POLITECNICO DI TORINO

MASTER's Degree in COMPUTER ENGINEERING

MASTER's Degree Thesis

ANALYSIS AND COMPARISON OF SPEECH-BASED TELEPORTATION TECHNIQUES FOR IMMERSIVE VIRTUAL REALITY

Supervisors

Prof. FABRIZIO LAMBERTI

Dr. DAVIDE CALANDRA

Dr. FILIPPO GABRIELE PRATTICÒ

Candidate

EMANUELE CERALLI

OCTOBER 2021

Summary

Virtual Reality (VR) is a technology which has significantly emerged in recent years in a large variety of fields (from the purpose of developing videogames to that of training), in which nowadays many companies are investing a huge amount of money. With the technology progress, Head-Mounted Displays (HMDs) have been created in order to cover the whole user's Field of View (FOV) and give him/her a real sense of immersion in the Virtual Environment (VE). Moreover, several hardware devices have been introduced to allow the user to perceive an increasing degree of immersion while he/she is operating in VR, like haptic gloves for tactile sensations and force feedback returned by the virtual objects that are present in the VE. However, many limitations are intrinsic to VR itself. For example, while for small areas it is possible, thanks to Room Scale VR, to detect and then reproduce the natural movements of the user as far as his/her translations and rotations are concerned, in a very large VEs like a hangar or a football field, it is not possible to offer a completely realistic interaction. When the physically available space is not large enough, the user necessarily has to find another way to navigate the virtual space in order to reach the desired places.

When considering locomotion in VR, several issues must be considered to give the user a comfortable experience. About that, it is important to reduce as much as possible the phenomenon of cybersickness; for example, the classical technique of using the controller's analog stick in order to move towards a specific direction in space creates a discrepancy between the visual feedback and the feedback provided by motion, producing nausea. Various alternatives have been studied during the years in order to avoid this problem. Many techniques involve a high physical effort from the user, like Arm Swinging (AS), Walking-in-Place (WIP) and the use of treadmills, but seem to offer a more realistic interaction with respect to others paradigms using controllers or gaze, like Point & Teleport. However, the approach that is mostly used today is teleportation, which allows to instantly change the Point of View of the user without emphasizing the movement. At the same time, this kind of operation requires a way to indicate the Point of Interest (POI) to be reached. Another technology which has significantly emerged during the years, not just in the context of VEs, is speech recognition. More and more people exploit

its functionalities in order to facilitate their everyday life, for example to make calls while driving or to perform other tasks in a comfortable and natural way. However, for many reasons, this kind of Human-Computer Interaction cannot be always considered very reliable. In fact, the quality of the speech recognition can be strongly influenced by the noise in the environment, or, in case of cloud services, by the possible weakness of the network connection. Despite these limitations, the use of speech is something on which many are working. For example, for the International Space Station (ISS), a prototype of CIMON, an on-board assistant for astronauts based on this technology, has been developed to let them have a more efficient communication during experiments or emergencies. Moreover, the so-called Silent Speech Interfaces, which take data from biosignals generated by the organs (e.g., muscles) involved in the speech action are being studied in order to give the possibility to people with speech disorders not to remain excluded from technological progress. The use of speech has been introduced also in many VR applications, with the purpose of substituting menus, manipulate objects and perform other kinds of tasks. A recent survey has assessed that speech is the most used interface for hands-free operations. Many times, it was found that this technology can improve the operational efficiency with respect to several other interactions methods. Among its uses, there is also teleportation.

The present study aims at finding which are the existing techniques using voice to teleport the user from one place to another in a VE. Starting from research works in the literature, three kinds of techniques have been identified as the most interesting, and two of them have been specifically considered. The first technique exploits speech alone, requesting the user to utter phrases like "Take me to the green bin" in order to directly give the system a semantic description of the POI. The second technique, instead, uses speech and head gaze together, by letting the user give a spatial indication with the latter and trigger the action by using a voice command not including a specific description of the POI; the user can say, for example, "Take me there" while pointing to the object he/she wants to reach. The third technique found in the literature behaved the same way as the last one, but used fingers for pointing to the destination; thus, it was discarded as it was considered as too similar to the previous one. In its place, a further technique was implemented, not derived from the literature, but combining the two approaches presented above in order to allow the user to give both a spatial indication and a semantic description about the desired destination and, consequently, take advantage from both; the user, however, can also use just one of the two components if he/she prefers. All these techniques exploit the possibility to solve ambiguities, which happen when two or more destinations can be referred according to the user's action. For example, with the first technique, a semantic description which can fit multiple POIs can be given and, consequently, the user must then specify a further characteristic to reach the one he/she actually intended. With the second technique, the user might

be pointing to multiple, overlapping POIs, and a choice has then to be done among them. For this purpose, a panel with images representing the options of the case (inspired from a work which introduced it in the context of gesture ambiguities) was used in these situations; in such a way, the user may easily understand which are the choices according to his/her indication.

Among the expectations from this study, there was the clear preference of the third technique, in which ambiguities could be more easily avoided thanks to the possibility for the user to provide a higher amount of information. For example, if a person utters "Take me there" while pointing to a bin which is fully occluded by an armchair, he/she would inevitably trigger the disambiguation, since the bin cannot be pointed without indicating the armchair too. If, instead, the user has, in addition, the possibility to say "Take me to the bin", the system automatically discards the armchair; on the other hand, with respect to the technique using voice only, in which, with the same phrase, the disambiguation would be triggered because of the presence of multiple bins in the scene, with the third one, only the bin that is pointed is considered and, for this reason, no disambiguation is needed.

Several tests with end-users were organized by using Unity as a game engine to create the experience and Microsoft Speech to recognize the voice commands according to a dynamically built grammar and give vocal feedback. An ad-hoc environment was built in order to give the users the possibility to experiment the possible limitations of the three techniques, as well as their advantages. For example, objects were classified depending on their dimension and were put at different distances from the user, so that the problem of pointing small objects with a certain precision could be analyzed. Objects with different characteristics in terms of placement and physical aspect were also introduced to produce a cognitive load and challenge the first technique. Overall, 15 people took part to the tests, which were performed in a within-subject mode. Participants were requested to perform eight teleportation steps, in a random order, for three times (i.e., one experience for each technique). After each experience with a given technique, they were required to fill a questionnaire composed by items from the SASSI tool (sections on system response accuracy, likeability, cognitive demand, annoyance, habitability, and speed), the SUS for usability, and a set of "custom" statements designed to study some specific aspects of interest. After having experimented all the three techniques, participants were also asked to indicate a ranking of preferences among them, with no ties. Some objective metrics were considered too, like the errors in selecting the destination for each experience and the time needed to complete each teleportation step.

Objective results indicated that errors resulted much lower in the technique using gaze without description, whereas the time needed to complete teleportation did not lead to significant differences among the three techniques. The subjective results showed that the technique using gaze without description was much less

appreciated than the other two methods in terms of likeability, annoyance and cognitive demand. Other expectations were confirmed, like the difficulty in pointing with such technique small and far objects or the lower awareness of the correct words to use with the technique using speech only. However, many unexpected outcomes were found too. For example, the hybrid technique did not take a clear advantage on none of the other two methods, only with respect to that using gaze. This result could be due to the fact that, even though its flexibility and freedom conceded to the user was very appreciated, the technique one using speech alone was found to be more natural and less confusing. It also emerged how the cognitive demand, which was expected to be much higher in the speech-only technique actually had a lower value than with the technique using gaze without description, for which the effort required in the action of correctly pointing the objects translated into a higher level of concentration required.

The latter outcome can be considered as a suggestion to improve the complexity of the environment in a further study, by putting some objects with positional and physical properties that make their recall by voice more mental demanding. Another aspect to be considered in a further study is that, since the technique using gaze resulted a bit unnatural, it can be substituted with a hand pointing-based method, especially after having noticed the importance that was given by the participants to the naturalness of interaction. Finally, it could be interesting to test the techniques in a multi-task experience, to see how much they interfere with the activity the user is focused on and understand which one is preferable depending on the case.

Acknowledgements

First of all, I want to thank my family for having supported me a lot during these years, especially in periods of difficulty.

Then, I would like to thank all my friends for the moments spent together, which helped me in taking some time off.

I also want to thank my company tutor, Federico, for his precious help during this work, and all the people who spent some of their time during the tests.

Finally, I thank my professor, Fabrizio Lamberti, and his Ph.D. students, Davide and Gabriele, for having stimulated me to do the best I could in this thesis.

Table of Contents

List of Tables

List of Figures

Acronyms

VR
Virtual Reality

VE
Virtual Environment

AR
Augmented Reality

HMD
Head-mounted Display

CAVE
Cave Automatic Virtual Environment

FOV
Field of View

CNC
Computer Numerical Control

CVE
Collaborative Virtual Environment

POV
Point of View

POI
Point of Interest

HCI

Human-Computer Interaction

IDE

Integrated Development Environment

ISS

International Space Station

SPA

Smart Personal Assistant

EMG

Electromyography

UI

User Interface

NLP

Natural Language Processing

WIP

Walk-in-Place

AS

Arm Swinging

WIM

World-in-Miniature

Chapter 1

Introduction

Virtual Reality (VR) is a technology which has become more and more widespread and in which several companies are investing nowadays. Starting from the second half of the 20th century, when Sensorama emerged as the first system able to immerse the user in a virtual world [1], in lots of fields this technology has been presented as innovative for their activities. One of those fields in which lots of people have dealt with it until now is clearly that of videogames. However, the development of consoles with VR capabilities like PlayStation and all the other VR apps that can be installed today in mobile phones, represent only a part of the VR world, which, independently from the application, is characterized by some general elements [1]. The first thing is that the Virtual Environment (VE) must be completely generated by a computer. This is what distinguishes it from Augmented Reality (AR), another important technology which, instead, adds virtual elements into the real world, that remains still visible to the user. What is also important, mostly in certain kinds of applications, is to give the user the highest possible level of immersion. In this context, the word "immersion", as mentioned by [1] indicates the factor which "makes the sensibility of the world as the user lives inside it and can touch it".

Based on this property, an important preliminary classification can be done on which are the different kinds of VR systems:

- Non-immersive systems are all those which also go under the name of "desktop VR", in which the user deals with a monitor to visualize the virtual world and exploits simple interfaces, like mouse and keyboard, to interact with it (1.2).

- Then, there are all those applications which generally include the use of a Head-Mounted Display (HMD) or exploit the Cave Automatic Virtual Environment (CAVE) technology. In this case, since the user's Field of View (FOV) is completely covered thanks to this device, he/she feels clearly more involved in

the scene that is represented. For this reason, this is called "Immersive VR" (1.1).

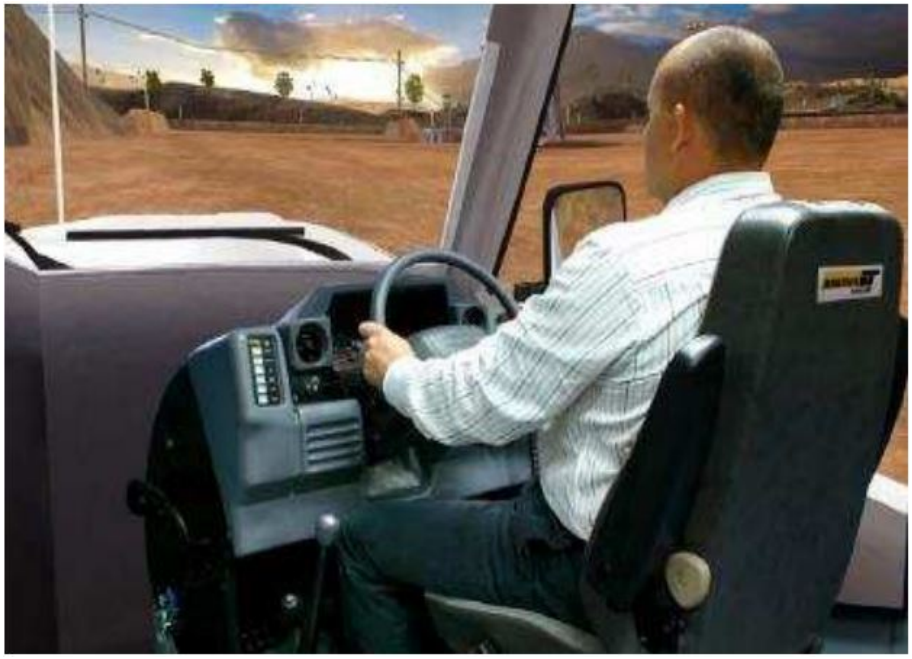

Figure 1.1: An immersive VR experience through the use of CAVE. - [1] Creative Commons Attribution 4.0 International License.

Of course, the degree of immersion that a virtual experience can give varies from one application to the other and is dependent also on the type of hardware that is used. Many devices have been in fact developed during the years in order to increase the user experience under this aspect. For example, in many applications, what can be very useful is to provide the user a tactile feedback on the objects he/she is interacting with in the VE. With this purpose, haptic gloves have been introduced, which can return both tactile sensations, like temperature and roughness, and/or a force feedback returning forces on user's hands, depending on the different objects he/she is interacting with [1]. However, the kind of application determines the different aspects of the experience that must be privileged and, consequently, the needed amount of immersion for the several senses.

As said before, VR is applied in a large variety of fields. In the context of training, for example, it can be an interesting solution to allow inexperienced people to practice the activities they will be involved in before they are put on the field, in order to achieve some confidence with tools, machines and procedures, thanks to an application which immerses the user in a virtual scenario reproducing the real one [1]. Several advantages can be achieved with this technology, including

Figure 1.2: A desktop VR experience. - [1] Creative Commons Attribution 4.0 International License.

benefits on safety [2] and costs [3].

- The user, first, avoids injuries which can happen due to inexperience. What can be considered is what happens if a person forgets a certain step in a procedure, and this carelessness potentially impairs his/her safety; if the user works in a VE, he/she is certainly out of this kind of risk.

- No damages can be brought to real tools and machines, which could be caused by an inappropriate use of them due again to inexperience; considering, for example, a Computer Numerical Control (CNC) machine, its cost can reach many hundreds of thousands of euros [4]. If a damage occurs due to an inappropriate use, companies may be forced to spend money for maintenance on it or to buy another one, besides the fact that this machine cannot be available for some time.

Another of the uses of VR that have been thought, and for which the CNC machines world can be taken again as an example, even though from another point of view, that is the one of a company which produces them, is relative to the exhibition of products. In fact, dealing with such large and heavy commodities to be carried to the specific venue can be very expensive [5]. The potentiality of VR can be to allow this companies to carry the necessary devices (e.g., headset,

controllers) in order to make the visitor perform a 360 degrees exploration on the virtual 3D model of the machine and maybe also to see a demonstration of its operation thanks to appropriate animations made with a certain fidelity, as shown in 1.3.

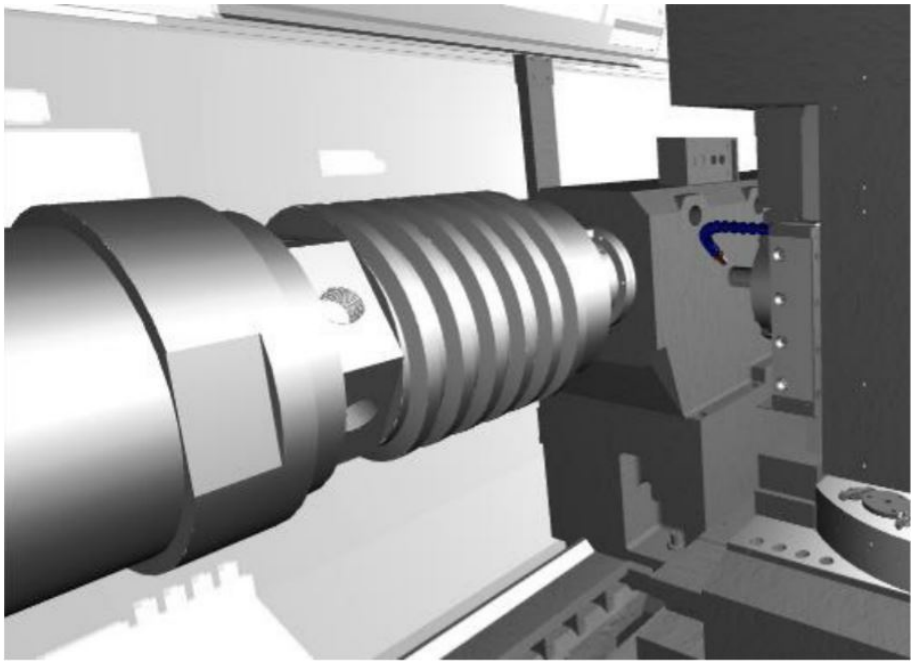

Figure 1.3: Virtual view of the internal part of a CNC machine. - [5] Creative Commons Attribution License 4.0.

In the domain of medicine too VR has been largely adopted. Besides the fact that it can be used to develop skills for surgeons [6], its use includes the treatment of many disturbs like anxiety, thanks to the fact that the user can be immersed in a world representing the feared environment and, in this way, the pathology can be somehow treated [7]. Another important aspect that has been studied is the possibility to make people interact in the same VE from different locations. Thanks to the technology progress, in fact, people all around the world can connect together and perform operations in a single Collaborative Virtual Environment (CVE), cooperating for the same purpose, for example in a training application [8].

A virtual experience, whatever is the domain that is considered, is supposed to provide the user the possibility to perform some operations. Especially, it is essential for him/her to stay concentrated on the task and not on how to use the technology itself. For this reason, it is important to give him/her a fast and intuitive way to perform all those actions which are not really pertinent to the scope of the experience and that can bother him/her (e.g., configuration of the

system, interaction with menus) [1]. One of these aspects could be the way of locomotion along large environments; besides the fact that it could represent a huge loss of time during a training session, there is also a physical limitation on it. To explain this, what must be considered is that the so-called Room Scale VR allows the user to move his/her body, reflecting the movements to the perspective on the virtual world. During his/her experience, he/she is allowed to rotate around himself/herself and there will be a correspondence between the rotation angle of the user and the rotation of the Point of View (POV) in the VE. The same happens if the user gets a bit down or changes his/her position; the same will be done by the virtual POV. With this very important feature it is in general possible to detect natural movements the user does in order to be allowed to interact with objects close to him/her. This is possible thanks to a 6DOF tracking system that can be external or even integrated on the headset, depending on the technology, understanding translational and rotational components of the movement [9].

Unfortunately, this is practically only feasible for little areas covered. In fact, despite the possibility to define the boundaries of the operational area for the user, in such a way that his/her movements into it are reproduced with a certain precision, there are the constraints given by the size of the room itself in which he/she is working. If the environment is very large (e.g., the size of an hangar) it is not possible to think that the user can have a space which is as large as it to move. This implies that he/she has to find a way to change his/her position without walking, because he/she cannot do it for long distances, since the locomotion would necessarily be impaired by physical barriers present in the room [10]. Most recent headsets like Oculus Quest allow also to make the user aware of when he/she is going out of the pre-defined operational area, in order to avoid him/her stumbling or bumping into the walls. For this reason, different techniques to reach the several points in the VE have been invented. The one that has been considered in this work is the *teleportation* [11], which is a technique that changes the user's POV without emphasizing the motion and seems to be the most suitable one in order to avoid time losses and other issues like cybersickness [12]. Anyway, leaving out the several advantages and disadvantages of this technique with respect to the others, there is a preliminary problem to solve while using it, which is how to specify the particular Point of Interest (POI).

In the years, an innovative way to interact with computing systems in general have been developed, which is the use of speech. It is very common, today, to find people exploiting voice commands to make calls, while driving, in order to avoid bothering themselves, or simply as a more comfortable way to interact with systems while doing other tasks [13]. In fact, this communication mode can be surely considered one of the most natural, since it allows the user to express his/her wills in a similar way with the one used in real life [14]. In VR application too, it has been decided to somehow introduce this feature in order to facilitate user's

interaction, as alternative to menus [15] or to execute several tasks [16], including teleportation [17, 18, 19]; in these cases, it has been often seen as a great way to improve the efficiency in the use of the application. In [18] one of the techniques also allowed to reduce the problem of cognitive load, which means that the user has to concentrate in order to find the correct words to express his/her intent. However, the speech recognition remains a technology which still presents many problems, it is not always reliable, since it depends on many factors which limit its performances (e.g., noise in the environment, amount of memory required) [14].

In this thesis it has been decided to work in an environment that is rather large, which is an hangar. The choice of the environment itself has been derived from the idea of Leonardo company, where an initial part of the work was done, to try to facilitate the experience on the virtual training for maintenance by avoiding the use of physically visible menus, which can bother the user, through the use of speech commands. The case study developed in this thesis with the VR@POLITO lab is designed to investigate a set of state-of-the-art techniques that include the use of speech: a user can activate the command which transfers him/her to a certain destination by pronouncing a sentence including an indication of it (e.g., the user says "Take me to the green bin" and he/she is transferred close to it). This method avoids the use of menus in the user view, which can distract him/her from the task, and allows a Human-Computer Interaction (HCI) which is quite intuitive, but certainly has some limitations when used alone.

For example, how can the problem of cognitive load be addressed? That means, how can the user remember the name of all the target positions to be pronounced when he/she needs to address them? This can be rather easy when a few POIs are present in the environment, but what happens when they are a lot? So, what if, instead of remembering a list of names, the possibility to directly point to them as they appear in the FOV is exploited? This seems to solve the problem, but what happens when, for example, the destination does not clearly appear in the FOV? As it can be expected, each technique has its pros and cons.

After a research on the state of the art to understand which are the techniques, involving speech, that have been used for teleportation, an environment was created using Unity3D, in order to implement an immersive application for the Oculus Quest 2 to compare them. In this application, the only purpose is to start from a place and teleport the user to another one multiple times, in order to analyze which are the issues encountered with the several techniques. In particular, among the ones that were analyzed, two have been derived from literature and the other one is a sort of "hybrid" version of the previous ones, and they are listed below.

- A speech-only technique [18], in which the user pronounces a phrase containing both the indication of the action of teleportation and the name of the destination he/she wants to reach (e.g., "Take me to the green bin", as exemplified above).

- A sensor-fusion technique [19], which uses the voice to trigger the action, while the effective destination is addressed by using the gaze. In this way, the user is able to visualize his/her head pointing to a target position and say "Take me there" and, then, the pointed object becomes the positioning target to which the person will be sent.

- The third technique is not derived from the state of the art, but developed as a union of the previous ones, which means that the user can both use them independently, as described before, but also together, with the advantage that both a spatial and a semantic information can be given, in order to restrict the range of possible addressed objects while talking with the system.

