Locomotion and Telepresence in Virtual and Real Worlds

VIII Edizione Premio MIMOS

Candidate

Alessandro Spada

Thesis Advisor

Alessandro De Luca

Sapienza
University of Rome

2019

Contents

Chapter 1

Introduction

1.1 Virtual Reality

Virtual Reality refers to a computer technology that aims at replicating a real environment in order to simulate user's presence. It is possible to imagine that VR was born during the 19th century, where some embryonic discovers were made. Indeed in that period it was demonstrated that the brain processes two images from the eyes in a unique tridimensional object. Subsequently during the first half of 900 some further dowels in the development are added, with the creation of a flight simulator.

During these years the research and development of new solutions are growing the fastest, mostly because we are capable to build relatively small and portable solutions with high computing power, high resolution, capable of deceiving human eye. Indeed more and more companies in the entertainment field are offering to the audience new devices to enrich multimedia experiences. At the same time, these technologies may be applied to new sectors, such as arts, culture and also medicine.

1.2 Sight in VR

Virtual reality was born with the purpose of creating the illusion of seeing something that actually does not exist. Nowadays we can distinguish technological solutions between those that are developed around the user and those that are in contact with him.

The first group refers to those devices that require a physical structure to create virtual environments, for example the C.A.V.E. This recursive acronym stands for *Cave Automatic Virtual Environment*, which is a room-size cube that uses projectors to show non existing scenarios.

However the user is supposed to be non moving, or at most with limited motion inside the cube. Besides he doesn't loses perception of the real world, as it perceives this situation as it is, a room.

A better situation is instead one where the user is able to substitute its environment with another one that is completely virtual. In other words his eyes should be fully or at most partially covered, in order to enable motion.

1.2.1 Head Mounted Displays

Another complete different approach to Virtual Reality is based on wearable devices, also called *head mounted displays*. Depending on the purpose of the device, these hmd may present different structures. In other words they may not be equipped strictly with an external display, but they may be show some informations directly on the lenses.

Figure 1.1. Samsung Gear VR

These headset provide distortion lenses and other sensors, as a imu, using the phone as a display. Thus there is no more need of a standalone vision device, letting every user get in contact with off-the-shelf solutions. Besides these devices have the advantages to be wireless, thus the user is able to move on his environment. As a consequence virtual or augmented reality applications become increasingly wider, from medicine to arts and culture.

1.3 Omnidirectional Treadmills

Locomotion interfaces refer to several devices involving the human body, with the purpose of getting natural or induced locomotion. Several approaches have been introduced, most of all capturing movements through a mechanical structure. In other words we have a passive conveyor belt which moves following user's movements.

This approach, as well as its similar, presents a flat walking surface. A completely different situation is shown if this surface is curved, such as Virtusphere or Cybersphere. Indeed the user is supposed to enter inside this circular structure, which is not scalable to every kind of people. Besides these systems don't capture other aspects of human motion, for example crouching. Thus the need to rethink the working principle and use instead sensorized systems.

Chapter 2

Cyberith Virtualizer

2.1 Hardware

Virtualizer is an omnidirectional treadmill developed by Austrian company *Cyberith*. The project of this device was presented on Kickstarter on 2014 and in small time raised far beyond the requested amount. Its structure is composed by a base plate, which is connected to three vertical pillars. Then a circular structure is connected to these pillars, which is composed by an inner ring which can rotate along an outer one. A harness connected to the inner ring completes the circular structure. The ring structure can move along the pillars in order to enable crouching.

Figure 2.1. Cyberith Virtualizer

As regards its sensors, it has three set of sensors. The first one is a group of six optical sensors responsible of measuring player's feet velocity and direction. The second set is mounted on the pillars and it is responsible of measuring player's height. The last set is mounted on the ring and gives player's orientation from the starting position. Finally a vibration unit mounted on the base plate completes the hardware.

