmd[7] = (byte)0xAE;
    cmd[8] = (byte)0x00;
    cmd[9] = port;
    cmd[10] = (byte)0x81;
    cmd[11] = speed;
    cmd[12] = (byte)0x1E; // Step 1: 30 degree
    cmd[13] = (byte)0x82;

    cmd[14] = (byte)0x1E; // Step 2: 30 degree
    cmd[15] = (byte)0x00;

    cmd[16] = (byte)0x82;
    cmd[17] = (byte)0x1E; // Step 3: 30 degree
    cmd[18] = (byte)0x00;
    cmd[19] = (byte)0x00; // without instant brake

    os.write( cmd );
}
```

Figure 4.4: Bytecode to move robot forward by rotating 90 degrees

Note that in Figure 4.4 the byte 19 is 0x00, what means that the actuator will not stop abruptly. This command is making the robot move forward by rotating both actuators 90 degrees, the first 30 degrees gaining speed, then 30 degrees at full speed and then, the last ones, losing speed.

In Figure 4.5 we can see the final robot, with images taken from a test session made during the development phase.

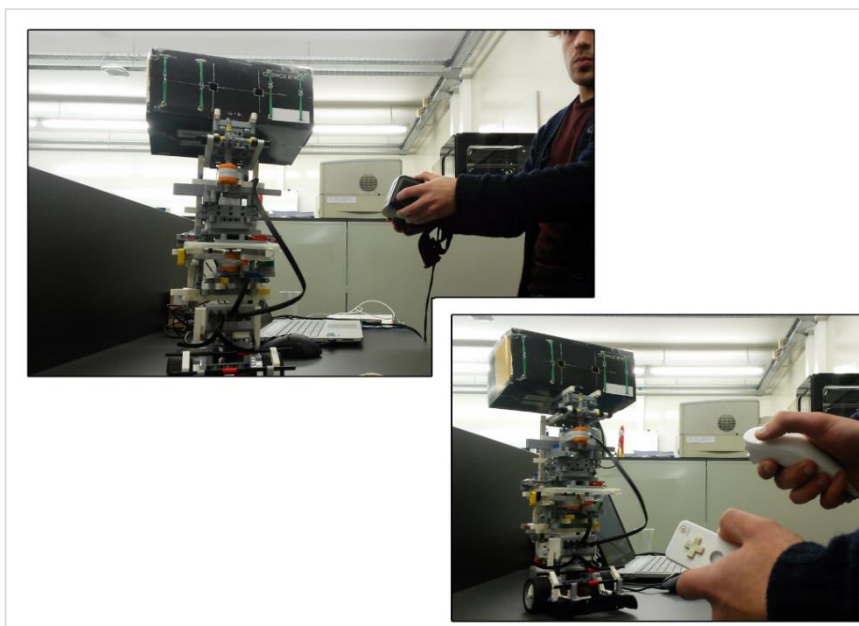


Figure 4.5: Robot Development

## Prototype Development

A class *BluetoothService* was developed and does all the work for setting up and managing Bluetooth connections with other devices. It has a thread that listens for incoming connections, a thread for the connection with a device and another for performing data transmissions when connected.

*BluetoothService* class implements *BluetoothSocket*<sup>31</sup> and *BluetoothDevice*<sup>32</sup>, both from Android SDK, representing the connection made and device connected, respectively. *OutputStream* and *InputStream*, both from *java.io*<sup>33</sup>, are used to exchange data with the device (robot). In Figure 4.4 *os.write(cmd)* is calling the instance of *OutputStream*.

---

<sup>31</sup> <http://developer.android.com/reference/android/bluetooth/BluetoothSocket.html>

<sup>32</sup> <http://developer.android.com/reference/android/bluetooth/BluetoothDevice.html>

<sup>33</sup> <http://developer.android.com/reference/java/io/package-summary.html>

## Experimental Procedure and Results

## **Chapter 5**

# **Experimental Procedure and Results**

In this chapter the experimental procedure and the results obtained are presented. This experiment had two major focus, the first was to evaluate if stereoscopic vision can have an impact in telepresence robots and the second was to test the robustness of the platform created, using the prototype.

### **5.1 Experimental Procedure**

In this Section we describe the developed experiment, where we examined the operator's performance and experience while using the immersive telerobotic platform. Each operator performed equivalent tasks, one with stereoscopic vision and the other without the stereo. With this approach we have been able to understand the impact and contribution of stereoscopic vision and HMDs in the teleoperation of the robot.

After performing the task the operators provided feedback about the system in general.

### 5.1.1 Prototype Equipment

The EV3 robot developed in the prototype, seen in Chapter 4, was used in this experiment, Figure 5.1.

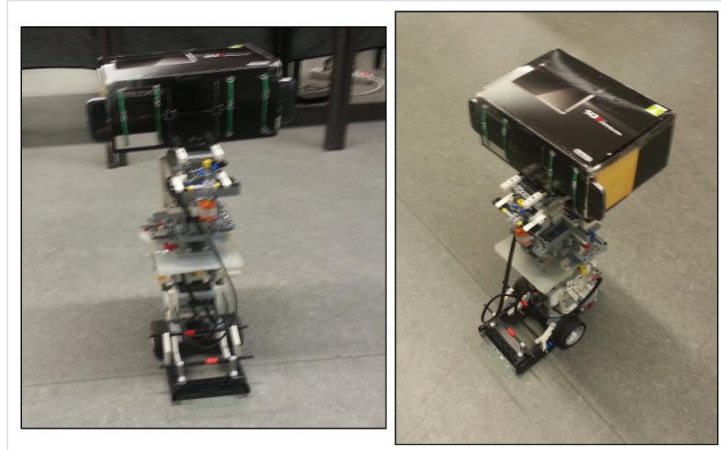


Figure 5.1: EV3 Robot

For the stereo vision we are using two Samsung Galaxy S4 Mini<sup>34</sup> smartphones. In this experiment we used a resolution of 640 x 480 pixels (per eye/smartphone) with a framerate of 30fps. With this configuration we were able to experience an average global delay of 60 milliseconds.

In Figure 5.2 we can see the operator's equipment.



Figure 5.2: Experimental Procedure – Operator Equipment

The participants used an Oculus Rift DK2 v0.4.2 and Wiimote with Wii Nunchuk, as shown in top left of Figure 5.2, below is the router used, a GO-RT-N300 Wireless N300 Easy Router, to establish the WLAN and, finally, at right the laptop used to run the server and web application, an Asus N550JK-CN102H.

<sup>34</sup> <http://www.samsung.com/uk/consumer/mobile-devices/smartphones/android/GT-I9195ZKABTU>

### 5.1.1.1 General Controls

The robotic head has a range of 180 degrees in axial rotation, and 20 degrees in the lifting and lowering movement.

Positional tracking maps body leaning movements to robot back and forth movements and also robot rotation. Following we have a list of positional controls.

