UNIVERSIDADE D
COIMBRA

Ricardo Filipe Matias

A VR Supported Study on Human-Robot Collaboration

Dissertação no âmbito do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores orientada pelo Professor Doutor Paulo Jorge Carvalho Menezes e apresentada ao Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Outubro de 2021

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

RICARDO FILIPE MATIAS

A VR Supported Study on Human-Robot Collaboration

Thesis submitted to the

University of Coimbra for the degree of

Master in Electrical and Computer Engineering

Supervisors:

Prof. Dr. Paulo Jorge Carvalho Menezes

Coimbra, 2021

This work was developped in collaboration with:

University of Coimbra

UNIVERSIDADE D
COIMBRA

Department of Electrical and Computer Engineering

deer.uc
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA
E DE COMPUTADORES

Institute of Systems and Robotics

INSTITUTE OF SYSTEMS AND ROBOTICS
UNIVERSITY OF COIMBRA

If you can't explain it to a six year old, you don't understand it yourself.

Albert Einstein

Acknowledgments

The author would like to sincerely thank all participants in the pilot study, which took some time of their busy lives to contribute to this work.

He would also like to thank the Institute of Systems and Robotics (ISR) for providing all the necessary equipment and allowing its use outside the laboratory and the ISR team for always being available to help.

The author would also like to give a very special thank you to its family and friends for all the support provided.

Abstract

Robots are gradually moving from environments where they work autonomously and alone to new ones where the operations are done with human presence to cooperate towards the same task. These new types of robots are known as collaborative robots. However, human-robot collaboration is far from being as safe and effective as human-human collaboration. The designation ”collaborative robot” refers in most cases to robots with enough safety measures to not harm the human collaborating with them, which often negatively affects the performance of both robots and humans.

In this work, an application (BaxterVR) was developed to study human-robot collaboration using Virtual Reality and adapted to two distinct scenarios. The first scenario was made in collaboration with ProAction Lab. Its main objective is to record human brain activity while cooperating with a robot in a virtual environment, particularly when the robot makes a mistake. With a deeper understanding of how the human brain activity varies in these situations, new technologies could be developed, such as an emergency stop for the robot using brain signal analysis. The second scenario was used to analyse how distractors affect a human performing a collaborative task with a robot. It was explored in a pilot study where the user's electrodermal activity and heart rate were recorded and studied. At the end of each experiment, the participants filled three questionnaires to evaluate their experience, revealing that the scenario was excellent overall but presented two main issues. The participants also reported feeling more distracted by the event that involved constant and prolonged sound. At the same time, the analysis of electrodermal activity signals collected during the experiments revealed that the events with a more intense initial impact presented the most reactions.

Resumo

Os robôs estão gradualmente a moverem-se de ambientes onde trabalham autonomamente e sozinhos para novos onde as operações são feitas com presença humana para cooperar na mesma tarefa. Estes novos tipos de robôs são conhecidos como robôs colaborativos. No entanto, a colaboração entre humanos e robôs está longe de ser tão segura e eficaz como a colaboração entre humanos. A designação de ”robô colaborativo”refere-se na maioria dos casos a robôs com medidas de segurança suficientes para não magoar o humano, o que muitas vezes afeta negativamente a performance tanto dos robôs como dos humanos.

Neste trabalho, foi desenvolvida uma aplicação (BaxterVR) para estudar colaboração entre humanos e robôs utilizando Realidade Virtual, sendo adaptada para dois cenários distintos. O primeiro cenário foi feito em colaboração com ProAction Lab. O seu principal objetivo é recolher atividade cerebral humana enquanto o humano colabora com um robô num ambiente virtual, particularmente quando o robô comete um erro. Com um melhor conhecimento acerca de como a atividade cerebral humana varia nestas situações, novas tecnologias podem ser desenvolvidas, tal como uma paragem de emergência para o robô através da análise de sinais cerebrais. O segundo cenário foi explorado num estudo piloto onde a atividade eletrodérmica e ritmo cardíaco dos utilizadores são gravados e estudados. No final de cada experiência, os participantes preencheram três questionários para avaliar a sua experiência, revelando que o cenário estava excelente no geral, mas apresentou dois problemas principais. Os participantes revelaram também que se sentiram mais distraídos pelo evento que envolvia um som mais constante e prolongado. Simultaneamente, a analise da atividade eletrodérmica recolhida durante as experiências revelou que o evento com um impacto inicial mais intenso apresentou mais reações.

List of Figures

List of Tables

Contents

1

Introduction

The creation of two systems to study human-robot interaction is presented as the topic of this M.Sc dissertation. The first system is used in collaboration with the neurosciences laboratory ProAction Lab in research to improve the safety of humans when interacting with a robot through a brain-machine interface. The second system is used in a pilot study to evaluate how the participants react to distractions in a simulated workplace and how they feel about the interaction with a robot. This section is meant to introduce the reader to the topics presented, mainly the most important aspects, related work and objectives.

1.1 Study Relevance

Robots are gradually moving from highly controlled environments, where they work autonomously and alone, to new ones where the operations are done with human presence to cooperate towards the same task. These new types of robots are known as collaborative robots [1].

The industrial environments are good examples of this situation, since many tasks are automated, leading to large areas where robots operate alone, however, in many cases, this still needs to be complemented by human manual work [2]. The assembly lines in the automotive and electronic equipment industries are good examples of these situations. Moreover, the repetitive nature of these tasks can cause fatigue and frequently lead to lower back pain situations or spine injuries [3]. Introducing a robot collaborator could be an effective measure to help solve these problems.

Human-robot collaboration is still far from being effective and safe, as is the collabo-

ration between humans. The International Federation of Robotics only allows robots and humans to work together if they do not share the same workspace or if they share it but do not act simultaneously. This is due to a significant flaw in current day robots, the lack of awareness of their surroundings, particularly of their partners' actions and feelings, which leads to the lack of ability to adapt accordingly. Robots in the near future will need to react and adapt to the working partner's presence and actions [4].

Similarly to the case of operators controlling large machines, awareness is a crucial issue for robots, particularly those that need a lot of strength and speed, since one wrong move can seriously injure or even kill someone in its vicinity. We may say that safety results from strong awareness of the surroundings, but even a careful car driver cannot always predict the next move of a distracted pedestrian. Speed limitations in urban areas aim to reduce the number of (fatal) accidents due to unpredictable human behaviours, and the same principle has been applied to collaborative robots. The designation *collaborative robots* refers in most cases to robots whose joint torque or force is reduced to a level in which collision with a human will make it stop while not causing any harm. Other cases are based on turn-taking, frequently enforced by two lateral buttons that the human must press simultaneously to ensure that his/her hands are not inside the robot operating volume while moving.

While being a new and exciting trend, this gathering may affect the performance of both robots and humans. Robots have to operate slower and be less powerful to avoid fatal accidents. Humans may need to adapt to these new situations. This adaptation may require unexpected cognitive and emotional efforts that eventually degrade their performance and physical or mental health. Therefore, it is essential to understand what influence a specific collaborative setup may have in humans, particularly along a workday period and how to improve the safety of the interaction, and consequently, its efficiency.

1.2 Main Objective

The goal of this work is to create a Virtual Reality application, BaxterVR, to study human-robot interaction and use it to create two distinct scenarios. Each scenario consists

of a simulated environment where the user can visualise and interact using a virtual reality-based setup.

The first scenario was made in cooperation with ProAction Lab, a laboratory of the Faculty of Psychology and Educational Sciences of the University of Coimbra, working on research related to Cognitive Neuroscience and Psychology. The main objective is to record human brain activity when cooperating with a robot in a virtual environment, allowing the possibility to study and develop adaptive cobots, particularly studying how the human reacts to errors from the robot. With a deeper understanding of how the human brain activity varies in accordance to errors made by the robot, the safety of an interaction would be substantially improved, allowing for the development of technologies such as an emergency stop for the robot using only the analysis of the brain signals from the cooperating human.

The second scenario is used to analyse how distractors may affect a human performing a collaborative task with a robot. It is explored in a pilot study where the user's reactions to the robot's actions, such as posture and physiological signals, namely, electrodermal activity and electrocardiogram, are measured and studied to evaluate the user's reaction to events inside the simulation. However, there is the question of knowing if the user behaviours in a VR-based environment are different from the ones in a real situation. Although other factors may contribute to the above mismatch, the immersion level and sense of presence are two crucial aspects. The results obtained in a VR-based test will not be accurate if the sense of immersion in that scenario and of presence of its elements is not achieved. For this reason, it is necessary to evaluate these factors as perceived by each user, along with the interactive task under analysis, to enable a prediction of the accuracy of the task-related parameters under estimation. This is commonly done through the application of self-response questionnaires such as UEQ and flow short scale questionnaires and by evaluating the user's reactions, as will be analysed later.

Another essential aspect to take into account is the realism of the robots' movements. To obtain results comparable to real situations, the simulated robot has to behave like a real one in a real environment. Whenever possible, using a controller that mimics or matches as close as possible that of the real robot will guarantee that what concerns the perception

of robot motions by the user will be similar to the actual case.

The use of Virtual Reality to study human-robot collaboration has the clear advantage of providing clean, repeatable and safe support to create counterparts of live experiences [5]. This is perfect for analysing how adequate an environment or an interactive system is for human use. Besides safety, these approaches may have economic advantages as they enable the validation of hypotheses without risking any kind of material losses due to inadequate usage or do any type of predictive analysis of what will be the outcome of the introduction of a change, such as a collaborative robot, in a specific point of the production process.

1.3 Related Work

Human-Robot interaction as a multidisciplinary field emerged in the mid-1990s and early years of 2000. It focuses on understanding the interactions between robots and people and how they can be shaped and improved. Over the years, robots have seen great improvement, with new models constantly making the old ones obsolete with new hardware, cognitive capabilities and more. The vision of this field is to make robots exist in our everyday lives, from helping in the industrial environment and household chores to social robots that can entertain or care for humans. As such, studies on this field focus not just on the short term interactions in the laboratory but also the long-term interactions with systems such as a robotic weight loss coach [6] that helps track calories consumption to combat obesity. The introduction of this robot in the participants' lives had them tracking their calories consumption and exercising almost twice as long, demonstrating the benefits that robots could have if correctly introduced in our lives. This introduction also raises the importance of the robot's appearance. Facial expressions are a big part of how the user perceives the robot. However, realistic facial expressions can be tough to achieve as humans can intuitively understand when something looks unusual. Leaning more towards the cartoonish and simplistic facial expressions could solve this issue. Systems such as KASPER [7] use this idea in the design to approximate the facial expressions without ultra-realism.

As mentioned previously, the growth of the HRI field leads to an increase in the number of collaborative robots being introduced in our society, which in turn raises the need to study and improve the security and efficiency of human-robot interactions. Several works validate the use of VR to reach this goal for different types of interactions, be it in handing over and receiving objects from a robot partner [5] or controlling a robot arm [8]. Regarding the improvement of HRI safety, different solutions are presented to this problem. Applications such as ”beWare of the Robot” [9] tackle the safety issues through ”emergencies” or warning signals in the form of visual and auditive stimuli. A study was conducted with thirty participants that revealed that the ”emergencies” system was found helpful and also validated the use of VR in the study of HRI.

Instead of warning the user, another way to tackle the safety issues is to adapt the robot's movements. This can be achieved through a ”digital twin” environment, where the physical environment is directly transcribed to a digital one, allowing for part or all of the logic to be in the digital environment. The robot can then use this system to avoid the human or even stop entirely [10].

The introduction of biological signals in HRI can improve the quality of the interaction. For instance, it can allow the user to control a robot using a brain-machine interface [11] or allow the robot to recognise the human's mental state and physical activities through sensors monitoring heart-rate, muscle activity, brainwaves through EEG, nose temperature and a head movement [12].

Most simulations dedicated to testing HRI have the participants fill a sense of presence questionnaire [13, 14]. Three main indicators can measure this sense of presence [15]: the sense of the user ”being there”, how much the simulation feels more natural or present than reality and how the user thinks about the simulation more as a place visited than a set of images. The measure of this feeling is crucial because it is directly related to how well the experience transfers to the physical world [16], [17] and, as such, how similar the results would be if the interaction were with a real robot. The analysis of the user's behaviour can also give valuable information about the sense of presence, for example, its posture, physiological signals and reactions to events within the simulation. The immersion of a user in a Virtual Reality Simulation is another aspect to take into account. It can be

described as ”a description of overall fidelity in relation to physical reality provided by the display and interaction systems” [13].

There are two types of factors that contribute to the sense of presence, external and internal. The external factors refer to the simulation aspects, such as its consistency, the quality of the information presented, being that the user's visual and kinesthetic systems are of great importance, the representation of the user within the simulation, the correlation of the user's real and simulated movements and the connection between cause and effect within the simulation, that needs to be simple enough for the user to be able to model over time. The internal factors are user-specific. They relate how different people react to the same stimuli and how the different mental models affect the perception.

Another aspect to take into account is to isolate the user from the external world, meaning that, for example, giving instructions to the user while already in the simulation negatively affects his sense of presence [15]. An interesting aspect to consider is that if it is necessary to make a scene change, it is more immersive for the user to put on a virtual VR headgear and then change the scene than opening the door to another scene. The technical specifications of the hardware should also be considered, such as the delay between the user's real and simulated movement, as they can significantly impact the level of immersion.

Besides evaluating the user's sense of presence, it is also essential to understand how the interaction with the robot felt. One of the possibilities is to evaluate the fluency in the human-robot collaboration [18]. This fluency refers to the coordinated mixture of activities between the members of a synchronised team. It evaluates if the action timings are precise and if the actions and plans change dynamically and appropriately without much communication. If the fluency of the interaction improves with time, as seen by the user, it might indicate that the robot adapted his behaviour successfully. More specifically, human-robot interaction has five main aspects to take into consideration [19]:

- The level of autonomy, which can be described as how much time a robot can be neglected;
- The nature of information exchange, which includes the interaction time, the cog-

nitive or mental workload of an interaction, the amount of situation awareness and the amount of shared understanding between humans and robots;

- The structure of the team involved in the interaction;
- The training of the people and the robot, including interactions that require minimal training, such as educational robots for children and interactions that require careful training, for example, for handling hazardous material;
- Task shaping process, which considers how a task changes when a new technology is introduced, in our case, the robot. Includes both tasks that could not be done before introducing this technology and tasks that are made easier.

The opinions of the participants obtained through the questionnaires are essential aspects, as mentioned. However, they tend to give information only about what the user perceives. Coupling these questionnaires with a measurement of the participant's biological signals can be used to explore both what the user perceives and its subconscious reactions. Examples of such signals are electrical brain activity and electrodermal activity.

About the electrical brain activity, one concern that may arise is that the signals obtained using simulated environments may not be reused reliably outside of them. Studies have proven that this is not the case [20], as the evoked potentials obtained while inside a simulation can be successfully recognised with high accuracy, confirming that they correspond to the ones obtained in the real world.

Works that measure the participant's brain activity while inside a simulation are also prevalent outside the HRI field. For example, in the case of patients that suffered a stroke and need to re-learn their motor skills. Translating the participant's intentions of moving a limb, captured with a BCI, to an actual limb movement of the user's avatar inside a VR simulation can be a very efficient therapy [21].

Another way to evaluate the user's subconscious reactions in the simulations, such as reactions to events, is by detecting their stress level. Electrodermal activity (EDA) is one of the leading indicators of stress. There are various techniques and open-source software that can be used to analyse such signals [22]. More recently, deep learning algorithms alongside the ”classical” statistical algorithms can be used to make more complex analysis

[23].

1.4 Contributions

The work described in this document focuses on creating a base system (BaxterVR) to study human-robot interaction in a virtual reality environment. Two virtual reality scenarios are created using the base system as a foundation, taking into account the sense of presence of the user's interacting with it and their feeling about the interaction with the robot. The first scenario is an essential contribution to the study and improvement of human safety in HRI using electroencephalography. The second scenario integrates the user's electrodermal activity and heart rate to better understand his reactions while collaborating with a virtual robot.

As an addition, an article about the second system, added in the appendix, was written and accepted in the GRAPP 2022 event.

1.5 Document Overview

This dissertation is organized as follows:

- **Chapter 1** describes the relevance of this work, its main objective and related works.
- **Chapter 2** explains key basic concepts needed to understand this work.
- **Chapter 3** presents the requirements, an overview, implementation and results of BaxterVR and the scenario created in collaboration with ProAction Lab to study human reactions to errors from the robot.
- **Chapter 4** shows an overview and implementation of a scenario to study the effect of distractors on a human when interacting with a robot. A pilot study, respective results and discussion are also presented.
- **Chapter 5** concludes the dissertation and indicates follow-up work, considered relevant in the future.

2

Background

This chapter aims to provide the reader with background knowledge about the key aspects of this work.

2.1 Baxter

The robot used in the human-robot interactions is the Baxter robot. This section aims to describe both the robot and the SDK provided by Rethink Robotics.

2.1.1 The Baxter robot

The Baxter robot (figure 2.1) was built by Rethink Robotics and presented in 2012. With the pedestal, it has 2 meters in height, which can be quite imposing. It has two arms, with seven degrees of freedom each. Each arm can be controlled independently. It resembles a humanoid figure with a screen able to present various faces. When a human is interacting with Baxter, these humanoid features can help create a connection with the robot. It was made to make simple and repetitive tasks in production lines, such as loading and unloading. Above the screen, Baxter has an array of sensors that allow it to detect nearby people. The security features it possesses make it very appealing for human-robot interaction studies. It can detect potential collision events and reduce the strength of the impact. Contrary to most industrial robots, Baxter does not have to be inside a cage while working due to all its security features.

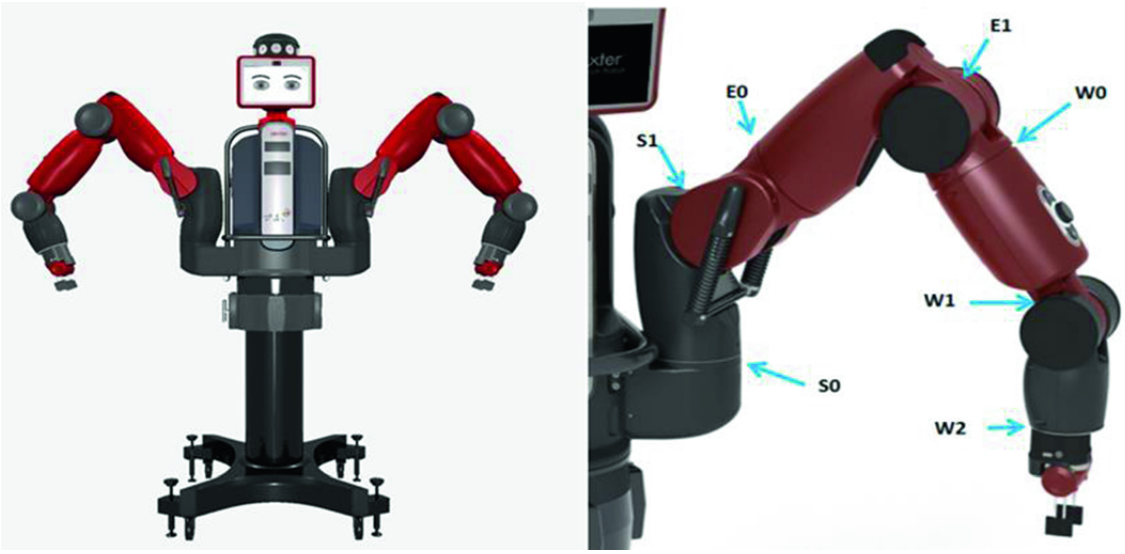


Figure 2.1: Baxter robot and arm joints

2.1.2 Baxter SDK

Rethink Robotics provides a Software Development Kit (SDK) for anyone interested in developing custom applications with the Baxter Robot. It is distributed in four main repositories:

- Common: contains the URDF, meshes and custom ROS messages that describe Baxter.
- Interface: contains Python interfaces and action servers to control the Baxter robot.
- Tools: useful operation and maintenance tools such as calibration scripts.
- Examples: examples of the SDK usage.

The URDF and meshes in the common repository were used to create the simulated Baxter robot presented in this document. The interface and tools repositories were used to communicate with the real Baxter and calibrate his arms. The software presented in this work was created using the most recent version of ROS (ROS Noetic), while the ROS version used by the real robot was much older. This version difference made it impossible to connect both and have the real robot and the simulated one running simultaneously due to incompatibilities such as message constitution.

2.2 MoveIt

For the user to be immersed in the interaction and to have realistic robot movements, the robot needs to consider the surrounding objects and respond to any changes, be it in the environment or the movements of any human with whom it is interacting. The motion planning framework MoveIt, built on top of ROS, is an excellent solution for this problem. Its integration with the proposed system will be described in the next chapter.