For the voice interface, Microsoft Speech has been used. As said, the environment is a hangar, with several target positions that can be reached. In order to have the possibility to efficiently challenge all techniques, taking into account the several limitations they present, a proper set of destinations was build, considering short, medium and large distances, as well as small, medium and big objects to point. In this environment, an experience was implemented, consisting in several steps of teleportation from a starting point to an indicated destination. 15 users were invited to take part to the test, which was designed in a within-subject mode, in such a way that all the people experimented the three techniques. During the experience, both objective and subjective results have been retrieved in order to assess several strengths and limitations of the three modalities. For example, the number of errors made in each experience was taken, as well as the time needed to complete each step, while average scores on the questionnaire filled by the users after each experience allowed to define the level of likeability, cognitive demand, and some other specific issues.

The thesis work has been structured in this way:

- In the next chapter, a general overview on the interaction techniques in VR, with a particular focus on locomotion, and on the speech interfaces is done, passing through examples of implementations and many comparisons which allow to understand strengths and limitations of the main ones.

- The following chapter presents the technologies that have been used in the development of the interfaces used for the study.

- The document proceeds with the design, the considerations on and the implementation of the case study.

- Then, it is made a presentation of the test procedure that has been thought in order to stimulate the limitations of the interfaces, with a group of people trying the application and returning some data.

- Once data have been obtained by the tests, these results are exposed and discussed to come to a conclusion on the study.
- Finally, the last chapter shortly presents the conclusion and lists some aspects related to this work that can be subjects for future studies.

Chapter 2

State of the art

2.1 Speech interfaces

One of the technologies that have been introduced in lots of devices in order to facilitate HCI is the use of speech. Being one of the most natural ways to communicate, it allows to easily trigger the desired action without, for example, visually being focused on the device the user is interacting with [14], or without requiring a particular knowledge of the device interface, that could be complex especially for certain kinds of people. The speech recognition tasks can be classified into two main groups [14]:

- Dictation mode, which allows to acquire a voice and translate it into a text.
- Command and Control mode, that is used when some particular expressions need to be recognized, exploiting a specific grammar to be referenced to.

Despite the usefulness of this kind of technology, there are several issues which need to be addressed to ensure success in this kind of interaction [14]. For example, the memory required to perform speech recognition is usually quite large, especially if a large variety of terms needs to be understood. When using an offline solution on a device, for example on a smartphone, what has to be considered is the amount of memory that can be exploited for this purpose; if it is too large, online solutions can also be used. However, these present a big disadvantage, which is the response time: calling the services which interpret the voice and give a response is not immediate and, moreover, the duration of the operation is not always predictable, since it is affected by the quality of the internet connection which, in turn, depends also on the specific environment in which the application itself is used.

Then, another consideration is that the place in which the user operates affects the quality of the input because, if it is noisy, the recognition task is more difficult to be performed [14]. For example, for the International Space Station (ISS), a

prototype of CIMON (2.1), an on-board assistant based on speech interaction, has been developed to be used by astronauts as a guide during experiments or emergencies, when some information could be required concurrently to the procedures they are performing and the connection with the Earth is unreliable [20]. In this case, clearly, the very noisy environment plays an important role and is an important consideration that must be taken into account for the design. The robustness of the system must be guaranteed, and, for this reason, a solution that has been proposed is to define a restricted set of instructions for specific situations to be assigned to several small systems instead of creating a single system with a large vocabulary.

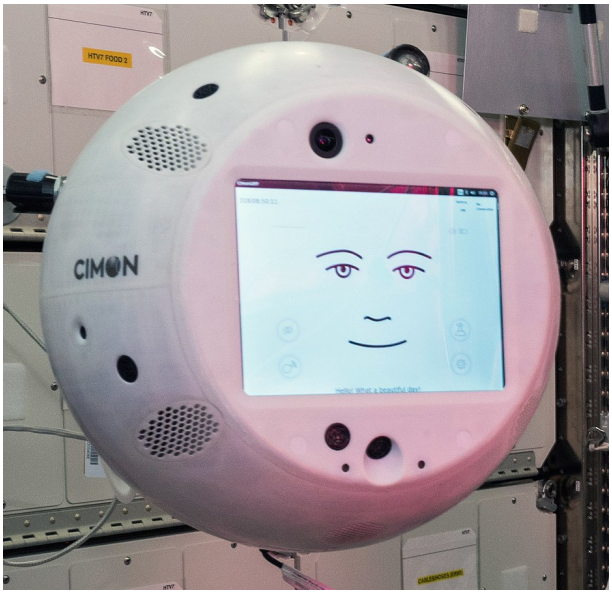

Figure 2.1: CIMON system. - NASA, Public domain, via Wikimedia Commons

Another example of HCI through the use of speech is the one described in [21], where, through a cloud-based speech processing, the human utterance is converted to an intent, which can be in turn converted to a machine-readable code for a lightweight robot. This is done by capturing the user's utterance with an Alexa Device, then, the Amazon Server is the one that allows to interpret the phrase and associate it to a specific intent, according to a defined interaction model. After that, a Control-PC associates the retrieved intent with a specific operation of the robot, which, in this case, can consists in linear and circular movements in a two dimensional plane. In this scenario, it is highlighted again how the noise is a problem which reduces the quality of the acquisition performed by the smart speaker. Another important factor is that, in this kind of task, the system needs to be very precise in order to be safe and to carry out appropriately its functionalities.

These kinds of speech services are in constant improvements since they acquire more and more data from all the users around the world; however, they require a certain time to respond and, for this reason, in a similar scenario, it could be better to use an offline, and so faster, solution.

The just mentioned Alexa Device is a typical example of a Smart Personal Assistant (SPA), which can be used by common people for several purposes. Despite the comfort it brings to the users, there are some important aspects which sometimes limit the potentialities of its use. For example, in [13] it is highlighted how, for some tasks, users do not have so much trust in the system's behavior; specifically, what is significantly considered is the aspect of security and privacy. Taking into account one of the scenarios analyzed in this work, which is the one of online shopping, it has been noticed how lots of people have a lack of trust given by the unawareness of where their data, provided by using voice, can go when transmitted.

In [22] it can be seen how the power of a HCI based on speech is such that it's being studied how to make this technology available for all people with different patterns, in particular for all those who stammer. In fact, it would not be possible for them to use nowadays technologies and several additional considerations should be done during the design of such an interface in order to avoid excluding this kind of people. More specifically, there are a lot of studies which are proposing solutions for several kinds of speech disorders through the so-called *Silent Speech Interfaces* [23]. These techniques exploit some of the several biosignals that are produced by the organs of the human body during the action of speaking, in order to capture the user's words. These can consist, for example, in the face muscles activity that is responsible of the production of the acoustic signal, which can be measured through Electromyography (EMG) [24], the same technique which has also been used in many works to improve the accuracy of the detection in noisy environments [25].

In general, a problem that can occur when dealing with speech recognition can be the one of ambiguity. In fact, it is possible that the information given by the user as a command is not sufficient to be associated to a specific response by the system. For this reason, disambiguation techniques should be implemented in order to facilitate user's life. For example, in [26], a disambiguation feature of Alexa is described. If a user wants to make a call to a person by saying "Call <name>", if <name> is ambiguous, because more than one contacts are addressable by it, Alexa asks to confirm a given suggestion, which, in this case, can include the surname (e.g., "Do you mean <name> <surname>?"). If multiple possibilities can match the name, Alexa can also propose the list of them and ask the user which one he/she wants to call. These are two of the possible methods which are used in order to avoid making the user repeat the command until something of unique is recognized.

2.2 Interaction techniques in VR

2.2.1 2D, 3D and speech

In recent years, a lot of interaction modes for virtual experiences have been developed. Many comparisons between some of them, that have been done in order to understand, depending on the specific kind of operation, which one is better and why, are exploited here to do a general overview. In [27], for example, many possible tasks have been introduced, which are selection, manipulation, positioning and rotation, creation of objects, text input, etc. For this purposes, three kinds of User Interface (UI) are used:

- First, a 2D UI which uses flat buttons with icons representing their functionality, selectable using a controller, with its orientation determining the item that the user is pointing; through a beam the direction of pointing is visible.
- Another kind of interaction is the 3D one, which aims at making the task as similar as possible to the real one. For example, the user is able to grab an object by pressing a button of the controller hold in the hand when a collision happens between the virtual controller and the virtual object in the scene, while releasing the same button the object is put down.
- Finally, a speech interface can be exploited, in which, by pressing a specific button of the controller, the voice recognition starts; the user utters the phrase and, after he/she has finished, which is detected by a silent time of a certain duration, the corresponding operation is done.

Time spent to perform the different operations were measured in order to retrieve an objective part of the result, while, in addition, questionnaires were given to the several users at the end of the experiment in order to assess the subjective quality of the three interfaces. Results showed that each of them could be the best choice for a specific scenario. For example, a conclusion that was obtained is that, when the priority is to make the user feel present in the environment and have fun, the 3D interface is preferred. The 2D one proved to be better when there are a lot of objects to be manipulated in a fast and accurate way. Finally, the speech interface presented the advantage of being easier to learn and, clearly, faster when long text inputs are used in the application.

2.2.2 Hands-free techniques

Considering the fact that in VR applications the user should often be able to interact with the environment without impairing other tasks he/she is currently doing and considering that hands are the main mean for performing these tasks, a

lot of ways to avoid the use of hands for system control, selection, manipulation, etc. have been studied in the years, as explained in [28].

This work, in fact, considers the different approaches that have been introduced in many circumstances to perform hands-free operations, that can be done in parallel with the one on which the user is focused. This work does not include the navigation task, which, by the way, will be discussed later. These kinds of interfaces are:

- Voice, going from simple standard commands to Natural Language Processing (NLP).
- Eye and head gaze, which, besides the purpose of pointing to a specific element of the scene, can include the possibility to detect eye or head movements to trigger the action. Among them, it is underlined how the head gaze is used by a major number of applications with respect to eye gaze, for its capability of being implemented without the need of additional hardware.
- Also, brain interface is very innovative, exploiting signals emitted when a user is visually focused on an element of the scene.
- Detection of face expressions or of mouth state.
- Foot movement.
- Arm position.
- Contraction of muscles, detected by EMG, which is the electrical activity produced by them.

It is clearly not always easy to integrate some of the more innovative kinds of interface considering, for example, that brain interface must be put on the user's head simultaneously with the HMD. However, the result given by this survey is that the voice is the most frequently used interface, in particular as far as system control tasks are concerned. Eye and head gaze follow it as the second most studied techniques, while a significantly smaller part of the studies use the other interfaces listed before.

2.2.3 Use of speech

Several studies have focused on how speech recognition is a powerful alternative to other input methods. For example, in [15], a comparison between a pie menu and the use of speech to access its same features is done.

The application focuses on the factory layout planning, giving the user the possibility to enable or disable a certain visualization and create items to be

introduced in the VE. Six tasks were chosen among all the possible ones to be performed, of which four of them belonged to the visualization configuration group, while the other two to the object manipulation; however, the whole set of tasks was active in order to keep the complexity of the application. The four operations of the first category belonged to different levels of the menu, in order to reproduce correctly the hierarchical structure of it, while, as far as the second category of tasks is concerned, the user should point to a certain place in the environment to make objects appear. At the end, the results of this experiments confirmed the hypothesis that speech solution is faster, but also rejected two previously made hypothesis:

- It was expected that the user experience was in general better, but this revealed to be wrong because, for example, the dependability resulted to be better in the pie menu considering users' feedback.

- Moreover, with the use of speech, a lot more errors were done, that is attributable to the fact that users should remember many domain-specific terms they were not used to know, but what was also expected was that this problem could decrease with practice.

For specific purpose applications, there are other several works that have been done testing the introduction of speech, like the one described in [16], which allows objects manipulation in a procedure of car assembly. What is done here is to visualize a car with the support of AR, and manipulate the several parts composing it. Using Windows Speech Recognition API, the user's voice input, which is a simple short phrase, is processed and compared with a list of possible phrases present in a grammar file, each one associated with a certain meaning in terms of intent. Consequently, the user is able to perform several operations in sequence with the aim of building the whole car; the main tasks introduced are to bring, select, move and assemble 3D objects together.

In the domain of maintenance, another work exploits VR with voice commands in order to perform a procedure on an engine [29]; in particular, speech is used in order to make the virtual personnel represented in the application reproduce some of the user's actions, like the one of grabbing an object. In this case, the system requires some specific rules on the command given, like the structure of the phrase, which must be composed by a subject, a predicate and an object, with a specific order and with the verb in the present tense. From this work emerged how the speech-based approach makes the interaction more natural and how, when using specific rules, the system is able to respond with a higher degree of accuracy and speed.

2.2.4 Locomotion techniques

The locomotion is the operation which allows the user to explore the virtual world and change his/her viewpoint with respect to it. Given the fact that, in general, the user can be placed in a room which has a smaller size than the VE [30], many techniques have been studied to allow this task. Currently, one of the most used is the teleportation, which is a modality that instantly changes the viewpoint without emphasizing the movement to reach the destination [11].

This technique has substituted in many cases the previous modality of using the analog stick present on the controller, which was exploited to give the direction of the navigation and move toward it. In fact, besides the fact that in many applications, especially with large environments, this can produce a useless loss of time [17], there is the big problem of cybersickness [12]. This issue is due in part to long exposure to a virtual application, but is increased due to the mismatch between the physical and virtual state of the user [12].

Eye gaze and controller point

Clearly, the teleportation mode needs a preliminary step of selection of the target destination, and this can be done by somehow pointing to it. In [31] a technique is presented and compared to a previously existing one; the latter is the classical selection using controller. In fact, from its orientation is given the point in which the line intersects a part of the environment representing the desired place to go. When a user triggers the thumb button of the controller, the teleportation mode is activated, and the place can be addressed. When it is released, the viewpoint change is performed.

The other technique is a modality for which the place is addressed by the direction the user looks at, exploiting what is called eye gaze (2.2). With the same mechanism, the user presses the button of the controller he/she holds in the hand, looks to the target place, and releases the button to transfer himself/herself.

A difference between the two methods consists in the shape of the ray: while in the first case it is an arc, in the second it is a straight line. However, after the users had the opportunity to try the application with both techniques, they were submitted to questionnaires and what resulted was that both methods presented low frustration, effort and motion sickness, with a high value of immersion and enjoyment. However, about the preferences, they preferred eye gaze as it was perceived as more natural and quicker, implying less need to think and, moreover, easier to use when destination points were more distant.

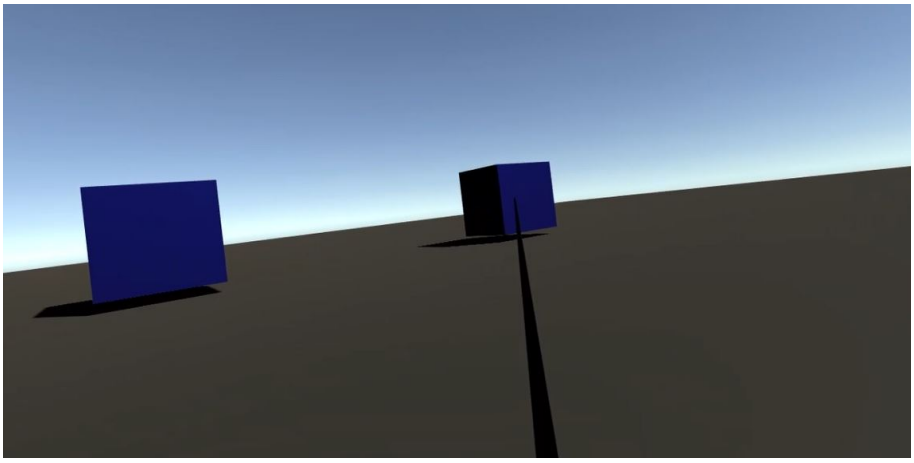

Figure 2.2: A simple implementation of the gaze technique. The ray indicates the direction in which the user is looking.

Pointing and Walking-in-Place

Nowadays, in many applications, developers choose to avoid the use of controllers. With the introduction of hand tracking technology, however, it is possible to perform tasks using virtual hands since the HMD could have the capability to reproduce user's hand using sensors integrated on it mapping the real one.

As an example, in the work presented in [11], the user can point to a specific target destination with his/her hand and, from the virtual reproduction of it, a straight line starts, which intersects the several points of the environment. The quality of the technique, that is called Point & Teleport (2.3), is compared with the one of other two modalities of selection, which are the classical locomotion using controller's analog stick and the one based on Walking-in-Place (WIP), performing an experiment in which the user passes through many places reachable with the presence of some obstacles, to better see the criticalities. WIP is a locomotion technique which uses inertial sensors to detect legs movements simulating the walking act [32]. Being a more natural kind of interaction, it can reduce motion sickness [33].

The points to be reached were predefined. The time to reach the destination was measured and other subjective parameters like usability, presence and motion sickness were considered in the questionnaires that were submitted to the users at the end of the experience.

Objective results showed that, without obstacles, the times needed to reach the destinations were similar, while, when obstacles were present, Point & Teleport time resulted much longer. This has been attributed to the fact that the way to trigger the task of teleportation consists in waiting for 2 seconds after the destination is

pointed; in fact, differently from the gaze method seen before, there was no joystick that could be used to immediately trigger the action.

Another parameter that was considered is the number of collisions, for which the significant differences were found between joystick and Point & Teleport and between Point & Teleport and WIP. In fact, Point & Teleport resulted in almost no collision happened.

About user comments, Point & Teleport was very appreciated in terms of easiness, but all the three techniques resulted optimal in terms of feeling in control. Finally, the higher effort was required by the WIP.

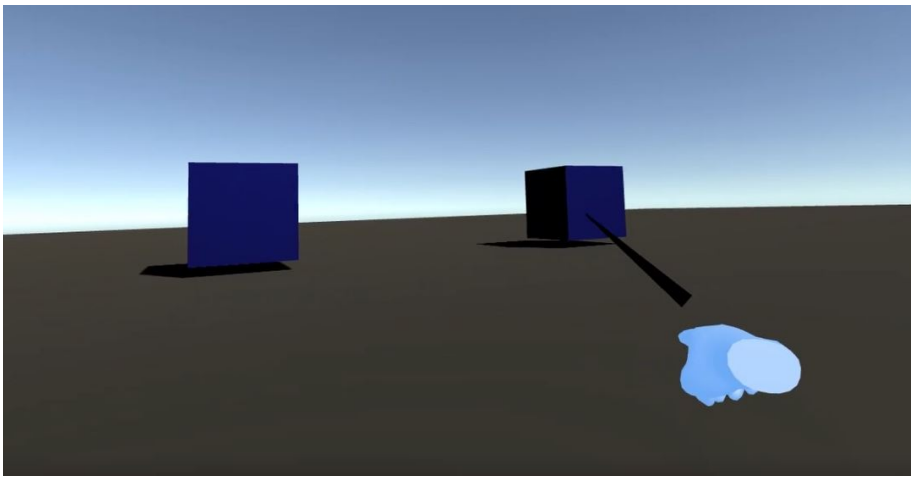

Figure 2.3: A simple implementation of the Point & Teleport technique. The ray indicates the direction in which the user is pointing with his/her hand.

Arm Swinging and treadmills

To extend the locomotion techniques overview, Arm Swinging (AS) can be considered. It consists in swinging the arms to somehow control the movement [32]. A work, [34], has compared its quality with respect to the previously cited WIP, through a sort of mini game, in which the user should navigate an environment to reach a ball and put it in a basket. The performances, clearly under the same locomotion speed, proved to be rather the same. Interesting observation have been made in terms of subjective results.

For example, WIP caused much more nausea than AS, while the latter produced a higher degree of disorientation on the user. Moreover, it was interesting to observe that the disorientation with WIP was even less than the one the user perceived in the idle state. Proceeding with the feedbacks of the users, no consistent difference was found in the two techniques as far as realism, possibility to act, and other

many subjective impressions on the quality of the interfaces. These two techniques, with respect to some of the others seen before, involve a higher energy expenditure. For this reason, it was seen which one caused the highest heart rate, and the result was that WIP required more energy [34].

AS and WIP have been applied in the context of serious games [35], for which also the use of treadmills (2.4) can be considered as another natural way of locomotion. In [36] is presented how a system which allows the user moving in any direction by walking with a particular kind of shoes sliding on a treadmill is a great solution for travelling in VEs.

A work which compares these three techniques, [35], dealt with an emergency procedure in a tunnel, in which the user had to perform different actions like stop incoming cars at a certain distance from the incident, turn off the engine, press alarm button, etc. Given the large environment, the scenario was suitable to test these kinds of techniques. The comparison was made based on subjective impressions given by the users, together with other objective stats like completion time and data about whether the user completed or not the whole procedure or a single task.

At the end, the treadmill solution seemed to be much worse than WIP and AS considering many usability factors, like locomotion, flexibility and error correction/handling, but proved to be very good in terms of simulation fidelity and interaction with objects. About the last aspect, in fact, what was considered is that in AS it resulted more difficult walking with something in the hands, being the hands the way to perform locomotion too. As far as other metrics like motion sickness and sense of immersion are concerned, no significant differences were perceived and, in an overall evaluation, it seemed to be difficult to establish the greatest technique.

Concerning comparisons, in [10], a testbed for locomotion was developed. This system, built starting from a detailed review of the literature, groups together all the important analysis factors regarding the task of locomotion in VR and, consequently, gives the user a complete evaluation method with which he/she can assess the quality of a specific technique for a given use case. This can be done thanks to the possibility to give specific weights for both the several objective (e.g., accuracy, error-proneness, operation speed) and subjective (e.g., input sensitivity, ease of use, mental effort) metrics considered and also for the importance of a specific feature of the application, in order to obtain a final ranking of the techniques.

Regard the features, what is meant is that, depending on the kind of application, some tasks are more important than others. In particular, the ones considered, which are experimented by the users, were grouped under 5 categories.

- Straight movements, starting from the request to follow a path staying as much as possible centered on a shown line along it or the one to follow

Figure 2.4: Treadmill for locomotion in VR - Virtuix,
CC BY-SA 3.0 https://creativecommons.org/licenses/by-sa/3.0, via Wikimedia Commons

a robot periodically changing its speed while trying to remain centered in correspondence of a circle.

- Direction control, which regards several tasks presenting the need of changing direction multiple times, starting from a sequence of multiple straight lines in sequence to then perform a real curved trajectory, till the request to climb a stair and a ramp.
- Decoupled movements, which aims at testing all those situations in which the user's walking direction must be independent from the ones of his/her gaze and hand movements. For example, it is requested to start walking in a direction while looking at another one, and then to control two robots taking coins beside the user with arm movements while keeping walking.
- Agility, consisting in testing the capabilities of the user to avoid collisions with incoming objects of different kinds while using the technique.
- Interaction with objects, which includes operations like grabbing objects o pressing buttons in different situations, from standing still in front of them till walking with the specific technique to follow them while moving.

As an example, the application VIVECraft [37] was considered, in this case comparing AS, WIP, Cyberith's Virtualizer (a kind of treadmill) and joystick in its use. The users were demonstrated about how to set the several weights for the

parameters of evaluation in that case, given the fact that the application, which was a game, required certain characteristics in the locomotion task. In that case, the joystick solution resulted the best, followed by AS, Cyberith's Virtualizer and WIP.