- **Lean Forward:** The robot moves forward with speed depending on the degree of slope
- **Lean Backward:** The robot moves backward with speed depending on the degree of slope
- **Lean Left:** The robot rotates around itself to the left, with the speed depending on the degree of inclination
- **Lean Right:** The robot rotates around itself to the right, with the speed depending on the degree of inclination

Head tracking maps head rotation, flexion and extension to the robotic head rotation, lowering and lifting movements. Following we have a list of head tracking controls.

- **Head Flexion (face downwards):** The robotic head will lower the cameras down to a maximum amplitude of 20 degrees and a threshold of 1 degree
- **Head Extension:** The robotic head will lift the cameras to a maximum amplitude of 20 degrees and a threshold of 1 degree
- **Head Rotation Left:** The Robotic Head will rotate the cameras to the left, with a threshold of 1 degree
- **Head Rotation Right:** The Robotic Head will rotate the cameras to the right, with a threshold of 1 degree

The gamepad configuration:

- **Analog Wii Nunchuk:** used to control the robot, moving forward/backward and rotating left/right
- **Wiimote's D-Pad:** used to control the robotic head
- **Wiimote's B Button:** enable/disable Head and Positional tracking

### 5.1.2 Scenario and Instructions

Our main focus was to test the impact that stereoscopic vision can have in telerobotics, but we also tried to evaluate the head tracking solution, comparing it with a classic robotic head manipulation through a D-Pad. Our tests focused on:

## Experimental Procedure and Results

- **Visual Search and Object Identification:** in a polluted or camouflaged environment try to identify an item
- **Distances Between Objects:** with the robot fixed try to identify the distance between objects and which one is closest
- **Head Tracking Control:** try to follow a bouncing ball controlling the robotic head

### 5.1.2.1 Types of Controls

For this experiment we prepared a visual search task with three major points of interest. We had three types of controls:

- **MHT (Mono Head Tracking):** Mono video on HMD, with Nunchuk to control the robotic body
- **SHT (Stereo Head Tracking):** Stereo video on HMD, with Nunchuk to control the robotic body
- **SWM (Stereo with Wiimote):** Stereo video on HMD, with Nunchuk to control the robotic head and Wiimote's D-Pad to control the robotic head.

### 5.1.2.2 Reconnaissance Course

The operator is brought to a control station, and asked to operate the robot in a remote environment, of which the operator has no prior knowledge, and cannot see due to visual barriers. As we can see in Figure 5.3, three tests are waiting in the remote environment.

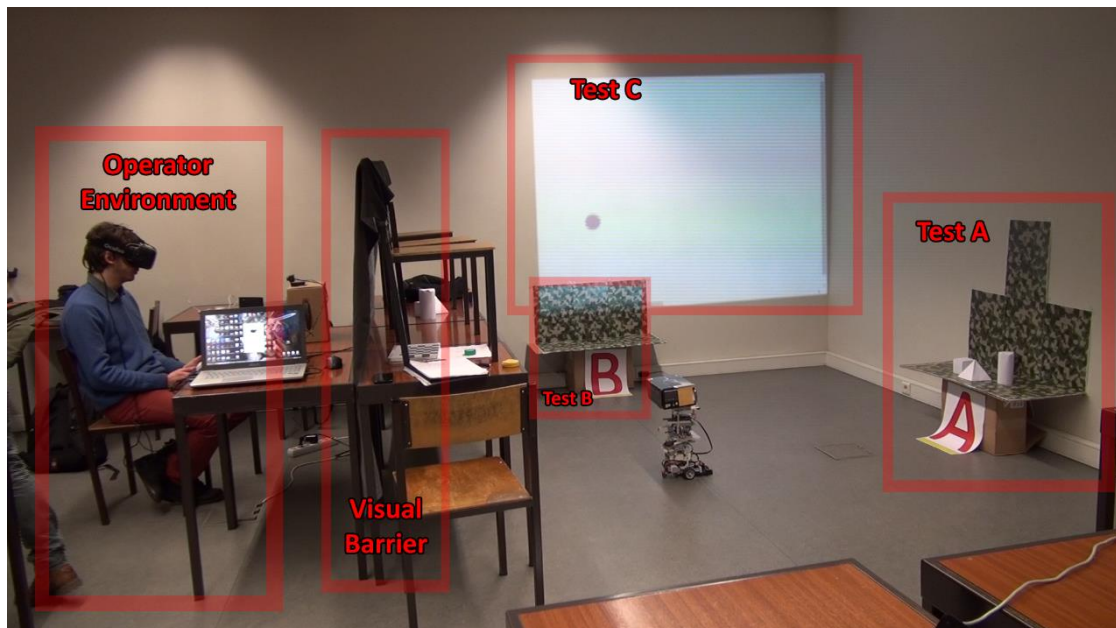


Figure 5.3: Reconnaissance course

## Experimental Procedure and Results

In Test A the objective was to evaluate the impact of stereoscopic vision in the identification of distances between objects, we had three white objects (a cube, a parallelepiped and a pyramid) and a camouflaged object (a cube), a total of four objects. They were arranged in two different layouts (A.1 and A.2) as we can see in Figure 5.4.



Figure 5.4: Test A – Both layouts

Both of them, A.1 and A.2, were randomly ordered in terms of distribution among operators and Stereo/Mono variant of the test.

In Table 5.1 the distribution order of these tests is shown per Participant.

Participant No.	Order of Test A Layouts
1, 3, 5, 7, 8, 9	A.1 (Mono) – A.2 (Stereo)
2, 4, 6, 10	A.2 (Mono) – A.1 (Stereo)

Table 5.1: Assignment of participants to layout setups

In Test A two questions were asked to each participant, in order to evaluate the accuracy of what they were seeing. This test was made with two types of controls, namely: MHT and SHT.

In Test B we had a visual search task and tried to evaluate the object identification accuracy in a polluted or camouflaged environment. When the operator could not see the camouflaged object, he was asked to approach until he get a glimpse of something, then the distance at which he could



## Experimental Procedure and Results

observe something was measured and collected. Figure 5.5 shows more in depth what the environment in Test B had camouflaged.