2.1.1 Working Principle

The working principle of the device is simple and at the same time complete. It is based on the principle of low friction, i.e. the user is supposed to push the hips slightly against the ring. In this manner one foot slides backwards, while the other is doing a step. The belt at the same time compensates the remaining friction, enabling running, walking and crouching with stability. Being a commercial product its full behaviour is not available, however in principle every set of sensors has an optical encoder and when the device is started they start doing measurements. Supposedly they are infrared sensors, i.e. they working coupling a light emitter with a receiver. The receiver will read the amount of the reflected light of the source and give accordingly a value. Then these values are stored in a memory inside the treadmill, which is then read by a thread running on the computer through external libraries working on every OS.

2.2 Software

This section will describe software modules used for this work and also Virtualizer's APIs. These functions cover almost every needed information to develop a walking algorithm for the robot, or at most every function to move it on a plane.

The set of API is pretty straightforward. It consists of few calls that cover every feature of the device. Indeed they retrieve player's height, which gives the current height of the ring construction in centimeters, player's orientation, which gives back the orientation of the inner ring compared the outer one. Concerning speed, we can retrieve player's speed, orientation. which tells if the player is moving right, forward or if it is not moving. Finally we can trigger a vibration signal to the platform.

One of the most pressing issue connected with a device like the Virtualizer is the quality of data that is returned. Indeed some measurements without any noise filtering may cause malfunctioning of the robot, which in case of humanoids may cause falls. However, as state by the developers of the device, those measurements undergo a series of noise filtering routines before being released to the user.

Chapter 3

Oculus Rift

The *Oculus Rift* is a virtual reality headset developed by Oculus VR, which is an american technology company. Like the other companion device on this work, it was proposed on Kickstarter in 2012. Turns out the campaign raised about US$2.5 million dollars. This project then rapidly gained fame and interest from the audience, until it was bought by Facebook.

3.1 Hardware

Figure 3.1. Oculus Rift CV1

Concerning internal components, the Rift has two amoled screen with a resolution of 1980x1020 per eye. It has headphones, a microphone, together with a magnetomer, gyroscope and accelerometer. The panels have a refresh rate of 90 Hz and global refresh, instead of scanning out in lines. The Rift has also low persistence, i.e. they display an image for 2 milliseconds of each frame. All these features leave to the user a natural watching sensation, without any motion sickness that may arise with previous versions.

3.2 Working Principle

The working principle of the Oculus Rift is based on co-working of the hmd with the infrared camera. Indeed around the device there is a constellation of leds that are invisible to the human eye, being installed inside the coverage. These leds are visible to the infrared camera, thus at each moment the software takes data from it coupled with inputs from internal sensors. Through a combination of these informations the sdk is capable of computing the position of the hmd and also to compensate some latency with a prediction phase.

Another important aspect of the working principle is how images are rendered on the Rift. Images are rendered on high-definition display inside the visor that are positioned near the eyes.

From the developer side, the Oculus Rift has its sdk developed mostly for game development platforms, such as Unity or Unreal Engine. This is because this device finds his place in the entertainment field. In a parallel way, the user is also able to develop other works with its `C++` sdk. This is indeed the adopted solution, in fact in this work the vision module has been developed using external IDE, skipping the above mentioned gaming tools.