World-in-Miniature

Another very efficient technique which has been developed, especially to solve the problem of large environments to be navigated, is the World-in-Miniature (WIM). It consists in having visible an additional scaled version of the whole environment [38].

In [39], for example, a system was developed to provide a third person POV additionally to the user's first person one while selecting the position to be transferred to. It is presented how the general WIM techniques suffers from several limitations like occlusion, disorientation, scalability, and usability, and in this work some further techniques have been proposed in order to solve them, for example the external POV for disorientation and many others for occlusions, like semi-transparency, which is also the same principle used in the work of this thesis.

Navigation techniques using voice

After having seen the use of voice in virtual experiences for interactions in general and the several locomotion techniques, a focus can be done, for the purpose of this study, on the works which have been done to integrate the speech for locomotion purposes.

A first work [17] demonstrates that, since lots of years ago, the experiment has been done. This application provided virtual training for operators in switching stations, in which the user could perform in complete safety all the necessary tasks. The speech input of the user could be interpreted to perform four categories of command, including navigation, which could be done by saying a phrase like "Take me to transformer T-2". As mentioned before while introducing the positive aspects of teleportation, the real advantage of having this kind of possibility in this work was considered the avoidance of noticeable losses of time while moving along the very large environment which composed the scene. In this way, the training task could be executed in a faster way.

This kind of HCI has continued to be implemented in recent works, for example in [18], where the same approach could be used to reach target places in the environment. However, this application presented some features that were trying to facilitate the user's experience by giving him/her the possibility to perform a sensor-fusion interaction modality. The features are four:

- The first one is positioning by using the same method just seen, which uses voice alone.
- The second one is to make teleportation without explicitly pronouncing the name of the destination, instead saying a phrase like "I'll go there" while pointing to the desired place.
- The third feature is the possibility to give a name that the user wants, according to his/her preferences, to a specific object that can be addressed later, to perform a recall with that specific name.
- Finally, it is possible to perform disambiguation when two similar objects which are close can bother the selection of one of them in particular, by giving, for example, a spatial information on the one the user is interested in.

The development of this application was for Oculus Quest, which allowed the hand tracking feature without the necessity of other particular devices (e.g., Leap Motion). It was underlined how, clearly, the sensor fusion technique required the correctness in the acquisition of two kinds of input by the system.

The use of voice alone for changing the place in which the user wants to be is present in another recent work, which is [40]. In this case, the only fact is that teleportation was not implemented as the mean for navigation. The application was thought, in fact, for people with motor disabilities, with the aim of giving them a realistic experience in the museum, including the movement from a room to another one that is emphasized. A virtual agent could be asked for questions by the user himself/herself, exploiting the speech recognizer integrated in the application, referenced to a grammar file containing examples of expressions for the intents. Among the several questions, the user could express the desire of changing room to reach another artwork. To make the person aware of what he/she could see, there was a form of natural conversation: the system was able to recognize the request in different forms and not only by using standard commands, thanks to the capability of elaborating the semantic of the utterance. This interaction could be started by asking which were the artworks that were present in the museum; once they were listed to the user, the latter could choose one of them and proceed with the tour.

To complete the literature overview about the locomotion techniques using speech mixed with another interaction mode, the work [19] can be considered. Here the purpose of the application was to transfer the user in an immersive world in which he/she could perform his/her activities, basing on the fact that a certain kind of environment could induce positive emotions on him/her. Several steps were performed once the application was started:

- The environment was created according to some of the user's characteristics, which could be demographic, psychographics, regarding interests, and so on.

- After the user was inserted in the virtual world, he/she could take some predefined items to enrich the scene, for example pets, people, etc.
- Then, the tools that the user wanted to use for his/her work were inserted in the VE, thanks to a camera mounted on the screen.

Once the setup was done, the user could navigate the just composed environment by gazing at a certain point of it and saying "teleport". So, in this case, there was the possibility to improve the speed limitations present in case, not having the controller, the user needed to wait for many seconds to trigger the teleportation task; here the trigger was, in fact, the detection of the voice command.

2.2.5 The problem of occlusions

Referring to techniques which involve the explicit selection of a target object and considering the selection mode, besides the voice alone, that is used in this thesis, which is ray-casting, there is a very relevant problem with the pointing of fully occluded objects [41]. The issue is moreover how can the user be aware of their existence when they are not visible from his/her POV.

The solution of having multiple viewpoints is a still underexplored technique [41]. Another one is to make occluding objects semi-transparent in order to make it possible to see what there is behind, while the technique of directly rearrange the potentially addressed objects in a different layout in order to make them all visible is also used.

In [41], all these aspects were considered to make some further experiments, creating other kinds of techniques which could be used while addressing the problem. Some examples are:

- Use the controller to move a cursor that dynamically represents a threshold for which, when an object is closer to the user with respect to it, disappears.
- Define an area with the controller and reorganize all the objects inside it in a different layout, in such a way that they are reachable.
- Make all the objects which could be potentially pointed slightly translate in order to make all visible.

2.3 Reference works

In this thesis, it has been decided to implement techniques derived from works related to speech for teleportation purposes in VEs, in order to compare them. Since the study was related to state-of-the-art experiences, the references about the three techniques that have been implemented were the following.

- [18], which has been considered being a recent work combining a pointing (in the case of this study head-gaze is used, not pointing with hands) feature with speech. As far as the voice alone is concerned, many other works were actually found, like [42]. Talking about the technique involving pointing and speech together, no other recent work was found using this kind of fusion.
- [19], that is a recent work found in literature implementing head gaze teleportation, with voice command detection as trigger event. Similarly to what explained in the previous point, also in this case it has not been found another recent work using the union of these two inputs to perform the teleportation action. For this reason, this work has been considered another reference.

Then, [43] gave a cue for what happens in ambiguity situations, which means, for the speech, when an item is referred with a too generic definition and, consequently, can represent different items, and, for gaze, when two or more potential destinations are pointed during the teleportation request. Taking into account the fact that the cited work, which deals with gestures, used a panel representing the possible gestures deriving from an ambiguous one given by the user, showing previews of them and allowing the user to choose the one he/she meant to do by using voice, in the case of this thesis something similar happens: when multiple destinations can be the result of a teleportation request, the several possibilities (i.e., POIs) are shown in the panel and the user chooses the intended one. In order to be coherent with all the three techniques, this disambiguation method has been implemented for all of them.

Chapter 3

Technologies

In this thesis work, Unity was used, in the version 2020.2.1, in order to create the VE, implement the three techniques and manage the several steps of the procedure for test purposes. The development of the whole experience was made thanks to C# language, that is supported by this engine. In particular, the code was written by using Visual Studio 2019, an Integrated Development Environment (IDE) which can be set as default editor from the Unity interface itself, in order to have a more comfortable interaction.

For the purposes of the speech recognition, Microsoft Speech was used. The two parts (the recognizer and the VE) communicate through a client-server architecture.

The headset is the Oculus Quest 2, which was interfaced to Unity thanks to SteamVR tool.

3.1 Unity and C#

The choice of Unity as the game engine for this thesis work was done according to several criteria, which made it resulting more suitable than the other largely used one, which is Unreal:

- The latter is more complex as far as the programming language is concerned. In fact, it uses C++, which stands at a lower level of abstraction than C#. Moreover, also the Speech recognition application was developed in C#, so, by using the same language, it would be easier to interface the client side with the server one.

- Unity's easiness of use is also due to the interface [44], which allows to perform many operations, like variables initialization, in a very intuitive way, as explained later.

- This complexity could be a compromise if photorealism was needed. Unreal, in fact, is known for its capabilities of higher graphical quality [44] with respect to Unity. Since this work is an experiment which aims at testing teleportation techniques, the easiness of implementation was preferred.

- Another aspect that was taken into account was the community. Besides the great documentation that is present online, with a very detailed description of each class with its properties and methods, Unity has a larger community which can be very useful when trying to solve a problem. It is very common, in fact, to find other people which have already encountered the same issue before and explain how to deal with it or give a cue to better understand why something works in a certain way.

3.1.1 Unity

As mentioned before, Unity [45] presents a very intuitive interface, as can be seen in Figure 3.1. In fact, once a project is created, several windows appear on the screen to be used for several purposes, which can clearly be organized in different ways according to the user's preferences.

In the center of the view, the *Scene* panel provides an overview of the environment that is being created by the user. The latter inserts the so-called *Gameobjects*, which are the several elements that can have a functional role in the environment or even just represent a visual presence in it: in fact, what gives a gameobject a certain behaviour is a set of properties which go under the name of *Components*, that can be added progressively by the user to it starting from something that is empty.

If we take as an example the lighting of an environment, this is obtained thanks to a gameobject, or a set of gameobjects, which own a specific component called *Light*. In the same way, the possibility that the user has to have a POV in the scene when the application starts is given to the fact that there is a gameobject owning the *Camera* component. To make a further example, the fact that an object is physically visible in the scene is possible thanks to the *MeshRenderer* component, and so on. Each component, moreover, includes several properties which are specific for it (e.g., the intensity, the color, etc. for the Light component).

One of the most important components in a VR application is certainly the *Collider*, which behaves as a boundary of a certain portion of space that, when collided, is able to trigger events. This because, in a C# script, it is possible to evoke an ad-hoc function whose content is executed once a collision is detected on the gameobject which owns it as a component. The majority of events are given by collisions, even when a physical interaction between objects is not visible by the user.

Talking about C# programming language, what needs to be said is that a specific version of it has been developed for the use in Unity. In fact, when a script is created directly from the Unity interface, what is automatically imported, thanks to the *using UnityEngine* directive, is a namespace including the definition of additional types, functions, etc. that are not present in the standard C# language and that can be exploited for VR development.

A very important aspect regarding the ease-of-use of Unity interface is the drag and drop mechanism. Besides the fact that this can be done to insert 3D models in the scene from the *Asset* panel, which is the section containing all the predefined elements that can be introduced, whenever the user wants and multiple times, in the scene, it also allows to facilitate the association between the several variables declared in the scripts and the gameobjects that are present in the scene. In fact, when a script is associated as a component to a gameobject, it is possible to visualize and to assign with a value (or with an object in general, depending on which is the type considered) all those variables which have been declared in the code as *public* or with the *SerializeField* directive. For example, when declaring in a script a Gameobject on which some kinds of operation must be performed, it is possible to simply drag and drop the corresponding item that has to be represented by it during the execution of the program in the field next to its name, without the need of matching the correspondence by coding.

Unity has an online store, the *Asset Store*, which allows to download free or buy many items, like 3D models, and import them in the project. The engine is not intended to create 3D objects (except for very simple ones like cubes, spheres, cylinders, etc.), which must be modeled with other specific programs and then imported in an FBX or in an OBJ format. What can be done is also to create materials to be assigned to the models; however, specific programs to perform this in a more accurate way, which brings a higher graphical quality, do exist.

Another feature that can be introduced is the one of animations: it is possible, in fact, to create a flow of different kinds of animation with a certain relationship, in such a way that each of them is triggered as a consequence of certain events during the game execution. Also in this case, the single animation can be created by specific programs if it is intended to be very complex (e.g., human body movements), only simple ones (e.g., translations, rotations) can be done by using Unity itself.

At the end of the development, it is possible to create an executable file of the application that can be run on multiple platforms.

3.1.2 Visual Studio

Visual Studio [46] (3.2) is an IDE available for Windows and Mac, which can be used to edit, debug, and build code. It allows developing scripts by using many different programming languages like C++, C# and Python.

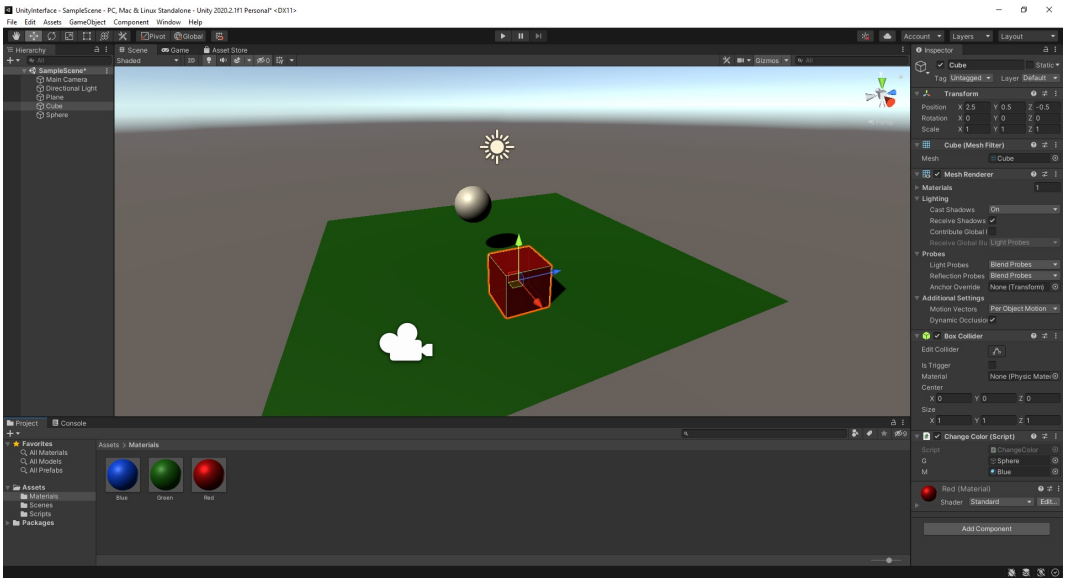

Figure 3.1: Unity 3D interface.

Besides the Microsoft documentation available on the web, in order to make the user more productive while coding, several features are available in this program:

- First, the code syntactic correctness is checked in real time, without the need of pressing a particular button to discover errors. If there are problems, the specific piece of code is marked with wavy underlines and a window below can be exploited in order to understand the reason of each of them with the corresponding line number.
- The *IntelliSense* feature is available to facilitate the user's acknowledge of what is inside the class he/she is currently dealing with. When he/she wants to access a member of a class, a box appears with the list of all of them, from which he/she can choose. Moreover, while each of the suggested solutions is currently selected, a short text description on what it represents is shown.
- *Quick Actions* are proposed by the system when an error is present somewhere: an icon with a light bulb appears with a suggestion on what can be done to solve it.
- *Refactoring* feature can be used to perform some organizational modifications to code, such as renaming variables, including pieces of code in a new method, etc.
- When a method is used, it is possible to directly move to its definition by right clicking and selecting the appropriate option, in order to reach the file and the

specific lines of code in which the function is described to better understand what it does.

- Finally, it is possible to group several projects together in a single solution, and these are all loaded when the solution itself is opened.

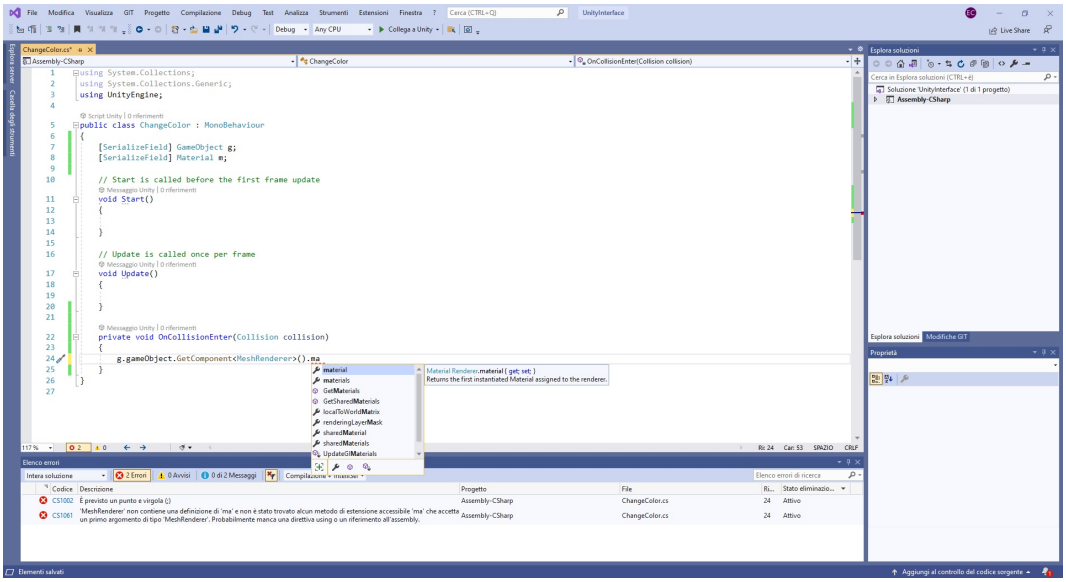

Figure 3.2: Visual Studio 2019 interface.

3.2 Microsoft Speech

The system that has been used to recognize the user's voice input is Microsoft Speech [47], which was chosen being a fast solution, since it does not exploit external services which may cause a certain latency. By writing some C# code, it is possible to instantiate a speech recognition engine based on the desired language and to load onto it a specific grammar referred to a GRXML file.

Microsoft Speech also allows to synthesize voice, using the ones installed in the device, which can be used together with the speech recognizer part to perform a HCI based on vocal conversation.

The features present in the libraries associated with the recognition part include different kinds of events related to when a general noise has been detected or when a specific phrase which was included in the grammar has been recognized. Moreover, data on what is the confidence with which the recognition was done is performed, referring to the probability, with respect to the other phrases planned in the grammar, that the detection was correct.

3.3 Oculus Quest 2

The Oculus Quest 2 [48] (3.3) is a device that belongs to the category of all-in-one headsets, which have introduced the possibility for the user to perform experiences even without the mandatory need of having other hardware to be linked with it, as happens with Oculus Rift S, for example, that must be physically connected to an external computing system. Specifically, it is not necessary to execute the application with an external computer, instead the executable file can be directly uploaded into the headset and executed from there, thanks to the presence of a memory up to 256 GB. For this reason, no wired connections are needed.

Moreover, it exploits an inside-out 6DOF tracking system, which means that no external sensors are needed to configure the space in which room-scale interaction is detected, since they are on the headset itself. Once the device is started, it is possible to manually define, using the controller (3.4), the level of the floor and the boundaries for room-scale interactions. During the experience, a possible violation of the previously defined area is made known to the user which, while getting closer to the boundaries, starts seeing the grid which defines them and the vision of the VE is substituted by the one of the real world, instantly recorded by a camera on the headset itself.

This headset also allows the hand-tracking feature, avoiding the use of the controller to detect a gesture and to perform some kinds of interactions.

Despite the possibility of using the stand-alone mode, it is still possible to adopt the use of a solution, the Oculus Link (3.5), with a cable to be connected to a computer in order to exploit its performances and have better quality. The cable should be at least 3 meters long to ensure good results and the computer too needs some specific requisites, like having a Windows 10 operating system installed and a RAM of at least 8 GB. Moreover, there is a wireless alternative to Oculus Link, the Air Link, for which the headset exploits the computer's computational power without physically being connected to it.

The Oculus Quest 2 has an LCD display and a resolution of 1832x1920 per eye and it is possible for users who wear glasses to keep them even during the experience. The immersion is guaranteed also in terms of audio, which is 3D positional directly integrated in the headset.

3.4 SteamVR

SteamVR [49] is a tool which allows the fruition of VR contents on several hardware types. It can be used to play an application independently of which headset and controllers the user owns, since it acts as a "bridge" between the application and the hardware itself, by defining some functions at a higher level of abstraction.

Figure 3.3: Oculus Quest 2 headset.

For example, if the user wants to trigger an event consequently to the pressure of a button on the controller, after having imported the SteamVR plugin in his/her project, he/she can use the Unity interface to create a new *Action*, in an existing configuration or after having created a new one, and then associate this action to a hardware event through the SteamVR interface, performing the so-called binding, as shown in Figure 3.6. Once this has been done, the user can check the state of each action that have been defined by using specific functions in the scripts available after having imported the *Valve.VR* namespace, and, consequently, trigger events in correspondence of them.

Among the several functions, it is also possible to create virtual hand poses in response to specific events, for example when an object with a particular shape is grabbed, in order to have more realistic experiences.

Figure 3.4: Oculus Quest 2 controllers.

Figure 3.5: Oculus Link cable.

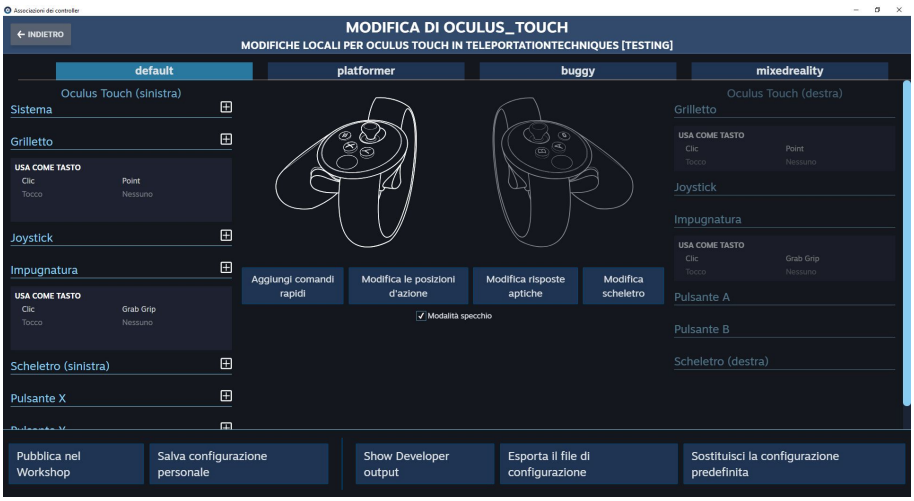

Figure 3.6: The SteamVR binding interface.

Chapter 4

Design and realization

4.1 Functional requirement: an industrial test case

During the activity in Leonardo, the purpose was to develop a voice interface that could be potentially adapted in an immersive VR application dealing with training for maintenance on aircraft. This work was done in order to discover the potentiality of this feature, but, concretely, there was no connection between the software produced in the thesis and any specific activity the Company was actually dealing with.

The system was thought in such a way that two elements are given as a result of the elaboration of the user input:

- An intent, which represents what is the kind of operation the user wants to do.
- An entity, representing what is the object of the action.

In particular, the intents considered were 5:

- Movement intent, to make the user teleport to different places in the hangar, depending on what he/she needs to do.
- Select tool intent, which means that, when the user needs to use one of them, like a wrench or a screwdriver, can utter an appropriate phrase in order to make it appear in the environment.
- Deselect tools, which is the opposite operation; when the user does not need the specific tool anymore, he/she can remove it from his/her view.

- Open schematic intent, in order to make a specific schematic appear on the user's view, like the one of the electrical power system or the one relative to the hydraulic power system.
- Close schematic intent, to make it disappear, again, when the user does not need it anymore.

This feature was obtained by generating a GRXML file containing the possible phrases, grouped under the several intents, to be uttered by the user and expressing the intents themselves, and then connected, at the beginning of the execution of the application, with the possible related entities (e.g., the cockpit for movement, a screwdriver for the selection and deselection of tools, etc.), in order to form a grammar that is loaded onto the speech recognition engine.