It computes the robot's movement's using motion planning algorithms. Simply put, motion planning consists of establishing a trajectory in a sequence of intermediate points, considering the path restrictions (environment and robot restrictions) and the time restrictions (acceleration, maximum speed) [24]. In contrast, using a simple inverse kinematics calculator without considering the surrounding environment leads to the robot bumping into objects or himself, although much faster since the collision checking operation accounts for close to 90% of the computational expense. Since MoveIt's motion planning algorithms are extremely well optimised, its calculations are very fast, and as such, the computation time does not negatively affect the interaction with the human.

MoveIt's motion planning allows for the generation of trajectories through cluttered environments while avoiding local minimums and considering the limitations of the robot in use. Furthermore, it is possible to attach an object to the robot's end-effector so that the computed trajectory also avoids the collisions between the attached object and the surrounding environment. This is an important detail, as it makes the robot seem more aware of its surroundings. Moreover, when a human interacts with a robot in a VR HRI scenario, seeing the object the robot is grabbing going through or bumping into other objects can negatively affect the user's immersion.

To communicate with the robot's controllers and get information about its state and sensor data, it uses ROS topics and actions. When no real robot is connected, it is possible to use fake controllers that publish the robot's information, such as joint states. When creating the MoveIt package for a robot, various launch files are created for different situations: working with a real robot, a Gazebo simulation or no robot at all, launching the fake controllers mentioned. This work uses the latter since no real robot is connected.

MoveIt provides three options to represent the objects in the robot's surrounding environment: geometric primitives, meshes or cloud point data. Geometric primitives have the advantage of being a lot faster to process. However, meshes are a good option when precision is critical or simple shapes cannot easily represent an object. Cloud point data is mainly used to represent real-world data gathered by the robot's sensors. A representation of the current state of the robot and surrounding environment can be visualised using RViz, which can be extremely useful for debugging purposes. This work uses only geometric primitives since they can easily represent all objects used.

To use MoveIt with a particular robot, it is necessary to create a MoveIt package for that robot. This can be done through the MoveIt Setup Assistant, where various aspects of the robot are specified: URDF, virtual joints, planning groups... Many robots already have MoveIt packages created with specific settings and customisations, such as custom inverse kinematics calculators, more correctly adapted to that robot. This is the case with the Baxter robot. In this work, a package was created to work with MoveIt. However, some aspects were not working in the best way possible, such as the inverse kinematics calculations. It was sometimes unable to compute a valid path, even though it should be possible. This was solved using the Baxter MoveIt package provided by Rethink Robotics which includes a custom inverse kinematics calculator.

2.3 Robot Operating System (ROS)

ROS or Robot Operating System provides services for robots, such as hardware abstraction, message-passing between processes and "tools and libraries for obtaining, building, writing, and running code across multiple computers" [25]. Simply put, it allows for different processes to communicate with each other even across multiple machines. For example, it provides the means for a robot running a ROS process to communicate with a computer that is controlling it.

The main ROS concepts are

- Nodes: a process performing some computation.
- Master: provides naming, registration and lookup for the nodes. For example, when

a node subscribes to messages from another node, the master notifies the publisher to send the messages to the subscriber. In addition, the master includes the Parameter Server, a central data storage.

- Messages: data structures used to communicate between nodes. They can have various types of fields, including integer, boolean and even arrays.

- Topics: nodes communicate with each other via a publisher/subscriber model, and topics are the names used to identify the specific types of information available. For instance, if a node wants to receive information about the robot's joint state, it subscribes to the topic where the robot node is publishing this data.

- Services: it is a server/client synchronous type of communication between nodes. It allows a client to make a request to a server and obtain a response synchronously, similar to an HTTP request.

As mentioned, when a node wants to send a request to another node and receive a reply, it can use ROS services. However, there are cases where the requesting node needs to cancel the previous request or receive feedback about how it is processing. In these cases, it is possible to use ROS Actions. They have three important messages: goal, feedback and result. When an action client wants some goal to be achieved, it publishes a message to the goal topic with the appropriate message. The action server processes this request and sends periodic feedback to the feedback topic. When the goal is achieved or aborted, the result is published to the result topic. The client can preempt the task being executed by the server at any time.

In a practical example, ROS allows for a node to request another node, such as an action server, to move the arm of a robot to a certain position. To do this, the requesting node sends a predefined ROS message to the goal topic of the action server. The action server node makes the necessary computations to move the arm and can provide periodic feedback about the arm's position. The requesting node can monitor this feedback, to make another request or cancel the previous one if necessary. It also monitors the result topic, since the action server will send a message to this topic once the movement is concluded.

2.4 Unity overview

Unity is a development environment used to create 2D, 3D, AR and VR applications, which can be deployed to the web, game consoles, mobile devices and personal computers. It can be described by two basic concepts:

- Scenes: each scene contains its environment, obstacles and decorations, which, in turn, are collections of game objects, essentially allowing the simulation organisation in pieces. Using scripts, a new scene can be loaded. The game objects within the scenes have various components that dictate their behaviour, such as gravity or appearance.

- Scripts: Unity allows the user to create runtime and design-time scripts using C#, with an object-oriented type of programming, using an implementation of the standard Mono runtime. A script can be applied to a game object as a component to control and monitor certain aspects, such as a joint angle or object position and orientation. Every script attached to an active game object in the current scene runs every frame in a predefined order.

It is the combination of the objects that compose a scene and scripts that control them that define each Unity game.

Another essential part of Unity is the assets. From the Unity website, [26]: ”A Unity asset is an item that you can use in your game or project. An asset may come from a file created outside of Unity, such as a 3D model, an audio file, an image, or any other types of files that Unity supports.” Without assets, the development of a project would take a lot more time. Besides providing the capability to decorate scenes without creating all the models from scratch, it also has another essential feature: easy use of the software necessary to use devices like the HTC Vive PRO virtual reality system. This project uses version 2019 of Unity.

2.4.1 Unity and Gazebo comparison

As mentioned, the main mean of interaction between the human and the robot in this work is through a VR simulation. When planning to build this simulation, two systems were considered, Unity game engine and Gazebo simulator. Both present clear advantages and disadvantages.

Gazebo was made to be used with ROS, and as such, the integration with MoveIt would be relatively simple. Furthermore, unlike the Unity game engine, it is a robotics simulator meaning that it has many built-in features needed to work with robots, such as a URDF importer. It offers integration for the Oculus Rift VR Headset DK2, which is available in the ISR laboratory IS3L. However, the integration with the HTC Vive PRO Virtual reality system, the planned headset for this project was not supported and would have to be developed.

Unity, on the other hand, does not have built-in support for ROS, albeit being possible through the use of the Rosbridge plugin, but supports a much wider variety of VR headsets, including the HTC Vive PRO virtual reality system.

After some consideration, Gazebo was chosen to begin development, and ROS integration was simple. When trying to use Oculus Rift VR Headset DK2, the motion from the user in the real world was not translated to the virtual world. Since even after many tries, the problem persisted, reassessment of the system to use was necessary. If a supported system was not working correctly, integrating an unsupported system more complex than the supported one could consume too much time. Following this logic, the development platform was switched to Unity.

In conclusion, Gazebo provides many functionalities to simulate robots, however, when integrating complex external systems, such as a VR system, Unity has the upper hand. Due to its immense popularity, most systems can be integrated with Unity with a simple plugin addition. Moreover, much interest has been put into using Unity as a robotics simulator over the last few years. As such, many of the disadvantages that come with using a game engine instead of a robotics simulator can be overcome with the plugins created. An excellent example of such a plugin is ROS#. Another significant advantage in using

Unity, which is also derived from its popularity, is the extensive and active community that overshadows Gazebo's community.

2.5 Equipment used

During this project, various equipment was used either due to the status of the global pandemic or the availability of the devices. The first piece of equipment used was Oculus Rift VR Headset DK2 because of the compatibility with Gazebo. As mentioned previously, it was not successfully used.

The first scenario developed used two systems: Razer Hydra (figure 2.2a) and HTC Vive Pro Eye (2.2b). Razer Hydra is a game controller that detects the controllers' absolute position and orientation with a weak magnetic field. It was mainly used to test and develop the simulation at home. HTC Vive Pro Eye, on the other hand, is the Virtual reality system used in the final version of the simulation. It was used for both the testing and development in the laboratory and experiments with the final product. It allows for high precision room-scale tracking through the use of two solid-state lidars. A useful feature of this HMD is the possibility to define boundaries in the room. While inside the simulation, if the user is not close to these boundaries, nothing will happen, however, when approaching them, the surrounding image inside the simulation will start to change from the simulation itself to the cameras in front of the headset. This feature gives the user a sense of security because he will know that no wall will stand in the way if the boundaries were made correctly and also allows for him to be fully immersed inside the VR simulation, with no need to be constantly aware of the real world.

The second system uses Oculus Quest as its virtual reality system, a Bitalino board and sensor to measure the user's electrodermal activity, and a Polar H10 heart rate sensor to measure the user's heart rate.

Oculus Quest (figure 2.2c) is another virtual reality system. The way this system tracks the surrounding environment differs from HTC Vive Pro Eye. While HTC Vive tracks the surroundings through external sensors, Oculus Quest relies on internal sensors and an array of cameras in front of the headset.

It presents a similar feature to HTC Vive relating the boundaries, with the same advantages, however, since it relies on the cameras and computer vision to identify the surrounding environment, it becomes much more portable. The only calibration needed is to specify the ground position by placing a controller on the ground and identifying the borders if Oculus does not recognise the environment.

Comparing the two HMDs, HTC Vive Pro Eye manages to track the user's movements with higher precision than Oculus Quest due to being made mainly by external sensors, namely two lidars. However, this also presents a disadvantage in some situations. Since Oculus Quest relies solely on internal sensors and computer vision, they are highly portable, without the need to set up a room specifically to use them. HTC Vive enables tracking the user's eye position with an accuracy of 0.5º-1.1º, which can be helpful or even critical in some researches. A critical aspect that distinguishes both HMDs is the price since HTC Vive Pro Eye costs more than three times as much as Oculus Quest.

In conclusion, Oculus Quest is a more affordable option that offers more flexibility while keeping the quality high. HTC Vive has higher quality, enables eye tracking, but has a significantly higher cost and needs a dedicated space to be used.

ProAction Lab had an HTC Vive Pro Eye VR equipment available, and the research involving the first scenario required it to be used. It enables the integration of eye-tracking in the future, and since it was already available, the cost was not an issue.

Concerning the second scenario, only the Oculus Quest VR equipment was available. Since no eye tracking was needed and the high portability of the equipment was essential to allow for in-home development, due to the global pandemic, we concluded that an investment in an HTC Vive Pro Eye was not necessary, and this scenario was developed using Oculus Quest.

2.6 Biological signals

Electrodermal Activity (EDA) or Galvanic Skin Response (GSR) consists of the variation of the skin's electrical conductance due to sweat secretion. This variation is not under our conscious control. Instead, it is an autonomous process managed by the Sympathetic

(a) Razer Hydra **(b)** HTC Vive Pro Eye **(c)** Oculus Quest

Figure 2.2: Systems used to interact with the simulations

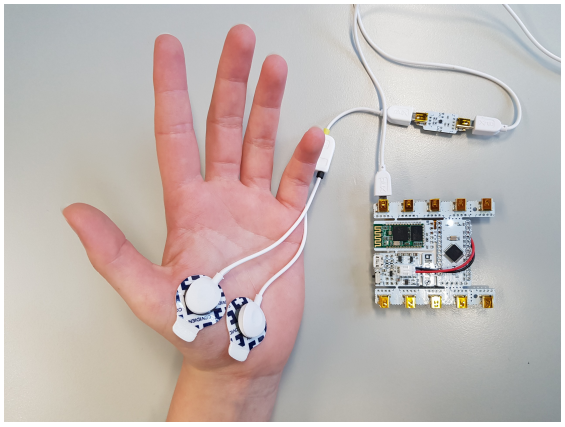
nervous system (SNS), which manages various psychological processes, including emotional arousal [27].

EDA can be related to the regulation of our internal temperature, but it also has a strong association with emotional arousal. It can be divided into two components [28]: the tonic component and the phasic. The tonic varies slowly over time, while the phasic component presents sudden variations in its value, often due to external stimuli. These sudden variations are called SCRs or Skin Conductance Responses. They can be used to evaluate the user's reaction to an event: the higher the response, the higher the arousal.

The BITalino (r)evolution Plugged kit (figure 2.3a) and the respective EDA - Electrodermal Activity sensors 2.3b) are used to gather the user's EDA signals. This kit features a board that allows Bluetooth connection to another machine and multiple sensors that can be connected to the mainboard through UC-E6 sockets. It also offers its software to monitor and record the biosignals, OpenSignals. This software, however, was only used in the development of the scenario.

Heart Rate (HR) consists of measuring the heart beats per minute. It is stimulated by both the parasympathetic nervous system (PNS) and the SNS. Input from the PNS tends to decrease the HR, while input from the SNS tends to increase it [29]. Variability on the HR can be prompted by psychological activity, and as such, this measure can be used to measure a human's response to external stimuli. An increase in HR or decrease in its variability is generally linked to a higher mental workload [12].

(a) BITalino (r)evolution Plugged kit



(b) Bitalino's EDA sensors

Figure 2.3: EDA sensors and Bitalino kit

To measure the heart rate in this work, Polar H10 heart rate sensor was used (figure 2.4).

Figure 2.4: Polar H10 heart rate band

Brain activity can be measured using an EEG sensor. It can be used to predict specific actions or reactions from a human through the analysis of certain brain waves, such as the intention of moving a limb [21]. Specific systems can then be developed to act based on these brain waves and act accordingly. For instance, it has been proven that with enough training, both humans and monkeys can move a cursor on a screen using only their brain activity [30]. To do this, they first start by moving it with a joystick while their brain activity is gathered and analysed. Once enough data has been analysed, they gradually move to a scenario where they have no joystick but still have to move the cursor. At this point, the analysis of their brain waves is enough information to indicate their intention of

moving the cursor, and thus they can control it with a brain-computer interface.

The same principle can be applied to human's recognising errors made by a robot. The first step is to gather enough EEG data to model and recognise the signals that indicate such an event, more specifically, error-related negativity (ERN) signals [31]. Hence it is crucial to have a safe environment, where the reactions of a human to errors from a robot can be recorded. These ERN signals can then be fed as a stop signal to the system controlling the robots.

2.6.1 Lab Streaming Layer

As mentioned, both the EDA and HR can be used to measure human reactions to events. For this to be possible, the acquisition of these signals needs to be synchronised with the occurrence of the events.

The Lab Streaming Layer (LSL) system provides this functionality. It handles the networking, time-synchronisation, real-time access and centralised collection, viewing and disk recording of the data.

It enables each data source to stream its data to the local network. The data from all streams are captured and recorded with common timestamps in a single machine using the LabRecorder application.

It also provides Matlab functions to import the recorded data, enabling subsequent analysis.

3

VR simulation to study human reactions to robot's errors in human-robot interactions

This system was made in cooperation with ProAction Lab. It is part of a study where the main objective is to understand if human brain waves, gathered through electroencephalography (EEG), can be used to detect errors made by a robot in a human-robot collaboration situation. The proposed system, BaxterVR, enables the user to interact with a simulated robot in a Virtual Reality environment. In the scenario presented, this is done throughout several trials. In each trial, one of two types of interaction occurs: the robot hands an object to the user, or the user hands an object to the robot. Both of these actions have variants where the robot makes a mistake, inducing the required reactions from the human.

3.1 Requirements specification

The design of the system followed a user-centred approach. An initial set of requirements was established by the researchers at ProAction Lab, which were then evaluated towards the planned system and adapted accordingly. Through several iterations, a final version of the requirements was established:

1. It should be possible to configure:
    - The number of trials to execute and configure each trial individually

- The maximum time a trial is allowed to take
- For each trial:
  - The type of interaction (robot handing the object versus robot grabbing the object).
  - Distance between the user and the robot (Safe distance versus user within robot reach)
  - Which arm the robot will use (left or right).
  - Turn on/off the robot's monitor (face).
  - The speed of the robot's arm.
  - The type of error (no error, reach too close or reach too far).
  - The error movements from the robot can either be visible from the beginning of the trial or start as regular movements and change to error movements.
  - The type of object to use.
  - The user's end-effector, between a hand or a Vive controller.
  - A time offset between the start of each trial and the moment the robot starts moving.

2. Trial and event markers should be inserted in a log file with respective time stamps, the human's hands and head position and the robot's active end-effector position.
3. Trials should end at a button press or when the time limit is reached.
4. The trial order is randomised.
5. The robot resets its position at the beginning of each trial.
6. The beginning of the trial is indicated by a sound.
7. The robot can either reach accurately, make a mistake and reach away from the user's hand or reach towards the user's body.

8. The user's hand movements can either be normal or suffer from a deviation when the hand gets close to the object to grab.

As part of the research, it is expected that the errors and surprise situations induce detectable potentials via an EEG device. Moreover, the position of the user's hands and head and of Baxter's active end effector is recorded every frame through the logging mechanism created, allowing the researcher to integrate it into the behavioural analysis performed after each experiment.

The customisation of the robot arm's starting motion in error trials (starting as an error motion or normal motion) leads to sudden and unpredictable changes in the movement trajectory, which violates the human participant's expectations, eliciting the required error-related neuromarkers [31].

3.2 System Overview

Given the requirements, BaxterVR can be split into two main features: a VR simulation, or scenario, and a simulated Baxter robot. The simulated robot includes a method to plan its movements, the robot's appearance, and its animation.

Starting with the VR simulation, the main objective is to elicit reactions from the human to the robot's errors during two types of interactions: the robot handing an object to the human or the human handing it to the robot, each one with error variants. Each simulation consists of several trials with specific configurations, following the requirements presented in the previous section.

One of two objects can be used in the interaction, depending on the configuration: an abstract object, which is a yellow bar, and a wrench.

The surrounding scenario is simple to avoid distractions and possible responses in the biological signals due to external factors. It does not represent a realistic scenario since it is not necessary at this first stage of the research, but, at the same time, to induce a high sense of presence, it feels natural and pleasant. It consists of a room with a couch, plants and the symbols of Coimbra University, the ISR and Proaction lab (figure 3.1). In the

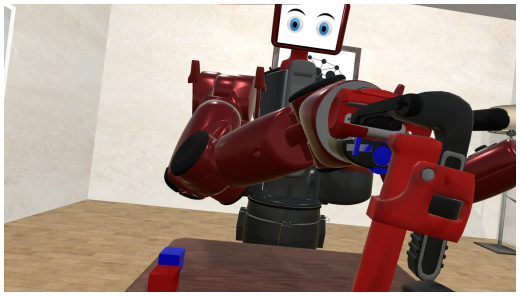
middle, there is a table with the object used in the interaction and three cubes that the user can interact with to get used to virtual reality. The Baxter robot is positioned on one side of the table, and the human interacting with it is on the opposite side. On the wall to the left of the user, there is a whiteboard with instructions. The objects are placed in their original position between each trial, and the robot arms move to a default position. When the objects fall to the ground, they are also placed back in their original position.

Figure 3.1: Surrounding environment

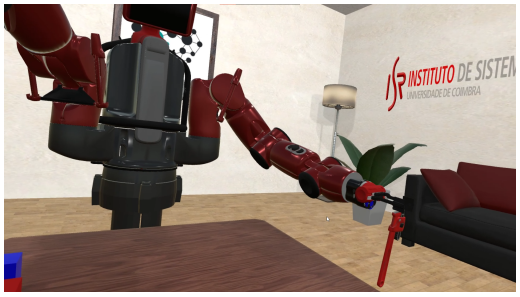
The table between the human and the robot creates a barrier between them big enough for the robot to move its arms freely. Depending on the trial configurations, the robot can be further from the table, at a safe distance from the user or closer, allowing it to reach the user with the arms extended.

The simulation configuration is a vital aspect to consider to give as much flexibility as necessary to the experiment. Two possibilities were considered, configuring through an external file or using a User Interface in the simulation. As mentioned, the development of this work follows a user-centred approach, so creating a User Interface could lead to unnecessary work since it could be necessary to change it in every iteration. The best process is then to use an external file, such as a CSV file. With a CSV file, the researchers also have the ability to automate the configuration with tools such as Excel.