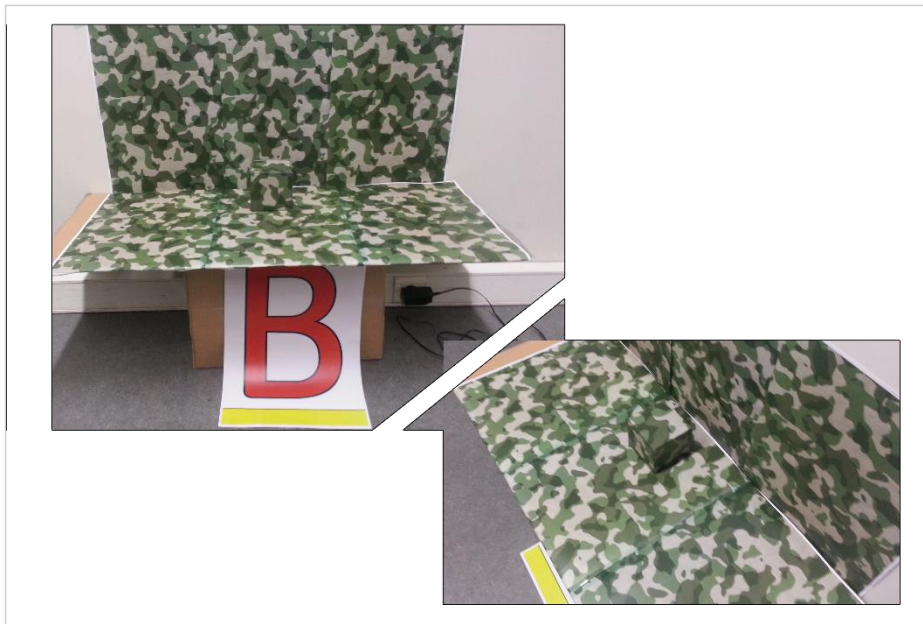


Figure 5.5: Test B – Camouflaged Cube

Test B was also made with two types of controls, being: MHT and SHT.

In Test C, regarding the head tracking module, we asked the participants to follow a bouncing ball with a manual control (with D-Pad of the Wiimote) and then try again with head tracking. Our objective was to get some feedback about the comparison of both.

We now present the questions asked and the order in which the tests were performed.

- **Test A:** In this test two questions are asked to the operator:
  - “How many geometrical figures can you see?”
  - “Sort them, in order of distance to the camera, from the nearest to the farthest.”
- **Test B:** After Test A, the operator is asked to rotate the robot by 90° to the left. Then one question is asked:
  - “How many geometrical figures can you see?”
    - However, if the operator is not sure, he will be asked to move as close as possible to scene, until he is sure of what he is seeing.
  - After doing both tests (A and B), the user is asked to repeat them both with another control type. If the first test was with MHT, the user switches to SHT and vice versa.
- **Test C:** Finally the user is asked to perform 2 tasks:
  - Follow the bouncing ball with the robotic head, but only with the Wiimote’s D-Pad, for 30 seconds.

## Experimental Procedure and Results

- Follow the bouncing ball with the robotic head, but only with the Head Tracking, for 30 seconds.

In the end of this reconnaissance course all the operators were asked to fill a survey about what they experienced, and also asked to leave testimony with tips, advice or reviews.

### 5.1.3 Participants

As it can be observed from Figure 5.6 the majority of the participants were under the age of twenty-three.

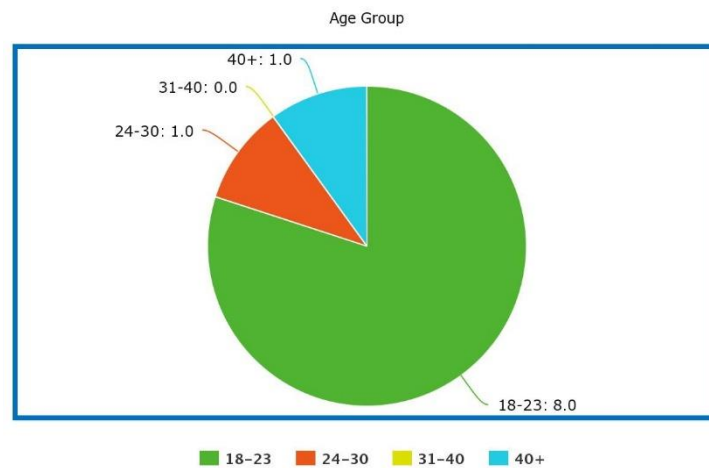


Figure 5.6: Age distribution

We tried to reach different areas of training in order to avoid potential users of this kind of systems, these distribution can be observed in Figure 5.7.

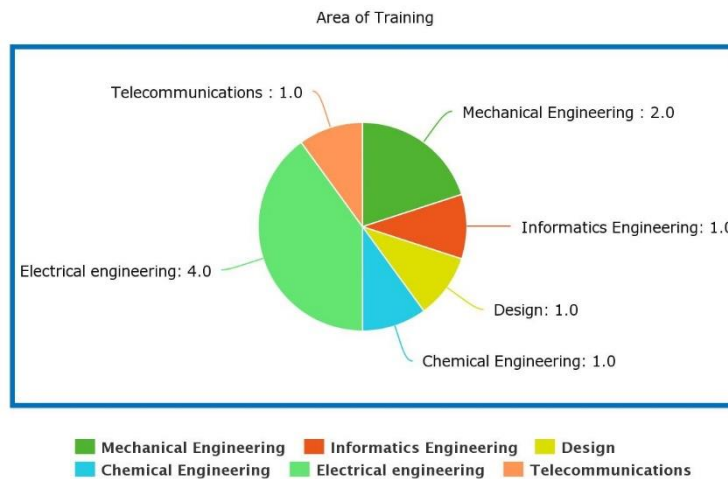


Figure 5.7: Area of Training distribution

### 5.1.4 Measured Results

The measures include performance, easy of use, and operator evaluations.

Performance measures include:

- The number of correct identified figures in Test A
  - Comparison between stereo and mono vision
- Number of correct order distances in Test A
  - Comparison between stereo and mono vision
- Distance needed with stereo and mono vision to spot the camouflaged object

#### 5.1.4.1 Test A

In this test we had three white geometrical figures, as observed in Figure 5.4, and a fourth one camouflaged in the scene. Should be noted that no one observed the same arrangement of figures two times. In Table 5.2 we can see the obtained results.

	<b>Mono Vision</b>	<b>Stereo Vision</b>
<b>1st Question</b> (How many figures?)	no one seen the camouflaged figure 0%	3 participants in 10 were able to see the 4 figures 30%
<b>2nd Question</b> (Distance order?)	correct answers: 4 40%	Correct answers: 10 100%

Table 5.2: Test A results

#### 5.1.4.2 Test B

In this test we had a camouflaged figure alone in a scene. At an initial distance of 170 centimeters from the robot.

	<b>Mono Vision</b>	<b>Stereo Vision</b>
<b>1st Question</b> (How many figures?)	2 participants have seen the figure at the initial distance 20%	10 participants have seen the figure at the initial distance 100%
<b>2nd Question</b> (Move closer)	3 participants were unable to see anything Average: 67.2cm	10 participants have seen the figure at the initial distance Average: 170cm

Table 5.3: Test B results

Something to be noted:

- With mono vision 3 participants were unable to see anything in Test A, even with the robot in front of the figure.
- With stereo vision no one had the need to move the robot, all participants immediately spotted the figure.

## Experimental Procedure and Results

- With stereo vision the ability to identify a hard to spot object, or one that becomes hard because of the surrounding environment, increases in, at least (since our initial position was 170cm), an amount of 60.5%.