The remaining of the document describes, what has been done, taking a cue from the structure of the grammar built, with the VR@POLITO lab. This additional work, which specifically focuses on the teleportation task by adding many newly developed features, was outside the purposes and scope of the work with Leonardo.

4.2 Features of interest

Based on the state of the art observed in the corresponding chapter, the application was made in such a way that it contained the following features.

- The user should have a certain set of utterances to express his/her will to transfer himself/herself to a target position. For example, different phrases could be "Take me to", "Go to" or "I want to go to". In this way he/she is not bound to remember the same word in order to proceed. On the other hand, when using the speech to refer to a specific kind of object, also its name should be expressed in multiple ways, depending on the characteristics it owns, which can be both physical and spatial.
- If the system understands correctly, the user must be transferred to the addressed position, otherwise, he/she must receive a vocal feedback expressing that the instruction was not recognized. This is done in order to make the user aware of what the system is receiving.
- The system must be able to disambiguate if, after the utterance is pronounced, the system is not able to associate the intent of the user to only one of the items that are present in the scene. In this case, moreover, besides listening to a voice asking for more details, the user must be facilitated in his/her work by having a graphical aid composed by a panel showing him/her a preview of the places among which the system can disambiguate, with an interaction that must remain coherent with respect to the currently used interface.

- When the disambiguation occurs, it is necessary that the user is able to revert the action and exit that state. This is important when he/she erroneously calls disambiguation and the set of items that are shown in the panel does not actually contain the one corresponding to the user's will.

4.3 Speech recognition system

In the case of this specific study, the intent that is actually always considered is the teleportation one, while the entity is associated with the destination point the user wants to reach. Specifically, for the speech only interface, it corresponds to a name with which the object in correspondence of the POI is called, while, when using gaze only, it consists in multiple ways to give the spatial indication (e.g., "there", "over there", "in that place", etc.).

These two items (the intent and the entity) are both important to be recognized by the system. In fact, to reproduce the scenario seen in [18], it is needed first that it understands that the user explicitly wants to perform the action of teleportation (i.e., the intent), in order to avoid executing the same actions when something else is said. If the user, for example, during his/her experience, says "This is the table", the system should understand that his/her will is not to perform a teleportation, even if the word "table" correspond to the list of the POIs, and this can happen because the intent is not recognized. Then, the other element represents what is the specific destination (i.e., the entity), which can be an expression describing a specific POI for the use of the speech only interface, or can be an expression like "there" or "over there", to be used for the head gaze technique, in order to indicate that the desired place to go is the one that is being addressed through head orientation.

4.3.1 Initialization phase

In order to have the features needed for the study, a grammar needs to be introduced in the speech recognition engine when the application starts, and this behavior was obtained in several steps:

- First, a GRXML file was created. This file contains a set of phrases that were grouped under two kinds of rules, each of which had an ID representing the intent. In particular, it was composed by the set of phrases the user can say before pronouncing the specific name of the destination. Even if the intent, in this work, is represented by just one kind of action, the second rule set was introduced to represent all those phrases to be pronounced in case of ambiguity, which could be a little bit different thanks to the fact that they allow to shorten the utterance, since the intent has been already expressed

before, in the phrase that caused the ambiguity. For example, if the user says "Take me to the table" and the system responds asking "Which table?", the user is not required to utter "Take me to the green one", but can also answer "To the green one", without explicitly saying the verb.

The GRXML file was created according to the World Wide Web Consortium (W3C) standard [50], which has defined a certain set of rules for all those files representing grammars for speech recognition. For example, the several rules can contain a single ID, which can be exploited to distinguish them. A third intent is actually present in the system, that is the one of closing the disambiguation panel; however, since the structure that is identified in a phrase expressing this intent is different, not including the presence of an entity, it was easily introduced directly in the code when necessary.

- After having created this file containing part of the grammar planned for the recognizer, the next step was to start writing code in order to load the grammar itself onto it. To do this, a C# script was created in a Visual Studio project, in which the Microsoft Speech package was included, in order to be able to exploit its features and use all the methods appropriately made to manage the recognizer itself.

  The first part that is executed at the beginning of the program uses a defined string containing the path to reference this grammar file, in order to read it and create an appropriate structure that can be used in a second moment to build the grammar with the data retrieved. In particular, it was found that a file with that structure could be easily read by using the *XmlTextReader* class. Each time a rule is encountered, it is added to a list, is be used to define the set of starting utterances included in the grammar.

- Then, since the whole phrase that can be pronounced by the user is not composed just by the part considered until now, which defines the intent, but also by a series of objects (i.e., the entities), two arrays of string containing all the several entities that the user could express were defined directly in the script. In particular, the entities, as can be expected from what was said before, were divided into two groups, the ones containig the POIs to be put in the scene, and the one containing the several synonyms for spatial indications. Then, depending on which technique is used, each of the elements contained in the specific array is passed in the execution of the application as a semantic key, in such a way that all the names of specific elements that have been introduced in the scene are linked to each of the phrases introduced at the previous point.

  This can be done thanks to the *GrammarBuilder* objects, which are the ones, defined in the *Microsoft.Speech* library, that are used to link element in order

to finally form a set of phrases composing the grammar, which, then, is loaded in the speech recognition engine by using the function *LoadGrammar()*. In this way, what is obtained is a sort of tree structure, in which each branch represents a potential utterance.

This piece of code was then included in a function in order to easily recall it when necessary. In fact, when starting the application, it is useful to rapidly select how the system is wanted to be initialized. This function takes as arguments first of all a string, representing the name of the intent, that is used mainly to create a dictionary in order to associate each utterance to a specific intent, but also in a previous step to check if the latter is effectively present in the GRXML file, otherwise an exception is thrown saying that that intent cannot be instantiated, and, consequently, the GRXML file is expected to be modified. The other parameter is the array of strings representing the names of the destinations.

4.3.2 Listening phase

After the application has initialized the grammar, the subscription to the event which is able to recognize the speech is done. When this event is triggered, an action can be performed as a consequence of it. In this case, what is done is to advise Unity that the user has pronounced a command.

In particular, as will be explained later an a more accurate way, the whole system including the graphical part of the application and the speech recognition system is a client-server architecture, in which the speech recognition system is the client sending requests on a local web-server exposed by Unity.

If the system, instead, cannot associate the user's input to a specific part of the grammar, the system triggers an appropriate event relative to that case. This is due to the fact that Microsoft Speech tries to assign a confidence value, between 0 and 1, to each of the phrases detected with respect to the grammar content; when this value is too low, the speech is rejected. In case this happens, a synthesizer was also instantiated in order to give the user a vocal feedback on what happened thanks to the text-to-speech.

4.3.3 Preliminary steps of the implementation

Despite the flow of the program described above, in a first phase of the project, the system was made in a simpler way for which, first of all, only the first grammar rule was introduced. In fact, a project in Visual Studio, which was not interfaced yet to Unity thanks to the web-server, was created and worked in such a way that, when the user pronounced a phrase, the system tried to understand what it meant

and deduced an intent and an entity, thanks to a part of the initialization phase described before.

To do this, different events included in the *Microsoft.Speech* library were exploited in order to write some code to verify the correctness of the system:

- The first step was to understand if the system was listening any kind of noise. In fact, this was done in order to avoid not understanding later why the system was not returning the intent and the entity while, for example, the microphone was off. For this purpose a string was printed by using the *Console.WriteLine* function, which displayed it on the console, when this event occurred.

- Once it was sure that the system was listening, another subscription was done to the event relative to the speech recognized. In this case, the handler does something when an association can be done between the utterance of the user and the element of the grammar introduced. From the returned result, it is possible to retrieve data regarding which specific words have been recognized with respect to the grammar previously introduced and which intent and entity have been detected among the possible ones. So, again by using the *Console.WriteLine* function, the entity and the intent recognized were printed on the console.

After having been able to verify that the recognizer worked properly, a further step was done by introducing a communication with the Unity environment. In particular, this very simple speech recognition application was made behave as a client of a local web-server exposed by Unity itself. In the script, on the Unity side, which then has been also used to manage all the events relative to the virtual experience (e.g., the teleportation of the first person character, the appearance of objects, etc.), a task was created in parallel from the predefined *Start* function of the Unity namespace, which is the one executed once at the beginning of the application. In this function, an *HttpListener* gameobject was instantiated in order to make it behave as a listener and, then, by using the *GetContext()* method in an infinite loop, the system was able to repeatedly wait for a request from the client (i.e., the Microsoft Speech recognizer).

In this part of the experiment, the web server did not give any answer to the recognizer; this because, for that moment, it was only important to verify that the Unity side received what was interpreted by the other part. In order to do this, a simple *print* instruction was inserted in the function after the *GetContext()* method in order to extrapolate the intent and the entity once something was received. The string was sent by the recognizer including the character "-" between the intent and the entity, in order to make Unity distinguish them. So, when the string was received, it was parsed, removing all the elements which belonged to the operation of transmitting a content, and then split in correspondence of the "-" character in order to retrieve an array with two elements, the intent and the entity respectively.

Later on, clearly, the necessity of having a response from the server was considered, in order to introduce the possibility to disambiguate. As explained before, in fact, there are two grammars which have been introduced, one for the normal state and one for the ambiguity one, which are a bit different in order to allow a slightly different kind of interaction depending on the case.

The fact that an ambiguity occurs or not, is detected by the Unity side by looking at how many objects in the scene can match the description given by the user. In this specific case, it is true that also the client part contains the list of the target points, but this is not enough to say that the change of the grammar can be done independently by the recognizer side itself. The spatial disambiguation, for example, must introduce objects in the grammar that can be identified only by Unity.

For this purpose, both an *HttpListenerRequest* and an *HttpListenerResponse* were introduced. The second one, in particular, could be used to construct a response to be sent to the speech recognizer side in case of ambiguities, in such a way that the latter was aware of this and could change the grammar.

So, to resume, the process works like this:

- At the beginning, the normal grammar used in order to address the several POIs by explicitly expressing the will of doing it (i.e., saying the intent, for example "Take me to the green bin") is loaded.
- When, based on what was recognized, Unity detects that an ambiguity is occurring, it responds in such a way that the recognizer too understands it.
- Consequently, the part of grammar, in the GRXML file, specific for the ambiguity case, is introduced, together with the several entities, performing a process that is the same done during the initialization.
- After that, the recognizer goes detecting the user intent according to the just loaded grammar and, again, it sends the result of the recognition to the Unity web server which, in turn, analyzes the situation and answers again.
- When the disambiguation stops, the grammar relative to the ambiguity is unloaded from the speech recognition engine.

Now, to better understand, in detail, how this process works, especially in relation to what happens on the Unity side, let's analyze the single interfaces that have been created one by one.

4.4 Speech-only interface

When the voice only is the means to trigger a teleportation action, each entity that is passed to the server side during the execution of the application is then

processed in a dictionary which is used to associate the entity itself to the element of the scene; in fact, there can be different ways of expressing the same gameobject present in the scene, and, for this purpose, this kind of structure was used. The entry of the dictionary can give two kinds of result which, in turn, are used as a response to the speech engine which, according to this, decides what to do:

- When the name of the destination entity is not ambiguous, which means that one and only one POI can be identified with that description, the output of the dictionary is a string representing the name that is given to that specific object in the Unity scene. For example, if the user says something like "Take me to the green bin", if there is one and only one green bin in the scene called "garbage_1", what is expected to happen is that there is an entry of the dictionary with the string "green bin", which corresponds to the potential entity label that is directly passed from the speech recognizer, and its corresponding output value in the dictionary is "garbage_1". In this case, this kind of association allows the teleportation operation to be performed successfully.

  At this point, the next step is to find in the scene a gameobject with that name and retreive the coordinates relative to the POI corresponding to that gameobject. In particular, each object in the Unity scene has an empty child, which means that no other component but the spatial one characterizes it. So, this gameobject, that is used to define the position, close to the POI, in which the user is teleported after having addressed it, has a name composed by a specific pattern which is "<name_of_the_parent_object>_position", where <name_of_the_parent_object> is the name of the object representing the POI. In the example made before, the gameobject called "garbage_1", physically representing the bin, has a child called "garbage_1_position".

  For this reason, what is specifically done after processing the incoming entity string in the process, is to concatenate the obtained output string with the standard string "_position" and look for a gameobject with that name in the scene by using the function *GameObject.Find()*; now, the returned item has some *Transform* properties, representing position, rotation and scale values, as it is for every gameobject in a Unity environment. The position, in particular, is used to be assigned to the first person controller, which is the Player prefab contained in the SteamVR plugin for Unity, in order to teleport it to the target position.

  It is important to underline that the dictionary presents multiple keys with the same value, and this has been done in order to allow the use of different synonyms to refer to the same gameobject. For example, the user can say "Take me to the green bin" or "Take me to the green trash can" and the value for both in the dictionary is "garbage_1".

- Another case is when the entity detected by the speech recognizer, and then passed to the Unity side, is a name that is supposed to be generic, since it can be referenced to different gameobjects in the scene and, so, for that moment, what has been pronounced by the user is not sufficient to trigger a teleportation action toward a specific POI. Taking as an example the same kind of object presented before, a bin, let's suppose that multiple bins are present in the scene, and each of them can be identified with a different color.

  In the previous case, the user, by saying "Take me to the green bin", gave all necessary parameters in order to point to the desired place, because only one object presented these characteristic. But let's now see what happens if the user says only "Take me to the bin", without specifying its color. This could occur just because he/she tries to perform the action rashly, temporarily not considering that there are multiple bins in the scene and so without the idea that the utterance is ambiguous. In this scenario, a disambiguation system is needed, and the mechanism of the conversation should go in a different way than the standard one, as shown in 4.1.

  For this reason, in the same dictionary mentioned before, also generic entities were introduced. When the ambiguous entity string, passed from the speech recognizer side, enters that dictionary, the corresponding output that is given in this case is not anymore the specific name of a gameobject present in the scene, but, instead, a fixed string labeled as "AMBIGUITY", which is given as a response to the client. So, for the example made before, there is a key of the dictionary represented by the string "bin" leading to this output. At this point, when the client side reads this response and identifies the "AMBIGUITY" string, it understands that the grammar must be changed as explained before, in order to change the set of phrases that can be said.

  Moreover, the Unity side, together with the "AMBIGUITY" string, also sends the name of the generic element from which the ambiguity has arisen, in such a way that the recognizer is able to understand which entities to consider among all the ones that are present. In the case seen before, what is sent is "AMBIGUITY-bin". Then, the grammar that is created for the specific ambiguity contains different patterns for the entities, which can be expressed in different, sometimes simpler, ways from the user.

  To obtain this, what was done is, starting from the generic entity name, to create a specific array of strings containing all the entities with that specific word. These are part of the ones that will be introduced in the grammar. Moreover, a set of entities given by themselves excluding the generic word are also created. This allows the user to avoid repeating the generic term. For example, when the user says "Take me to the bin", the system looks for all the defined entities containing the word "bin" (e.g., "red bin", "green bin", "yellow

bin"), keeps them as potential entities, but creates some more by removing the generic word "bin", in this case "red", "green", "yellow". In this way, in the ambiguity scenario, the user can say both "the red bin" or "the red one".

On the other hand, on the server side, when the user solves the disambiguation, Unity needs to receive anyway an entity expressed in the same way as if the disambiguation hadn't occurred, in order to avoid the need to process the string before introducing it in the dictionary. Considering that, during the disambiguation state, the recognizer, for the reason just explained, could understand the attribute of the object without necessitate its complete name, a processing phase on the client side is needed before sending the string to the server.

For this purpose, the client does the following operation. After having received the entity that led to the ambiguity, it temporarily saves it in a string. Then, when the user solves the disambiguation, the system check if the utterance pronounced by the user contains the generic entity too. If this does not happen, it concatenates the pre-saved string with the attribute, otherwise the string is sent without this processing phase.

To make an example, if the user says "Take me to the bin", the system understands that "bin" is a generic word. Consequently, it temporarily saves it and then observes what the user pronounces to solve the ambiguity. If the user says "The green one", since the detected entity is "green", before sending the entity to the Unity side it concatenates it with the word "bin"; if, instead, the user says "The green bin", being the detected entity "green bin", it understands that the word "bin" is already present and, consequently does non concatenate it, otherwise the whole element obtained would be "green bin bin". With this mechanism, in both cases, the entity that is sent is "green bin" and Unity can directly use what it receives to process it in the dictionary, independently from the fact that it derives from an ambiguity scenario or not.

4.4.1 Graphical aid

Besides the aspects relative to the mechanisms which allow to interact in a vocal way with the system depending on the two situations presented before (in the standard case or after the detection of an ambiguity), there is a further issue. In fact, it is important to give the user the possibility to visualize the several objects in order to be able to identify a property which distinguishes them and solves the ambiguity. For example, if the user says "Take me to the bin", the disambiguation system should start. But when the system asks "Which bin?", the user should be able to remember that there was a red bin in the scene and that it is the one he/she wants to address.

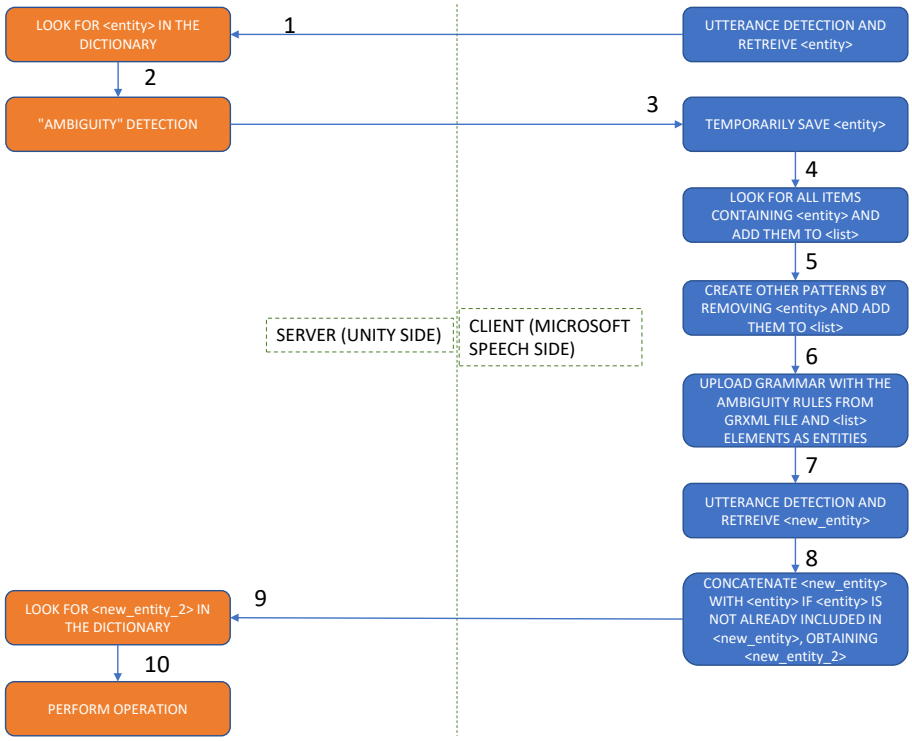

Figure 4.1: The communication between the two parts in case of speech ambiguity.

For this reason, inspired by [43], a panel was introduced in correspondence of the disambiguation event in order to present to the user the several possibilities according to the generic entity, as shown in 4.2. This panel has been created by using a *Canvas* with a certain number of "containers", each of which includes in turn an image of the object.

In order to dynamically create the panel, which can contain a different number of elements depending on the case, for each kind of generic object a list of textures was created in the script. These were initialized with the several images that should be shown to the user. The panel has a certain width and a certain height and, for this reason, based on appropriate computations depending on the number of elements that need to be shown, the dimensions and the positioning of them is dynamically made.

The panel, by using an appropriate script, was made in such a way that it is put at a fixed height and rotates following his/her head with a certain latency and a certain speed, which has been set very low, for two reasons:

- Generally, the reason of this behaviour is due to avoid sickness in the user. If it was fixed on his/her view, many problems of this kind could occur.

- Moreover, the choice of a very low speed while following the user's position was done in order to make him/her able to act with the gaze technique during the disambiguation phase. In fact, as can be seen later, the same panel is used for disambiguation while using the gaze technique; while, if the user talks, he/she does not need to rotate his/her head, and so the panel can remain stable in a certain position, if the user needs to move his/her head to hit a point, the panel follow this rotation; if the speed with which the panel itself follows the user is too high, the latter could have difficulties to select the target.

Figure 4.2: The disambiguation panel which can occur after the user says "Take me to the bin".

4.5 Speech with head gaze

With the head gaze technique, the user is able to point to a generic direction in the space by moving his/her head and, consequently, select the object intercepting an imaginary ray which goes toward that direction starting from the position of his/her head.

As the user's head is directed toward an object, he/she can pronounce a phrase like "Take me there" and, depending on the selected POI, he/she is teleported in the corrsponding place. About the grammar, the vector that is passed to initialize it, clearly, is not anymore the one containing the semantic descriptions of the POIs, but a certain number of expressions expressing approximately the same thing, since all of them denote an indication (e.g., "There", "Over there", etc.).

The approach used with this technique was to use the *RayCast* function included in the Unity namespace, which returns a boolean value corresponding to true when

an object, that must have a collider in order to be detected, is intercepted, false otherwise. In particular, it takes as parameters many elements.

- First, it is defined what is the starting point of the ray, expressed by a *Vector3* object, which represents 3-dimensional spatial coordinates. For the head gaze technique, an empty object was created and placed as a child of the camera contained in the Player prefab of the SteamVR plugin, in order to exploit its *Transform* component to define this parameter. The fact of putting the reference target as a child of the head implies that its position, as well as the rotation, is always the same with respect to the center of its parent object. The result is that the ray always starts in the same point with respect to the user's view and rotates together with it as well.

- The second parameter is the direction of the ray. The same empty gameobject used to determine the origin of it, as explained before, has always the same orientation with respect to its parent, and, for this reason, it is used in order to define what is the direction in which the ray goes, depending on the orientation of the head.

- The following parameter is the distance the ray reaches while going toward the direction determined by the previous parameter. In particular, this means that all the objects that are more distant with respect to that threshold are ignored, while the ones which are closer are considered. In this case, it was chosen to set a very high value, in order to allow the user to even point an object from a corner of the hangar to the opposite one.

- The last parameter that could be used is the *LayerMask*, which is a way to exclude some of the objects that are pointed in such a way that they cannot be detected. The starting reasoning was that, apparently, only the objects which can be POIs were useful, so, in a first time, it was thought to exploit this functionality. However, a further consideration was that also the other objects in the scene should be detected, because, if they represented an obstacle with respect to a potential POI, they had to become semi-transparent in order to allow to see it. In fact, as explained more in detail later, it was important that the user could see what was behind an obstacle, independently on the fact that it represented a POI or not.

  For this reason, given the need to act on both, as will be explained below, different tags were used just to distinguish them. In particular, to all those objects representing potential destinations the tag "POI" was assigned. All the other ones, except from the floor and the walls of the hangar, were assigned with the tag "OBSTACLE".