When handing the object to the human, the robot starts by picking it up from the table. After grabbing the object, the robot moves to one of the predefined positions. If in an error trial, to a position too far (figure 3.2b) or too close (figure 3.2a) to the human and if in a



(a) Baxter delivering the object too close to the human



(b) Baxter delivering the object too far from the human

Figure 3.2: Baxter delivering an object in the wrong position

normal trial, to a position in front of the human with the adequate distance. It can also start moving towards the normal position and then change its movement to a wrong one when its end-effector gets within a certain distance from the human's hand. To prevent a situation where the human takes the object, not allowing the robot to grab it, thus possibly stopping the simulation indefinitely, the user cannot grab the object during this time. Moreover, suppose the robot still cannot grasp the object after a certain number of tries. In that case, it is repositioned to where the robot can grab it, allowing for the experiment to proceed, even if an error occurs at some point.

When the human is handing over an object to the robot, he starts by picking it up from the table then pressing a button on the controller. If in an error scenario, the robot moves to one of the error positions without grabbing the object. Otherwise, it grabs the object handed. In the latter case, it needs to evaluate if it can reach the object and only move in that situation. Moreover, it is almost impossible for the user to extend its arm to give the object and keep it still enough so that the robot only needs to compute the movement once. As such, the object's position needs to be constantly monitored, and the robot's movements repeatedly cancelled and recomputed for the new position.

The constant recomputation of the robot's movement imposes two crucial conditions on the system responsible for simulating the robot's actions. It needs to allow for the cancellation of previously started movements and be fast enough to start a new one swiftly and continuously with unnoticeable transitions. In a first attempt, a simple Inverse Kinematics calculator was implemented. However, it became evident that this solution was not ideal, as another essential aspect to consider in the robot's movements are the objects in the en-

vironment and the robot itself. These considerations would prevent two main situations that break the user's immersion: the robot arms going through stationary objects such as the table or the rest of the robot's body and going through the object it is grabbing. Given these restrictions, it is required to consider a more sophisticated approach that includes Motion Planning.

Besides the necessary motion planning features mentioned, if the system simulating the robot allowed for easy connection to the real robot, the presented approach could be easily adapted to studies involving both robots, such as a "Digital Twins" system.

For the reasons presented previously, the motion planning framework MoveIt is used to compute the robot's movements as it complies with all the requirements.

3.3 Implementation

As mentioned, BaxterVR can be divided into two central systems (figure 3.3): the VR simulation made with Unity and the Baxter simulator. Both systems were made to be independent. Building them in such a way makes it possible to efficiently use BaxterVR's Baxter Simulator with various VR scenarios or use the created VR scenarios with different robots. Figure 3.4 presents the integration of this application with the external systems and respective communications.

3.3.1 Baxter Simulator

The Baxter simulator consists of the visual representation of the Baxter robot in the VR simulation, its animation and the movement planning, which, as mentioned, is made with MoveIt.

3.3.1.1 Rosbridge and ROS#

Since MoveIt controls the robot's movements and Unity presents them to the user, both of these systems need to be connected. MoveIt is a ROS based framework and Unity does not directly support the connection to ROS. However, many efforts have been made over the last few years toward this goal [32], and it is now possible to use Rosbridge on the

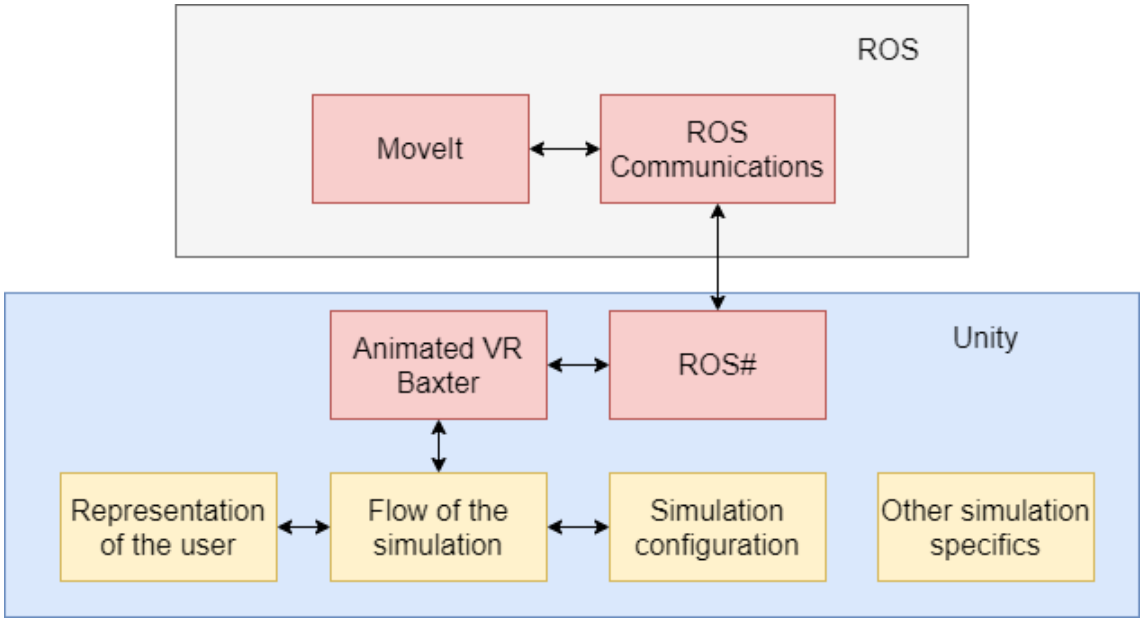


Figure 3.3: Different features implemented. The red blocks represent the Baxter simulator system. The yellow blocks represent the VR simulation.

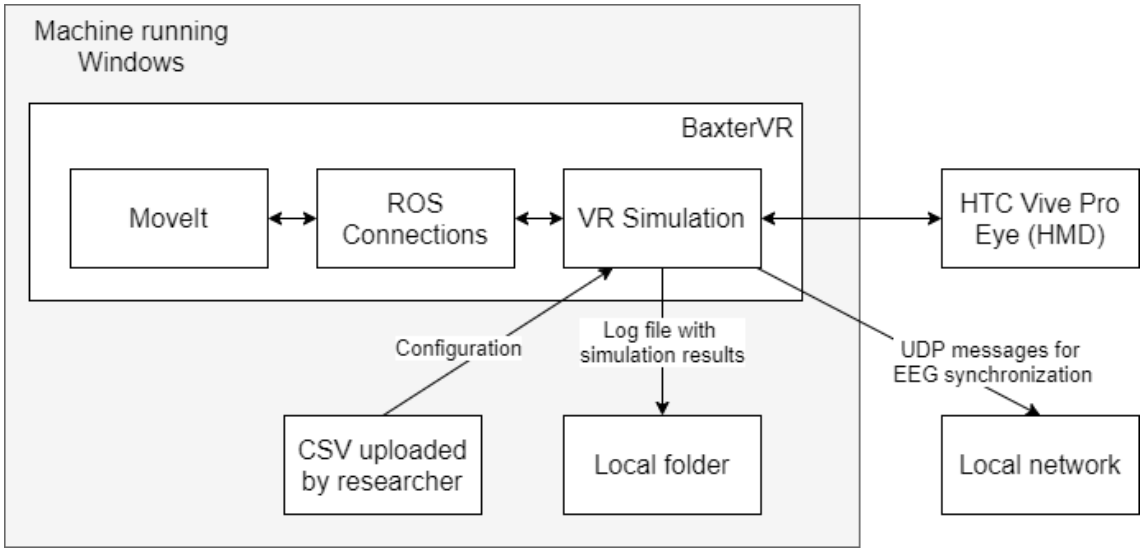


Figure 3.4: Integration and communications of the BaxterVR application.

ROS side and ROS# on the Unity side to make this connection.

Rosbridge (or rosbridge_suite) is a standard ROS library that provides a JSON interface so that non-ROS programs can send commands to ROS. ROS# makes use of this interface using a Unity Asset Package to give Unity-based applications a library with base functions that allow communication with ROS. This includes publishing and subscribing to topics, communicating through services and actions, and using custom messages. The

communication with the custom ROS nodes mentioned in the next section was achieved by creating a custom script for each one using this library.

ROS# also introduces important robotics functionalities such as importing a robot through a Universal Robot Description Format (URDF) file. In this project, this functionality was used to import Baxter's model.

3.3.1.2 ROS and MoveIt

To handle the communications between the VR Baxter and MoveIt, a total of three ROS nodes were created (figure 3.5). Their purpose is to handle the robot joint angles to the simulations, transmit information about the VR objects, such as their positions, and orientations to MoveIt, and handle requests to move the robot using an action server.

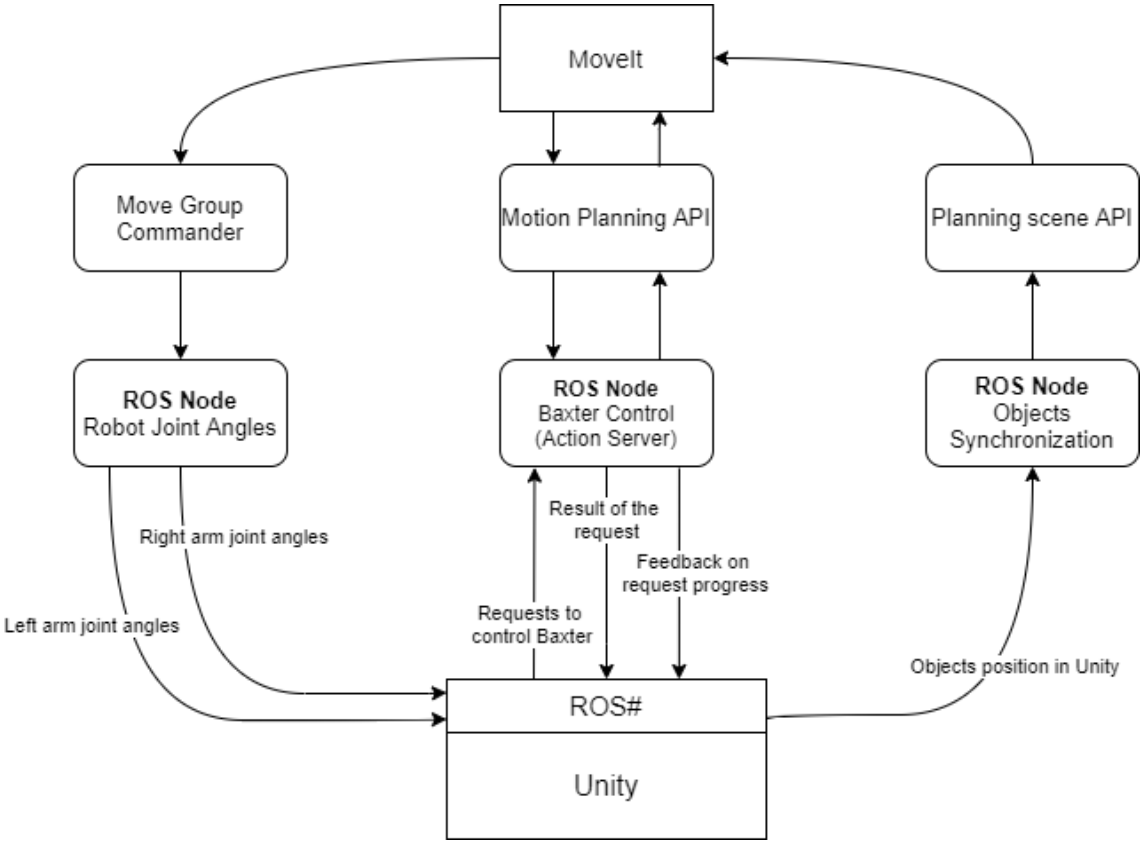


Figure 3.5: ROS Nodes

VR Baxter publishes the necessary information about the objects in the environment, such as position, orientation and size to the corresponding ROS node, which uses the Planning Scene ROS API to add them to the planning scene. The scene objects that are

relevant for the movement calculations are represented in MoveIt as boxes (figure 3.6). Object positions are updated only if their changes are above a given threshold to reduce the communication overload.

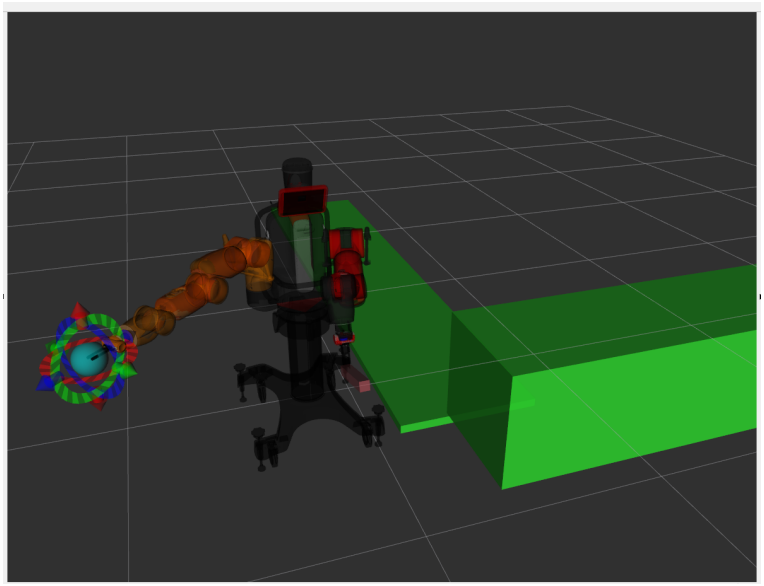


Figure 3.6: Rviz Visualization.

To publish the robot arms joint angles to VR Baxter, two separate topics are used, one for each arm, with a custom message containing seven values corresponding to each of the seven joints. The current values of the robot joints are obtained using a *MoveGroup-Commander* object, which provides an interface to a move group or a group of joints of the robot. The MoveIt package in the Baxter robot's SDK supports two move groups, one for each arm, allowing simple access to the joint values.

VR Baxter needs to be able to make requests to MoveIt to control the robot arms. It also requires the ability to preempt or cancel these requests, for example, when the object being picked up moves. Periodic feedback is also necessary to synchronise the actions on both sides, such as attaching a picked-up object to VR Baxter's end effector at the same time it is attached in MoveIt. A ROS action is the perfect fit for the previous description, and as such, a ROS action server was created to control the simulated Baxter robot. When it receives a request, it sends the necessary information to the motion planner using its API. The action server can receive the following requests:

- Move the end-effector to a specific position.

- Move the end-effector to a specific position using a Cartesian path.
- Pick up an object, given its position.
- Detach an object from a robot end-effector.
- Change the speed of the arms.

The action taken when picking up an object can be divided into four main stages:

1. Moving to a pre-grasp location: a position close to the object with an orientation that allows the robot to grab the object with one simple straight movement.
2. The grasp movement: a Cartesian movement, or a straight movement, towards the object to grab. After this movement, the object should be between the grippers of the robot.
3. Attach the object to the end-effector.
4. A retreat motion: a Cartesian movement away from the surface where the object is resting, or the support surface, to lift it.

As mentioned previously, the robot needs to react to changes in the environment, such as movements of the object being picked up. The Motion Planning API makes it possible to make asynchronous requests to the motion planner and cancel executing requests. However, it does not provide a simple way to know the status of a request being executed. A solution to this problem is to monitor the result action topic from the action that the API uses to communicate with the motion planner. This topic is only published when the motion is concluded while also giving information about its success.

3.3.1.3 Animated VR Baxter

The URDF importer from ROS# allowed for an initial model of the robot to be imported to Unity. This model, however, was not working properly. The main reason was that Unity could not properly process the collisions between the colliders defined in the URDF description. Removing the colliders solved the issue but also removed the ability to detect collisions between objects in the simulation and the arms and torso of the robot. Since the

human will not be touching the robot directly, this is not a problem. Some other minor issues were easily solved, such as incorrect hierarchy between the various robot pieces.

Regarding the robot's animation, each joint is controlled by a Hinge Joint component, which allows rotation around a single axis. Three options were tested to move them: using springs, motors and moving the joints in small increments using a script. The springs presented the best results, producing movements similar to the real Baxter.

Three components were created to animate each of Baxter's arms. They communicate hierarchically, meaning that a script that controls the VR simulation only needs to access the top one to set specific variables such as which object is being picked up. The top component of the hierarchy is the Arm Controller, followed by the Gripper Finger Controller and the Gripper Finger Collision Detector. The Arm Controller is responsible for the arms movements. It uses the ROS# component that is subscribed to the ROS topic transmitting the joint values to set the correct angles to the joints of the simulated Baxter. It also constantly monitors the action being executed by the action server and the feedback topic to close the grippers if needed. Lastly, it is also responsible for producing the sound when the robot moves. The Gripper Finger Controllers control each of the gripper fingers in that arm according to the position instructed by the Arm Controller. The Gripper Finger Collision Detector constantly monitors for a collision between the tip of the gripper finger and the object being picked up. If this happens, they signal the Gripper Finger Controllers to stop the movement and avoid situations where the grippers move to the inside of the object.

Animating Baxter's screen is made by placing various textures in front of it. Each texture represents a face, with a total of four faces (figure 3.7): Baxter looking straight, looking down, with its eyes closed and with a sad face. It allows for the VR simulation to choose between each of the faces. To give the feeling that Baxter is blinking its eyes, Baxter's face switches from its current one to the eyes closed and 0.2 seconds later changes back. This happens periodically with a random time interval between five and ten seconds. When Baxter's screen is turned off, this texture is not rendered.

(a) Looking straight. **(b)** Looking down. **(c)** Looking sad. **(d)** Eyes closed.

Figure 3.7: Baxter's different faces.

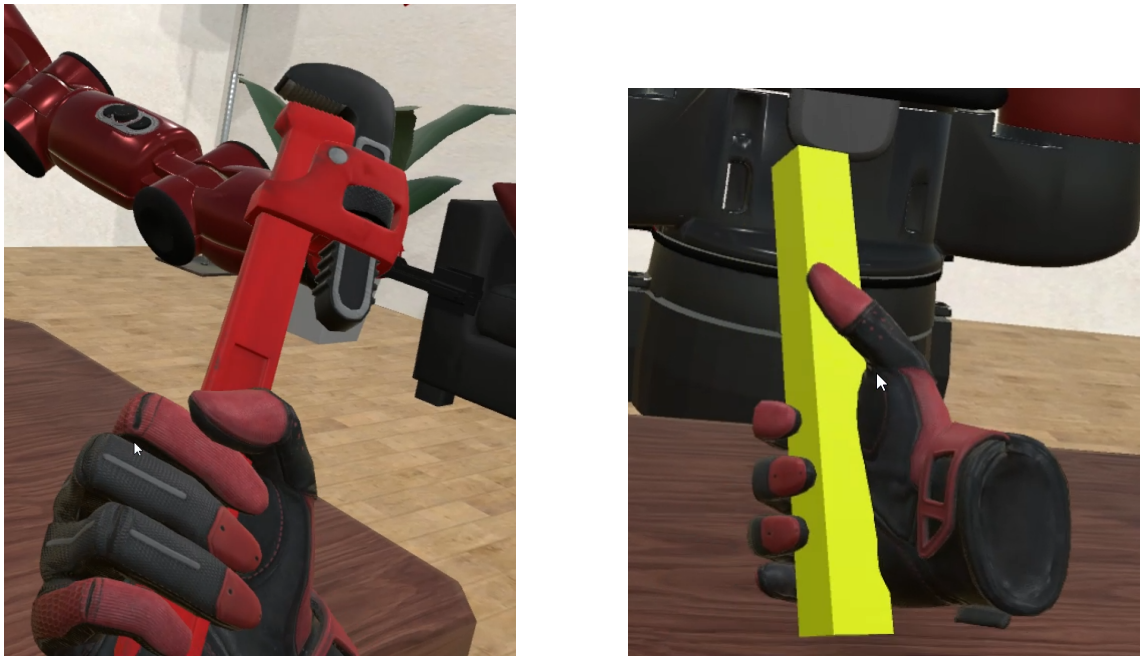
3.3.2 VR simulation

The VR simulation contains a series of important aspects: a scenario where the human and robot interact, the representation of the user in the virtual world and the control of the simulation flow (configuring each trial, requesting the robot to move...).

3.3.2.1 Representation of the human in the virtual world

The representation of the user in the virtual world is mainly achieved by using the HTC Vive Unity plugin, which communicates with the VR equipment and translates the real-world movements to movements inside the simulation.

This plugin provides two different camera rigs, which are two different ways to represent the player inside the simulation. In one of them, the user's end effectors are represented by the Vive controllers and in the other by hands. These fit perfectly with the requirements stated previously regarding the configuration of the user's end effector. The two camera rigs have different ways of interacting with the objects. In the case of the Vive controller, the interaction does not come included and had to be implemented. This was done by attaching an invisible sphere around the controller. If an object is inside it and the user presses the grabbing button on the controller, the closest object gets attached (figure 3.9). In the case of the hands, allowing the user to interact with an object comes down to simply adding a specific component to that object. However, this interaction is straightforward by default. When the user grabs an object, it gets attached to its end effector, but the hand disappears. This is not the desired behaviour to induce the highest immersion possible. To overcome this, a particular component can be added to an object that allows for the manual configuration of the hand's position and skeleton when grabbing that specific object. This component was added to both objects used in the interaction, and the



(a) User grabbing the wrench **(b)** User grabbing the abstract object

Figure 3.8: User's hand gripping the objects with custom skeleton positions

hand skeleton was configured to grab them in the most natural way possible (figure 3.8).