Chapter 4

Telepresence

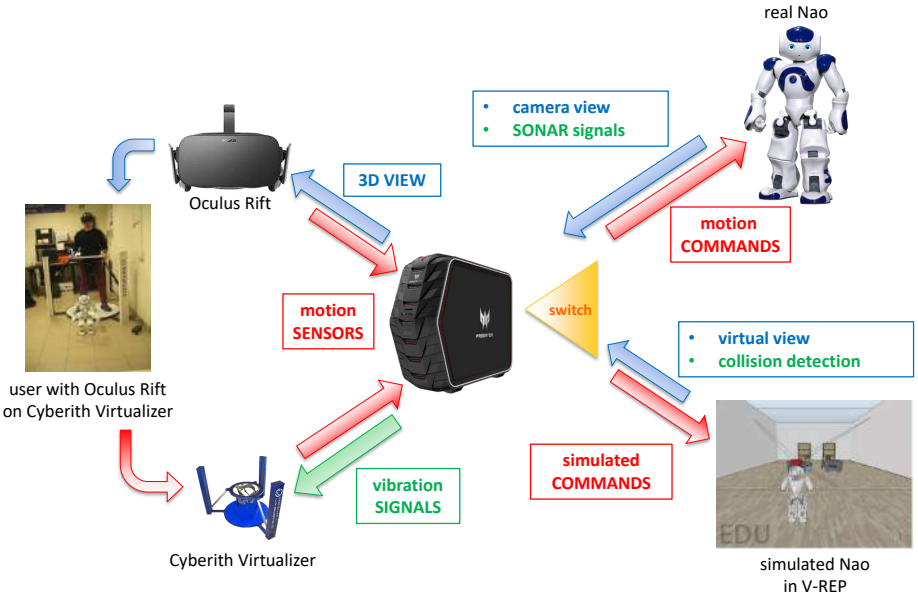


Figure 4.1. Overview of the Implementation

4.1 Framework Overview

The core of the algorithm can been described as two parallel threads working together. The first thread is responsible of **streaming** the visual output. To this end the user has the chance to stream a stereo camera mounted on the head of the robot to the hmd (see 4.3), or internal one if not comfortable with the multidimensional sensation. The same thread responsible for the video streaming, has the purpose of **mapping** user's head movements to robot's head. Thus it is responsible of grabbing pitch, yaw angles from the Oculus Rift and yaw from robot's head and send it to the robot, and at the same time grabbing one or more frames from robot's internal

camera or from the external stereo one and render on the Oculus Rift.

On the other side there is another running thread, which is responsible of taking input from the Virtualizer and send it to the robot. Thus at each iteration of its associated loop it will take user's orientation and feet velocity from the Cyberith, apply thresholds and control to these values and finally send the modified values to robot.

Another feature that is presented in figure 4.1 is collision detection: as already presented in 2, the treadmill has a vibrating unit on the base plate. A walker that is navigating with a robot can be then warned of a collision through robot's sensors. See 5.4 for more details.

4.2 Robot motion

From previous sections it was introduced that the Virtualizer returns velocities, precisely meters per second, as output of its API. Thus it is possible to use these informations as input to control a robot, through a precise human/humanoid velocity scaling or scaling the world, i.e., shrinking objects from a scene in order to approximate a robot to a human.

4.2.1 NAO Robot

For this work a NAO robot was considered, developed by french company Aldebaran. Its own OS enables the user to program with several options, even sending velocity commands. An important aspect that has been considered developing both modules is that all functions must be non-blocking, i.e. the robot must be able to receive multiple instructions, also interrupting the actual one if present.

One of the main problem of the motion mapping phase is that the Virtualizer does not return both linear and angular velocities, but only the linear one and the angle of the ring construction from the starting position. Thus the need of a control law that should bring the robot in the desired position, transforming the two informations from the treadmill onto two velocities, on linear component v_{nao} and the angular component ω_{nao}. Besides it is not possible to control robot's position in a non-blocking fashion, thus a velocity controlled law is adopted.

4.3 Stereoscopy

Another crucial aspect of telepresence is visual immersion, in other words the user needs to see the surrounding environment of the robot. One important problem is that the robot has two cameras, but they are not placed in order the create a stereo one. These cameras are positioned vertically, the upper one to identify goals or balls, the lower one to ease dribbles. Thus stereoscopy has been obtained only in a simulated environment with a virtual robot, using two vision sensors positioned at a distance comparable to the human eyes' one.