The value returned by the *RayCast* object can be used to detect whether something has been intercepted or not, but, especially, a *RayCastHit* element is also returned in order to retrieve data from the item itself. In particular, from it, it is possible to access some particular parameters, like the collider of the object that has been hit. Exploiting this fact, it is possible, in turn, to come up to the gameobject itself. In this way the latter can be manipulated in several ways.

However, there was an important aspect to be considered: the *RayCast* function gives information on the first intercepted element, while what must be addressed was also the problem of having occluded objects that can be pointed. For this reason, a variant of *RayCast* has been inserted, which is *RayCastAll*, returning an array of *RayCastHit*, one for each object that has been encountered by the ray going toward the defined direction and the maximum distance defined. Once, while pointing in a direction, all the objects intercepting the ray are accessible, it is possible to manage all those that belong to occlusion scenarios and, possibly, act on them in different ways, depending on if they represent destinations too or not.

The raycast function is clearly something that is put in the *Update* function of the Unity script, because each instant it is important to understand the state relative to what is being pointed.

4.5.1 Ray perception and objects highlighting

An aspect that was considered fundamental was to make the user aware of where he/she was pointing. For this reason, a gameobject with the *LineRenderer* component was created. This kind of component contains some parameters that include a starting point and an end point, plus a specific material to make a visible connection between them, in order to generate a visible line.

In a first moment, it was thought to just make the line visible when pointing to an object, in order to avoid bothering the user while performing other tasks. This was considered also a way to make the user know when he/she was pointing and when he/she wasn't, in such a way that he/she could understand the moment in which it could be possible to trigger the action of teleportation.

Then, since, as will be discussed later, the experience consists in the teleportation task only, and, moreover, it was noticed how it was rather impossible to address a small object very far without being aware of where the user was pointing each time, it was decided to make it always visible. However, it was still important, at the same time, to find a way to let the user distinguish the case in which he/she was currently pointing to an object from the one in which he/she was not doing it, and the method that has been introduced in order to do this was to make the objects highlight when hit by the ray.

The way to obtain this effect was the following one: to each of the objects a script was assigned containing two variables in the form of an array (a single

element was not enough since many objects could have multiple materials when rendered in their original form).

- The first variable is the one containing the default materials. They were declared as *SerializeField* in order to be able to assign them with a drag and drop operation in the Unity interface, after having created the material itself from the *Asset* panel.
- The second one is representative to the materials that are set to the objects when they are pointed. The process was identical to the previous case in order to perform the assignment.

Specifically, the materials for the highlighting were chosen to be the following ones:

- For the POIs, it was decided to make some graduations of blue, in order to allow a better view of occluded objects too, as shown in 4.3. Obviously, the materials, as anticipated before, were at the same time semi-transparent for the main reason.
- For the objects which could only represent obstacles but not POIs, the material was again semitransparent but, this time, with a white color, in order to prevent the user from concentrating on it, that is what they are supposed to do, instead, with the ones representing potential destinations. This is the case of the scenario in 4.4.

An important aspect to be faced was how to assign the several materials to the gameobjects depending on whether they were pointed or not. The operation needed to be performed in the *Update* function of the general script managing all the events of the scene, which is updated every frame, and the approach was the following one.

First of all, two lists of objects were created starting from the ones returned each frame by the *RayCastHit* vector:

- The first one contained the list of objects with the "POI" tag, representing all the destinations, which, consequently, means all those objects which should become blue.
- The other one was filled with the objects with the "OBSTACLE" tag, representing all those that should become semi-transparent.

After having filled them, all the items contained in each of the vectors were assigned with the proper materials.

At the end of the frame, however, a new update should be done, and, for this reason, these vectors were copied in another one which, at the following frame,

would be compared to the newly filled in order to know which ones had previously changed color. In that instant, in fact, all the elements contained in the "past" list were used to check if they were present in the current list of objects hit by the ray. If this was false, they were assigned with the default materials, since that meant that they were not pointed anymore. This process was important because, if it wasn't done, the effect would have been that, once an object was hit, it changed material, and then it would have kept that one, as it was permanently pointed.

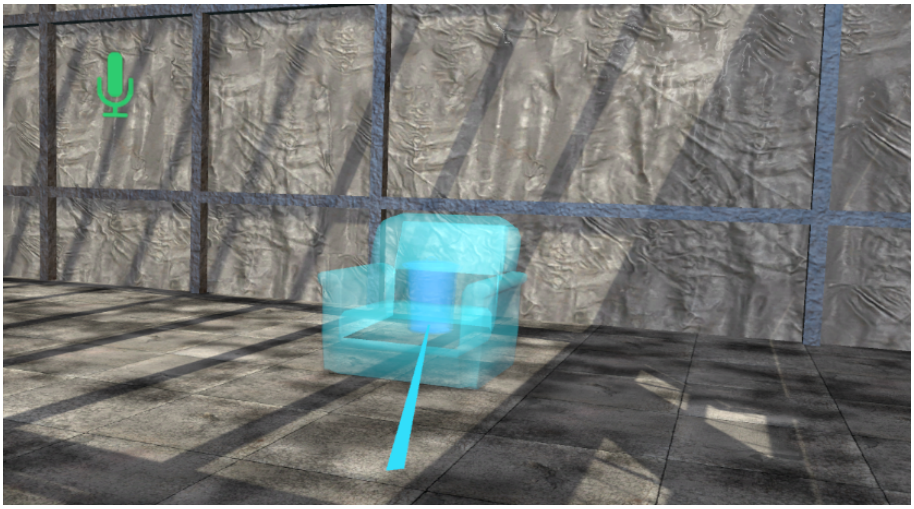

Figure 4.3: An occlusion scenario with gaze technique. The user is currently pointing at two elements, an armchair and a bin behind.

4.5.2 Disambiguation management

As far as the disambiguation case is concerned, it is first important to understand what it means for this kind of interface. In this case, in fact, it is not possible to find ambiguities in what the user says. In every case, to say something like "there", or "over there" cannot lead to different POIs. What can do this is the action itself of pointing with the head.

This has origin on the necessity to reach occluded objects, underlined by [41]. If the user wants to point at one of them, but there is another one which fully occludes it, the user must be able to reach it anyway. While the voice does not impair this task, this technique would physically impede it.

For this reason, the system was thought in this way: when the user points to a certain object, besides highlighting, it also becomes semi-transparent, in such a way that the user can see what is behind. At this point, he/she is able to point that object, but, at the same time, he/she is inevitably selecting more than one. So, when the action is triggered thanks to the voice, if more than one object is

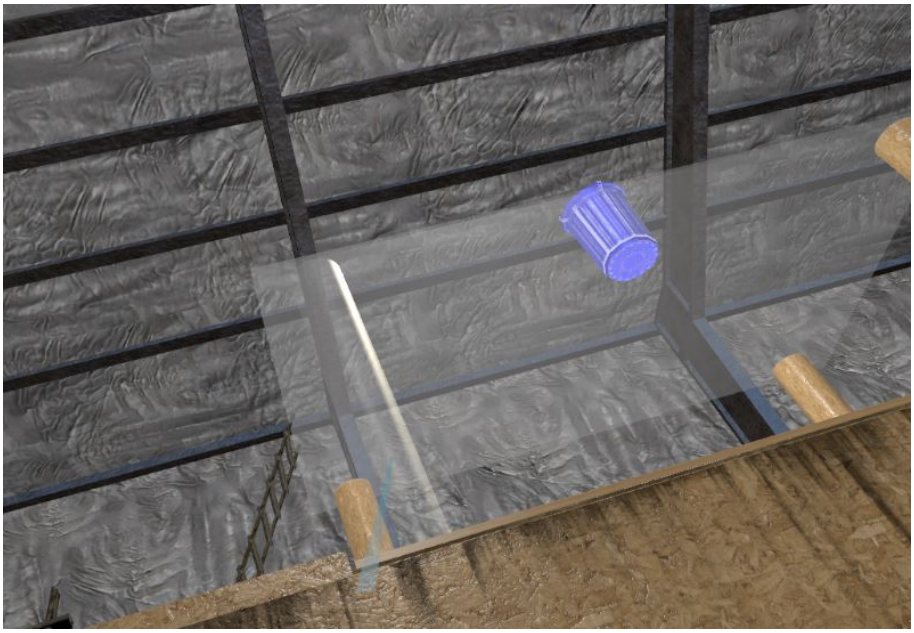

Figure 4.4: When pointing toward an object which is not a POI, this becomes semi-transparent to allow the vision of potential POIs.

selected, the same panel described before appears, this time with the several items that were selected contemporarily. Now the user can choose among them, this time by pointing with the ray, and, consequently, he/she can be teleported there.

As far as the grammar change is concerned, the concept is similar to what has been done with the speech alone (4.5): for each frame in which the user is pointing, a vector is created with the several ones that belong to the class of the destinations (i.e., which own the "POI" tag), while all the others are this time ignored. What is retrieved, in particular, besides the single elements that are then useful for building the appropriate grammar, is the number of objects contained by it, because this is the element which is able to make the system understand if the ambiguity scenario is going to occur; in fact, if this value is major than 1, when the voice triggers the action, the disambiguation panel appears and Unity sends a fixed string, similarly to what happened with the speech-based technique, to communicate to the speech recognizer that an ambiguity has been found and the grammar must be changed. This time, as anticipated at the beginning, it is not possible anymore to retrieve data about the single objects on the client side. Instead, they must be sent from the Unity web server one.

Selection of the objects to be put in the disambiguation panel

In order to understand which objects to be considered for the spatial disambiguation and, consequently, to be put on the panel, the approach can be more easily understood after having seen the mechanism of highlighting for the pointed ones seen before. Also in this case, in fact, a vector of gameobjects is created and the tags are checked. Now, clearly, only the ones with the "POI" tag are considered, as the obstacles are not addressable for the teleportation.

This is done in the *Update* function together with the operation of highlighting and, also in this case, a copy of the vector is done in order to understand, at the following frame, which ones are still present and which ones are not. Also in this case, it is very important to do this comparison, because, besides checking which ones are not involved anymore in a potential incoming disambiguation, it is worth considering that, if an object was present at the previous frame, it must not be added to the list again at the following one; if this consideration is not done, what happens is that, for example, if the user stays for 1 second pointing to an object, each frame within that time the same gameobject is also added to the list of disambiguation elements and, when the panel appears, it would contain a huge number of duplicates of the same object. Practically, disambiguation would be always triggered.

A further aspect must be considered as far as how the layout of the panel is created, in particular about how the vector obtained before, the one containing the several gameobjects, is translated into the visualization of them. Differently from what happened with the speech-only technique, in which the name of the generic entity could be processed and used in order to trigger the visualization of a fixed vector of images, depending on the case, here the situation is a bit more complex, since another approach has been used. In fact, the entity that is perceived by the recognition system does not contain any information on which objects have been pointed.

For this reason, the approach was the following:

- A general vector of textures, which represent the images themselves, was created, with all the possible objects present in the scene that could be representative of POIs.
- Another "universal" vector of texts was also created in order to make a correspondence to the images themselves: once these were inserted by pointing with a specific index, at the same index of this vector the text was retrievable.
- The important aspect that allowed a direct association with the objects pointed is that the images were named in the same way as the corresponding gameobjects. For example, the image representing the red bin was called "garbage_4" since the same was the name of the gameobject. At this point, it

was easy, looking at the names of the gameobjects contained in the vector, to retrieve the corresponding image and assign it to one of the "containers" of the panel.

Disambiguation solving modality

In order to be coherent with the kind of interaction from which the disambiguation had emerged in this case, a gaze technique was introduced to select among the elements represented on the panel. In particular, when the disambiguation starts, the objects in the environment cannot be pointed anymore, while the images can. The user, now, can say "Take me there" by pointing with his/her head rotation the corresponding image. In this way, he/she can teleport to the corresponding place.

In order to associate the image to the object in the scene to which the user wants to teleport, the process was approximately the inverse of what had been done in order to put the several images on the panel. In fact, for each of the objects represented in the panel, in the corresponding container, which includes its image, an empty gameobject with a *Collider* component was also put, in order to be able to be identified by the ray. Clearly, its dimension corresponds to the one of the container itself. Being this gameobject a child of the latter, by using *transform.parent* it was possible to access it and then, with the method *GetChild()* it was possible to retrieve the gameobject containing the image. At this point, the name of the image is taken which, as explained before, it is equal to the one of the corresponding item in the scene. So, when the action is triggered, by following this path, the destination can be addressed.

4.6 Union of the two techniques

The last technique implemented uses the combination of the two previously described techniques in order to facilitate the user's interaction. This one is clearly expected to have the advantage of giving the user the possibility to exploit two kinds of information, which are the semantic one, given by the speech, that can describe the object, and the spatial one, which is able to give an extra information about where the user is looking for the object he/she is describing.

For example, let's suppose that multiple bins are present in a scene, with different colors and in different places of the environment. If the voice only is admitted, the user, in order to avoid the disambiguation scenario, is obliged to give all the necessary information describing that object, for example by saying "Take me to the red bin", otherwise, if the user just says "Take me to the bin", the system must retrieve additional information about which one is addressed. If, instead, the user is able, at the same time, to point it with a ray to that specific bin, the system has

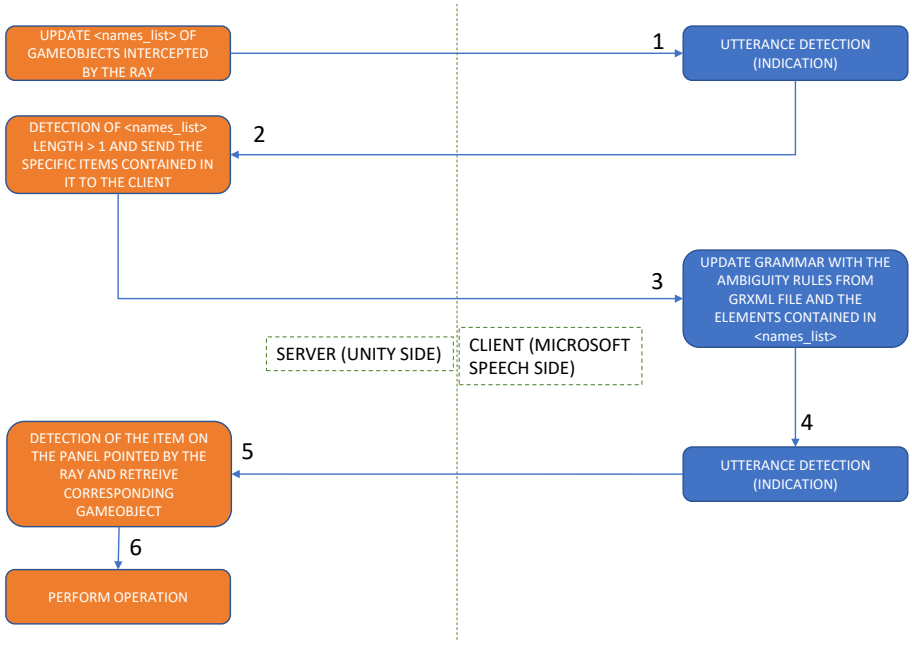

Figure 4.5: The communication between the two parts in case of spatial ambiguity

all the necessary information to understand that the intended bin is that one, even if the user does not specify the color.

A further example showing the expected advantage with respect to the gaze technique alone is the following one. Let's suppose that the user wants to address an object that is fully occluded by another one, for example a bin occluded by an armchair. If the user says "Take me there", the disambiguation inevitably starts since the two objects are both hit by the ray, he/she cannot point to the bin without hitting the armchair too. However, since these objects are of different kinds, he/she can give the semantic information, again by saying the same phrase as before, "Take me to the bin", and, in this way, the system can discard the armchair since it does not match the verbal description.

So, these are two cases in which the disambiguation can be avoided. The approach to implement the technique is now explained.

In the first phase, the system behaves as the gaze technique since, while pointing to a certain number of objects, it fills a vector containing them. As the user pronounces a phrase containing an entity which does not belonging to the class of the ones expressing a spatial expression (i.e., "there", "over there", etc.), but it's a phrase for which the entity is contained in the dictionary relative to the objects (the one described during the explanation of the speech only interface), all the elements that can match the description are taken and put in a second vector. Then, the two vectors are compared and only the intersection is finally considered.

For example, as shown in 4.6, if two bins, an armchair and a table are aligned, and the user, while pointing all of them, says “Take me to the bin”, then the system proceeds like this.

- It cycles all the keys in the dictionary in order to create a vector containing all the gameobjects representing bins, by considering all the strings containing the entity received; clearly, the elements must not be replicated, and, for this reason, each time an element could be added to that vector, it is first checked that it is not already contained in it. In this case, 4 elements are returned.
- In the meanwhile, the vector of the previously pointed objects is filled with 4 elements, which are the two bins, the armchair and the table.
- After this is done, the system compares the two vectors and “filters” the second one by leaving only the objects that are contained in the first one too, which practically means to make an intersection of the two vectors; in this case, the two bins are the objects which belong to both vectors and, for this reason, a panel appears showing these elements.

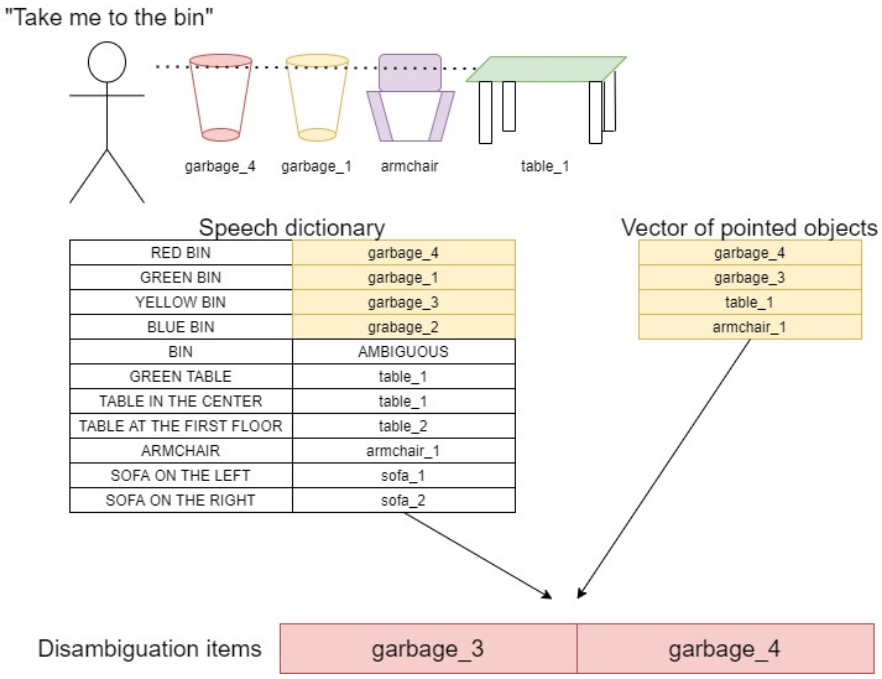

Figure 4.6: Scenario with information combination with the two techniques together.

In general, three different situations can occur after the filtered vector is obtained.

- If the size of the obtained vector is equal to 1, it means that the user could be referred to one and only one item, so the disambiguation is not performed and the user is transferred to the corresponding place.
- If the size of the vector is major than one, it means that more than one object could match the combination of the physical description and the spatial information, so the disambiguation occurs, showing these objects contained in the vector on the panel.
- If the size of the vector is 0, then it actually means that the user is in a situation in which no advantage was obtained in combining the techniques and, for this reason, the entity is considered and, if it semantically represents a physical object, then it behaves as the voice-only interface, with the possibility of incoming in a disambiguation scenario if the description is too generic, while, if the entity is an indication like "there" or "over there" it means that the user has not applied this last technique presented.

However, it is important to underline that, even when this modality of interaction is used, the other two techniques can be used independently, since they represent its components.

4.6.1 Disambiguation panel

In order to be coherent with interaction metaphor that is used in normal conditions, if with the speech technique the disambiguation is solved only with speech and in the gaze one it is solved with gaze, in this case it is possible to do it in both the ways. For example, if the user is trying to disambiguate between two objects, a bin and a chair, and he/she wants to choose the second one, the system will ask ("Where do you want to go?") and the user can both say "To the chair" or pointing to the corresponding image and say "There".

Chapter 5

Experiment design

The person which performs the experiment is first invited to fill a demographic questionnaire, in which age and gender are asked. Moreover, two questions, to be rated from 1 to 5, are relative to the experience with VR and the one with speech interfaces in general (e.g., Alexa, Siri, voice commands used while driving).

After that, the person is instructed on the general purpose of the experiment and on the several aspects which are common with the three experiences (e.g., use of voice, how things are graphically represented, which steps the experience is composed by). Among these aspect, some important points were noticed as necessary to be stressed while performing some pre-test experiences.

- Underline the fact that the teleportation task must be reached in one step only. In fact, while doing some pre-test experiences in order to validate the protocol, it was noticed how many people tried to reach the required destination with multiple steps. The lack of this information would clearly have impaired some statistical data.

- Another problem was found in understanding how to deal with disambiguation panels in the several cases. This happened in particular with techniques including gaze functionalities, for which some people tried to answer by using voice to describe the object they were interested in, instead of using the same kind of interaction used in 3D space.

Then, the user can start the several experiences and, after each of them, he/she fills a questionnaire. The techniques are marked as follows:

- S: speech-only interface, using description of the POI;

- SG: speech with gaze, referring to the one which does not allow the description of the POI;

- SGD: speech with gaze, with the possibility to also describe the object to be reached.

The order of the experiences is not always the same for the several users. Instead, all possible combinations are done before repeating an already occurred sequence; this is done in order to avoid favoring more a certain technique with respect to another.

5.1 The experience

The experience was thought to be fully dedicated to teleportation tasks. Even if, in a first moment, the idea was to combine it with a pick & place activity, then it seemed better to let the user concentrate on the single operation for which the study was being done.

In particular, the virtual experience was made by 8 pairs of starting points and destinations to be reached. While the initial project was to make the user follow a path for which every destination point would have been the staring point of the following step, then it was decided to make all them independent, in the sense that each starting point does not have a certain relationship with the previous destination reached. In this way, it was easier to build the scenario. Moreover, in order to make the experience non-deterministic, even if the 8 steps were the same for all the users, they were sorted in a random order for each of them. This experience was repeated three times, in order to make the user test all the three techniques; the sequence was defined by the Latin square [51].

The persons had controllers in both hands in order to be free, in a certain moment of the scene, to choose when to move to the next step, but this can be understood better after having described the procedure, which is the following.

1. The user is initially put in a starting scene in which he/she is not operative yet. In particular, an explanation is done about how the starting place and the required destination are represented each time. Once he/she feels ready, by pressing a button of the controller, he/she goes to the first step.