Another aspect to take into consideration, as defined in the requirements, is the possibility to induce a deviation to the user's hand when it reaches close to the object to grab. This offset only starts occurring when within a certain distance of the object (D) and intensifies the closer the hand and the object get up to a specific value (K), following equation 3.1. Where d is the distance between the hand of the user and the object.

$$\frac{K(D-d)}{D} \tag{3.1}$$

Both D and K can be configured. Moreover, to define the direction the deviation occurs, it is possible to configure an angle in degrees. When the user is facing the Baxter, 0 degrees of deviation correspond to an offset in the up direction, towards the ceiling. In the same situation, an offset of 90 degrees corresponds to an offset to the user's right, towards the right wall.

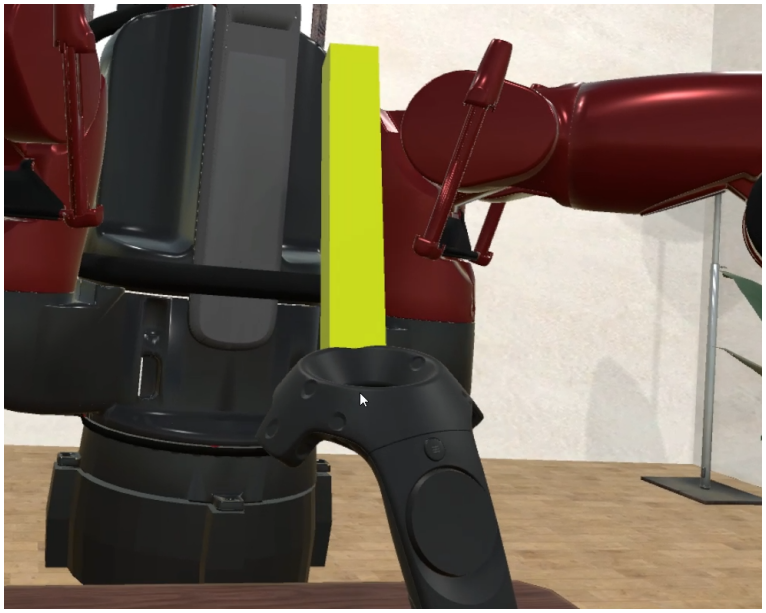


Figure 3.9: User grabbing an object with the Vive controller

3.3.2.2 General flow of the simulation

The flow of the entire simulation is controlled by the state machine presented in figure 3.10.

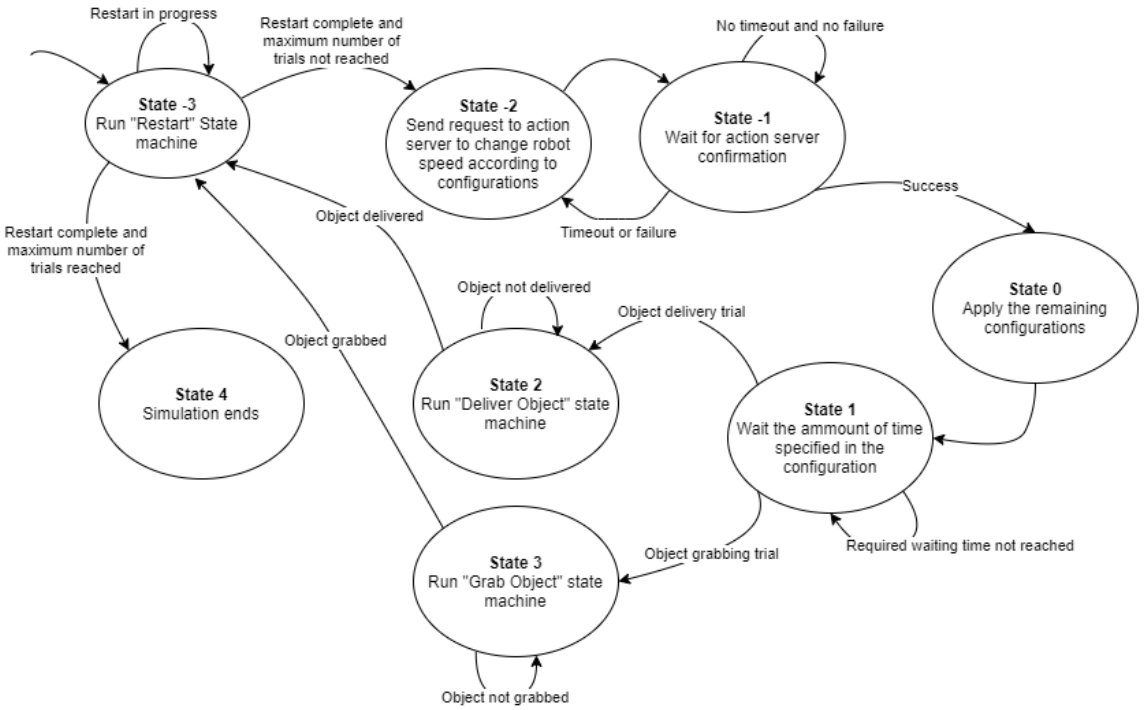


Figure 3.10: Global State Machine

At the beginning of each trial, the objects in the simulation are placed back in their

original position, and Baxter moves his arms to a default position so that they always start in the same position in every trial (State -3). This resetting of the simulation is made by the Restart state machine. The global state machine remains at state -3 until it is concluded.

After the reset, the specific configurations of that trial are applied. This is split between three states (State -2, -1 and 0) because changing the speed of Baxter's arms requires communication with the action server, while the remaining configurations are related solely to the VR simulation. The communications between VR Baxter and the action server are always made in three steps. VR Baxter starts by making a request, then waiting for the action server to confirm the reception of the request (Pending state). This state means that the action server received the request but is not processing it yet. A new request is made if the action server does not confirm the reception after a certain time. After obtaining reception confirmation, the next step is to wait for confirmation that the request is being processed (Active state). If this confirmation is not received after a certain time or if an error occurred in the request processing, a new request is made.

The configurations and actions taken at State 0 are listed below:

1. Change the text on the whiteboard to display the correct number of trials and instructions.
2. Set Baxter's position further or closer to the table, or, in other words, at a safe distance from the user or within reach.
3. Set the user's end effector to a hand or a Vive controller.
4. Activate or deactivate the textures on Baxter's screen.
5. Activate the object used in the interaction (abstract object or wrench) and deactivate the other.
6. Get the pre-defined pre-grasp and grasp positions for that object.
7. Set the object used in the interaction in the various scripts that need it. This includes the Arms Controller script mentioned in section 3.3.1.3.
8. Play a sound that signals the beginning of a trial.

9. If a trial timeout was specified, start a coroutine that is always monitoring how much time has passed. If it reaches the timeout time the state of the global state machine is changed to -3, effectively ending that trial and starting the next one.

Upon applying the configurations, no action happens for a certain amount of time, specified in the configurations (State 1). This gives time to the participant to look at the instructions board and understand what to do next.

After the waiting time, there are two possible courses of action the robot can take: grabbing an object from the table and handing it to the human (State 2) or grabbing an object handed by the human (State 3). Each one of these interactions has its state machine. The global state machine remains at one of these states until the respective interaction is complete.

If at any point the researcher presses the Spacebar key, the currently running trial ends and the next one starts. This is achieved by setting the global state to -3, similar to a trial timeout.

3.3.2.3 Delivering an object to the human

The state machine in figure 3.11 is used to deliver an object to the human.

The first state (State -1) is used to perform certain actions that should only be done once before the delivery action begins. These include inserting a line in the log file, sending a signal to the Python server, changing the texture of Baxter's screen to eyes looking at the object being picked up and changing the value of the flag that sets the permission for the human to grab the object to false, not allowing him to grab it.

After the previous actions are complete, a request is sent to the action server to pick up the object (State 0). As mentioned previously, the reception of the request is monitored (State 1) and in case of a timeout, a new request is sent.

Upon receiving confirmation of reception from the action server, VR Baxter waits for it to confirm that it is processing it or that it failed to execute the request (State 2). The first case means that the arm is currently moving while the latter most likely means that MoveIt could not find a trajectory to grab the object that fulfilled the restrictions. The most

likely reason for this is that somehow the object was moved and it is now out of reach of the robot (for example the human moved it with another object). For this reason, before sending another request to the action server, when the robot fails to grab it, the object is moved to its original position. If the request is successful and the robot arm is moving, the object position is constantly monitored. If it moves beyond a threshold, a new request is issued with the updated position.

When Baxter successfully grabs the object from the table (figure 3.12), the next step is to deliver it to the human. As usual, this consists of a request to the action server to move Baxter's end-effector to a certain position followed by waiting for an acknowledgement (States 3 and 4). The position requested depends on the type of trial. It can either be an error position or the expected delivery position. At this point, the texture on Baxter's screen is also changed to look at the human.

If the request was successful and the arm is moving to the delivery position, VR Baxter waits until the movement is complete (State 5). Moreover, if defined in the configurations that the movement should change from expected to error, the distance between the user's hand and the robot end-effector is monitored. When it reaches a certain value, a new request is issued with the defined error position.

On successful completion of the delivery movement, a request is sent to MoveIt to detach the object and the user is now allowed to grab it (State 6). After this, no action is taken until the user grabs the object (State 7).

The trial ends three seconds after the user grabs the object handed by the robot. The time between the user grabbing the object and the trial ending (which means resetting all the objects and Baxter's arms) gives a more natural feeling to the simulation. Without it, as soon as the user grabbed the object, it would be instantly placed in its original position.

3.3.2.4 Grabbing an object handed by the human

The other possible type of interaction is the robot grabbing an object handed by the human (figure 3.13). It is described by the state machine in figure 3.14.

Similarly to the state machine to deliver an object, it starts by performing the actions

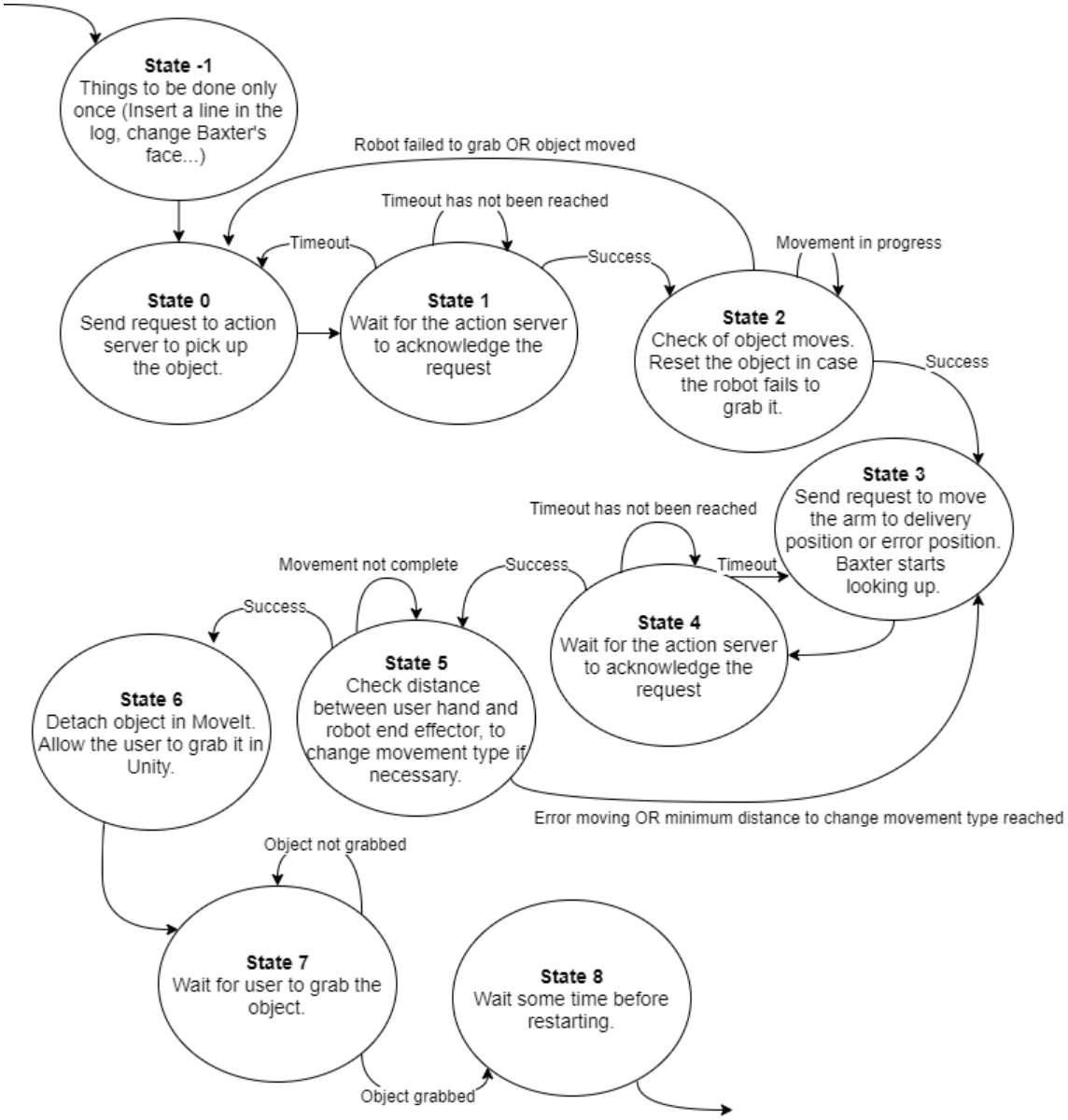


Figure 3.11: State machine to deliver an object to the human

that should only be done once (State -1): writing to the log file, signalling the Python server, allowing the user to grab the object and setting the screen texture of the Baxter robot to eyes looking at the human.

After performing the initial actions, VR Baxter waits until the user is holding the object, presses a button on the controller, and the object is inside the grabbing area which corresponds to the area where the robot can grab the object (State 0).

When the previous conditions are met, a request is made to the action server to move the arm (State 1). It can either be a request to simply move the arm to an error position or

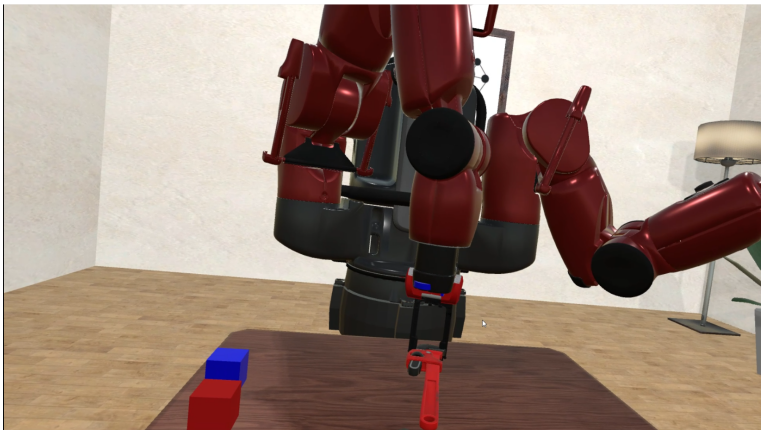


Figure 3.12: Baxter grabbing an object to deliver to the human

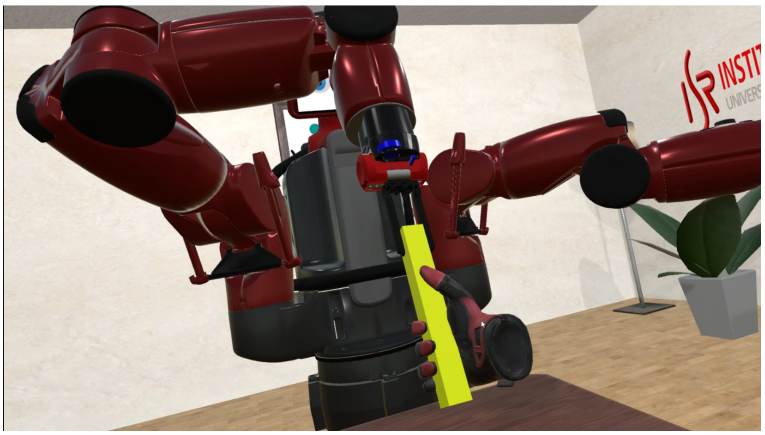


Figure 3.13: Baxter grabbing an object handed by the human

grab the object in the user's hand. This is followed by waiting for an acknowledgement (State 2). If in a normal trial, and the robot is grabbing the object from the user's hand, the object position is recorded at this state.

After receiving acknowledgement from the action server, VR Baxter waits until the movement is complete (State 3). If in a normal scenario, the object's position is constantly being compared to the one saved in the previous state. A new request is issued in the case of error or if the object moves beyond a certain threshold. If in a trial where the robot's motion starts as normal and changes to error, the distance between the user's hand and robot end effector is monitored. When it reaches a certain value a new request is issued to move the robot arm to the error position. Before changing to error motion, the robot arm behaves as normal, adjusting itself to the user's movement. These adjustments are not performed anymore in the error movement, which consists in just one movement to

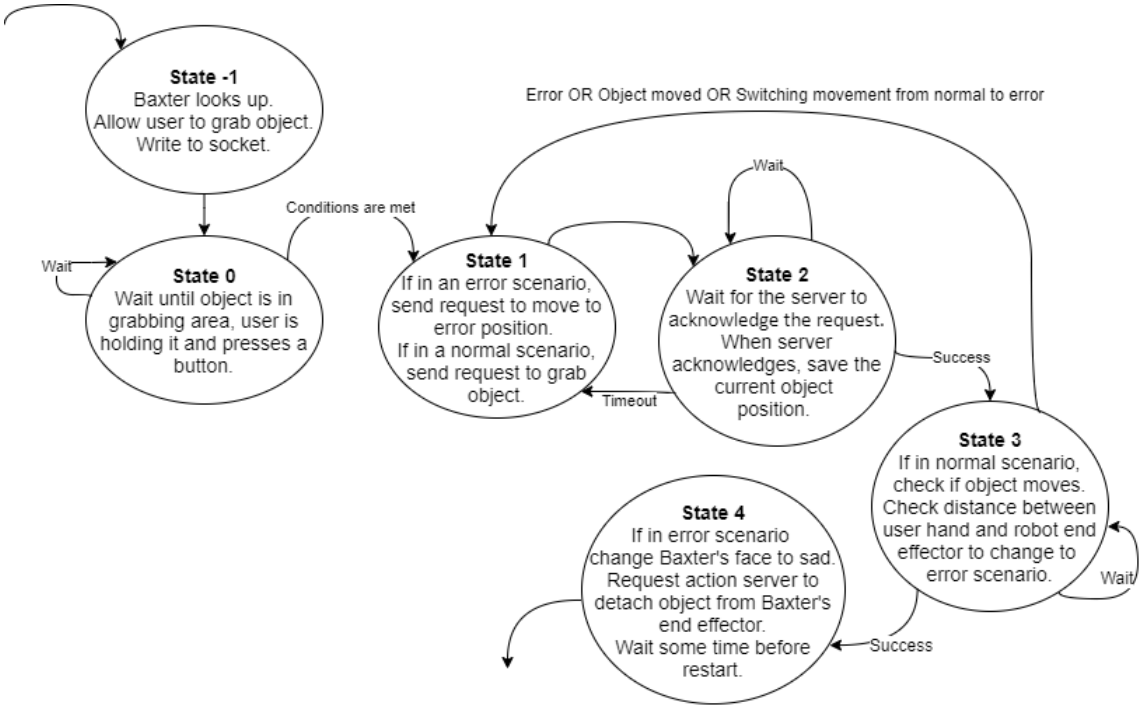


Figure 3.14: State machine to grab an object handed by the human

the error position.

When the motion of the arm is complete, either grabbing the object or moving to an error position, a request is sent to the action server to detach the object from the robot's end-effector and after three seconds, the trial ends (State 4). If in an error scenario Baxter's face is changed to sad.

3.3.2.5 Restarting the simulation

The state machine responsible for the simulation restart is presented in figure 3.15. It starts by setting the robot arm's speed to 0.7, which is fast enough not to consume too much time in the restart movement (State 0). The objects in the simulation are then positioned in their original position. A request is sent to the action server to detach any object that could be attached to the robot's end-effector (States 1 and 2) and another to move the arms to the default position (States 3 and 4). Finally, the number of trials completed is incremented (State 5).