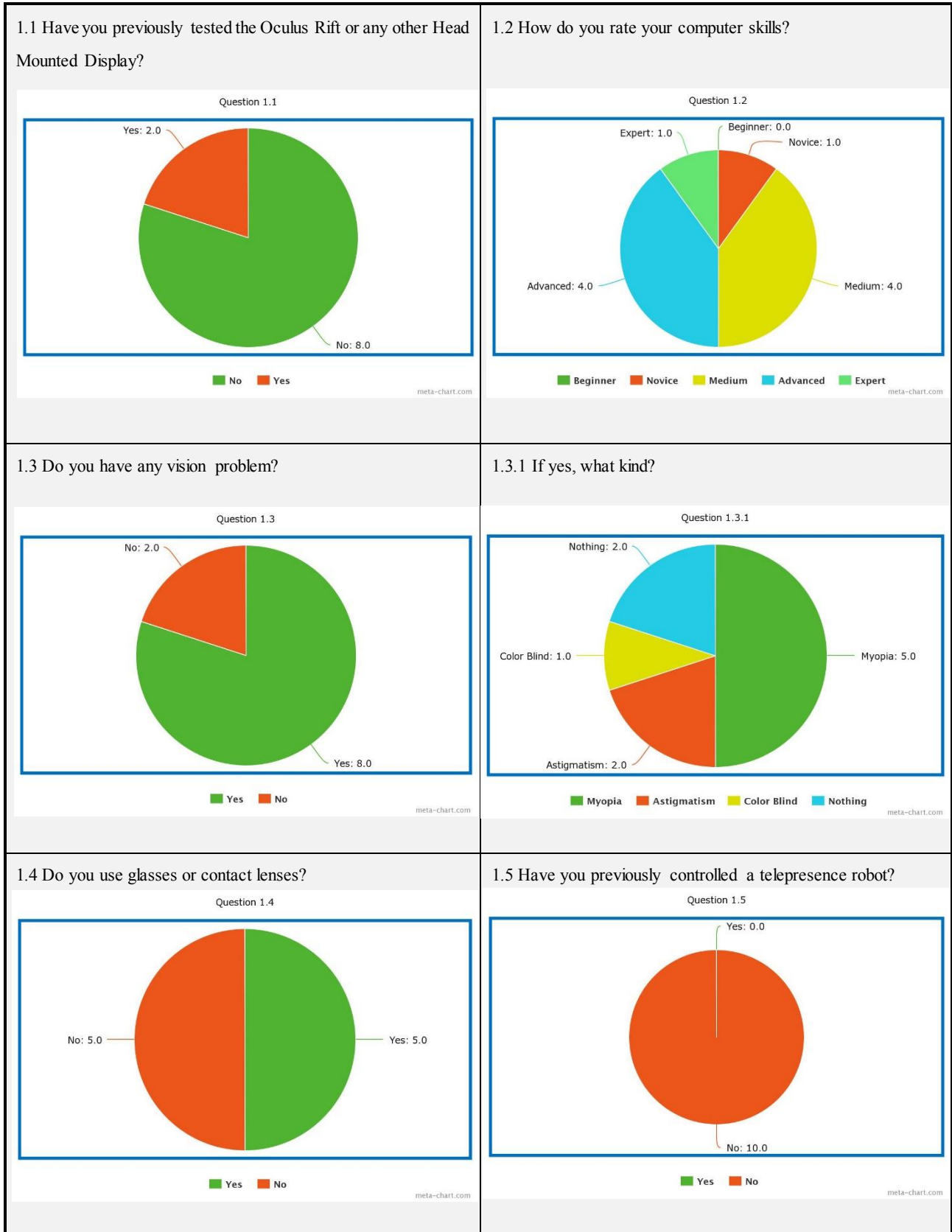
### **5.1.5 Survey Presented**

The survey, in Appendix A, consisted of four parts, the first three directly related to the experienced tests and the last one with a more global approach. The objective of the survey consisted in trying to understand whether the participant's results corresponded to their perception. We also try to get feedback about the prototype and overall experience feedback.

#### **5.1.5.1 Survey Results**

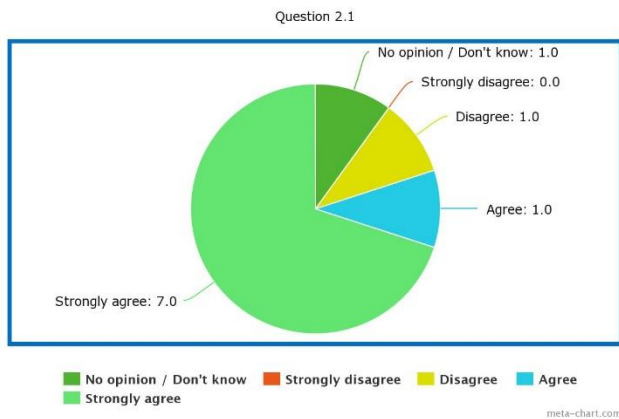
In this Section we present all the results obtained from the survey and display its information in Table 5.4.

## Experimental Procedure and Results

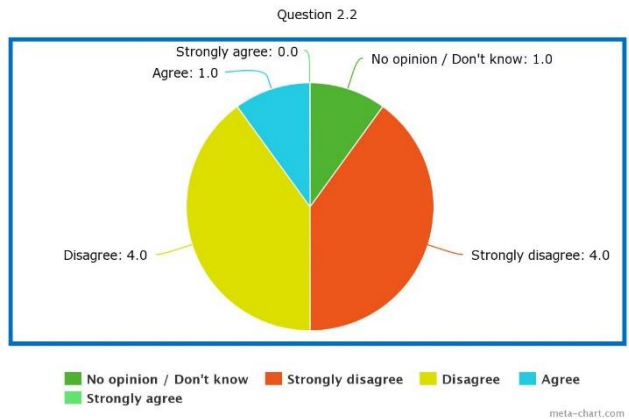


## Experimental Procedure and Results