2. Now the user finds himself/herself in a place which is located at a major height, in such a way that he/she is allowed to see the following couple of starting point and destination. In particular, the first one is indicated by a blue animated circle, changing its size between two values. The destination, instead, is shown to the user by highlighting its borders and making their width animated; also an arrow is put above it to emphasize the position. The border is coherent with the color of the object in such a way that, for all of those that are characterized by it, the user would not confuse.

At the beginning it was thought to make this step last 10 seconds, in which the user could understand where the target object was placed with respect to the following starting position. Anyway, it was observed later that it could be better to give the user all the necessary time in order to analyze the environment and to let him/her choose when to go on by pressing a button of the joystick, in such a way that situations in which the user cannot find the two positions are avoided.

3. When pressing the button, the user is automatically moved in the starting position, marked with the blue circle, and has the possibility to perform the first teleportation task with the current technique.

4. The user is expected, sooner or later, to complete that task and, consequently, he/she is transferred in the destination for 3 seconds.

5. After these 3 seconds has elapsed, the user is moved again in such a way that he/she can have a top-view of the environment and, again he/she is free to observe the following destination to be reached and the corresponding starting point for the desired time, and then push the button to proceed.

   A risk was found in the fact that the user presses the button in the previous step, forgetting that the transition is automatic, and then continues doing it multiple times until the scene changes. Then, when this happens, the risk is to directly move to the starting point without having had an overlook on the environment. For this reason, for a minimum of 3 seconds from the beginning of the top view, it is not possible for him/her to proceed. After these 3 seconds, the button input is "unlocked".

   With this step, the cycle restarts with the same technique and the new destination.

6. After the last destination is reached, the experience with the current technique ends and the user can fill the questionnaire relative to it before starting the same process with the following one.

5.2 Organization of the environment

The speech-based technique and the other two with head gaze, are very different. For this reason, based on their characteristics, it was made an analysis of which could be their weaknesses and their strengths.

The third technique, being able to exploit the advantages of both the other two, was expected to give better results. For this reason, the main considerations on how to build the environment, were made considering advantages and disadvantages of

the first two. Then, it was also thought to insert situations in which the power of the third technique could be exploited.

- As far as the speech-only interface is concerned, the problem was expected to be mainly due to the cognitive load. The user, in fact, must think about how to call the target destination he/she wants to reach. For this reason, all the several objects present in the scene should have a different way to be referred to, in order to avoid a total ambiguity between two or more objects representing different POIs. However, sometimes, it is not sure that the user finds an immediate way to distinguish them. In some cases, as can be observed later, the user was stimulated to give indication that are not limited to a physical aspect, but also to a positional property or with some details which characterize them.

- On the other hand, the head gaze technique, used together with the voice, allows the user to avoid thinking about a specific name for the object, since the only thing he/she has to do is to look at it and utter the command. However, this technique suffers from another problem, which is the one of occlusions. In fact, while by using voice alone every place can be reached without problems under this aspect, in the case of gaze it is not such easy to do that. In some cases, especially when the object is fully occluded, the disambiguation mechanism is not avoidable: the user inevitably hits two or more objects, independently on his/her capabilities.

  Then, contrarily to the previous technique, the one of gaze is expected to involve many complications in addressing small objects at a high distance.

  In this technique, moreover, what can be considered too is the fact that the achievement of the task depends on two inputs, differently from what happens with the voice only: both the actions must be correctly performed, otherwise the task could be not completed in a successful way.

Based on these considerations, the environment was built with these characteristics.

- In order to stimulate the cognitive load, four bins with different colors (red, green, blue and yellow) were put in the scene, and all of them were addressed during the experience. Two of them, the yellow one and the green one, were also put very close in an angle of the hangar, in such a way that the user could be invited to say a phrase like "Take me to the bin in the corner". The red bin, instead, was put behind an armchair when looked from the POV of the starting point, and so the user could say a phrase like "Take me to the bin behind the armchair". The position of this object is also intended to create the possibility to exploit the power of the third technique, which, in this case, can avoid a disambiguation that is necessarily triggered by the head

gaze technique without description. The last one, the blue bin, was put at a different height, major than the one of the floor, in such a way that, when the user performed the operation by exploiting head gaze, the inclination of the head could give results about discomfort.

Moreover, being the bins relatively small, two different scenarios have been created in order to emphasize the difficulty of pointing at them. In particular, being the yellow and the green bin very close, for two times they have been addressed, in particular for medium and large distance. The probability of an error was expected to be higher when the user was very far.

- Two tables were put in the scene, one more or less in correspondence to the center of the hangar, and the other one at a different height; however, the latter was never addressed. The one which was chosen as a POI could be referred by saying "Take me to the table in the center".

- Two sofas were put relatively close along the short side of the hangar, and both were addressed during the experience. On one of them, the one on the left, a ball was put, in such a way that the user could say both "Take me to the sofa on the left" or "Take me to the sofa with the ball". The one on the right had no objects on it, so it could be addressed by saying "Take me to the sofa on the right" or even "Take me to the sofa without the ball". This is another case in which the user is invited to think more about how to recall objects, since the difference is not trivially given by a color.

- Finally, two chairs were put in a different spatial configuration, one on the left and further with respect to the user, and one on the right and closer to the user's starting point. For this reason they could be indicated by saying "Take me to the chair on the left" and "Take me to the chair on the right", but also "Take me to the closer chair" and "Take me to the further chair". Moreover, near the one on the right a plant was put, so that chair could also be addressed by saying "Take me to the chair close to the plant/to the vase".

5.3 Programming of the experience

As anticipated before, the experience required that the user could understand what he/she had to do, without providing explicit suggestions regarding how to perform it with the available technique. For this reason, it was important to emphasize the starting point and the destination each time he/she was in the top-view and, moreover, each time he/she was at the starting point and had to transfer himself/herself to the destination one. To obtain this, DOTween was used in order to create animations for the several objects.

In particular, at each step, the user could see a blue circle varying its radius between two values in an interval of 0.5 seconds. In the same moment, an outline was inserted in the object in order to make the destination always visible. In fact, this component allows the view of an object even when it is occluded, since the border of its shape is always put in the foreground with respect to any other object in the scene.

Actually, after having done some experiments with this technique, it was noticed that, sometimes, is was still difficult to find that specific object when using the outline only. In fact, considering that the color, in order to avoid confusion in the user when it was occluded, was set coherent with the one of the object itself, it resulted difficult to notice it from long distances, even if a brighter version of that color was set. For this reason, later, it was decided to add an animated arrow in correspondence of the object itself, in particular above.

Besides the graphical aspects, also the functional part was important. In fact, the user was required to perform 8 steps along the environment. Since these steps were decided to be predefined, in order to make all the users perform the same experience, it was worth to prevent them from reaching a different destination from the correct one. For this reason, it was decided that, when the user chooses a wrong one, the teleportation is not performed and a voice gives him/her a feedback to make this explicit. Regard this fact, a modification was done at a certain point with respect to the initial idea. In fact, the application was initially thought in such a way that the disambiguation process, happening when two or more objects could be involved in the request, occurred independently from the fact that, among them, the correct destination was present or not. After that, it was considered to avoid from the beginning the teleportation in this case, and let the user disambiguate among the potential places only if one of them was effectively correct.

Another important feature was to allow the user to reach the destination sooner or later after a certain number of attempts, in such a way that he/she couldn't remain blocked in the same place for a long time; to do this, a panel appears after a wrong destination selection or a not understood command occurs for 3 times.

In order to allow the user to better understand when it was possible to talk and when it was not, an icon representing a microphone was put on the left of the user's FOV, changing among three colors [52].

- The green color was put when the user was allowed to talk, which is when it was in the starting point and could effectively trigger the operation.

- The grey color, instead, represented that the system was not listening to the user, so every command was practically ignored. This happened, for example, when the user was put to see an overview of the environment before executing the teleportation step.

- The red color was set as a graphical feedback when the user's command was not understood, together with the voice explaining that.

5.4 Questionnaire

The questionnaire used for this study was structured in this way.

- The first part, as anticipated before, was a demographic questionnaire used to understand the target user involved in the experience.
- Then, the SASSI questionnaire [53] was introduced. In particular, it seemed to be very suitable for the study since it was originally designed for speech interfaces. Clearly, in this case, some questions had to be interpreted in such a way that the "system" considered is each time composed by all the features of the interface, and not only by the speech component. For example, when asking if a technique is precise, the gaze component could make the difference in the answers. These questions should be rated from 1 to 7 on a 7-point Likert scale, where 1 meant "strongly disagre", while 7 "strongly agre".
- The 10 questions contained in the SUS questionnaire [54] were also introduced in order to have an indication of the usability. These statements should be rated from 1 to 5 on a 5-point Likert scale, where 1 meant "strongly disagre", while 5 "strongly agree".
- Then, a "custom" section was introduced in order to investigate some aspects that were not considered in the previous questionnaires, being very specific for the application and created to investigate specific problems the several techniques were supposed to present, like the difficulty to point to far and small objects with the techniques involving head-gaze.
- After for three times the SASSI and the SUS questionnaires were filled, the user was asked to express an order of preference among the three techniques.
- Finally, some additional comments were asked to the user in order to have a better idea on his/her sensations about the three techniques.

Chapter 6

Results and discussion

In this chapter, results about the tests performed, like the one shown in Figure 6.1, are presented. To facilitate the explanation, three acronyms will be used to indicate the three different techniques.

- S: the technique based on speech only, in which the user has to describe, by using voice, the object representing the destination.
- SG: the technique using speech combined with gaze, in which the voice can be used just to say phrases like "Take me there" while pointing with head orientation to the destination, without the possibility to describe it.
- SGD: the hybrid technique composed by the functionalities of both S and SG.

6.1 Demographic data

As can be seen in 6.2, most of the 15 users which were involved in the tests were between 20 and 30 years old ($\mu = 28.5$). The scores representing the level of experience with VR and with voice interfaces in general resulted to be both, on average, around 3; in particular, as far as the first one is concerned, more than half of the people resulted with a score being 4 or 5, while six of the users declared those scores in the question related to the experience with voice interfaces. Two users declared to have no experience with VR and one had no experience with any kind of voice interface.

6.2 Objective metrics

The objective metrics that were considered in this study and then analyzed for each of the techniques were three:

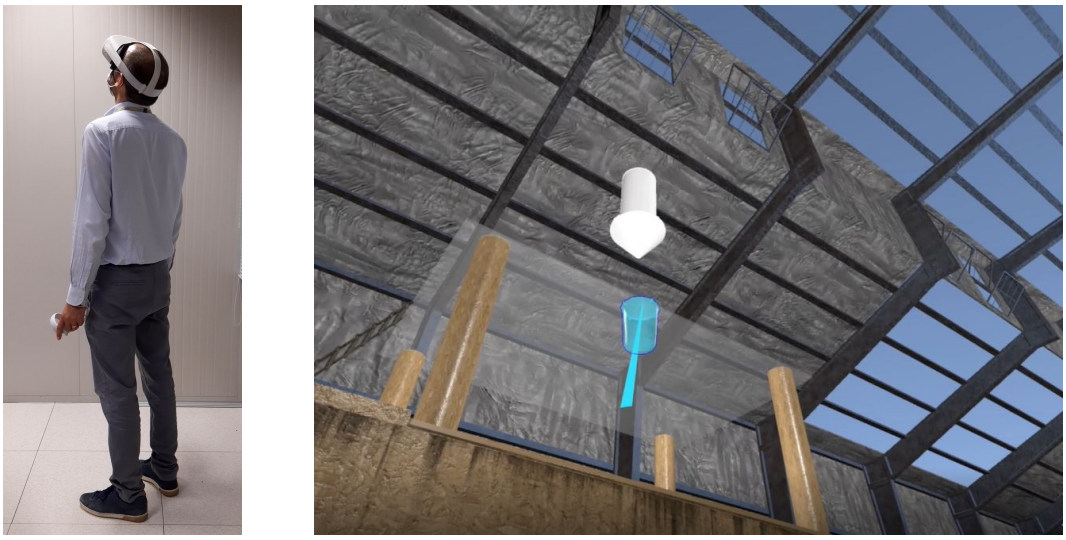

Figure 6.1: On the left, the user pointing to the destination with the SG technique. On the right, the user's POV in the VE.

- average time spent for each destination;
- number of wrong destination occurrences in all the experiences;
- number of invalid commands triggered in all the experiences.

Among them, the only one which resulted in statistically significant differences between at least two techniques was the one of the wrong destination occurrences. In fact, with SG, only 1 error was made in the whole group of people. Instead, for 20 times, in S, errors of this type were made, while, in SGD, the number of occurrences was 17.

The statistical significance of this data was verified by performing a repeated measures ANOVA [55] with an *alpha* value of 0.05, which led to a significance value of 0.0017, followed by paired sample t-tests [56] leading to $p = 0.0025$ between S and SG and $p = 0.0032$ between SG and SGD. In both cases, since the techniques compared are three, the Bonferroni post-correction [57] was applied to take into consideration the problem of multiple comparisons, and the null hypothesis would be rejected with $p < 0.025$.

The significance on the number of wrong destination occurrences related to SG with respect to the other techniques can be explained with the fact that, with the mentioned technique, the user always knows what he/she is currently pointing before triggering the action with voice, thanks to the visual feedback given by the highlighting of the object itself, so it is much more unlikely to make mistakes

Figure 6.2: On the top, the ages of the several users that took part to the test. On the center, the declared level of experience with VR for each of them. On the bottom, the level of experience with voice interfaces.

leading to the selection of another POI; instead, by using voice only (i.e., with S), the user is not aware of what is going to happen after his/her utterance (i.e., he/she

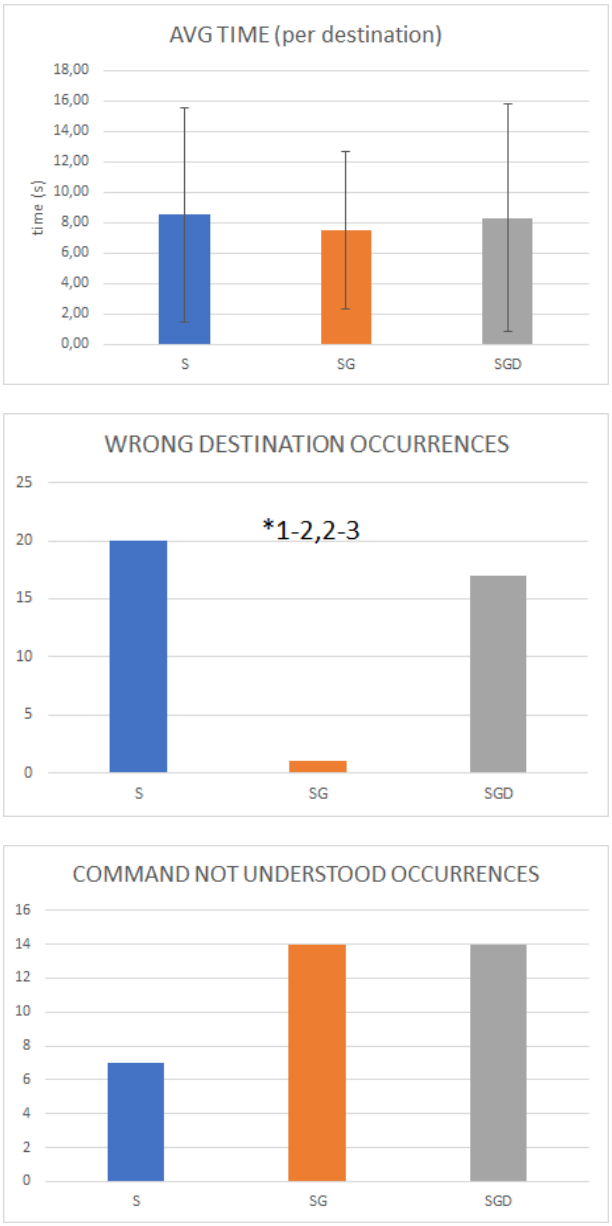

Figure 6.3: On the top, the average time needed for each step of the experience. On the center, the number of wrong destination selections. On the bottom, the number of invalid commands detected by the system. The symbol * indicates a significance.

doesn't know if the system is going to correctly interpret it), so he/she cannot correct himself/herself before triggering the command. In the SGD technique, the user has many times preferred the use of voice to describe the object, so the same

consideration can be done about the significance of the data related to SGD with respect to the one of SG.

6.3 SASSI questionnaire results

The 34 questions contained in the SASSI questionnaire [53] are classified in 6 sections:

- system response accuracy (6.5);
- likeability (6.6);
- cognitive demand (6.7);
- annoyance (6.8);
- habitability (6.9);
- speed (6.10).

For each of them, average statistics were retrieved by considering the contributes of average results (i.e., scores from 1 to 7) to each of the questions contained in it, in order to understand which were the general aspects under which the techniques differed (6.4). Then, by analyzing the single statements one by one, it was possible to find some more specific aspects for which the three techniques were perceived as different by the users. In particular, it resulted that 13 over the total 34 statements led to significant differences between at least two of the three techniques. Moreover, 3 of the 6 sections of the SASSI did the same. The details are presented below.

6.3.1 System response accuracy

About system response accuracy section, all the techniques resulted having, on average, a high score, with no significant difference among the three.

However, taking into consideration the statement 9, it resulted that S was perceived as more efficient than SG (6.60 vs 5.73, $p = 0.0069$), probably because many times the user had difficulties in being understood when pronouncing short words like “lì” (“there”) during the disambiguation scenario.

6.3.2 Likeability

As far as likeability is concerned, statistically significant differences were found between S and SG (6.30 vs 5.67, $p = 0.0006$), and between SG and SGD (5.67 vs 6.24, $p = 0.0189$).

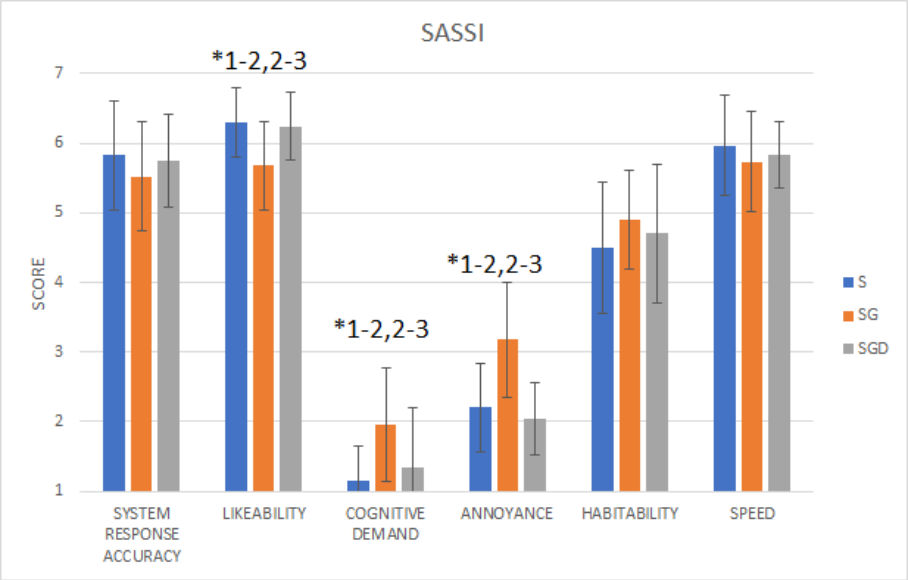

Figure 6.4: Results of the average scores for the several SASSI sections. The symbol * indicates a significance.

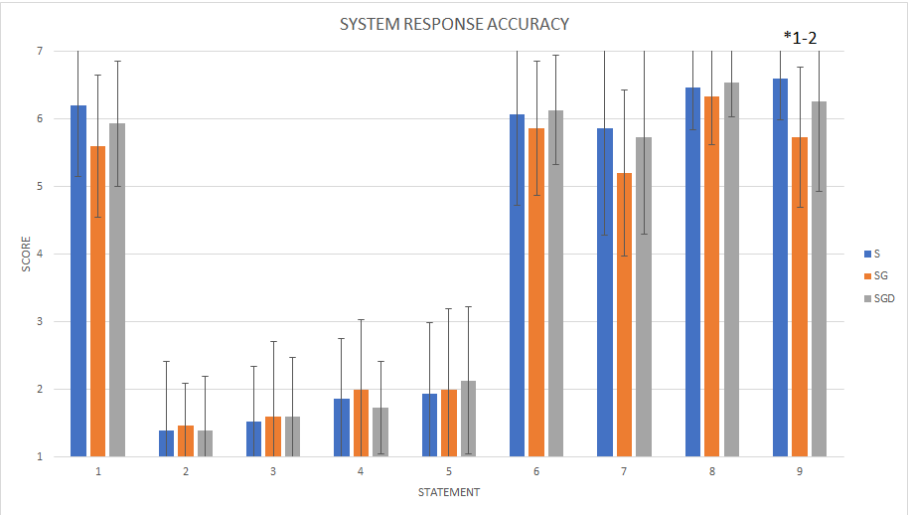

Figure 6.5: Statistics about system response accuracy for each statement. The symbol * indicates a significance.

Inside this section, five of the nine statements brought to statistically significant differences. The statement 11 made SG less enjoyable than both S (5.07 vs 6.20, $p = 0.0034$) and SGD (5.07 vs 6.20, $p = 0.0024$), while answers to statement 12 showed that the same is also less friendly than S (5.47 vs 6.20, $p = 0.0104$) and

SGD (5.47 vs 6.27, $p = 0.0053$); these two results can be explained by the fact that the technique was not so natural, requiring obligatorily head movements. Then, it also resulted from statement 14 that the average user liked S and SGD more than SG (for both, 6.47 vs 5.47, $p = 0.0028$ and $p = 0.0131$, respectively), while in question 17 they declared that they would use S more likely than SG in VR (6.07 vs 4.73, $p = 0.0039$), results that can be justified with the same reason expressed for the other two statements, with the difference that, in the last case, the flexibility of SGD was somehow appreciated. Finally, in question 18, the users told they were more in control with interaction when using S than when using SG (6.20 vs 5.13, $p = 0.0007$), probably for the fact that the avoidance of ambiguities depends on the users themselves (i.e., on what they say), while in SG, with fully occluded objects, is unavoidable.

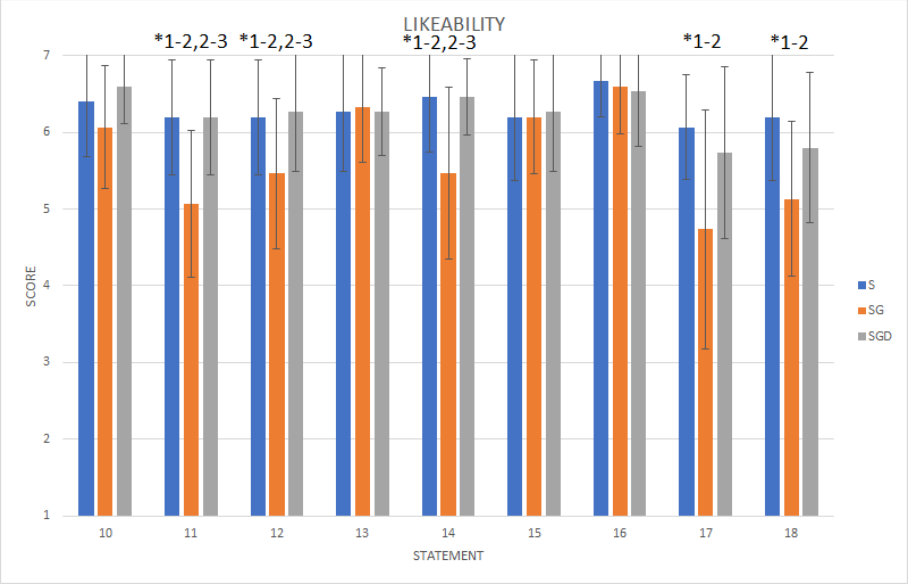

Figure 6.6: Statistics about likeability for each statement. The symbol * indicates a significance.