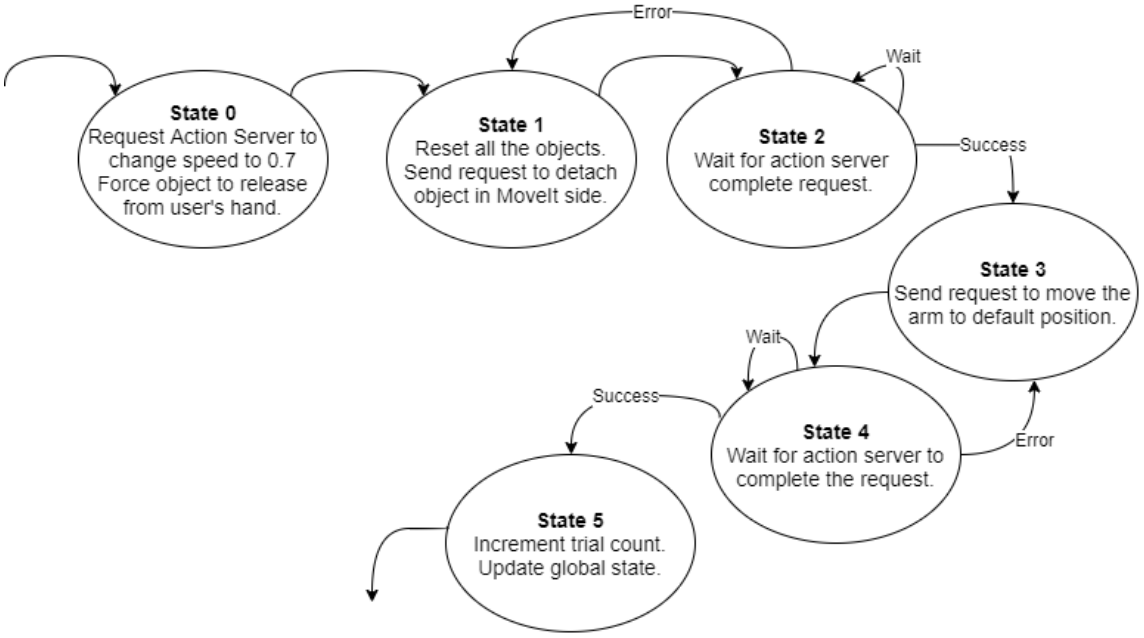


Figure 3.15: Restart state machine

3.3.2.6 Main menu

In addition to the main scenario where the experiments take place, a main menu was also created (figure 3.16) to allow for the researcher to upload the CSV file with the configurations and choose the folder where the log file will be saved. In addition, it also allows him to connect Unity to a server running on the localhost, in charge of sending signals to the EEG for synchronisation purposes. This connection is not necessary for the simulation to work. When everything is ready, the researcher can press Start to advance to the next scene.

3.3.2.7 Simulation configuration

As mentioned, the simulation is configured through a CSV file with a structure similar to the one in figure 3.17.

The NumberOfTrials defines how many trials will happen in the simulation. The timeout is the maximum time between trials in seconds. After that time, it will skip the current trial. If the value is 0, no timeout is considered. The 2 previous configurations are for the simulation itself, while the next 13 configurations are for each specific trial:

- RobotArm: indicates what arm the robot should use ('r' for the right arm and 'l' for

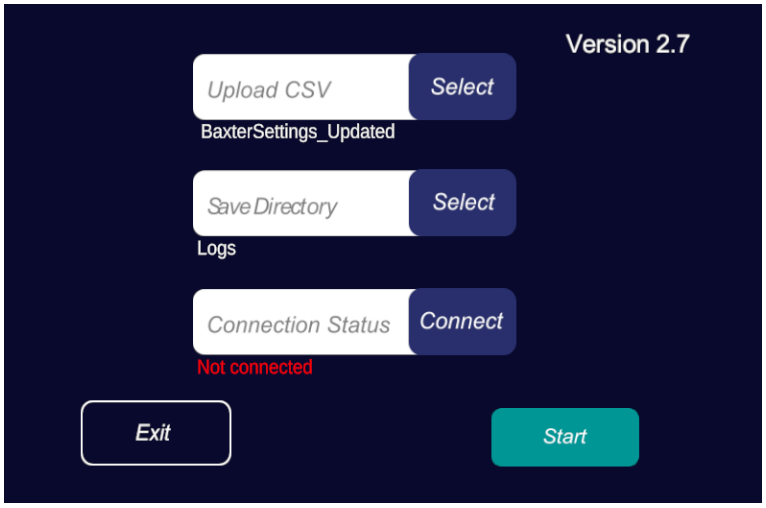


Figure 3.16: Main Menu

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	NumberOfTrials	Timeout											
2	2	0											
3	RobotArm	TrialType	RobotDistance	RobotMonitor	Speed	ErrorType	ErrorStart	ObjectType	HandType	Start_Time_Offset	User_Hand_Deviation_Angle	K (Max_Deviation)	D (Minimum_Distance_To_Deviation)
4	r	0	0	1	0.7	0	0	0	1	3	45	0.5	0.2
5	r	0	1	0	0.7	0	0	1	1	6	270	0.5	0.2
6													

Figure 3.17: Example of a CSV to configure the simulation

the left arm)

- TrialType: indicates if that trial corresponds to the robot grabbing an object from the user or the user grabbing an object from the robot. (0 for the robot delivering an object to the user and 1 for the robot grabbing an object delivered by the user)
- RobotDistance: indicates if the robot is close or far from the human (0 for close and 1 for far)
- RobotMonitor: turns on or off the robot's monitor (0 for OFF, 1 for ON)
- Speed: the speed of a robot arm (a value between 0 and 1, excluding 0. 1 corresponds to the maximum speed of the arm)
- ErrorType: the type of error the robot is going to make. 0 corresponds to no error, 1 delivers towards the user or tries to grab the object near the user and 2 delivers far away from the user or tries to grab the object far from the user.
- ErrorStart: In case of an error scenario, defines if the robot starts with an error motion (0) or a normal movement and then switches to an error motion when it gets close to the object (1).

- ObjectType: The object the robot and the user are going to interact with. (0 for an abstract object and 1 for a wrench).
- HandType: The end effector of the user. (0 for the controller and 1 for the glove)
- Start_Time_Offset: The time, in seconds, from the start of the trial to the moment when the Baxter starts moving
- User_Hand_Deviation_Angle: The angle of deviation of the user's hand when trying to grab the object in degrees.
- K (Max_Deviation) and D (Minimum_Distance_To_Deviation). Setting any of these values to 0 disables the deviation of the user's hand.

A parser was created to translate the configurations uploaded in the CSV file to information usable by the simulation. If any error is detected during this parsing, it is presented in the main menu (figure 3.18). This prevents situations where the researcher would start the simulation with incorrect configurations and possibly have to start over when something goes wrong.

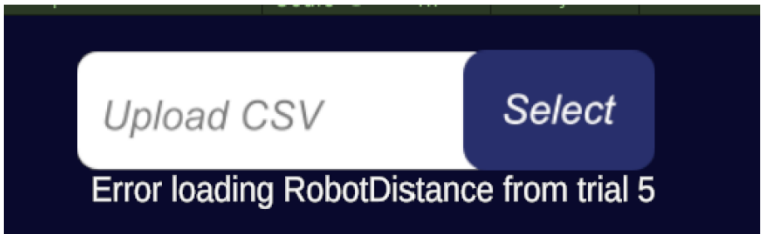


Figure 3.18: Example of the information presented to the researcher when an error occurs in the parsing of the CSV file

If no errors are detected in the configuration file, the trials configurations are saved in a randomised order.

3.3.2.8 Logging mechanism

The logging mechanism allows the researchers to integrate information such as the user's head movements in the behaviour analysis performed after each experiment. The log file is saved in the folder specified in the main menu in a CSV format.

A line is inserted in the log file in each simulation frame, which is the maximum possible recording speed.

The structure of the log file is presented in figure 3.19. It includes a time field with the format hour:minute:second:millisecond, a trial field corresponding to the number of trials ran in the simulation (first, second). Since the trial configurations from the CSV file are applied in a random order, for the researcher to know the configuration of that specific trial, the Configuration field presents that value. The positions from the right and left user's hands, Baxter's active end effector and user's head are saved in three columns each, representing the X, Y and Z coordinates. Finally, Head Rotation has four columns corresponding to the X, Y, Z and W of the quaternion relative to the rotation of the user's head.

Several events are also inserted in the log file (in the details column). Events are specific actions that occur in the simulation that can be important to the researcher's analysis. Examples of such events are a trial start, a reset of the simulation, or the robot's movement changing from correct to error.

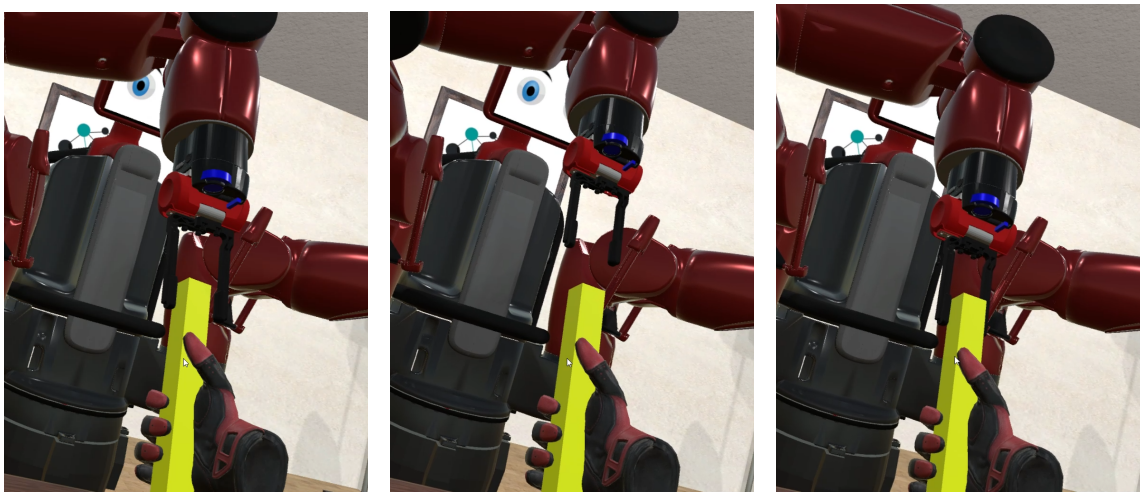
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Time	Trial	Config	Right Hanc	Right Hanc	Right Hanc	Left Hand	Left Hand	Left Hand	Baxter Active	Baxter Active	Baxter Active	Head Pos	Head Pos	Head Pos	Head Rot	Head Rota	Head Rota	Head Rota	Details
2	14:41:29:556	1	1	0,25	1	0	-0,25	1	0	-1,074481	1,139006	0,3905232	0	1,139006	0,3905232	0	0	0	1	
3	14:41:29:615	1	1	0,25	1	0	-0,25	1	0	-1,074481	1,139006	0,3905232	0	1,139006	0,3905232	0	0	0	1	
4	14:41:29:631	1	1	0,25	1	0	-0,25	1	0	-1,069251	1,118563	0,3957529	0	1,118563	0,3957529	0	0	0	1	
5	14:41:29:640	1	1	0,25	1	0	-0,25	1	0	-1,069251	1,118563	0,3957529	0	1,118563	0,3957529	0	0	0	1	
6	14:41:29:646	1	1	0,25	1	0	-0,25	1	0	-1,069251	1,118563	0,3957529	0	1,118563	0,3957529	0	0	0	1	
7	14:41:29:654	1	1	0,25	1	0	-0,25	1	0	-1,063306	1,097244	0,4016984	0	1,097244	0,4016984	0	0	0	1	
8	14:41:29:660	1	1	0,25	1	0	-0,25	1	0	-1,063306	1,097244	0,4016984	0	1,097244	0,4016984	0	0	0	1	
9	14:41:29:667	1	1	0,25	1	0	-0,25	1	0	-1,056577	1,075106	0,4084268	0	1,075106	0,4084268	0	0	0	1	
10	14:41:29:673	1	1	0,25	1	0	-0,25	1	0	-1,056577	1,075106	0,4084268	0	1,075106	0,4084268	0	0	0	1	

Figure 3.19: Example of a Log file

3.4 Results

To effectively perform the tests, the simulation was built to a final product, and an installation file was created using Inno Setup Compiler. The researchers at ProAction Lab then proceeded with the testing and were overall very satisfied with the product. Since the research is still in progress, no results from the EEG signals have been obtained yet.

The final version of the created system complies with all the requirements set by the researchers, and as such, can be considered a success. One of the most complex tasks was to have the robot act dynamically and naturally when grabbing an object handed by the



(a) Baxter failing to grab the object due to a movement from the user.

(b) Baxter readjusting to the new position.

(c) Baxter successfully grabbing the object in the new position.

Figure 3.20: Readjustments of Baxter's arms due to a movement from the user.

human. This requires various verifications, such as checking if the object moved or if it is between both grippers when they close. The final version of the system does this task perfectly, causing the interaction to feel very natural, with seamless, constant readjustments of the arm position (figure 3.20).

4

Support for the analysis of human factors when interacting with collaborative robots

The main objectives of this system are to validate the use of VR to study human-robot collaboration and analyse how distractors may affect a human performing a collaborative task with a robot. To achieve these objectives, a VR simulation is first created and integrated in the BaxterVR system and then used in a pilot study. The participants then evaluate the simulation and their reactions to the distractors through questionnaires. Furthermore, the participant's electrodermal activity (EDA) and heart rate (HR) are recorded during the experiments to complement the analysis of their reactions to the distractors.

4.1 A Collaborative Robot Scenario

An industrial environment is one of the most natural places for humans to collaborate with a robot, and as such, the designed scenario mimics one. As with any natural environment, it includes unrelated events that may distract the user, contributing to errors and fatigue.

The basis of the interaction consists of the robot picking objects from a conveyor belt and handing them to the human partner. The human, in turn, has to grab these objects and place them in boxes following a given protocol.

The Baxter robot has two different approaches for interacting with the human: it can

either hand the object and wait for the human to grab it or drop the object on the ground as soon as it reaches the delivery position (figure 4.1). Dropped objects break into pieces once they touch the ground, so the human must wait until Baxter picks and hands another one, adding another element to stress the user.

Figure 4.1: Baxter dropping an object.

The scenario is organised in increasing difficulty levels to analyse their effect on the user performance. The task of delivering an object was made purposely complex so that it requires the user to focus on it to complete a level successfully.

As previously mentioned, one of the objectives of this experiment is to understand how distractions in the surrounding environment affect the user. With this in mind, three types of events were created to redirect the participant's attention during the experiment.

Event type 1: Thirty seconds into the experiment, and every minute after that, a door opens at one side of the room, with the appropriate sound, and a robot car appears, transporting a box (figure 4.2). It moves across the room until it reaches another door, exiting the room. As the door opens, a sound of people talking on the other side gets progressively louder.

Event type 2: Another event is a box that appears in a conveyor belt on the corner of the room (figure 4.3). A loud alarm can be heard when it appears as well as the sound of the conveyor belts moving.

Event type 3: Lastly, when there are only thirty seconds left to complete the level, the clock on the wall starts ticking and flashing red (figure 4.4).

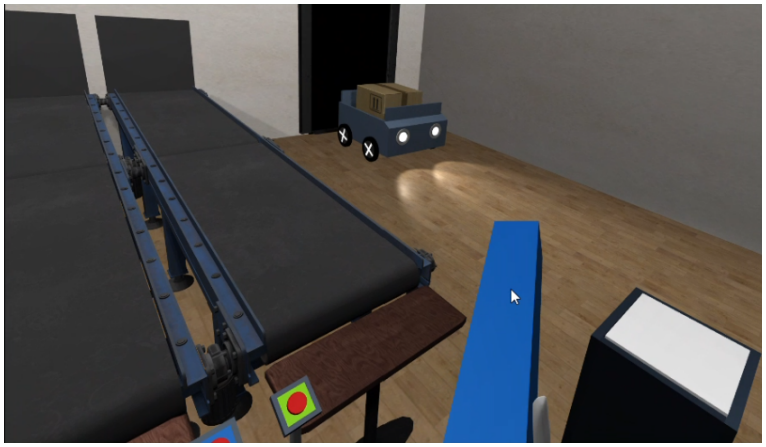


Figure 4.2: Transporter robot.

Figure 4.3: Distracting box.

Despite failing the level when the timer runs out, the user does not automatically fail the entire test. Each level is evaluated individually, and even after failing, he must deliver all the objects to pass to the next level.

This scenario was designed to be simple (figure 4.5) so that it is processed fast enough to not cause any discomfort to the user while at the same time containing elements similar to the ones found in an industrial environment such as conveyor belts or transporter robots.

As already mentioned, the main objective is to grab the objects handed by Baxter and place them inside a box (figure 4.6). The objects are simple rectangular bars and can be of four different colours, blue, orange, green or indigo and have to be placed inside a box with the corresponding colour. To obtain a box, the user must press a button with the corresponding colour.

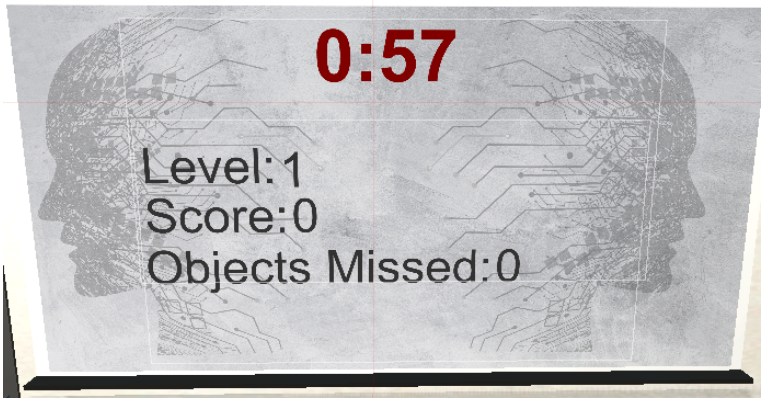


Figure 4.4: Score Board

Figure 4.5: Surrounding environment.

Once a box has one or two objects inside, to score, the user has to deliver it by placing it on top of an elevator and pressing a button (figure 4.7). On one of the walls, there is a whiteboard that displays information about the current level, more specifically, how much time is left, the score, objects missed and the difficulty level (figure 4.4).

The simulation has a total of five levels. The robot speed increases with each level, starting at 50% of the maximum speed, raising 10% per level. To successfully complete a level, the user must deliver a preset amount of objects within two minutes. The number of objects to deliver increases one per level and starts at one object.

The objects handed by Baxter arrive periodically on a conveyor belt next to it, being their sequence of colours of random order (figure 4.8). When an object gets to the pre-defined position, close to Baxter, the conveyor belt stops for a moment, waiting for the robot to grab it (figure 4.9). If Baxter starts the grabbing movement, the conveyor belt does not move until the object is grabbed, otherwise, two seconds later, it starts moving

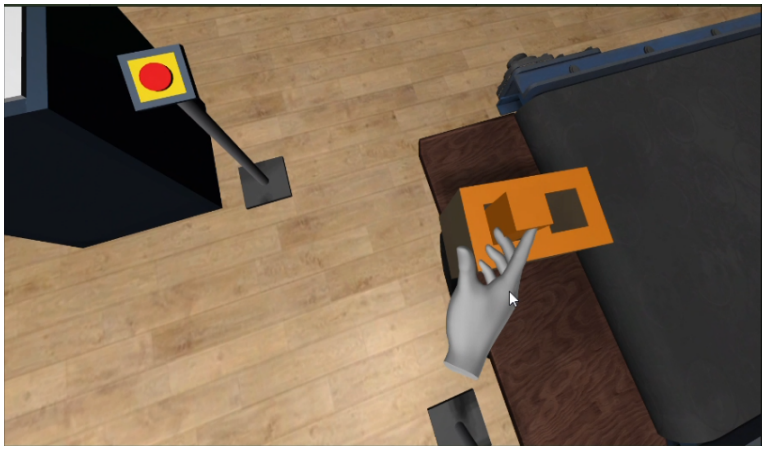


Figure 4.6: User placing an object inside the box.