Chapter 5

Implementation

5.1 Tools

Making the simulation working has been a bit tricky, indeed the Rift is developed only on Windows machines,besides there is also no direct implementation of the Virtualizer's sdk in linux, thus Windows has been preferred again. On the other side the robot simulation is working only on linux, then we had to use a parallel linux machine with the simulation tool running.

As concerns the development environment, Visual Studio is the adopted IDE using as programming language is `C++`. One useful feature of this program is that is able to build projects with previous compiler: this is indeed the case of this work, as everything was compiled with 2010 version. The adopted simulation tool is V-REP, which is multi-platform robot simulation environment, developed by Swiss Company Coppelia.

Once obtained a stable version of an algorithm using the simulation tools, a real robot was used to walk inside the laboratory and in the corridor. Seen from the robot side, the implementation was effortless.

5.2 Locomotion module

The first module is responsible of Nao's motion: at each iteration of a loop it will acquire player's speed vel and orientation θ. Then three motion cases are distinguished:

- if $vel \sim 0$ and angular speed $\theta \neq 0$, then the player is rotating on himself without walking. Due to the fact that the robot cannot be position-controlled in a non blocking fashion, then there is a velocity correction until the robot gets on the right position.
- if $vel \neq 0$ and $\theta \sim 0$ the player is walking forward thus send calls will have only components on the x-axis.
- both values not null, i.e. the player is walking while turning. In this case the send will have both linear and angular component.

Obviously these values won't ever have a perfect 0-value, thus a threshold system in applied. These thresholds are not given a priori, but must be computed through some tuning.

5.3 Visualisation module

This module is responsible of returning a visual feedback to the user. Depending on the initial configuration, the streaming of the images can be shown onto the oculus or to the screen.

Regarding the stereo camera, at the moment it is possible to stream a 3d scenario of the Rift only in a simulated environment, mostly because Nao's wireless card is not strong enough to stream heavy data from two photographic lenses. This thread at the same time has the purpose of retrieving pitch and yaw angles from the Rift together with Nao's head yaw angle and send to the robot the corrected values.

Figures 5.1 and 5.2 show visual output in case of a real or a virtual robot is operating.

Figure 5.1. Oculus Rift output with a real Nao

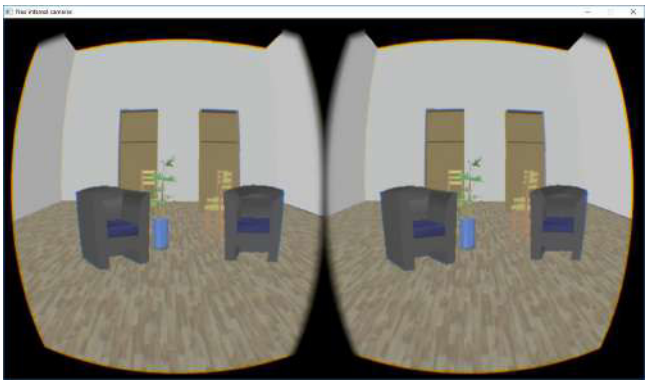


Figure 5.2. Oculus Rift output with a simulated Nao

5.4 Other features

This is actually a part of 5.2, but it has a different purpose. Indeed this module is responsible of testing if the robot is colliding with objects in its environment. In case of a collision is detected it triggers a vibration to the base plate. As soon as the collision is avoided, the vibration stops.

In particular we need to distinguish two cases, simulation or not. This is crucial because collision detection is completely different, mostly because the simulator does not emulated Nao's sensors. Thus in the first case collision is done using an internal system of the simulator, based on distance computation between the robot and another object. Then it is necessary to do the test with every single object, in other words there isn't collision detected with all items of the scene tout court.