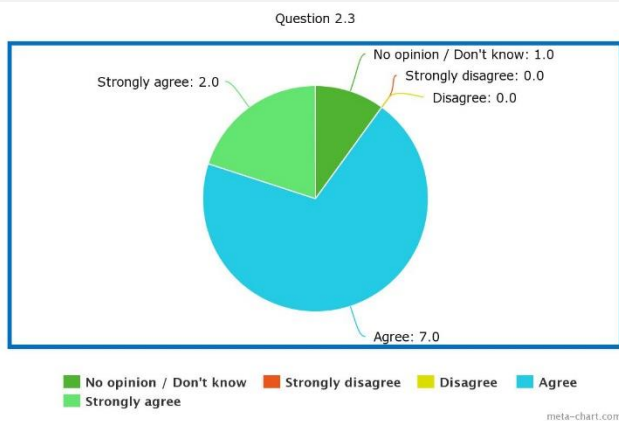
2.1 I've noticed that stereoscopic vision has increased my depth perception.



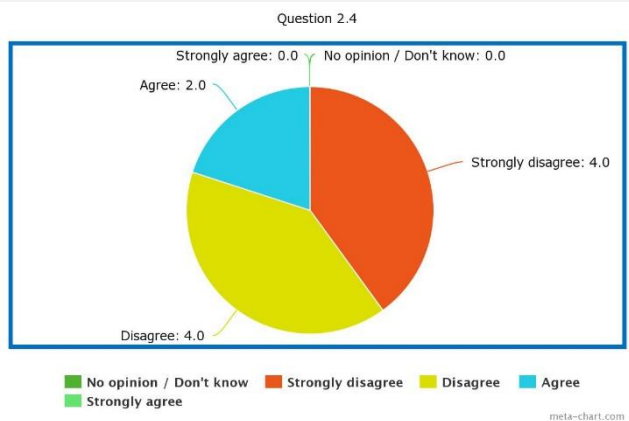
2.2 It was easy to figure out the correct order of objects with mono vision.



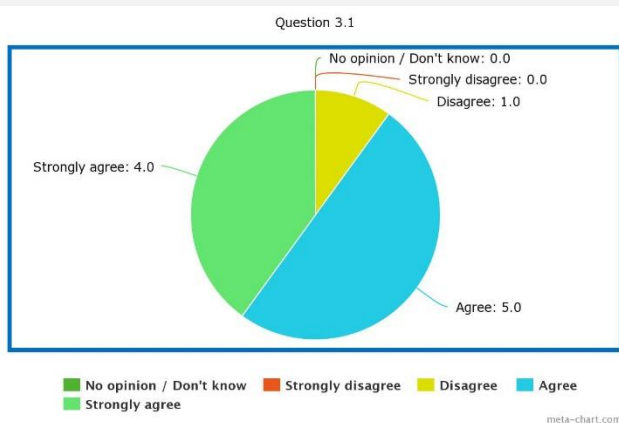
2.3 It was easy to figure out the correct order of objects with stereoscopic vision.



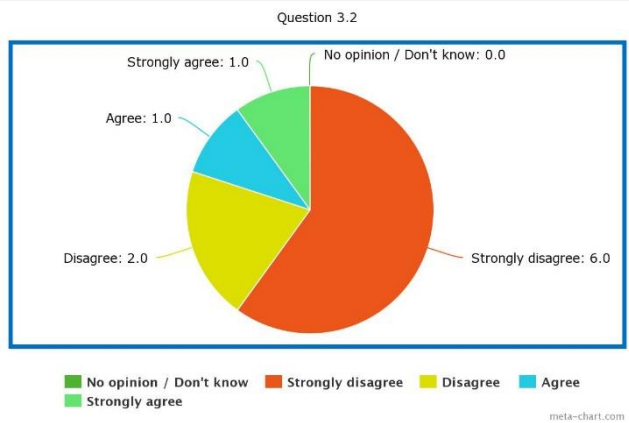
2.4 With stereoscopic vision I felt competition between eyes, resulting in double vision.



3.1 I had no problem identifying the camouflaged object with stereoscopic vision.

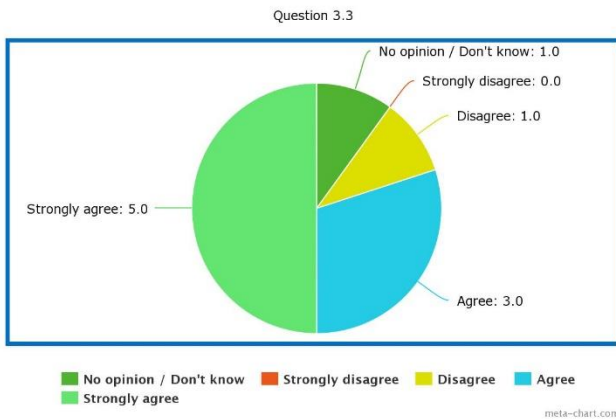


3.2 I had no problem identifying the camouflaged object with mono vision.

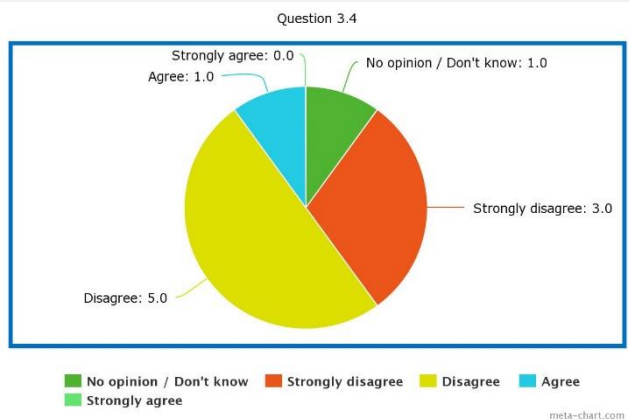


## Experimental Procedure and Results

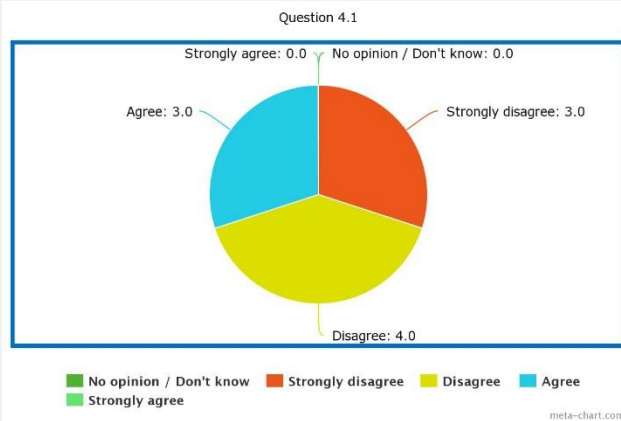
3.3 I've noticed that stereoscopic vision has increased my depth perception.