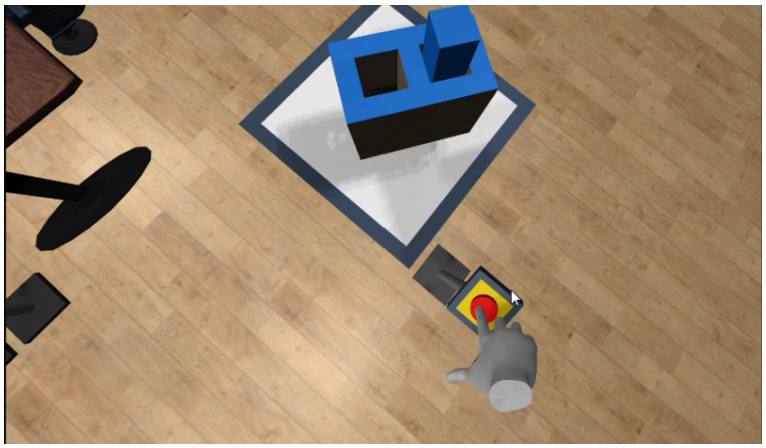


Figure 4.7: User delivering a box with one object.

again, and the object is lost. Suppose the robot picks the object and hands it to the user. In that case, the user must grab it so that Baxter can pick up another or if at level three or above, the human must pick the object before the robot reaches the delivery position. Otherwise, it will fall to the ground and get broken into pieces.

4.2 Implementation

The proposed system can be split into four main features, the BaxterVR system, which consists on the Simulated Baxter and the VR simulation, the Biological signals collection systems, and their synchronisation with BaxterVR, which is achieved using Lab Streaming Layer (LSL). The BaxterVR system presented in the previous chapter was made with reusability in mind and as such, adapting it to the presented system consisted on adapting

Figure 4.8: Conveyor belt that delivers the objects to Baxter.

Figure 4.9: Baxter picking up an object.

the VR simulation. A diagram explaining the integration of these features is presented in figure 4.10.

4.2.1 Integrating Oculus Quest

A vital part of a VR simulation is the representation of the user within it. The display of the VR scenario to the user and the capture of his actual movements and following translation to the virtual world is achieved using Head-Mounted Displays (HMDs).

As mentioned previously in section 2.5, the HMD used to create and test this scenario was Oculus Quest. Although being made mainly for wireless use, presenting high capabilities in the enclosed processing power, battery autonomy, and display quality, it offers the possibility to use a tethered version. This enables the use of high-end graphics-enabled computers both to process and display contents without requiring extensive optimisation

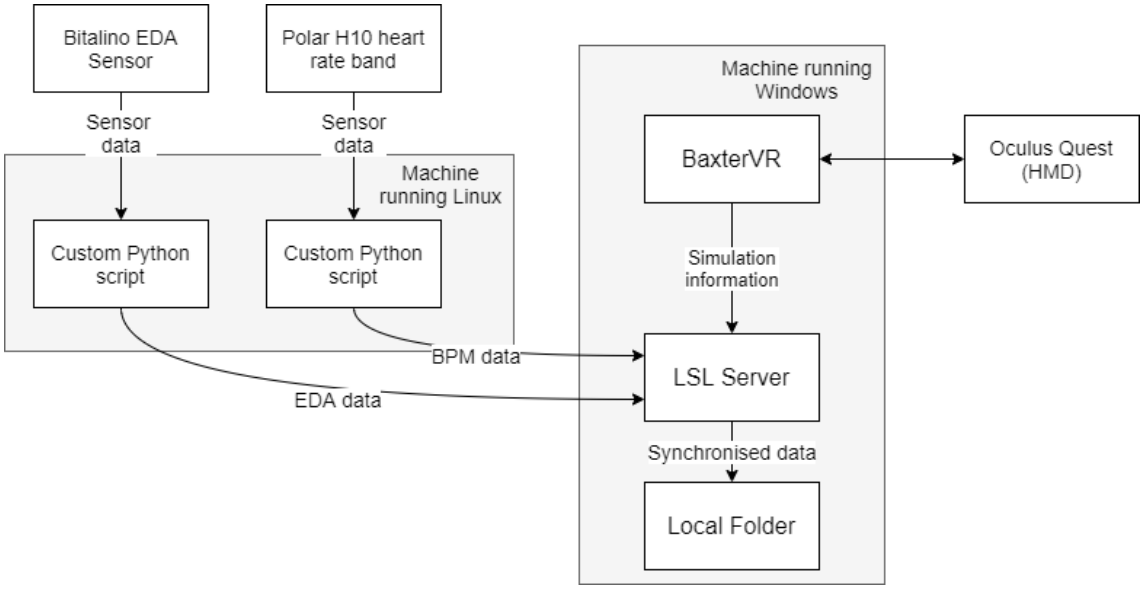


Figure 4.10: Integration and communications of the BaxterVR application with the external systems.

efforts or sacrificing any aspect of graphical quality. Another essential advantage of this PC connection is the possibility of establishing simple communications with other software packages running on the same machine. These reasons led to the development of the proposed system targeting Oculus Rift™, exploring the compatibility mode of Oculus Quest devices through a USB-C (Oculus Link) connection with the computer running the application. This is particularly convenient as it enables ROS-MoveIt, LSL-data collectors, and the VR application to share the same powerful machine and exploit simplified and reliable communications.

4.2.2 VR simulation

The flow of the simulation presented in the previous chapter is similar to a straight path. Everything happens in succession with almost no actions overlapped. As such, it can be controlled mostly by the state machines described. In contrast, the VR simulation presented in this chapter does not have a straight flow. For instance, the user can press a button while Baxter is moving to deliver an object, the conveyor next to it moves with another object, and the transporter robot appears. As such, its implementation consists of many independent features that are then synchronised by the main script.

4.2.2.1 Shapping the VR interactions

By following the steps the user needs to take to complete its task, it is possible to understand some of the implemented features that allow the user to interact with its surroundings.

Starting by the point when Baxter is at the delivery position with an object. The user has to reach over and grab it. This requires him to be able to move, see and interact with the virtual world. As mentioned, Oculus Quest is the used HMD. To integrate it with Unity, the plugin Oculus Integration was used. Similar to HTC Vive's plugin, it provides a Camera Rig object that represents the player in the VR world. To enable the user to grab particular objects, a grabber component provided by the plugin is placed on both users' hands. Each object that can be grabbed is then equipped with a grabbable component.

After grabbing the object handed by Baxter, the user presses a button to obtain a box via conveyor belt. The button itself detects when an object collides with it and moves as if pushed, moving back to its original position 0.2 seconds later. A black cube on the wall attached to the conveyor belt, resembling an entrance, is constantly monitoring the button to create a box when it is pressed. The conveyor belt then moves the box to near the user. To move the objects, each conveyor belt has a small invisible parallelepiped on top of it that detects collisions and moves any object that touches it, such as the box, in the direction the conveyor belt is "moving".

The boxes are created with the objects already placed in the holes but deactivated so that the holes seem empty. When the boxes detect that the user inserted the correct object in one of the holes, that object is deleted, and the one in the box is activated. This creates an effect where it appears that the object inserted by the user snaps into place, guaranteeing that it is always attached in the correct position. The object inserted in the box is considered correct if it is the same colour as the box and is in the list of objects the user can score. This list is used to monitor the currently active objects delivered by Baxter to the user and, consequently, can be used by the user to increase the score. These objects are automatically removed from the list when they are destroyed.

Finally, when the user places a box on the elevator, he has to press a button for the

elevator to go down. Its behaviour is similar to the one described previously. The only difference is that the elevator's platform is now monitoring its state. When the button is pressed, the platform moves down. A dark square without colliders sits in the middle of the elevator. It is only after the platform and box go through it that the box is deleted and the player's score increases. This detail makes the user feel that the box goes down the elevator and then continues its journey instead of just seeing it disappear when it reaches the bottom. Such details can have a significant influence on the immersion of the participant.

At anytime when performing its task, the user can drop the object on the ground, causing it to break. Breaking an object consists of detecting its collision with the ground and then replacing that object with a version of it made by several small cubes.

4.2.2.2 Controlling the flow of the simulation

The state machine responsible for controlling the simulation is presented in image 4.11. Its primary function is to control the simulation level, configure each one, and run the conveyor belt and robot state machines responsible for their control.

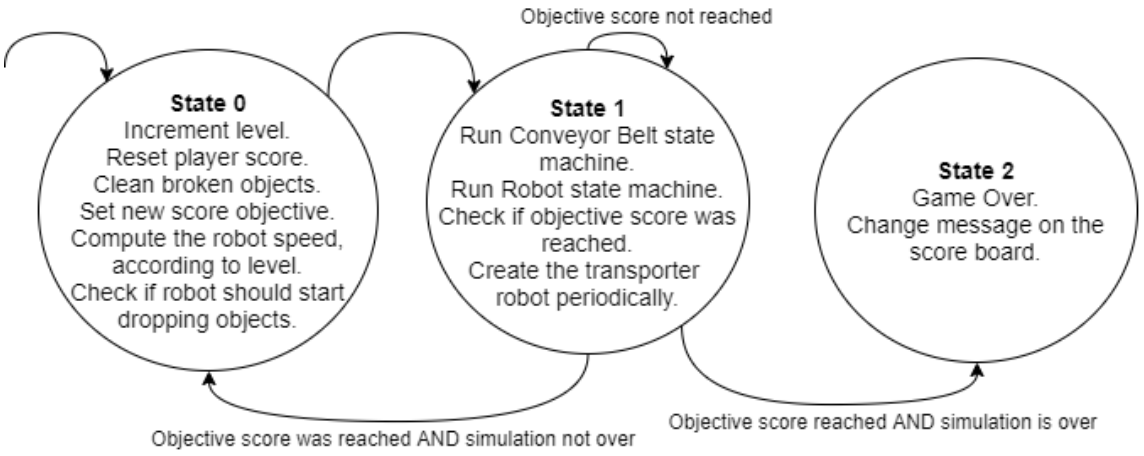


Figure 4.11: Global state machine.

The conveyor belt is controlled by the state machine presented in figure 4.12. It starts by creating an object and updating the object Baxter tries to grab (State 0). It is created inside a black cube resembling an entrance at the beginning of the conveyor belt. This conveyor belt then moves the object to a position close to Baxter using the previously described method. The conveyor belt stops when it gets close to Baxter, and a timer starts

(State 1). If Baxter does not start the motion to grab that object until the timer runs out, the conveyor belt starts moving again, and a new object is created (State 0).

Each time an object is created, it is added to the list containing all the objects that the user can use to increase its score. A black cube similar to the entrance sits at the end of the conveyor belts deleting any object that touches it, giving the feeling that it went into another room. If the object deleted is in the list mentioned previously, the counter of objects missed by the user is incremented.

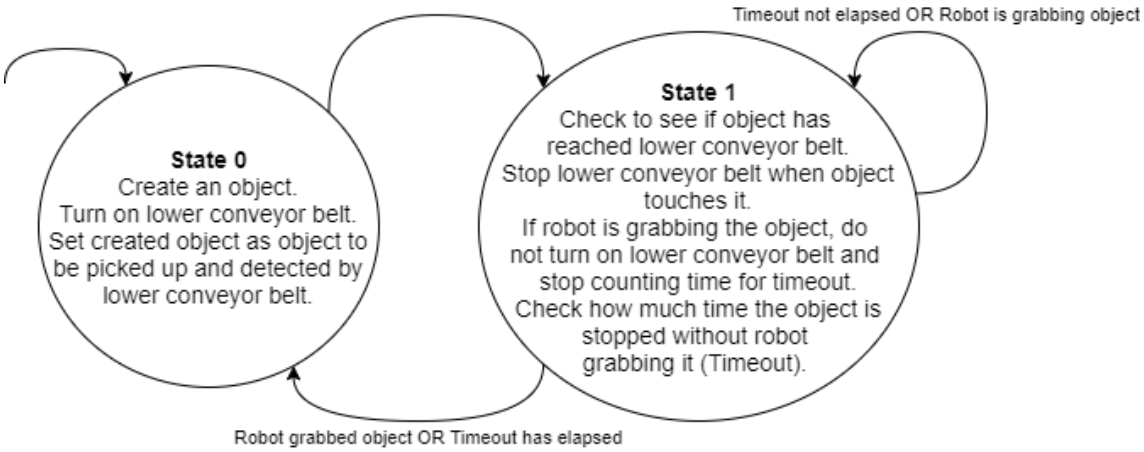


Figure 4.12: Conveyor belt state machine.

Baxter's movements are controlled by the state machine presented in figure 4.13. It starts by changing Baxter's speed to the one defined by the global state machine (States -2 and -1). Then, it waits until an object is in the correct position in the conveyor belt (State 0) and, when this happens, runs another state machine to grab the object (State 1). If this state machine fails multiple times, the object is placed in a default position, where the robot can grab it. After grabbing the object, Baxter delivers it by running the corresponding state machine (State 2). When it reaches the delivery position, it either waits until the user grabs the object or drops it on the ground (State 3) and goes back to waiting for an object to be in the correct position on the conveyor belt (State 0).

The state machines to grab and deliver the object are similar to those presented in the previous chapter. The state machine to grab the object consists of a request to the action server to pick the object while constantly monitoring its position and making a new request if it changes. At the initial state, Baxter's head turns in the direction of the conveyor belt by changing the value of the Hinge Joint connecting its screen to the rest of the body, and

its face changes to eyes looking down, giving the appearance that Baxter is looking at the object. The state machine to deliver the object consists of a request to the action server to move the end-effector to a specific position. At its initial state, Baxter's screen turns back to its original position, and its eyes change to looking straight to give the feeling that it is looking at the user.

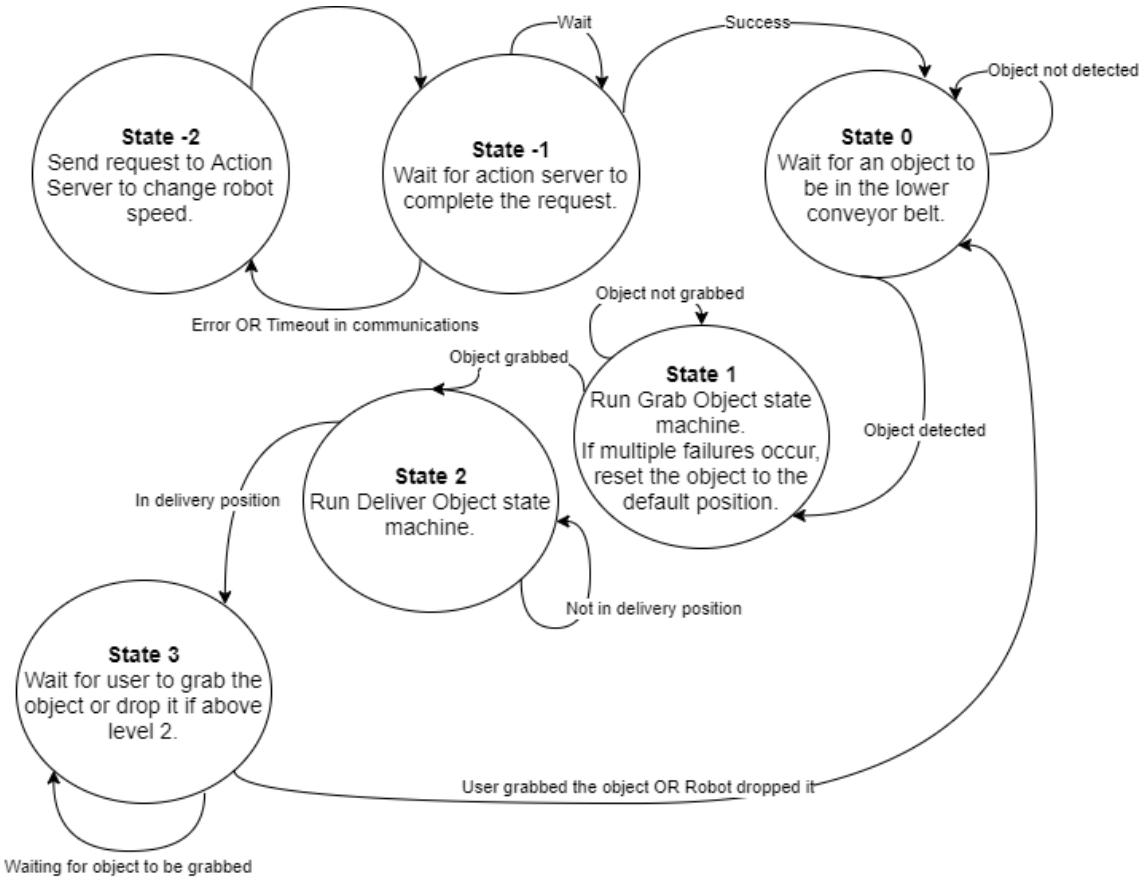


Figure 4.13: Baxter state machine.

4.2.3 Synchronous Biosignals Acquisition

To identify the user's reactions to the distracting events mentioned, both the user's heart rate (HR) and electrodermal activity (EDA) are acquired and stored. EDA is acquired using a Bitalino board [33] with a sampling rate of 1kHz. HR frequency is obtained using Polar H10 heart rate sensor. Both of them use in-house developed applications that establish the LSL bridge with these Bluetooth-enabled devices.

The development of the Unity Based application included support for Lab Streaming

Layer (LSL) to capture the evolution of hand and head coordinates and register the occurrence of the above-described events. This LSL support was also included in the development of both the Polar H10 and the Bitalino application.

The synchronous signals acquisition is handled by LabRecorder that receives the timestamped streams and stores them. One of the significant advantages of this framework is that all the recorded data streams can then be related as they share the same clock reference.

Figure 4.14 presents a diagram of data aquisition system.

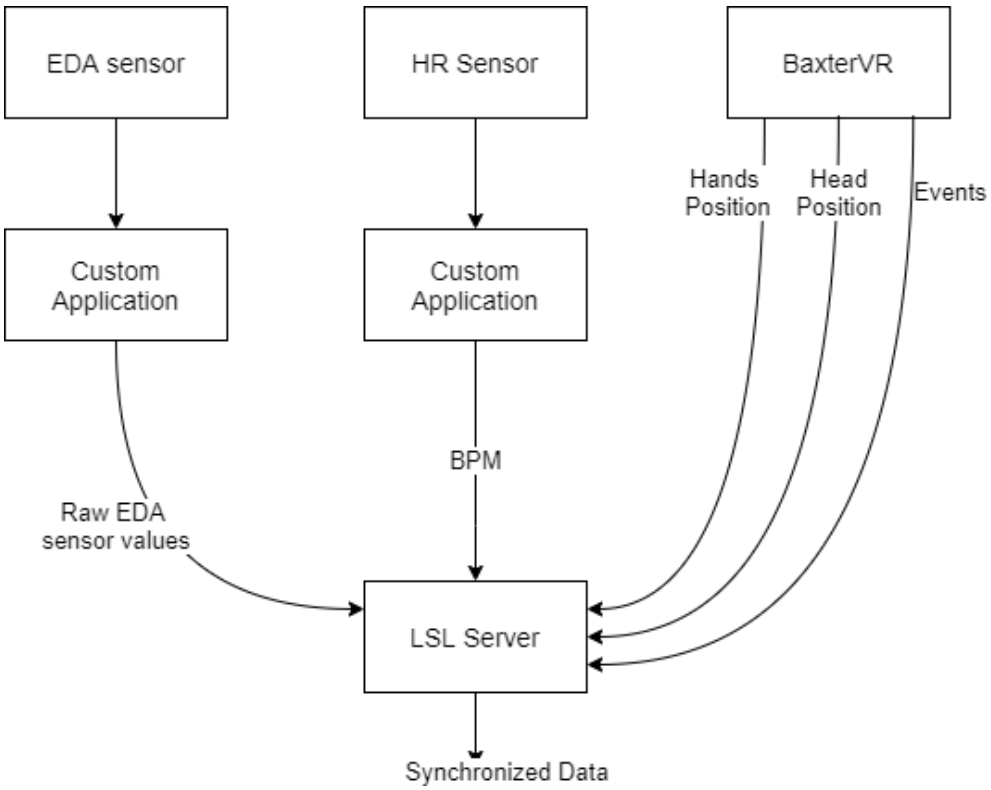


Figure 4.14: Data aquisition diagram

4.3 A Pilot Study

Using the second simulation created, a pilot experiment was conducted to understand how the users react to distractions while interacting with a robot in an industrial environment and how they feel about the interaction. This chapter aims to describe the pilot study and present the results.

In this study, the participants interact with the simulated robot described previously,

while their biological signals are recorded. In the end, they answer a UEQ questionnaire [34], a Flow Short Scale Questionnaire [35] and a few open answer questions to allow the participants to give any opinion that cannot be established by the questionnaires, such as if they had motion sickness and what was the cause.

This pilot experiment had the participation of 20 volunteer students of MSc and PhD courses, with a gender distribution of 35% female and 65% male (figure 4.15), and ages between 18 and 25 years old (figure 4.16). The participants had no previous experience with this application.

Figure 4.15: Gender distribution of the participants.