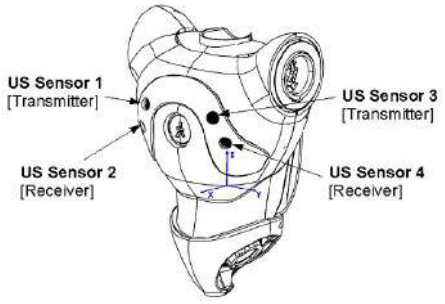


Figure 5.3. NAO's sonars

On the other side the real robot is equipped with two sonars on the chest, see fig.5.3. Using its related APIs, it is then easily tested the distance of the nearest object. If this value is below $30cms$, then a vibration is triggered. However those sensors actually are very energy consuming, thus for a better autonomy is it advisable to switch-off this feature.

As a recap, once the algorithm has taken the desired configuration from the GUI, it starts two threads. The first is the motion one, in which the user will be asked to crouch in order to start the navigation. In this manner it is possible to enter the Virtualizer after starting the program. Once this condition is satisfied, then the Nao will stand up and communicate to be ready to walk. In a parallel way the second thread, i.e., the visualisation one, will start the exact routine telling the user only if the Rift was activated or the internal camera of the robot on the screen. Figure 5.4 shows a visual representation of this algorithm.

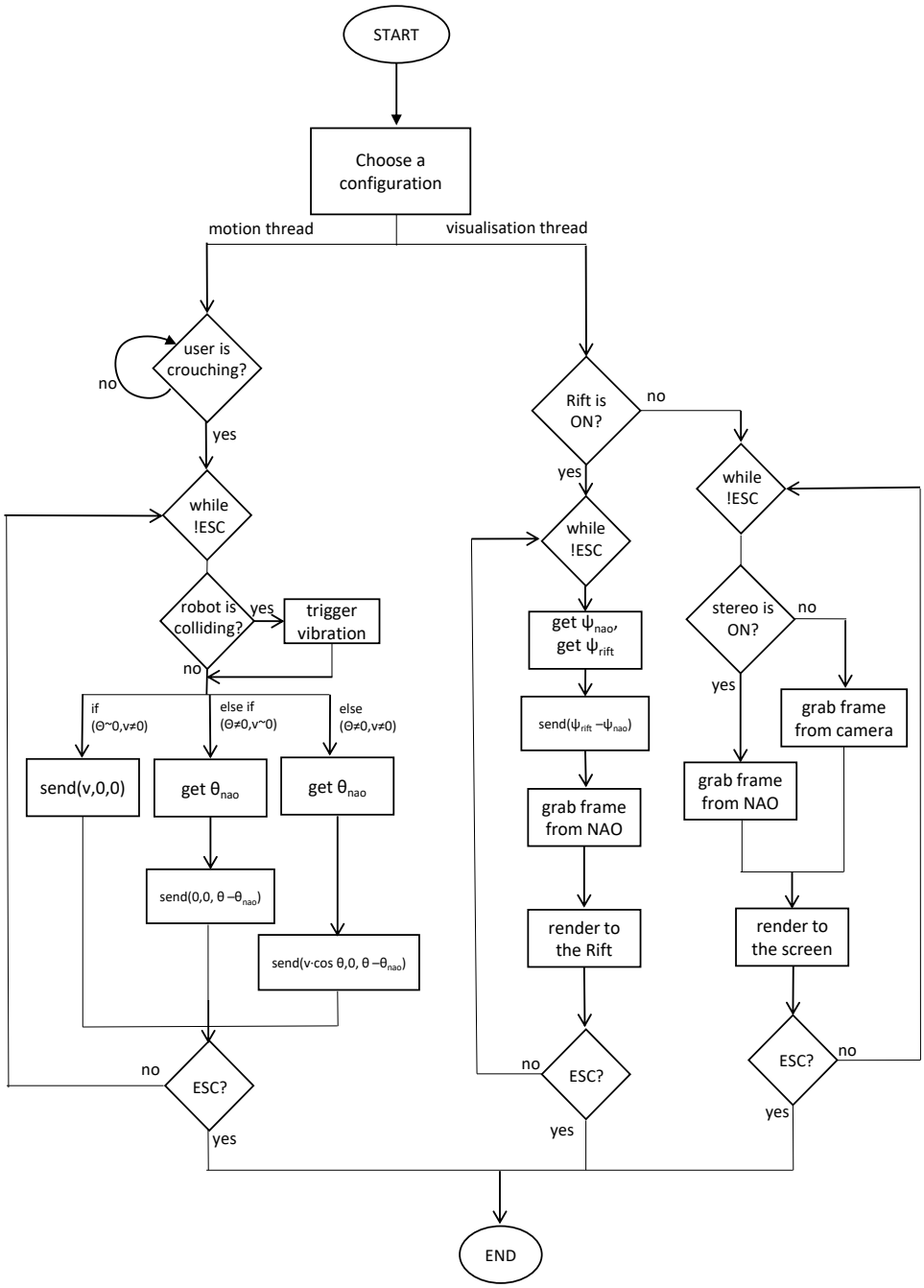


Figure 5.4. State diagram of the algorithm: at the start the user is asked to choose a configuration from the GUI. After this phase two threads are started, one for the motion and the other for the visualisation. The first one then starts a loop, in which it reads data from Virtualizer and sends it to the robot. Depending on the motion type, it will retrieve from the robot its orientation. The second one has two routines: the first one is started if the Rift is activated, the other one in the other case. The latter will in turn differentiate if a stereo camera is considered, or Nao's internal one.

Chapter 6

Test and Results