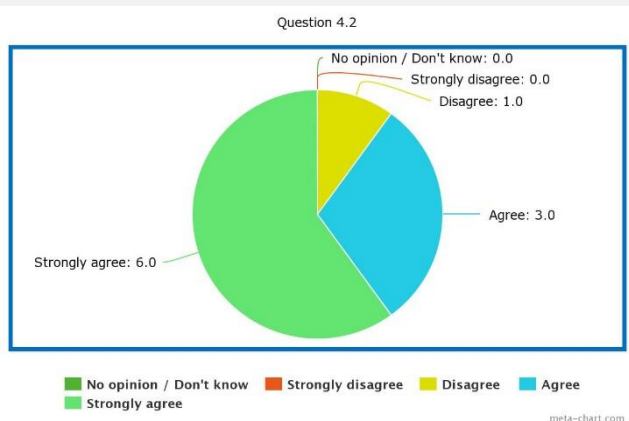
3.4 With stereoscopic vision I felt competition between eyes, resulting in double vision.



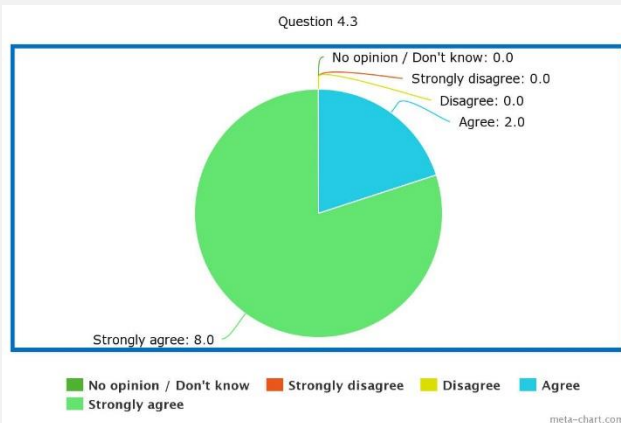
4.1 The Wiimote was extremely helpful in the overall maneuvering of the robotic head.



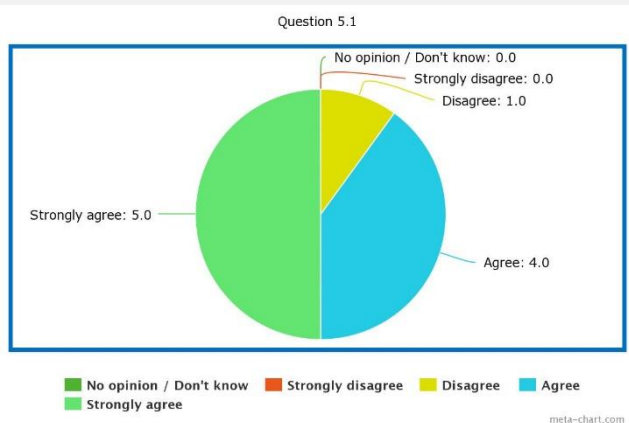
4.2 The head tracking was extremely helpful in the overall maneuvering of the robotic head.



4.3 I think that a head tracking module would be more intuitive than a Wiimote/Gamepad control in a real world situation.



5.1 I would prefer a stereoscopic vision solution in a real world search and rescue (SAR) situation.



## Experimental Procedure and Results

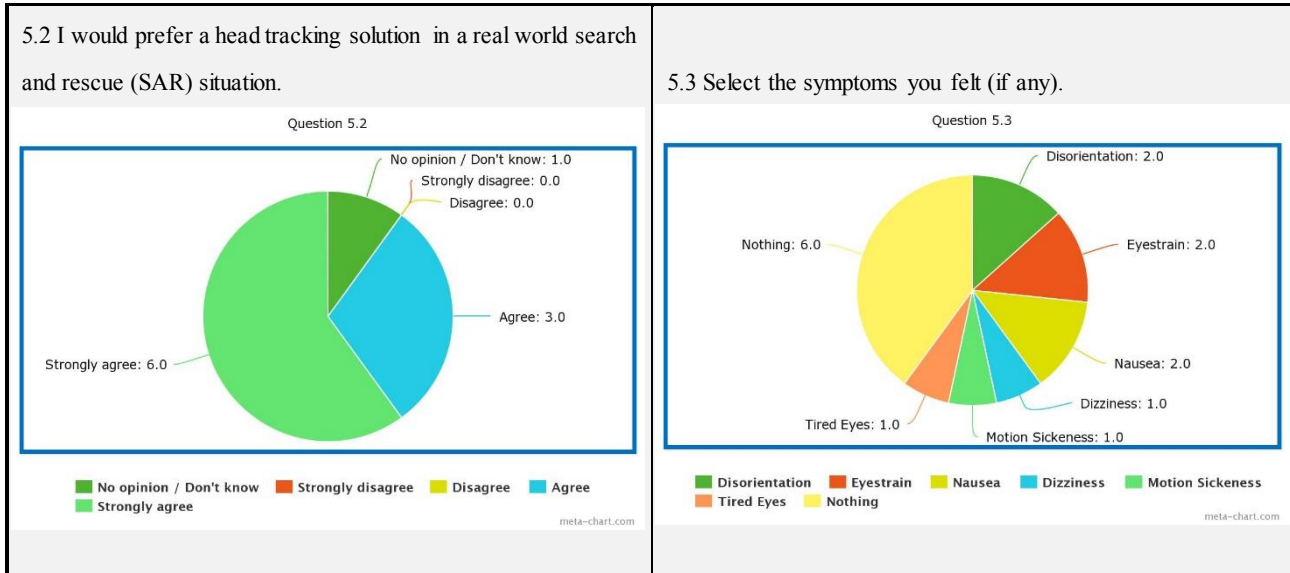


Table 5.4: Survey results

### 5.1.5.2 Comments and Testimonies

General comments by the participants, about the system, included descriptions such as *"going the good way"*, *"intuitive"*, *"clear vision"* and *"a natural depth perception"*.

A few less positive comments were made during the tests about the flexion and extension range of the robotic head or, by other words, the vertical amplitude of the robotic head. But this was a known limitation induced by the hardware used to develop the prototype, seen in Chapter 4, and by no means it interferes with the success achieved in the development of the platform.

The written testimonies were also very positive, and quoting a full testimonial by a more experienced participant:

*"This project has a magnitude comparable to an extremely high cost project developed at Instituto Superior Técnico (IST), with an hardware solution. In TrekProbe, with 3D mode vision, I've not been able to see a difference, in terms of perception of things in 3D, when compared with the IST equipment solution."*

### 5.1.6 Experiment Conclusions

As we seen in Test A, in a polluted environment is impossible to see certain objects, while recurring to stereo vision we see an increase from 0% to 30%. The distances between objects are much more easily to spot with stereo vision, being 60.5% higher than without it.



## Experimental Procedure and Results

In Test B, regarding Visual Search and Object Identification in a polluted or camouflaged environment, stereo vision has an improvement of 80% over mono vision. The distance that it is possible to identify an object also increases by 60%.

In Test C, regarding head tracking module, when asked about what would be better in a real world situation between a gamepad/joystick and head tracking, 80% of the participants were in favor of the head tracking.

Generally speaking, the experiment revealed itself a success, where all of the participants stated that the stereo vision is a big leap from mono and classic implementations. The results of the experiment were a success, and it was also possible to imply, with the survey presented, that not only the results were good but the participants were also aware of this.

## Conclusion

## Chapter 6

# Conclusion

Our main motivation came from the fact that state-of-the-art solutions in telerobotics have a lack of use of immersive solutions. A secondary motivation came from the fact that currently researchers end up wasting too much time and money building specific solutions each time they plan to test a specific case, which influenced the creation of an extensible framework. The expansion of technologies in areas like mobile, immersive technologies and web-based solutions led us to a low-cost platform based on a real-time software based solution.

Despite the inclusion of different types of technologies, both from software and hardware, which greatly increased the complexity of the project, the platform development went extremely well, achieving all the proposed objectives.

The prototype developed had two objectives, test the platform developed and evaluate the impact that immersive components can have in telerobotics. The platform extension, with the chosen robot was also a success, despite some hardware limitations that we already knew from the start that we were going to find.