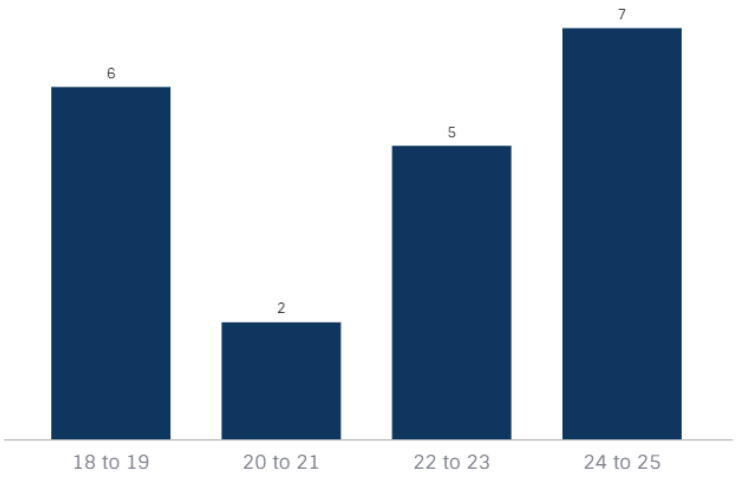


Figure 4.16: Age distribution of the participants.

The protocol followed in the experiment was the following:

1. Disinfect all the equipment used.
2. The participant is introduced to the experiment and explained what he/she has to do to conclude each level successfully and what data will be collected, and how.
3. The participant reads and signs a consent.
4. The EDA and HR sensors are placed on the participant.
5. The participant puts on a hygienic mask for the VR system and the VR system itself.
6. The simulation is started, and the participant should play it following the given rules and trying to complete each level.
7. If at any time the participant starts feeling uncomfortable with the experiment, the procedure should immediately stop. Otherwise, he/she should finish the simulation without any interruption.
8. When the simulation ends, either because the participant concluded the final level or because it was stopped early, ask if he/she wants to play again and is asked his/her opinion on the simulation.
9. The participant fills a User Experience Questionnaire and a Flow Short Scale Questionnaire to evaluate the simulation and answers four open answer questions.

4.3.1 User Experience Questionnaire (UEQ)

The User Experience Questionnaire (UEQ) is frequently used to measure the user experience with a certain product. Through 26 items with the form of a semantic differential, the data obtained can be used to evaluate the product performance in six different categories:

- Attractiveness: refers to the overall impression and how users like it.
- Perspicuity: tells how easy it is to get familiar with and learn how to use.
- Efficiency: refers to how easy it is to use without unnecessary effort.
- Dependability: translates how in control of the interaction the user feels.

- Stimulation: reveals how exciting and motivating it is.
- Novelty: indicates how innovative, creative and interesting the product feels.

The answer distribution from the UEQ questionnaire is presented in figure 4.17. Each item on the questionnaire (Y-axis) has a percentage of responses (X-axis). The responses are mainly positive, with two items (speed and pleasant) showing less good appreciation. The speed appreciation can be related to some users' perception of the robot as too slow. The pleasant appreciation can be related to motion sickness that some users felt. The participants reported both the perception of the robot being too slow and the motion sickness in the open answer questions, as will be presented in section 4.3.3. The predictability item also shows some variability in the answers; however, this does not translate to a less good appreciation because the simulation presented unpredictable aspects to the users, such as the distractors and the robot dropping the object. Lastly, it is also possible to observe inconsistencies in the creativity, annoyance and confusion categories where there is only one less good answer in each item. The most likely reason for these inconsistencies is errors made by the participants while filling the questionnaire.

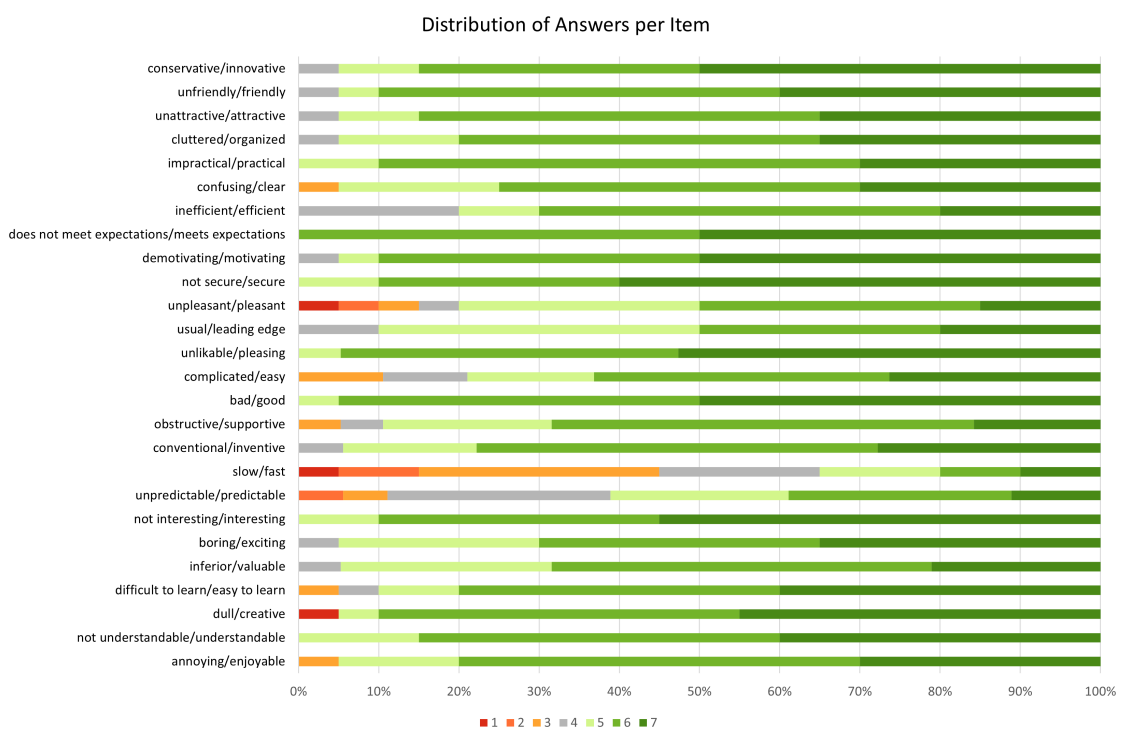


Figure 4.17: UEQ answers distribution.

The means of each category evaluated by the questionnaire are presented in figure 4.18. The values of the scales vary between -3(horribly bad) and +3(extremely good). A value above 0.8 represents a positive evaluation, while above 1.5 is an excellent evaluation. With the exception of the Efficiency category, which scored 1.5, every other category scored above this value, meaning that the system has an overall excellent rating.

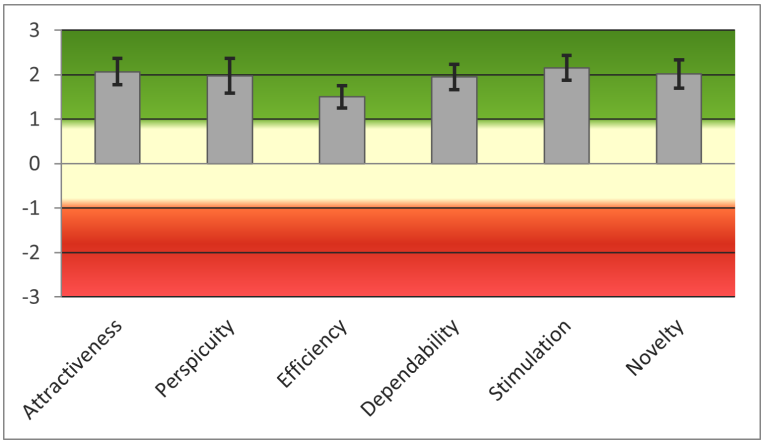


Figure 4.18: Means and deviations of the scales for the experiment.

The relatively low rating of the Efficiency category can be related to the speed of the robot. When asked about the simulation, most people said that the robot seemed slow on the later levels. In reality, the robot was moving faster than in the first levels; however, the participants seemed to get progressively better in the task they had to perform and ended up finishing it before the robot grabbed another piece. The efficiency category can also be thought of as the efficiency of the interaction since the perception of the robot being slower than it should, translates into a low efficiency rating.

Figure 4.19 presents the measured scale means in relation to existing values from a benchmark dataset containing data from several product evaluations. The highest rating categories were dependability, stimulation and novelty with ”Excellent”, then attractiveness and perspicuity with ”Good”, and finally, efficiency with ”Above Average”. These are overall excellent results indicating that the experience is considered valuable.

4.3.2 Flow Short Scale Questionnaire

The Flow Short Scale questionnaire evaluates three aspects of the simulation on a scale of 1 to 7: flow, anxiety level and challenge level. Flow level indicates if the user is feeling

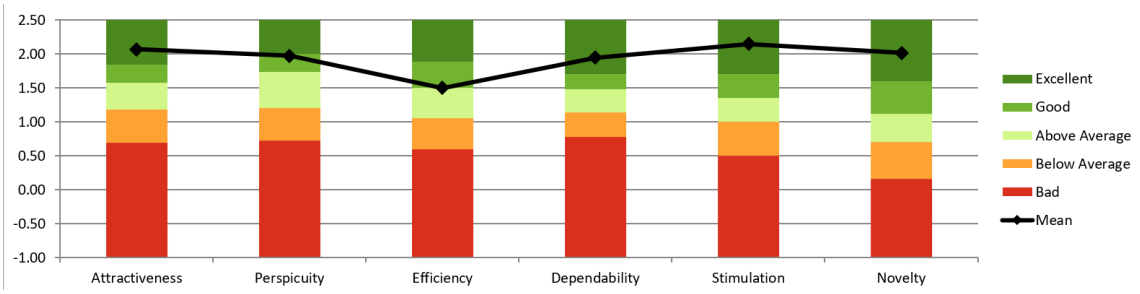


Figure 4.19: Benchmark for the experiment.

engaged and focused while engaged in the activity. If the flow level is high, the participants find the activity intrinsically interesting and take pleasure and enjoyment when involved in it. Anxiety level translates to how much anxiety the users felt and challenge level indicates if the challenges presented were too difficult (7) or too easy (1).

Table 4.1: Results from the Flow Short Scale Questionnaire.

	Average	Standard Deviation
Flow	5.525	0.667
Anxiety	3.85	1.512
Challenge	4	0.459

Table 4.1 presents an average and the standard deviation of the flow, anxiety and challenge level. According to the results, the challenge level seems to be perfectly adequate, with a score of 4 and a relatively low standard deviation of 0.458. The anxiety level has the lowest value with 3.85, meaning that the participants were not in high levels of anxiety during the simulation. However, it presents a significant standard deviation of 1.512. This is partially due to some of the anxiety scores being a lot higher than the average, with values reaching 5, 6 or even 7. Most of these high scores are from the participants that were using the joystick for moving in the VR environment, and consequently, reported motion sickness. On the other hand, most of the lowest scores on anxiety were reported by the participants that felt that the robot was too slow. Lastly, a flow score of 5.525 out of 7 means that the participants liked the activity and the simulation and felt engaged and focused, although some aspects could be improved.

4.3.3 Open answer questions and Simulation Performance

In the open answer questions, the users were asked to describe the interaction with the robot, the main difficulties, what caused the most distraction, and, at last, suggestions to improve the simulation.

When asked to evaluate the surrounding distractions, the result was that the ticking clock and some of the surrounding noises were the most distracting elements. Some participants even ignored the existence of the transporter robot completely. This is because they were focused on the interaction and the task they had, which involved much visual analysis.

As mentioned previously, concerning the interaction with the robot, most people felt like it was too slow, especially in the later levels. Some people also reported that it felt like something was wrong when the robot stopped waiting for them to grab the object, dropping it on the ground instead. This interaction is intended and is meant to test if it is more natural to have the robot behave like it is aware of the human, waiting for him to grab the object, or behaving like a traditional robot that is just doing its task without considering the collaborating partner. The results are within the expectations. Even though interacting with a non-sentient robot, the human still feels like the interaction is more natural if the robot reacts to his actions instead of doing a pre-planned task.

One of the main complaints about this experiment was related to the movement inside VR. The participants have two ways to move inside VR, either by using a joystick or moving in real life. The actual space where the experiment takes place was roughly the same size as the working space inside the simulation. This led to two kinds of behaviours from the participants, some would move in real life, concluding the experiment without touching the joystick, others moved almost exclusively with the joystick because they were afraid to bump into something in real life. This last type of movement induced motion sickness in most participants.

The open answers questionnaire also included one question where the participants were asked to evaluate their experience with VR on a scale of 0 to 6. The results are presented in table 4.2. This table also shows the performance of each participant in the simulation. The levels failed column refers to the number of levels the participant did not complete

within two minutes. Objects missed indicates the number of objects that Baxter did not grab from the conveyor belt. A higher number means that Baxter waited for the user for a more extended period. Baxter starts dropping the objects on the floor at level three and above without waiting for the user. Thus, it always grabs the objects from the conveyor belt, which means that the number of objects missed translates to the performance of each user in levels one and two. Finally, this table also shows how many objects each participant broke in the experience, including the ones dropped by Baxter. Overall, a participant performed better if these three numbers are lower and worse if they are higher. As expected, a higher experience with VR generally translates to better performance and a lower experience to worse performance.

	Levels failed	Objects Missed	Objects Broken	VR Experience
Participant 1	2	14	9	0
Participant 2	0	9	3	0
Participant 3	2	9	9	6
Participant 4	0	7	2	5
Participant 5	1	37	6	2
Participant 6	0	16	4	0
Participant 7	0	5	2	5
Participant 8	0	0	2	4
Participant 9	1	14	6	1
Participant 10	4	41	15	0
Participant 11	0	6	5	6
Participant 12	0	8	3	0
Participant 13	3	24	15	0
Participant 14	1	16	4	2
Participant 15	0	11	2	0
Participant 16	1	14	3	4
Participant 17	1	19	7	0
Participant 18	1	25	8	
Participant 19	0	7	2	1
Participant 20	5	39	19	0

Table 4.2: Performance of each participant

4.3.4 Individual EDA signal analysis

Before starting the analysis of the EDA signals, the raw data collected from Bitalino's EDA sensor is converted to microSiemens using a formula provided in Bitalino's Electro-

dermal Activity (EDA) Sensor User Manual [36]. Figure 4.20 (top) shows an example of a recorded EDA signal and figure 4.21 the HR of the same participant. The blue vertical lines on figure 4.20 correspond to a specific event: the clock starts ticking.

As one of the objectives of this work is to identify the influence of simulation events on the subjects' anxiety and stress, we expect to observe the corresponding response in the acquired bio-signals. Nevertheless, these signals do not have any absolute scale as their baselines and responses vary from person to person.

EDA signals can be divided into the tonic component, or Skin Conductance Level (SCL), which changes only slightly within tens of seconds, and the phasic component or Skin Conductance Response (SCR). The SCR represents a rapid change that happens shortly after the onset of a stimulus, usually 1 to 5 seconds, and thus, it can be used to detect a reaction to an event. When a response occurs in the absence of a stimulus, it is called a non-specific SCR.

To perform analysis on this signal, the free, open-source software Ledalab was used [28, 37]. The recorded signal is noisy and recorded at a high sample rate, so before the actual processing, it is necessary to downsample it and apply a moving average smoothing.

The filtered signal can then be analysed using a Continuous Decomposition Analysis, extracting the tonic and phasic activities (figure 4.20 (bottom)).

In the case of the example given (figure 4.20), event-related activations are detected in the first, second and fourth events. We can confirm these results through visual analysis. After the first event (figure 4.22), a sudden change in the skin conductance is verified with latency between 2 and 3 seconds. This is in accordance with the results obtained, which indicate an SCR after the first event, with a latency of 2.7 seconds.

In figure 4.23 it is possible to recognise the reaction of five participants the first time the clock started ticking. A vertical black line represents the event. In a window of 1 to 4 seconds after the event, it is possible to see a positive slope variation in the EDA values. These correspond to SCRs, which indicate a reaction to the ticking clock.

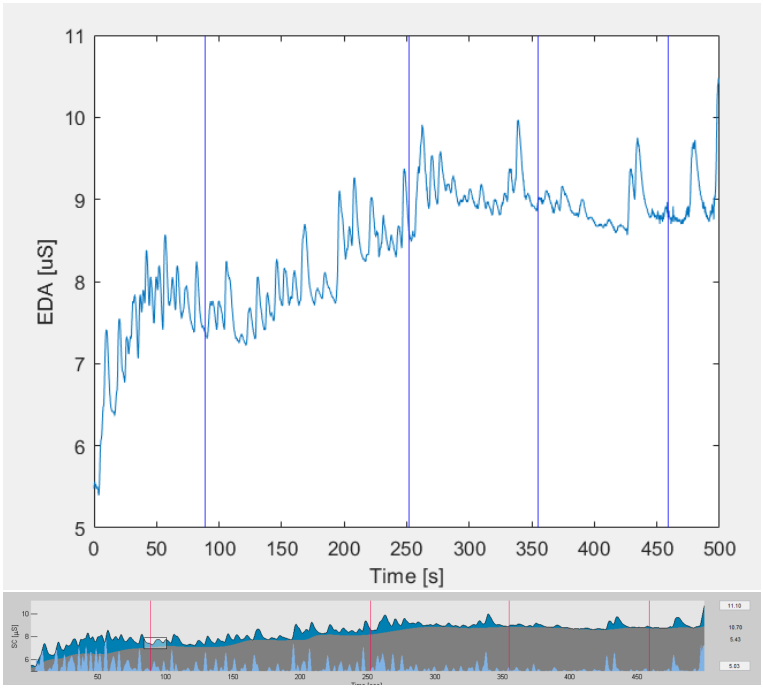


Figure 4.20: Top: EDA signal of a participant. Blue lines represent an event occurrence. Bottom: Continuous Decomposition Analysis of the EDA signal. The red lines represent the occurrence of an event.

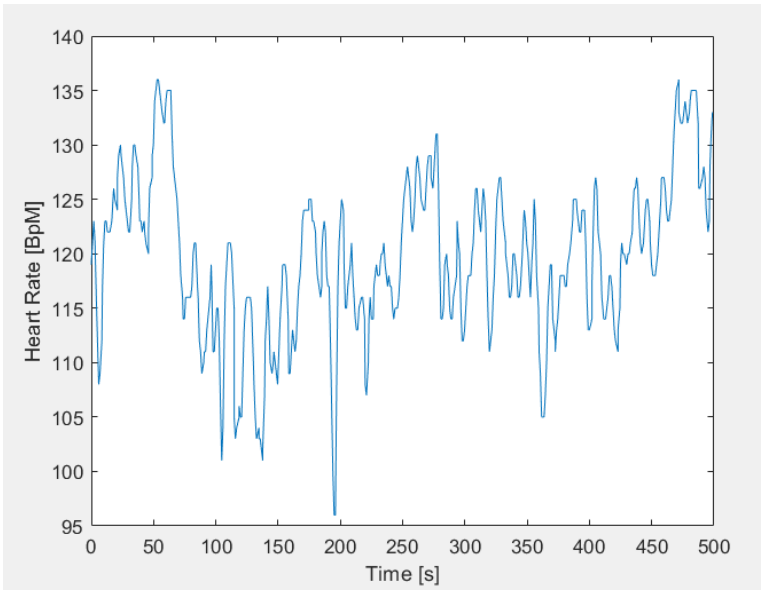


Figure 4.21: HR of a participant.

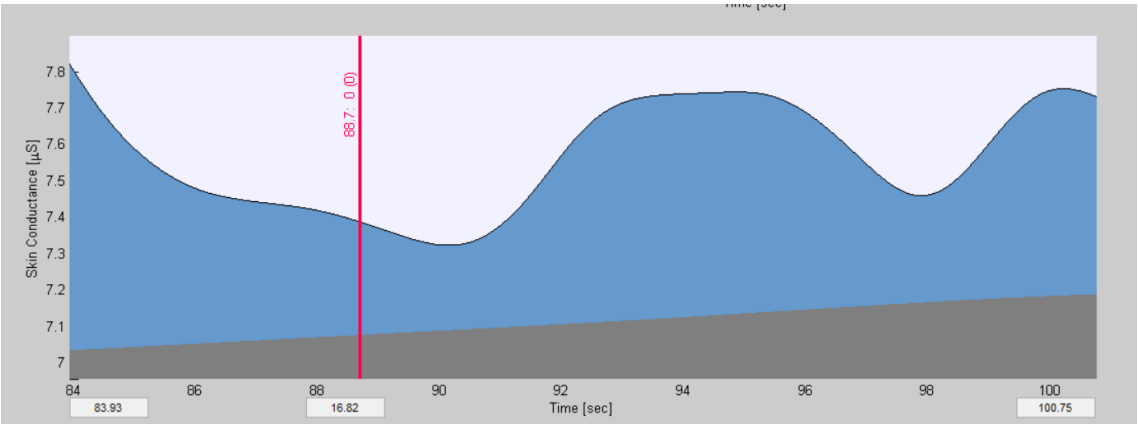


Figure 4.22: EDA response decomposition after the first event: Gray: tonic component, Blue: phasic component, Red line: event.