6.3.3 Cognitive demand

Under the aspect of cognitive demand, it emerged how SG was considered more cognitive demanding than both S (1.96 vs 1.16, $p = 0.0004$) and SGD (1.96 vs 1.33, $p = 0.0126$).

In particular, statistically significant differences were found in question 21 about the calm had while using the specific technique when comparing S and SG, since the user declared to be more calm in the first case (6.33 vs 5.47, $p = 0.0134$), probably

for the fact that they did not need to concentrate on pointing. In fact, when dealing with the aspect of concentration required during the task of addressing the POI (statement 22), in SG the score was higher than in both S (3.47 vs 1.87, $p < 0.0001$) and SGD (3.47 vs 2.33, $p = 0.0016$); in SG, in fact, the users have frequently found it difficult to point immediately with a certain precision the objects.

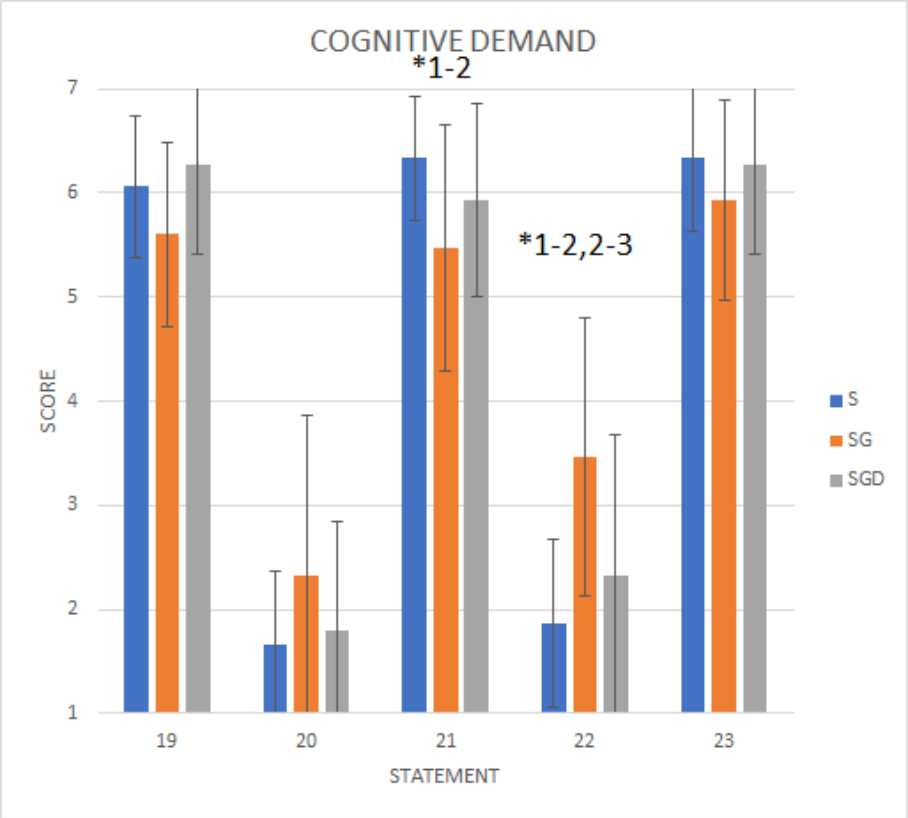

Figure 6.7: Statistics about cognitive demand for each statement. The symbol * indicates a significance.

6.3.4 Annoyance

As far as annoyance is concerned, statistically significant differences emerged between SG and S (3.17 vs 2.20, $p = 0.0004$), and between SG and SGD (3.17 vs 2.04, $p = 0.0008$).

Four over the five statements were involved in this significance. Two of them were, in particular, relative to the repetitiveness (statement 24) and boredom (statement 25) in the interaction; for the first aspect, SG resulted more repetitive than both S (5.40 vs 4.07, $p = 0.0109$) and SGD (5.40 vs 3.73, $p = 0.0007$). For the

second one, again, SG had a higher value than S (3.33 vs 2.27, $p = 0.0122$) and SGD (3.33 vs 2.07, $p = 0.0032$). This two results can be surely attributed to the fact that the voice commands were always rather the same in SG, differently from S, which required a specific description of the POI every time, and SGD which gave the possibility to alternate the way to address it. Then, statement 27 expressed that the SG technique was much more frustrating than S (2.13 vs 1.20, $p = 0.0207$), result that, again, can be attributed to the difficulty in pointing some kinds of objects and to the one to avoid ambiguities sometimes (i.e., with occlusions). Finally, it was found that SG was perceived as more inflexible (statement 28) than both S (3.13 vs 2.20, $p = 0.0207$) and SGD (3.13 vs 1.67, $p = 0.0003$), result that can be justified by the explanation already done about the inability to avoid sometimes disambiguation.

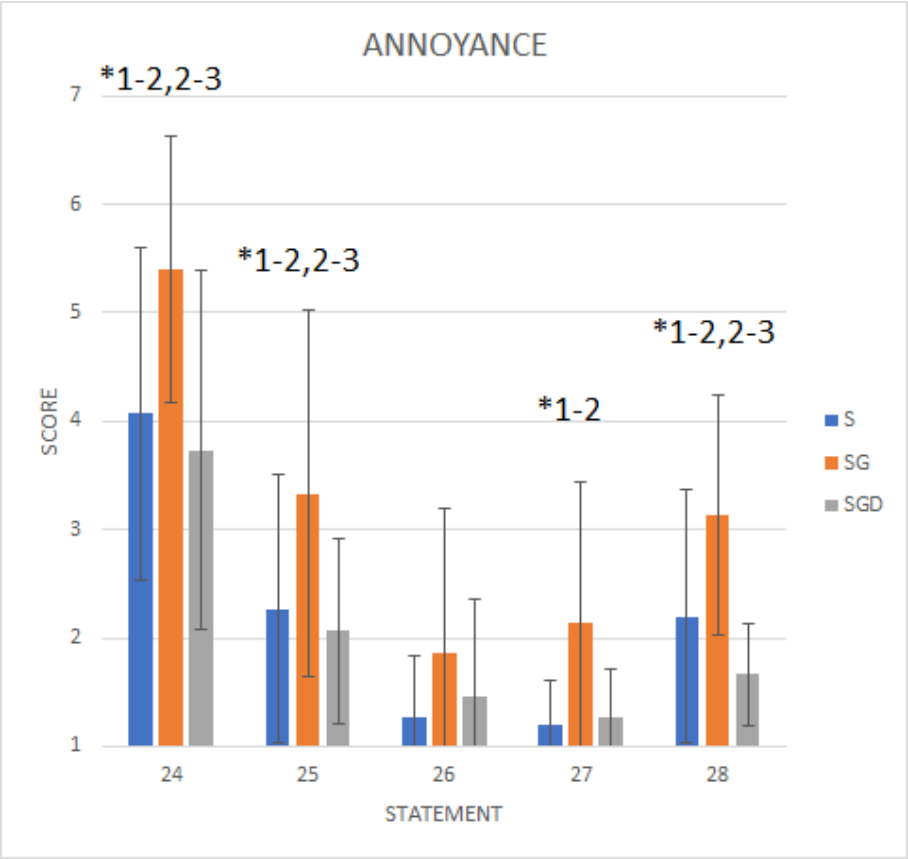

Figure 6.8: Statistics about annoyance for each statement. The symbol * indicates a significance.

6.3.5 Habitability

Only one of the four questions about habitability resulted in a significant statistical difference, and it was about the awareness of the correct words to be used in order to communicate with the system (question 29). In fact, in the case of S, the user was significantly more prone to ask himself/herself if he/she was using the correct words with respect to both SG (4.47 vs 2.80, $p = 0.0007$) and SGD (4.47 vs 3.13, $p = 0.0009$), given the fact that the way to describe the objects had to be found and could be not always correct.

About the whole section, on average, no significance was found.

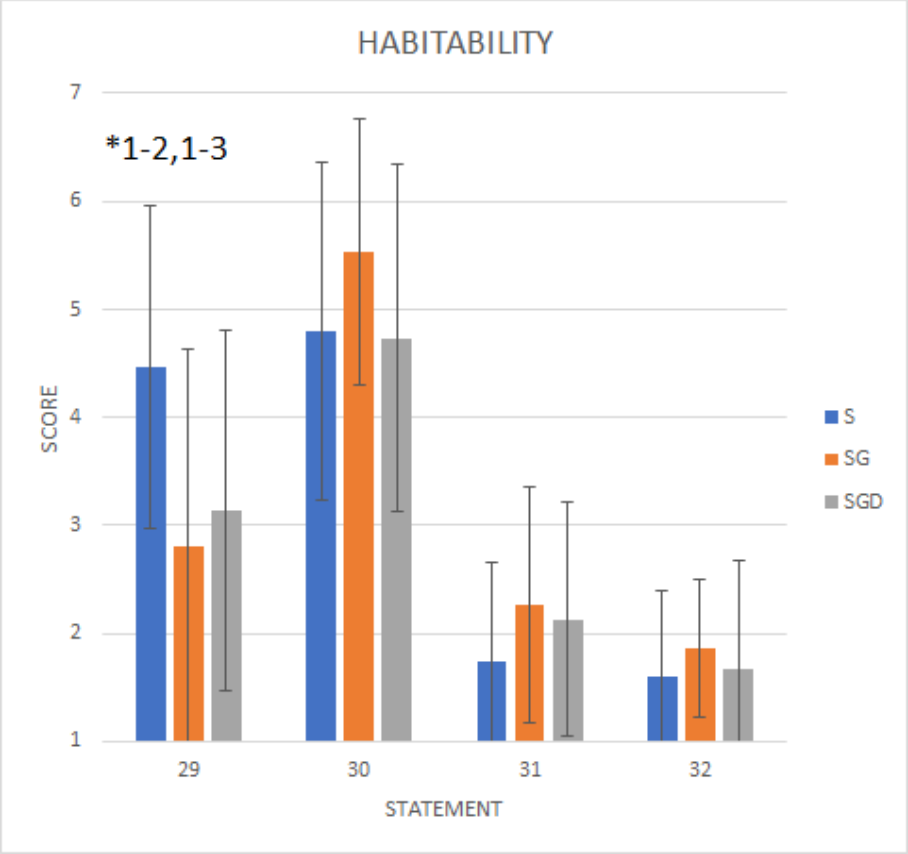

Figure 6.9: Statistics about habitability for each statement. The symbol * indicates a significance.

6.3.6 Speed

In the case of speed, no significant differences were found in the answers to the two statements composing the section. The average values were, however, rather high.

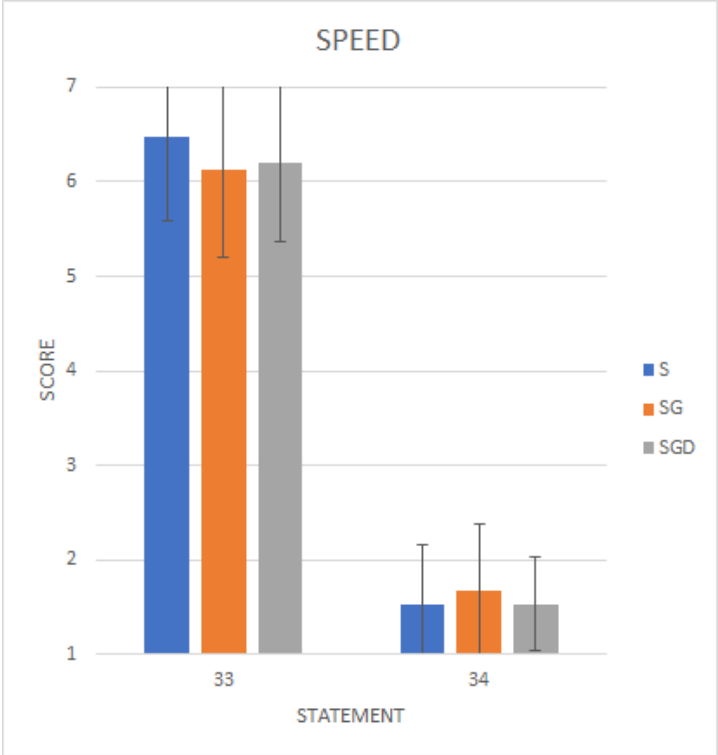

Figure 6.10: Statistics about speed for each statement.

6.4 SUS questionnaire

About the SUS questionnaire [53], over the ten statements composing it, only one of them led to a statistically significant difference between at least two techniques (statement 35) (6.11). In particular, it stated that the user would like to use the specific technique frequently in VR. What emerged in this case was that a significant lower score was given to SG with respect to S (3.00 vs 4.27, $p = 0.0003$) and SGD (3.00 vs 4.00, $p = 0.0104$), coherently to the observations done in the SASSI's likeability section.

In general, however, as shown in Figure 6.12, all the total scores were rather high (89.33 for S, 82.00 for SG, 86.50 for SGD) and, consequently, it can be stated that the three techniques are sufficiently usable.

6.5 Custom part of the questionnaire

About the last set of statements, which concentrated on aspects related to problems which could arise in the specific scenario, many of them led to differences between

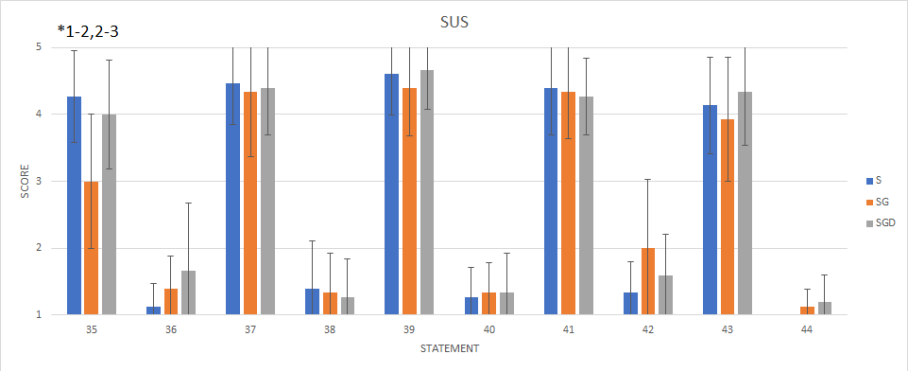

Figure 6.11: The SUS scores for each statement. The symbol * indicates a significance.

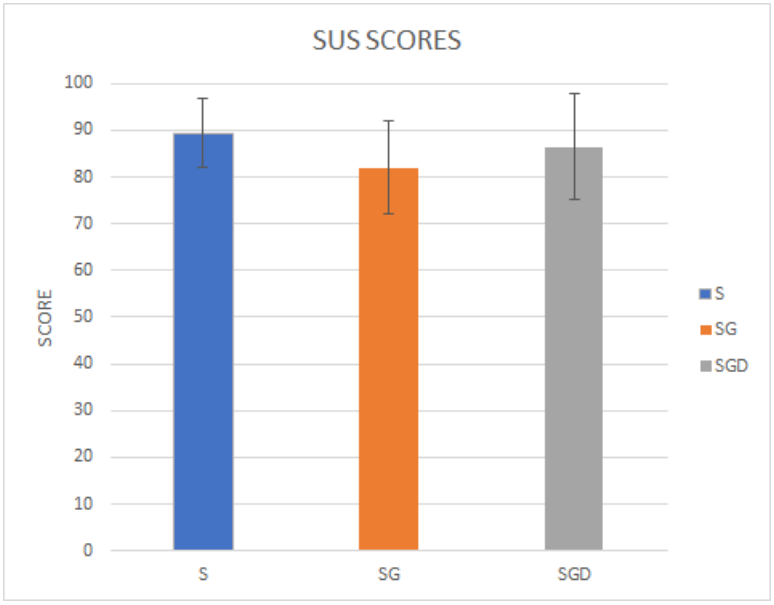

Figure 6.12: SUS average scores for the three techniques.

at least two techniques (6.13). In particular, the questions were ten, and five of them were significant.

Almost all the users, for example, expressed that they found it difficult to reach small and far objects (statement 45) with the SG technique, with a significant difference with respect to both S (3.2 vs 1.4, $p = 0.0001$) and SGD (3.20 vs 1.80, $p = 0.0009$). This was expected as S does not involve the action of pointing, while SGD allows to use the most comfortable techniques depending on the user's choice.

Another aspect that differentiated the SG technique was the higher level of difficulty in understanding how to reach occluded objects (statement 47), this with

respect to both S (2.73 vs 1.60, $p = 0.0075$) and SGD (2.73 vs 1.53, $p = 0.0025$). In fact, while both S and SGD allowed the use of voice, which overcomes the problem of occlusions, with S it could be difficult to understand how to avoid an occluding object when trying to point to the one behind.

The statement 48 told that it was difficult to avoid ambiguities while using the specific technique; in this case, the score of SG was significantly higher than the one of SGD (2.53 vs 1.67, $p = 0.0069$). In fact, the possibility that SGD gives to combine a spatial information and a description or to use description only, allowed to overcome some situations in which the ambiguity was difficult or impossible to avoid.

There was, then, a question which led to differences between all the three pairs of techniques, which was the felt necessity to make references to spatial indications, details or close objects relatively to the POI. The scores, in this case, showed that S overwhelmed SG (4.53 vs 1.80, $p < 0.0001$) and SGD (4.53 vs 2.80, $p = 0.0001$); at the same time, SGD was still significantly higher than SG (2.80 vs 1.80, $p = 0.0059$). In fact, the kind of interaction of SG was not prone to give descriptions, while SGD could make the user do this more, when the speech only component was chosen by the user; however many times in SGD the user decided to use the pointing too.

Finally, SGD also resulted causing the users to be confused about its behaviour, significantly more than S (1.87 vs 1.07, $p = 0.0053$), mainly for the fact that the use of two techniques merged together was not considered so comfortable as expected.

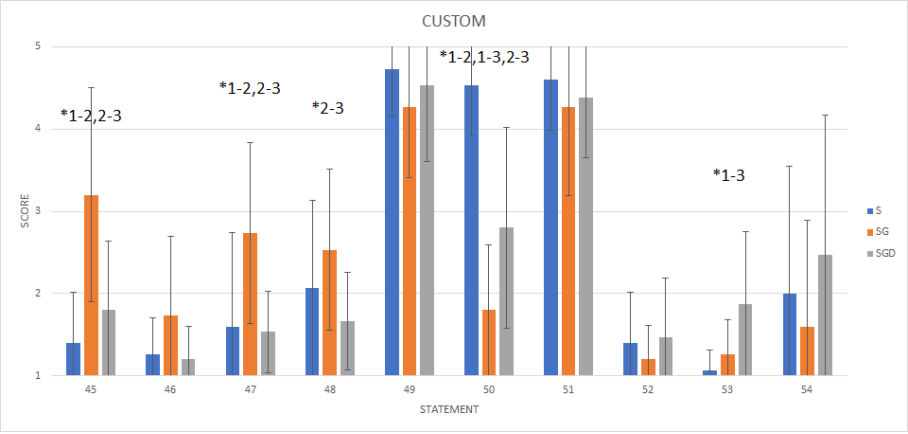

Figure 6.13: The average scores of the several statements belonging to the custom section. The symbol * indicates a significance.

6.6 Preferences

As mentioned before, after the user had filled the questionnaire by giving the scores to the several statements for all the three techniques, it was asked him/her to define a ranking of preferences among them. The results are shown in Table 6.1, where it can be seen that very often SG was the less preferred, for nine times the first choice was SGD, while for six times it was S; however, considering the whole trend of the three techniques, no statistically significant difference was found between the preference of S and the one of SGD.

PREFERENCES RANKING			
USER	1	2	3
1	SGD	S	SG
2	SGD	S	SG
3	S	SGD	SG
4	S	SG	SGD
5	SGD	S	SG
6	SGD	S	SG
7	S	SG	SGD
8	SGD	S	SG
9	S	SG	SGD
10	S	SGD	SG
11	SGD	S	SG
12	SGD	S	SG
13	SGD	S	SG
14	SGD	S	SG
15	S	SGD	SG

Table 6.1: The several preferences for each of the users.

6.7 Discussion on subjective results

As far as the subjective aspects are concerned, in general, what emerged from this study was the clear lower appreciation of SG with respect to S and SGD. This is also demonstrated by the fact that, over the 15 users, none of them expressed his/her preference for SG over the other two, and, moreover, it was almost always chosen as the least preferred one in the ranking.

In fact, this technique was considered characterized by low levels of efficiency

and flexibility. As was noticed by one of the users, "While by using voice it's up to me to say the correct thing to directly address the place I want to reach, for SG, sometimes, the disambiguation event is unavoidable, I have no chance to avoid the appearance of the panel when dealing with a fully occluded object."

Moreover, it was considered frustrating with respect to the other two techniques, mainly for the difficulty the user experienced when trying to point to an object which was small and far, as demonstrated by statement 45 of the custom section. As a user observed, "SG is very comfortable for close objects, but is terrible for the further ones", and many other people declared something similar. Considering that an environment, in general, can be very large, it can be understood that is not acceptable to use a technique which creates this kind of problem.

This explains also why the cognitive demand was perceived as higher in SG than in S, which was initially against the expectations; most of the users did not make a lot of errors while doing the operation of pointing, but then declared that they had to concentrate a lot to perform that task, as expressed by statement 22 of the corresponding SASSI's section. As one of them observed, "SG made me concentrate more on keeping the head in a stable position while pointing than on what to say". However, what is worth to observe is that, according to many of the users, the scenario seemed to be not enough cognitive demanding for S; a typical kind of comment was "In this case it was rather easy to know how to avoid ambiguities, since, for example, the bins were of completely different colors".

As expected, the SG technique was also the most repetitive according to the questionnaire results relative to the annoyance section; it is clear that repeating always the same kind of phrase (e.g., "Take me there", "Let's go there") instead of having to describe the object is perceived as less various.

An aspect that was taken into consideration before the experience but, actually, did not bring to significant results, is the discomfort in pointing to objects at different altitudes; this could be explained by the fact that the experience was not long enough to cause this kind of issue.