Due to logistic limitations the sample size of participants in the experimental procedure was somewhat limited, but was enough to imply that our initial claim was in the right direction. The experiment itself went extremely well with the results exceeding the expectations. All of the participants stated that the stereo vision is a big leap from mono vision and the more classic implementations.

It was possible to show that on a polluted environment is really hard to see objects that easily camouflage in common areas, but with stereo vision we see an increase from 0% (with non-stereoscopic solution) to 30% in the global identification of this kind of objects. And when in a

## Conclusion

camouflaged environment not polluted by other objects, this gain increases even more, to a value of 80%.

The distances between objects are also much more easily spotted with stereo vision, being 60% higher than without it.

Regarding the range at which the operator is able to identify a scene with some security of what he's able to see, stereoscopic vision increases this range, at least, by 60.5%.

The practical component was also accomplished with success, as seen in Chapter 4 and proved in Section 5.1.5. General testimonies by the participants about the platform were positive and included descriptions such as *"going the good way"*, *"intuitive"*, *"clear vision"* and *"a natural depth perception"*.

But not only of positive statements lives our platform. We have a modular platform that can be extended with multiple kinds of robots. These robots being controlled with gamepad/keyboard, head and positional tracking through Wi-Fi network. We achieved stereoscopic vision with everyday accessible hardware (two Android devices) using software synchronization and rectification. Currently we are able to achieve a maximum resolution of 1280x720 pixels, with a framerate of 30fps, 40-50ms of delay and a packet loss below 0.9%.

Regarding future work, we believe that some modules of this platform can, and should, get some refactoring. Due to the nature of this project, involving many and different technologies, sometimes we lacked time to perform a process of refactoring with care.

The communication API between the mobile and web applications can be extended with some additional capabilities.

Maybe developing a new prototype, with a more robust robot, could be really helpful in order to show features that currently are hard to see or even think about.

# References

- [TelePrs] Marvin Minsky “Telepresence”, OMNI magazine. June 1980
- [RT12] Linda R. Elliott, Chris Jansen, Elizabeth S. Redden, Rodger A. Pettitt. “Robotic Telepresence: Perception, Performance, and User Experience”. 2012
- [PlatRb13] H.M.; Mouser, C.J.; Ye Gu; Weihua Sheng; Honarvar, S.; Tingting Chen, “An open platform telepresence robot with natural human interface,” Cyber Technology in Automation, Control and Intelligent Systems (CYBER), 2013 IEEE 3<sup>rd</sup> Annual International Conference. May 2013
- [BSK12b] Kechavarzi, B.D.; Sabanovic, S.; Weisman, K., “Evaluation of control factors affecting the operator’s immersion and performance in robotic teleoperation” RO-MAN, 2012 IEEE. Sept. 2012
- [RMT13] Kechavarzi, B.D.; Sabanovic, S.; Weisman, K., “A Review of Mobile Robotic Telepresence”. 2013
- [StDisp09] Marc Lambooi, Marten Fortuin, Ingrid Heynderickx, and Wijnand IJsselsteijn. Visual discomfort and visual fatigue of stereoscopic displays: a review. *Journal of Imaging Science and Technology*, 53(3):30201– 1, 2009.
- [LCStVi07] C. Murphy, D. Lindquist, A. M. Rynning, T. Cecil, S. Leavitt, and M. L. Chang, “Low-Cost Stereo Vision on an FPGA,” in *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, April 2007, pp. 333–334.
- [FPGA06] D. K. Masrani and W. J. MacLean, “A Real-Time Large Disparity Range Stereo-System using FPGAs,” in *Proceedings of the IEEE International Conference on Computer Vision Systems*, 2006, pp. 42–51.
- [FPGA10] S. Jin, J. U. Cho, X. D. Pham, K. M. Lee, S.-K. Park, and J. W. J. Munsang Kim, “FPGA Design and Implementation of a Real-Time Stereo Vision System,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 1, pp. 15–26, January 2010.
- [EpiLR96] Papadimitriou, V.; Dennis, T.J., “Epipolar line estimation and rectification for stereo image pairs,” *Image Processing, IEEE Transactions on*, vol.5, no.4, pp.672,676, Apr 1996

- [ImgRect] Olivier Faugeras, “Three-Dimensional Computer Vision: A Geometric Viewpoint”, MIT Press. 1993
- [CamCal04] Z. Zhang, G. Medioni and S.B. Kang, (2004) “Camera Calibration”, Emerging Topics in Computer Vision, Prentice Hall Professional Technical Reference, Ch. 2, pp.443.
- [ROSR13b] Lee Garber, “Robot OS: A New Day for Robot Design”. December, 2013.
- [ARLM13] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, Denis Fisseler. “Analysis of the accuracy and robustness of the leap motion controller”. 2013
- [ARLM14] Jože Guna, Grega Jakus, Matevž Pogačnik, Sašo Tomažič, Jaka Sodnik. “An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking”. 2014
- [DMTF03] Al-Mouhamed, M.; Toker, O.; Iqbal, A, “Design of a multi-threaded distributed telerobotic framework”. Electronics, Circuits and Systems, 2003. Dec. 2003
- [ITAH12] Terrile, R.J.; Noraky, J., “Immersive telepresence as an alternative for human exploration”, Aerospace Conference, 2012 IEEE. 2012
- [MeBot10] Adalgeirsson, S.O.; Breazeal, C., “MeBot: A robotic platform for socially embodied telepresence,” Human-Robot Interaction (HRI), 2010 5<sup>th</sup> ACM/IEEE International Conference. March 2010
- [3DSAR09] Martins, H.; Ventura, R., “Immersive 3-D teleoperation of a search and rescue robot using a head-mounted display”, Emerging Technologies & Factory Automation, 2009. Sept. 2009
- [EyeRbt08] Breejen, E. den · Jansen, C., “EyeRobot TBI unmanned TelePresence reconnaissance mission”. 2008
- [LMP14] Leap Motion Controller. 10 Julho, 2014. <https://www.leapmotion.com>
- [HSRem08] Sangyoon Lee, Gerard Jounghyun Kim, “Effects of haptic feedback, stereoscopy, and image resolution on performance and presence in remote navigation”. 2008
- [AdvR07] Manuel Ferre; Martin Buss; Rafael Aracil; Claudio Melchiorri; Carlos Balaguer, “Introduction to Advances in Telerobotics”, Springer Tracts in Advanced Robotics Volume 31. 2007
- [SocRbt11] Coradeschi, S; Loutfi, A; Kristoffersson, A; Cortellessa, G; Eklundh, K S, “Social robotic telepresence”, Human-Robot Interaction (HRI), 2011 6<sup>th</sup> ACM/IEEE International Conference. 2011
- [ROSAbt] ROS - Robot Operating System. ROS - About. Junho, 2014. <http://www.ros.org/about-ros/> .