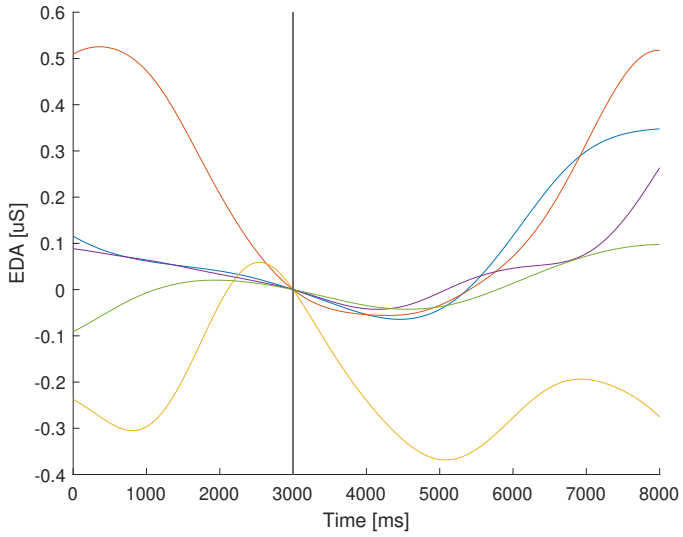


Figure 4.23: EDA responses to the clock ticking event (black line), for 5 participants.

	Reactions	Number of Events	Reactions (%)	Average Amplitude
Participant 1	3	4	75	0.2704
Participant 2	1	1	100	0.3168
Participant 3	3	4	75	1.0508
Participant 4	1	1	100	0.1128
Participant 5	1	3	33	0.1468
Participant 6	1	2	50	0.2135
Participant 7	1	1	100	0.9966
Participant 8	1	1	100	0.1245
Participant 9	1	1	100	0.2357
Participant 10	0	4	0	0
Participant 11	1	1	100	0.3377
Participant 12	2	3	66	0.0903
Participant 13	1	3	33	0.7209
Participant 14	0	2	0	0
Participant 15	1	1	100	0.1323
Participant 16	0	2	0	0
Participant 17	0	2	0	0
Participant 18	0	3	0	0
Participant 19	1	1	100	0.796
Participant 20				

Table 4.3: Reactions to the Clock Ticking Event

4.3.5 Results of the EDA analysis

Each signal was processed according to the procedure described in the previous section. The reactions of each participant to the three events are presented in tables 4.3, 4.4 and 4.5. Table 4.3 corresponds to the reactions of each participant to the Clock Ticking event, table 4.4 to the Box Spawning event and table 4.5 to the Transporter Robot event. Each table shows the total number of events that occurred in the trial of each participant and the number of reactions the participant showed to each event. It also presents the percentage of events to which the participant reacted, and the average amplitude of the event-related SCRs. The data from one of the participants (participant twenty) was corrupted, so only the data from the remaining nineteen participants was analysed.

The average amplitudes of the event-related SCRs can be compared to the average amplitude of each participant's total SCRs to estimate how intense the reaction was. Table 4.6 shows the average, maximum and minimum values of the SCR amplitudes to each par-

	Reactions	Number of Events	Reactions (%)	Average Amplitude
Participant 1	11	20	55	0.7240
Participant 2	12	15	80	0.5284
Participant 3	22	25	88	0.6411
Participant 4	9	13	69	0.3461
Participant 5	15	26	57	0.1246
Participant 6	9	17	53	0.3267
Participant 7	12	14	85	0.3718
Participant 8	13	14	92	0.3229
Participant 9	10	16	62	0.2355
Participant 10	24	33	72	0.7903
Participant 11	12	16	75	0.3442
Participant 12	6	18	33	0.3026
Participant 13	17	32	53	0.4793
Participant 14	4	18	22	0.2520
Participant 15	11	14	78	0.4365
Participant 16	9	20	45	0.2917
Participant 17	10	21	47	0.2166
Participant 18	4	21	19	0.1318
Participant 19	11	16	68	1.338
Participant 20				

Table 4.4: Reactions to the Box Event

	Reactions	Number of Events	Reactions (%)	Average Amplitude
Participant 1	7	9	77	0.5677
Participant 2	5	6	83	0.3403
Participant 3	7	10	70	0.8403
Participant 4	5	6	83	0.3509
Participant 5	6	11	54	0.1325
Participant 6	5	8	62	0.2498
Participant 7	5	7	71	0.3436
Participant 8	4	6	66	0.4363
Participant 9	3	6	50	0.3254
Participant 10	7	12	58	0.5043
Participant 11	4	6	66	0.6927
Participant 12	4	9	44	0.3134
Participant 13	9	12	75	0.4316
Participant 14	4	7	57	0.1377
Participant 15	4	6	66	0.3915
Participant 16	3	8	37	0.1347
Participant 17	1	8	12	0.0546
Participant 18	0	9	0	0
Participant 19	4	6	66	1.969
Participant 20				

Table 4.5: Reactions to the Transporter Robot Event

	Average SCR amp	Max SCR amp	Min SCR amp
Participant 1	0.5119	1.8665	0.0660
Participant 2	0.4894	1.5607	0.0571
Participant 3	0.7280	3.4311	0.0501
Participant 4	0.4416	2.1301	0.0519
Participant 5	0.1154	0.2770	0.0519
Participant 6	0.3612	1.0737	0.0552
Participant 7	0.3352	1.1059	0.0502
Participant 8	0.3538	1.0035	0.0513
Participant 9	0.2740	1.0420	0.0537
Participant 10	0.2740	1.0420	0.0537
Participant 11	0.3319	1.3052	0.0520
Participant 12	0.3198	1.4644	0.0511
Participant 13	0.4894	1.5630	0.0573
Participant 14	0.2665	1.0723	0.0501
Participant 15	0.4867	1.7965	0.0500
Participant 16	0.3216	1.2357	0.0522
Participant 17	0.2537	0.8734	0.0534
Participant 18	0.1239	0.3088	0.0529
Participant 19	0.4669	1.9946	0.0575
Participant 20			

Table 4.6: Average, maximum and minimum SCR amplitude for each participant

	Ticking Clock	Distracting Box	Transporter Robot
Bellow Average	58%	21%	42%
Within Average	21%	58%	26%
Above Average	21%	21%	32%

Table 4.7: Percentage event-related SCRs above, within and bellow the average SCR of each participant

ticipant. Table 4.7 presents the percentage of participants that showed an SCR amplitude to the event below, within or above its average SCR amplitude. It is considered above or below average if it differs more than 0.05 uS of the average amplitude.

Overall, most participants presented a high percentage of reactions to the events, with an average of 59% reactions to the ticking clock, 60% to the Box and 57% to the Transporter robot. The spawning box manifested the highest percentage of responses and the highest average SCR amplitude, although being the event that the participants reported as less distracting. This is most likely due to the sudden and intense alarm sound that played before the boxes appeared, which was the most intense and sudden auditory queue out of

all the events. Since the participants were focused on the task at hand, directing most of their visual attention to it, it is possible that they did not associate the alarm sound with the box entrance, thus inducing a reaction in the EDA signal, but not a distraction. This suggests that the stress spikes detected by the EDA signals should not be directly associated with how much the participant felt distracted but instead with how surprised they were with the particular stimulus, even if it is an unconscious reaction. Most participants reported the Ticking Clock event as the most distracting one; however, this could be due to the amount of time it was ticking (30 seconds), which can stress the participant over time but not induce an immediate high-intensity response when it starts. The Box event also exhibited the highest percentage of responses with an amplitude within or above the average SCR amplitude of each participant (table 4.7), which can also be related to it having the most sudden and intense sound.

It is also interesting to notice that the Transporter Robot event and the Ticking Clock event display most of the SCR amplitudes below average and the Box event within average. It indicates the existence of several non-specific SCRs that induced a response with an amplitude above average. Because the environment surrounding the experiments was not ideal, with many participants bumping into objects, getting tangled in the cable or being exposed to random outside stimuli such as people talking, these results are expected. It is anticipated that bumping into a tangible object while inside a virtual environment induces a significant stress response. Moreover, stimuli within the simulation, such as moving with the joystick or breaking an object, could also cause substantial reactions.

The results vary significantly between participants but are mostly consistent for each one. It is possible to identify participants that show significant EDA reactions, despite having a good performance in the simulation, such as participant 8, and others with less significant responses, such as participant 18. Participant 18 shows almost no reaction to the events, with 0% reactions to the ticking clock, 19% to the box event and 0% to the transporter robot. The most likely reason for such results is that this participant naturally does not present significant EDA responses. It demonstrates how each person reacts differently and the need to make relative comparisons to obtain meaningful results, such as considering the average SCR amplitude per participant.

4.4 Discussion

From the pilot experiment, we can extract some valuable information.

First, from the UEQ questionnaire, we conclude that the overall looks and interaction in the simulation is satisfactory. This is a crucial aspect to take into account because the more people test a product, the more reliable the results are, and having them enjoy the time spent with that product is a means to attract more people into testing it.

Analysing the answers, the high results of the attractiveness, perspicuity and stimulation categories, together with the participants' opinions, leads to the conclusion that the objective of providing an adequate VR-based alternative to real scenario experiments was achieved. As mentioned previously, the lower score on the efficiency category reveals that the interaction is less efficient than desired. Given that the robot arm is moving at the maximum recommended speed in the last difficulty level, one way to speed up the interaction and improve its efficiency would be to use both of Baxter's arms. In an ideal situation, the robot would be waiting for the human to complete his task and not the other way around. This way, the human would not feel like the robot was holding him back but instead feel like it was helping him.

The flow short scale questionnaire answers lead to the same conclusion as the UEQ questionnaire that the objective of providing an adequate VR-based alternative to real scenario experiments was achieved. The challenge lies at a perfect score of 4, meaning that the participants considered the task adequate. The flow level is good but could be improved. As mentioned previously, one of the best ways to make this improvement would be to have Baxter use both arms and speed up the interaction.

The fact that most participants almost ignored the existence of the transporter robot and the distracting box was not expected. However, it reveals that their focus was on the interaction and the task at hand, which ultimately is desirable. On the other hand, the ticking clock and surrounding sounds causing the most distraction suggest that the most efficient distractions in these experiments rely heavily on auditory queues.

Another aspect to take into account when dealing with virtual reality applications is the

surrounding space. Although having enough physical space to move around, some users still prefer to move with a joystick due to the fear of bumping into an object. In most cases, moving with the joystick causes motion sickness, mainly if the user relies exclusively on this type of motion. One detail that also induced stress to some users, essentially the same users that used only the joystick movement, was the wire that connects the Oculus Quest to the computer and the fear they would become entangled. To have the best possible results, the space that the users have to move around should be bigger than the one in the simulation. Ideally, the simulation should run using only a wire-free headset so that the users can move around freely without worrying about cables.

Concerning the biological signals, they present interesting results. They reveal the unconscious reaction the participants had to the events presented. It is possible to define two types of reactions induced. On one side, the moving robot and especially the ticking clock had a constant and stressful presence for an extended period. This is what caused most participants to feel distracted. However, their initial impact is not particularly intense, which can be seen in the biological signals. On the other side, the distracting box had the opposite effect. It was a sudden and obnoxious event that spawned over a short period but had a significant initial impact. This translates to most participants not feeling distracted but demonstrating more significant reactions in the biological signals.

5

Conclusion and Future Work

One of the main objectives of this work was to create a base system to test human-robot interaction that could be easily expanded to create custom scenarios. This objective was achieved as the base system described sped up the creation of the two simulations significantly.

The first system created is still being tested at Pro Action Lab, so it was impossible to present visible results. However, given that the researchers were pleased with the initial testing, the objective of creating a simulation that induced reactions to the participants when the robot made a mistake was achieved. This work is still in progress, and as such, further improvements or bug corrections to the system can be made.

The second simulation revealed interesting results. By recording and analysing the biological signals of the participants, it is possible to capture and understand unconscious reactions that happen when performing specific tasks and interactions. Together with the participants' opinions gathered by the questionnaires, it presents a much more comprehensive view of how each one felt during the experiment.

In the future, the base system can be used to easily create other scenarios and explore a variety of topics, such as controlling the robot through artificial intelligence. With the biological signal recording system, it is possible to receive these signals in real-time. If an analysis was made as soon as they were received, the robot could have an idea of how the participant reacted to a specific action and improve its behaviour.

One of the objectives of integrating LSL in this work was to record as much information as possible without much effort. With this in mind, when implementing this integration,

instead of focusing only on the information needed for the planned analysis, the objective was to gather as much as possible. Some of this information was not analysed, such as the heart rate and head and hand movements of the participants. In the future, the participants' heart rate could be related to the EDA signal to understand if there is a correlation between the two. The head and hand movements could be used to connect the events to specific actions of the participants. For example, link the apparition of the transporter robot to the participants looking at the door. In simulations similar to the first one, where the participant's reactions to the robot's mistakes are analysed, the head and hands position could also be used to detect escape movements.

Bibliography

[1] M. Knudsen and J. Kaivo-oja, "Collaborative robots: Frontiers of current literature," *Journal of Intelligent Systems: Theory and Applications*, vol. 3, pp. 13–20, 06 2020.

[2] H. Liu, D. Qu, F. Xu, F. Zou, J. Song, and K. Jia, "A human-robot collaboration framework based on human motion prediction and task model in virtual environment," in *2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 1044–1049, 2019.

[3] J. Krüger, T. Lien, and A. Verl, "Cooperation of human and machines in assembly lines," *CIRP Annals*, vol. 58, no. 2, pp. 628–646, 2009.

[4] B. Çürüklü, G. Dodig-Crnkovic, and B. Akan, "Towards industrial robots with human-like moral responsibilities," in *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 85–86, 2010.

[5] M. Duguleana, F. G. Barbuceanu, and G. Mogan, "Evaluating human-robot interaction during a manipulation experiment conducted in immersive virtual reality," in *Virtual and Mixed Reality - New Trends* (R. Shumaker, ed.), (Berlin, Heidelberg), pp. 164–173, Springer Berlin Heidelberg, 2011.

[6] C. D. Kidd and C. Breazeal, "Robots at home: Understanding long-term human-robot interaction," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3230–3235, 2008.

[7] M. Blow, K. Dautenhahn, A. Appleby, C. L. Nehaniv, and D. Lee, "The art of designing robot faces: Dimensions for human-robot interaction," in *Proceedings of the*

1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction, HRI '06, (New York, NY, USA), p. 331–332, Association for Computing Machinery, 2006.

[8] A. de Giorgio, M. Romero, M. Onori, and L. Wang, "Human-machine collaboration in virtual reality for adaptive production engineering," *Procedia Manufacturing*, vol. 11, pp. 1279 – 1287, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[9] E. Matsas and G.-C. Vosniakos, "Design of a virtual reality training system for human–robot collaboration in manufacturing tasks," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 11, no. 2, pp. 139–153, 2017.

[10] C. Maragkos, G.-C. Vosniakos, and E. Matsas, "Virtual reality assisted robot programming for human collaboration," *Procedia Manufacturing*, vol. 38, pp. 1697 – 1704, 2019. 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing.

[11] L. Bi, F. xin'an, and Y. Liu, "Eeg-based brain-controlled mobile robots: A survey," *Human-Machine Systems, IEEE Transactions on*, vol. 43, pp. 161–176, 03 2013.

[12] A. Al-Yacoub, A. Buerkle, M. Flanagan, P. Ferreira, E.-M. Hubbard, and N. Lohse, "Effective human-robot collaboration through wearable sensors," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 651–658, 2020.

[13] M. Sanchez-Vives and M. Slater, "From presence towards consciousness," 01 2021.

[14] B. Witmer and M. Singer, "Measuring presence in virtual environments: A presence questionnaire," *Presence Teleoperators amp Virtual Environments*, vol. 7, 07 1998.

[15] M. Slater, M. Usoh, and A. Steed, "Depth of presence in virtual environments," *Presence*, vol. 3, pp. 130–144, 01 1994.

[16] M. Slater, V. Linakis, M. Usoh, and R. Kooper, "Immersion, presence, and performance in virtual environments: An experiment with tri-dimensional chess," *ACM Virtual Reality Software and Technology (VRST)*, 06 1999.

[17] G. Clemenson, L. Wang, Z. Mao, S. Stark, and C. Stark, "Exploring the spatial relationships between real and virtual experiences: What transfers and what doesn't," *Frontiers in Virtual Reality*, vol. 1, 10 2020.

[18] G. Hoffman, "Evaluating fluency in human–robot collaboration," *IEEE Transactions on Human-Machine Systems*, vol. 49, no. 3, pp. 209–218, 2019.

[19] M. Goodrich and A. Schultz, "Human-robot interaction: A survey," *Foundations and Trends in Human-Computer Interaction*, vol. 1, pp. 203–275, 01 2007.

[20] J. D. Bayliss and D. H. Ballard, "A virtual reality testbed for brain-computer interface research," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 2, pp. 188–190, 2000.

[21] C. Claucich, L. C. Carrere, and C. B. Tabernig, "Virtual reality interface built using unity3d for rehabilitation with bci systems based on motor imagery," in *2018 IEEE Biennial Congress of Argentina (ARGENCON)*, pp. 1–5, 2018.

[22] E. Lutin, R. Hashimoto, W. De Raedt, and C. Van Hoof, "Feature extraction for stress detection in electrodermal activity.," in *BIOSIGNALS*, pp. 177–185, 2021.

[23] S. A. H. Aqajari, E. K. Naeini, M. A. Mehrabadi, S. Labbaf, A. M. Rahmani, and N. Dutt, "Gsr analysis for stress: Development and validation of an open source tool for noisy naturalistic gsr data," 2020.

[24] S. Liu and P. Liu, "A review of motion planning algorithms for robotic arm systems," November 2020. This is an author-produced version of the published paper. Uploaded in accordance with the publisher's self-archiving policy. Further copying may not be permitted; contact the publisher for details.

[25] "Ros introduction." `http://wiki.ros.org/ROS/Introduction`. Accessed: 2021-06-06.

[26] "Unity assets." `https://unity3d.com/quick-guide-to-unity-asset-store`. Accessed: 2021-06-12.

[27] M. van Dooren, J. G.-J. de Vries, and J. H. Janssen, "Emotional sweating across the body: Comparing 16 different skin conductance measurement locations," *Physiology & Behavior*, vol. 106, no. 2, pp. 298–304, 2012.

[28] M. Benedek and C. Kaernbach, "A continuous measure of phasic electrodermal activity," *Journal of Neuroscience Methods*, vol. 190, no. 1, pp. 80–91, 2010.

[29] G. Miller *et al.*, "Psychophysiological methods in neuroscience," *Psychophysiological methods in neuroscience*, 2017.

[30] M. A. Lebedev, J. M. Carmena, J. E. O'Doherty, M. Zacksenhouse, C. S. Henriquez, J. C. Principe, and M. A. L. Nicolelis, "Cortical ensemble adaptation to represent velocity of an artificial actuator controlled by a brain-machine interface," *Journal of Neuroscience*, vol. 25, no. 19, pp. 4681–4693, 2005.

[31] Q. Moreau, M. Candidi, V. Era, G. Tieri, and S. Aglioti, "Frontal and occipito-temporal theta activity as marker of error monitoring in human-avatar joint performance," *bioRxiv*, 2019.

[32] A. Hussein, F. Garcia, and C. Olaverri Monreal, "Ros and unity based framework for intelligent vehicles control and simulation," 07 2018.

[33] H. P. da Silva, J. Guerreiro, A. Lourenço, A. Fred, and R. Martins, "Bitalino: A novel hardware framework for physiological computing," in *Proceedings of the International Conference on Physiological Computing Systems - Volume 1: PhyCS,*, pp. 246–253, INSTICC, SciTePress, 2014.

[34] B. Laugwitz, T. Held, and M. Schrepp, "Construction and evaluation of a user experience questionnaire," vol. 5298, pp. 63–76, 11 2008.

[35] F. Rheinberg, R. Vollmeyer, and S. Engeser, "Die erfassung des flow-erlebens," 2006.

[36] PLUX - Wireless Biosignals, "Electrodermal activity (eda) sensor user manual." `https://bitalino.com/storage/uploads/media/electrodermal-activity-eda-user-manual.pdf`, 2020. Accessed: 2021-09-27.

[37] M. Benedek and C. Kaernbach, "Decomposition of skin conductance data by means of nonnegative deconvolution," *Psychophysiology*, vol. 47, no. 4, pp. 647–658, 2010.