On the other hand, SGD was the more preferred, with nine users choosing it in the first place of the ranking proposed at the end of the questionnaire; for six people, finally, S seemed to be the best. By analyzing the average answers of the users to the several statements and taking into account some of their comments, it is possible to understand what are the several reasons that led to these results.

In fact, S was very appreciated for its naturalness, since the possibility to have a single kind of input, which is voice, has been seen as a great way to make decisions in an intuitive way. For many reasons, it was sometimes even preferred to SGD, which, at the beginning, was thought to have a clear predominance on the others for its flexibility (i.e., the possibility to avoid disambiguation and the one to use the preferred component depending on the case).

Regard this fact, even if a lot of users appreciated the freedom to choose the best

way to reach the destination depending on the situation, a lot of them faced the problem of being confused by this combination, which was also confirmed by the answers to statement 53 in the custom section. In particular, some comments were "I find SGD a bit redundant and confusing, I find it better to use a single technique." or "When I use SGD I always need to think for a moment about which of the components of the technique to use before performing the operation." However, the statement 35 of the SUS resulted coherent with the likeability average in the SASSI, as far as significant differences are concerned.

In the habitability section, it was expected that the user would declare that it was less easy to find the correct words to address the POI (statement 29) with the technique S than with SG, and the result confirmed the expectations, being SG a technique in which no description has to be given.

About the statement relative to the need of giving additional references by voice when using the technique (custom section, 50), it was clear that the score would be very high for S, for which it was usually necessary to do that in order to avoid ambiguities (e.g., "the chair close to the plant", "the sofa with the ball", "the sofa on the right", etc.). For SG the score is negligible, since it was not possible to do that (i.e., the voice command type is always the same), while, for SGD, the score stands approximately in the middle between S and SG; this can be explained by the fact that, when using this technique, sometimes the user chose to use the S component, while in other cases he/she chose the SG one, depending on what he/she considered more comfortable at the specific moment; moreover, many users decided to use one component much more than the other in general.

Considering all those sections for which no statistical significance was found, the results were mostly coherent to our expectations. For the system response accuracy, for example, high values were desired for all the three techniques, which meant that, besides the several advantages and disadvantages, they were not implemented in a bad way. The same can be said for speed, which resulted in no statistically significant differences among the three. In this case, the fact that SG has not resulted in being slower than S (coherently with the objective results), despite the significant lower value obtained for efficiency in the system response accuracy section, can be explained with the fact that the speed metric was more related to the average perceived time needed to reach the destination; instead, the efficiency can be more interpreted in relation to the possibilities the technique gives to the user to perform the operation correctly; for example, many times the user had difficulties in being understood when pronouncing short words like "lì" ("there") during the disambiguation scenario, and this could have impaired the value obtained for the SG technique itself.

As far as habitability is concerned, it can be considered positive that statements like 31 and 32, expressing a certain unawareness from the user of the system behaviour, have not shown negative aspects, since, again, this could have been

considered a defect on the design of the interface in terms of usability.

Chapter 7

Conclusions and future works

This study presented a comparison of three techniques derived from literature, using somehow voice commands to perform the teleportation task in immersive VR.

The results, which have seen the predominance of S and SGD over SG, showed that, in general, what is mostly appreciated by the users is the flexibility and the naturalness of the interaction. For this reason, SG, even if it could be considered easier to use in terms of voice commands, was not appreciated for all its limitations with respect to both these two aspects (i.e., flexibility, because of the impossibility to avoid occlusions, and naturalness, since the head movements were considered a bit forced). S was appreciated a lot for its naturalness, since, for many of the users, the possibility to recall by using voice only the POI was seen as a great advantage and, in the end, the cognitive load expected resulted overrated. However, what must be said is that a more precise study can take into consideration the creation of some less intuitive characteristics of the objects (e.g., different gradations of the same color or put many of them in different places, which do not allow to simply distinguish between "left" and "right"), in order to stimulate more the cognitive demand for this technique.

Another further study can be done in the future by performing other kinds of comparison between speech-based teleportation techniques (even not derived from literature), by removing the SG one, which was not successful enough, and implementing in its place something that could be considered more natural, for example, a technique exploiting eye-gaze. In fact, this could be a better idea because, as a user observed "When I want to address a place, I look at it, I don't rotate my head in its direction."

A fourth technique that could be worth to consider could be the one of pointing

by hand, which, in this study, has been discarded since it was considered too similar to SG, using the same kinds of input. Actually, after having understood the importance of naturalness, it can be deduced that the use of hand to point to an object could result more intuitive; this was, in fact, also a suggestion of one of the users: "I would have preferred to point by using the finger instead of the head."

Moreover, a different kind of experience can be built in order to understand the power of the several techniques. For example, a scenario in which the user cannot only address some specific POIs, but he/she is also free to move wherever he/she wants in the space, can give different interesting results. For example, the user could say "Bring me forward 2 meters" or "Take me 1 meter on the left of the chair", giving some more specific details about his/her desired destination point. In fact, a comment of a user was "A limitation I found in this experience is the impossibility to reach a point if no reference object is present there." With this possibility, it is probable that different techniques present other kinds of strengths and limitations in performing this action.

Finally, what can be observed is that this study kept the user focused on the task of teleportation, but what happens, for example, if, meanwhile, the user has to perform other kinds of task? How can the naturalness of the interaction or the flexibility of it affect an experience that is based on a specific task to be completed, in which teleportation is just an exigence in order to carry it out? Will one of the compared techniques result more frustrating when interfering with the actions on which the user is concentrated? Different kinds of scenarios can be implemented (e.g., manipulating objects, visiting a museum) in order to asses the best way to perform teleportation for each of them. In this case, it could be useful to make an improvement of all those techniques exploiting ray-cast, which is, as suggested by a person, to make the ray activatable and deactivatable by the user during the experience, depending on his/her exigences.

Appendix A

Questionnaire

A.1 Demographic section

- ID
- Age
- Experience with VR (from 1 to 5)
- Experience with the use of voice interfaces (from 1 to 5)

A.2 SASSI (1: totally disagree, 7: totally agree)

A.2.1 System responce accuracy

1. The technique allows the system to interpret my commands accurately
2. The technique is unreliable
3. The use of the technique brings to unpredictable results
4. The technique didn't always allow me to do what I wanted
5. The technique didn't always bring the system to do what I expected
6. The technique is dependable
7. The technique allows making few errors
8. The interaction through the technique is consistent
9. The interaction through the technique is efficient

A.2.2 Likeability

10. The system is useful in this scenario
11. The system is pleasant
12. The system is friendly and user supportive
13. I was able to recover easily from errors committed
14. I enjoyed using the system in this context
15. It is clear how to speak to the system
16. It is easy to learn to use the technique
17. I would use this technique frequently in VR
18. I felt in control of the interaction with this technique

A.2.3 Cognitive demand

19. I felt confident using the technique
20. I felt tense using the technique
21. I felt calm using the technique
22. A high level of concentration is required when using the technique
23. The technique is easy to use

A.2.4 Annoyance

24. The interaction through the technique is repetitive
25. The interaction through the technique is boring
26. The interaction through the technique is irritating
27. The interaction through the technique is frustrating
28. The technique is too inflexible

A.2.5 Habitability

29. I sometimes wondered if I was using the right word

30. I always knew what to say to the system

31. I was not always sure what the system was doing

32. It is easy to lose track of where you are in an interaction through this technique

A.2.6 Speed

33. The technique allows to rapidly reach the destination

34. The system responds too slowly to the execution of the action through the technique

A.3 SUS (1: totally disagree, 5: totally agree)

35. I think that I would like to use this technique frequently in VR

36. I found the technique unnecessarily complex

37. I thought the technique was easy to use

38. I think that I would need the support of a person already capable of using this technique to be able to use it

39. I found the various functions of the technique were well integrated

40. I thought there was too much inconsistency in this technique

41. I would imagine that most people would learn to use this technique very quickly

42. I found the technique very cumbersome to use

43. I felt very confident using the technique

44. I needed to learn a lot of things before I could get going with this technique

A.4 Custom section (1: totally disagree, 5: totally agree)

45. I found it difficult to reach destinations represented by small and far objects
46. I found uncomfortable to reach destinations placed at different altitudes
47. I found it difficult to understand how to reach objects which were occluded by others
48. I found it difficult to avoid ambiguities in selecting the destination
49. I found it easy to understand how to use the technique when the panel appeared
50. When using the technique, I felt the necessity to give additional references (e.g., spatial, closeness to other objects, characterizing details) in order to reach the destination
51. The way to interact with the disambiguation panel is coherent with the way to interact without it
52. I kept making mistakes while using the technique
53. I was confused by the behaviour of the technique
54. I didn't need to use all the functionalities offered by the technique

A.5 Ranking

Insert the three technique in order of preference (1: most preferred, 3: less preferred). There is no possibility of a draw.
1:
2:
3:

A.6 Comments

Bibliography

[1] Asmaa Alraizzah, Foaud Lamya, and Lamia Fattouh. «Environments and System Types of Virtual Reality Technology in STEM: A Survey». In: *International Journal of Advanced Computer Science and Applications* 8 (June 2017). DOI: 10.14569/IJACSA.2017.080610 (cit. on pp. 17–19, 21).

[2] Etienne van Wyk and Ruth Villiers. «Virtual reality training applications for the mining industry». In: Jan. 2009, pp. 53–63. DOI: 10.1145/1503454.1503465 (cit. on p. 19).

[3] Sharon Farra, Matthew Gneuhs, Eric Hodgson, Burhan Kawosa, Elaine Miller, Ashley Simon, Nathan Timm, and Jackie Hausfeld. «Comparative Cost of Virtual Reality Training and Live Exercises for Training Hospital Workers for Evacuation». In: *CIN: Computers, Informatics, Nursing* 37 (June 2019), p. 1. DOI: 10.1097/CIN.0000000000000540 (cit. on p. 19).

[4] URL: https://www.manufacturingtomorrow.com/article/2018/12/2018-used-cnc-machine-price-average-cnc-machine-price/12676/ (cit. on p. 19).

[5] Yung-Chou Kao, Chung-Shuo Lee, Zhi-Ren Liu, and Yu-Fu Lin. «Case study of virtual reality in CNC machine tool exhibition». In: *MATEC Web of Conferences* 123 (Jan. 2017), p. 00004. DOI: 10.1051/matecconf/201712300004 (cit. on pp. 19, 20).

[6] Randi Mao, Lucy Lan, Jeffrey Kay, Ryan Lohre, Olufemi Ayeni, Danny Goel, and Darren de SA. «Immersive Virtual Reality for Surgical Training: A Systematic Review». In: *Journal of Surgical Research* 268 (July 2021), pp. 40–58. DOI: 10.1016/j.jss.2021.06.045 (cit. on p. 20).

[7] Jessica Maples-Keller, Brian Bunnell, Sae-Jin Kim, and Barbara Rothbaum. «The Use of Virtual Reality Technology in the Treatment of Anxiety and Other Psychiatric Disorders». In: *Harvard Review of Psychiatry* 25 (May 2017), pp. 103–113. DOI: 10.1097/HRP.0000000000000138 (cit. on p. 20).

[8] Sharad Sharma, Phillip Devreaux, David Scribner, Jock Grynovicki, and Peter Grazaitis. «Megacity: A Collaborative Virtual Reality Environment for Emergency Response, Training, and Decision Making». In: *Electronic Imaging* 2017 (Jan. 2017), pp. 70–77. DOI: 10.2352/ISSN.2470-1173.2017.1.VDA-390 (cit. on p. 20).

[9] Valentin Holzwarth, Joy Gisler, Christian Hirt, and Andreas Kunz. «Comparing the Accuracy and Precision of SteamVR Tracking 2.0 and Oculus Quest 2 in a Room Scale Setup». In: Mar. 2021 (cit. on p. 21).

[10] Alberto Cannavò, Davide Calandra, Filippo Gabriele Pratticò, Valentina Gatteschi, and Fabrizio Lamberti. *An Evaluation Testbed for Locomotion in Virtual Reality*. Oct. 2020. DOI: 10.1109/TVCG.2020.3032440 (cit. on pp. 21, 34).

[11] Evren Bozgeyikli, Andrew Raij, Srinivas Katkoori, and Rajiv Dubey. «Point & Teleport Locomotion Technique for Virtual Reality». In: Oct. 2016, pp. 205–216. DOI: 10.1145/2967934.2968105 (cit. on pp. 21, 31, 32).

[12] Yasin Farmani and Robert Teather. «Evaluating discrete viewpoint control to reduce cybersickness in virtual reality». In: *Virtual Reality* (Dec. 2020). DOI: 10.1007/s10055-020-00425-x (cit. on pp. 21, 31).

[13] Noura Abdi, Kopo Ramokapane, and Jose Such. «More than Smart Speakers: Security and Privacy Perceptions of Smart Home Personal Assistants». In: June 2019 (cit. on pp. 21, 27).

[14] Puneet Mittal and Navdeep Singh. «Speech Based Command and Control System for Mobile Phones: Issues and Challenges». In: Feb. 2016, pp. 729–732. DOI: 10.1109/CICT.2016.150 (cit. on pp. 21, 22, 25).

[15] Sebastian Pick, Andrew Puika, and Torsten Kuhlen. «Comparison of a speech-based and a pie-menu-based interaction metaphor for application control». In: Jan. 2017, pp. 381–382. DOI: 10.1109/VR.2017.7892336 (cit. on pp. 22, 29).

[16] Djamel Aouam, Samir Benbelkacem, Nadia Zenati, Sardi Zakaria, and Zaoui Meftah. «Voice-based Augmented Reality Interactive System for Car's Components Assembly». In: Oct. 2018, pp. 1–5. DOI: 10.1109/PAIS.2018.8598516 (cit. on pp. 22, 30).

[17] A. Veh, R.J. Marceau, A. Malowany, P. Desbiens, A. Daigle, E. Garant, R. Gauthier, A. Shaikh, and J.C. Rizzi. «Design and operation of a virtual reality operator-training system». In: *Power Systems, IEEE Transactions on* 11 (Sept. 1996), pp. 1585–1591. DOI: 10.1109/59.535701 (cit. on pp. 22, 31, 36).

[18] Jaisie Sin and Cosmin Munteanu. «Let's Go There: Voice and Pointing Together in VR». In: Oct. 2020, pp. 1–3. DOI: 10.1145/3406324.3410537 (cit. on pp. 22, 36, 39, 51).

[19] Rohit Mehra, Vibhu Saujanya Sharma, Vikrant Kaulgud, Sanjay Podder, and Adam Burden. «Immersive IDE: Towards Leveraging Virtual Reality for creating an Immersive Software Development Environment». In: Oct. 2020. DOI: 10.1145/3387940.3392234 (cit. on pp. 22, 23, 37, 39).

[20] Hans-Christian Schmitz, Frank Kurth, Kevin Wilkinghoff, Uwe Müllerschkowski, Christian Karrasch, and Volker Schmid. «Towards Robust Speech Interfaces for the ISS». In: Mar. 2020, pp. 110–111. DOI: 10.1145/3379336.3381496 (cit. on p. 26).

[21] Christian Deuerlein, Moritz Langer, Julian Sessner, Peter Heß, and Jörg Franke. «Human-robot-interaction using cloud-based speech recognition systems». In: *Procedia CIRP* 97 (Jan. 2021), pp. 130–135. DOI: 10.1016/j.procir.2020.05.214 (cit. on p. 26).

[22] Leigh Clark, Benjamin Cowan, Abi Roper, Stephen Lindsay, and Owen Sheers. «Speech diversity and speech interfaces: considering an inclusive future through stammering». In: July 2020, pp. 1–3. DOI: 10.1145/3405755.3406139 (cit. on p. 27).

[23] José Pérez-Córdoba, Juan Martín-Doñas, Angel Gomez, Alejandro Gomez-Alanis, and Jose Gonzalez Lopez. «Silent Speech Interfaces for Speech Restoration: A Review». In: *IEEE Access* 8 (Sept. 2020), pp. 177995–178021. DOI: 10.1109/ACCESS.2020.3026579 (cit. on p. 27).

[24] Stan Jou, Tanja Schultz, Matthias Walliczek, Florian Kraft, and Alex Waibel. «Towards continuous speech recognition using surface electromyography.» In: Jan. 2006 (cit. on p. 27).

[25] Bradley Betts and Charles Jorgensen. «Small Vocabulary Recognition Using Surface Electromyography in an Acoustically Harsh Environment». In: (July 2008) (cit. on p. 27).

[26] URL: https://developer.amazon.com/en-US/docs/alexa/alexa-auto/communication.html (cit. on p. 27).

[27] Daniel Hepperle, Yannick Weiss, Andreas Sieß, and Matthias Wölfel. «2D, 3D or Speech? A Case Study on which User Interface is Preferable for what Kind of Object Interaction in Immersive Virtual Reality». In: *Computers & Graphics* 82 (June 2019). DOI: 10.1016/j.cag.2019.06.003 (cit. on p. 28).

[28] Pedro Monteiro, Guilherme Gonçalves, Hugo Coelho, Miguel Melo, and Maximino Bessa. «Hands-free interaction in immersive virtual reality: A systematic review». In: *IEEE Transactions on Visualization and Computer Graphics* 27 (May 2021), pp. 2702–2713. DOI: 10.1109/TVCG.2021.3067687 (cit. on p. 29).

[29] Zhiyi He, Chuan Lv, Di Peng, and Dequan Yu. «A speech recognition-based interaction approach applying to immersive virtual maintenance simulation». In: July 2017, pp. 1–5. DOI: 10.1109/ICRSE.2017.8030764 (cit. on p. 30).

[30] Markus Funk, Florian Müller, Marco Fendrich, Megan Shene, Moritz Kolvenbach, Niclas Dobbertin, Sebastian Guenther, and Max Mühlhäuser. «Assessing the Accuracy of Point & Teleport Locomotion with Orientation Indication for Virtual Reality using Curved Trajectories». In: Apr. 2019, pp. 1–12. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300377 (cit. on p. 31).

[31] Andreas Linn. «Gaze Teleportation in Virtual Reality». MA thesis. 2017 (cit. on p. 31).

[32] Heni Cherni, Natacha Métayer, and Nicolas Souliman. «Literature review of locomotion techniques in virtual reality». In: *International Journal of Virtual Reality* 20 (Mar. 2020), pp. 1–20. DOI: 10.20870/IJVR.2020.20.1.3183 (cit. on pp. 32, 33).

[33] Sam Tregillus and eelke folmer eelke. «VR-STEP: Walking-in-Place using Inertial Sensing for Hands Free Navigation in Mobile VR Environments». In: May 2016. DOI: 10.1145/2858036.2858084 (cit. on p. 32).

[34] Yun Suen Pai and Kai Kunze. «Armswing: using arm swings for accessible and immersive navigation in AR/VR spaces». In: Nov. 2017, pp. 189–198. ISBN: 978-1-4503-5378-6. DOI: 10.1145/3152832.3152864 (cit. on pp. 33, 34).

[35] Davide Calandra, F. Lamberti, and Massimo Migliorini. «On the Usability of Consumer Locomotion Techniques in Serious Games: Comparing Arm Swinging, Treadmills and Walk-in-Place». In: Sept. 2019. DOI: 10.1109/ICCE-Berlin47944.2019.8966165 (cit. on p. 34).

[36] Heni Cherni, Nicolas Souliman, and Natacha Métayer. «Using virtual reality treadmill as a locomotion technique in a navigation task: Impact on user experience – case of the KatWalk». In: *International Journal of Virtual Reality* 21 (May 2021), pp. 1–14. DOI: 10.20870/IJVR.2021.21.1.3046 (cit. on p. 34).

[37] URL: http://www.vivecraft.org/ (cit. on p. 35).

[38] Richard Stoakley, Matthew Conway, and Randy Pausch. «Virtual Reality on a WIM: Interactive Worlds in Miniature.» In: Jan. 1995, pp. 265–272 (cit. on p. 36).

[39] Samuel Truman and Sebastian von Mammen. «An Integrated Design of World-in-Miniature Navigation in Virtual Reality». In: Sept. 2020. DOI: 10.1145/3402942.3402994 (cit. on p. 36).

[40] Andrea Ferracani, Marco Faustino, Gabriele Giannini, Lea Landucci, and Alberto Del Bimbo. «Natural Experiences in Museums through Virtual Reality and Voice Commands». In: Oct. 2017, pp. 1233–1234. DOI: 10.1145/3123266.3127916 (cit. on p. 37).

[41] Difeng Yu, Qiushi Zhou, Joshua Newn, Tilman Dingler, Eduardo Velloso, and Jorge Goncalves. «Fully-Occluded Target Selection in Virtual Reality». In: *IEEE transactions on visualization and computer graphics* PP (Sept. 2020). DOI: 10.1109/TVCG.2020.3023606 (cit. on pp. 38, 64).

[42] Adlin Shaflina, Farhan Mohamed, Chan Vei Siang, and Muhammad Ismail. «Virtual Reality 360 UTM Campus Tour with Voice Commands». In: Dec. 2020, pp. 1–6. DOI: 10.1109/ICIDM51048.2020.9339665 (cit. on p. 39).

[43] Di Chen, Ravin Balakrishnan, and Tovi Grossman. «Disambiguation Techniques for Freehand Object Manipulations in Virtual Reality». In: Mar. 2020, pp. 285–292. DOI: 10.1109/VR46266.2020.1581293094740 (cit. on pp. 39, 59).

[44] Paul Dickson, Jeremy Block, Gina Echevarria, and Kristina Keenan. «An Experience-based Comparison of Unity and Unreal for a Stand-alone 3D Game Development Course». In: June 2017, pp. 70–75. DOI: 10.1145/3059009.3059013 (cit. on pp. 40, 41).

[45] URL: https://docs.unity3d.com/ (cit. on p. 41).

[46] URL: https://visualstudio.microsoft.com/it/ (cit. on p. 42).

[47] URL: https://docs.microsoft.com/ (cit. on p. 44).

[48] URL: https://www.oculus.com/quest-2/ (cit. on p. 45).

[49] URL: https://valvesoftware.github.io/steamvr_unity_plugin/index.html (cit. on p. 45).

[50] URL: https://www.w3.org/ (cit. on p. 52).

[51] URL: https://en.wikipedia.org/wiki/Latin_square (cit. on p. 72).

[52] URL: https://icons8.com/icon/59836/microphone (cit. on p. 76).

[53] Kate Hone, Ub Ph, Robert Graham, and Alencon Link. «Towards a Tool for the Subjective Assessment of Speech System Interfaces (SASSI)». In: *Natural Language Engineering* 6 (July 2000). DOI: 10.1017/S1351324900002497 (cit. on pp. 77, 82, 88).

[54] John Brooke. «SUS: A quick and dirty usability scale». In: *Usability Eval. Ind.* 189 (Nov. 1995) (cit. on p. 77).

[55] URL: https://mathworld.wolfram.com/ANOVA.html (cit. on p. 79).

[56] URL: https://mathworld.wolfram.com/Pairedt-Test.html (cit. on p. 79).

[57] URL: https://mathworld.wolfram.com/BonferroniCorrection.html (cit. on p. 79).