- [AtkRb13] Information Week, Attack of the Telepresence Robots, Novembre 2013. <http://www.informationweek.com/applications/attack-of-the-telepresence-robots!/d/d-id/1108137>
- [FuzzL10] Vickers, S.; Coupland, S., “Soft computing head tracking interaction for telerobotic control,” Computational Intelligence (UKCI), 2010 UK Workshop on , vol., no., pp.1,6, 8-10 Sept. 2010
- [WRTC14] Cisco Inc. “WebRTC – Bringing Real Time Communications to the Web Natively.” [Online]. Available:<http://blogs.cisco.com/openatcisco/webrtc-bringing-real-time-communications-to-the-web-natively/>. [Accessed: 11-September-2014].
- [RelRR14] Gallastegi, Akaitz, “Web-based Real-Time Communication for Rescue Robots.”, Linköping University, Software and Systems. 2014
- [WebRTC] “WebRTC 1.0: Real-time Communication Between Browsers.” [Online]. Available: <http://www.w3.org/TR/webrtc/>. Accessed, 29, May 2014.
- [Node14] “Why The Hell Would I Use Node.js? A Case-by-Case Tutorial” [Online]. Available: <http://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js>. Accessed, 20, August 2014.
- [Distort14] “Distortion (optics)” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 04 October 2014. Web. 10 October 2014. <[http://en.wikipedia.org/wiki/Distortion\\_%28optics%29](http://en.wikipedia.org/wiki/Distortion_%28optics%29)>.
- [AndTools] “Android SDK: Tools Help” [Online]. Available: <http://developer.android.com/tools/help/index.html>. Accessed, 25, August 2014.
- [H5Rocks] “WebRTC in the real world: STUN, TURN and signaling” [Online]. Available: <http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/>. Accessed, 20, September 2014.
- [VisualF09] Marc Lambooi, Marten Fortuin, Ingrid Heynderickx, and Wijnand IJsselsteijn. “Visual discomfort and visual fatigue of stereoscopic displays: a review”. Journal of Imaging Science and Technology, 53(3):30201– 1, 2009.
- [motionJS] “JavaScript motion detection” [Online]. Available: <http://www.adobe.com/devnet/archive/html5/articles/javascript-motion-detection.html>. Accessed, 18, October 2014.
- [iRobot] “iRobot Corporation” [Online]. Available: <http://www.irobot.com/>. Accessed, 10, July 2014.
- [epipGeom] “OpenCV: Epipolar Geometry” [Online]. Available: [http://docs.opencv.org/trunk/doc/py\\_tutorials/py\\_calib3d/py\\_epipolar\\_geometry/py\\_epipolar\\_geometry.html](http://docs.opencv.org/trunk/doc/py_tutorials/py_calib3d/py_epipolar_geometry/py_epipolar_geometry.html). Accessed, 18, September 2014.

- [LegoProg] Lego Group, "LEGO® MINDSTORMS® EV3 Communication Developer Kit", provided by lego. 2014
- [kinect12] Khoshelham, Kourosh; Elberink, Sander Oude. 2012. "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications." *Sensors* 12, no. 2: 1437-1454



Appendix A

Experiment Survey



Universidade do Porto  
**FEUP** Faculdade de Engenharia

TrekProbe requests your help. Please complete the following survey based on the prototype you just tested. Thank you for your time.

Project Name:

TrekProbe: An Immersive Telerobotic Modular Framework using stereoscopic HMD's

Date:

January 23, 2015

Name: \_\_\_\_\_ Age: \_\_\_\_\_

Course and Academic Year: \_\_\_\_\_

E-mail: \_\_\_\_\_

**1 Previous Knowledge:**

**1.1 Have you previously tested the Oculus Rift or any other Head Mounted Display?**

Yes  No

**1.2 How do you rate your computer skills?**

Beginner  Novice  Medium  Advanced  Expert

**1.3 Do you have any vision problem?**

Yes  No

1.3.1 If yes, what kind:

---

**1.4 Do you use glasses or contact lenses?**

---

Yes       No

**1.5 Have you previously controlled a telepresence robot?**

---

Yes       No

**1.6 How do you rate your computer skills?**

---

Beginner       Novice       Medium       Advanced       Expert

## **2 Related to Test A (Distances Between Objects):**

**2.1 I've noticed that stereoscopic vision has increased my depth perception.**

---

No opinion / Don't know       Strongly disagree       Disagree       Agree       Strongly agree

**2.2 It was easy to figure out the correct order of objects with mono vision.**

---

No opinion / Don't know       Strongly disagree       Disagree       Agree       Strongly agree

**2.3 It was easy to figure out the correct order of objects with stereoscopic vision.**

---

No opinion / Don't know       Strongly disagree       Disagree       Agree       Strongly agree

**2.4 With stereoscopic vision I felt competition between eyes, resulting in double vision.**

---

No opinion / Don't know       Strongly disagree       Disagree       Agree       Strongly agree

### **3 Related to Test B (Visual Search):**

**3.1 I had no problem identifying the camouflaged object with stereoscopic vision.**

---

No opinion / Don't know     Strongly disagree     Disagree     Agree     Strongly agree

**3.2 I had no problem identifying the camouflaged object with mono vision.**

---

No opinion / Don't know     Strongly disagree     Disagree     Agree     Strongly agree

**3.3 I've noticed that stereoscopic vision has increased my depth perception.**

---

No opinion / Don't know     Strongly disagree     Disagree     Agree     Strongly agree

**3.4 With stereoscopic vision I felt competition between eyes, resulting in double vision.**

---

No opinion / Don't know     Strongly disagree     Disagree     Agree     Strongly agree

### **4 Related to Test C (Head Tracking):**

**4.1 The Wiimote was extremely helpful in the overall maneuvering of the robotic head.**

---

No opinion / Don't know     Strongly disagree     Disagree     Agree     Strongly agree

**4.2 The head tracking was extremely helpful in the overall maneuvering of the robotic head.**

---

No opinion / Don't know     Strongly disagree     Disagree     Agree     Strongly agree

**4.3 I think that a head tracking module would be more intuitive than a Wiimote/Gamepad control in a real world situation.**

- 
- No opinion / Don't know     Strongly disagree     Disagree     Agree     Strongly agree

**5. Global:**

**5.1 I would prefer a stereoscopic vision solution in a real world search and rescue (SAR) situation.**

- 
- No opinion / Don't know     Strongly disagree     Disagree     Agree     Strongly agree

**5.2 I would prefer a head tracking solution in a real world search and rescue (SAR) situation.**

- 
- No opinion / Don't know     Strongly disagree     Disagree     Agree     Strongly agree

**5.3 Select the symptoms you felt (if any):**

- Eyestrain
- Motion sickness
- Nausea
- Disorientation
- Dizziness

Other: \_\_\_\_\_

*Comments / Testimonial:*

[Add your comments here.]

Thank you very much for taking the time to complete this survey. Your feedback is valued and very much appreciated!