Concerning experiments, several situations were tested with the purpose of validating the strength and reliability of the code. Of course tests are chosen so that to apply the proposed control laws for the body and the head. The other tests are mostly a visual manner, thus there is no need to plot informations. The situations that are tested ar the following: straight walk, on place rotation, walk and rotation and corridor navigation.

In the following sections will be presented some plots regarding the results of the control laws in some on these situations. In particular these will report the yaw angles of the body and of the head in the second and third case. The last one is left as a visual experiment, while the first does not require any application of the proposed laws in sec.4.2.1.

6.1 Simulated robot

As regards the simulation, results are expected to be more precise because the error is estimated through the simulator. The first test is a simple rotation of $\frac{\pi}{2}$ clockwise, which outcome is shown in fig.6.1.1. The next test is also a rotation, coupled with a walk: in this case the robot is supposed to describe a curve. Results are in fig.6.1.2.

6.1.1 Simple Rotation

As shown in fig.6.1 and 6.2, the phase where the robot follows the Virtualizer is the slowest. This is because there are more factors that influence robot's reactivity, among which the gain that slows convergence. However with a more accurate observation of the plot relative to head's movement, it is possible to observe how the motion is more precise. Thus another cause of this slowness may be connected with substantiation of a movement by the robot.

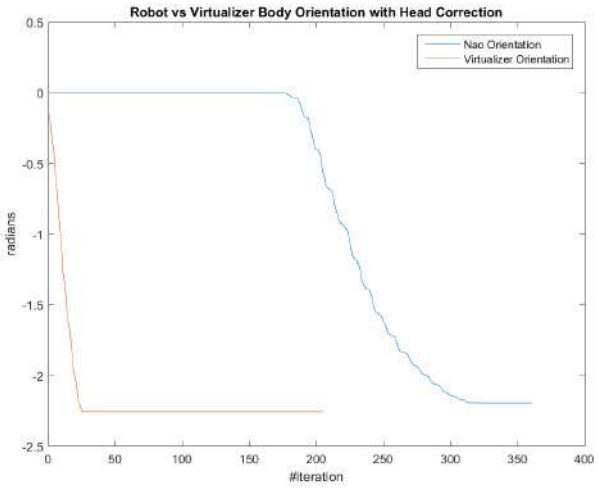


Figure 6.1. Signals from Nao's body and Virtualizer during a rotation, with head correction: the delay is clearly shown on the figure. This may be decreased with a better connection.

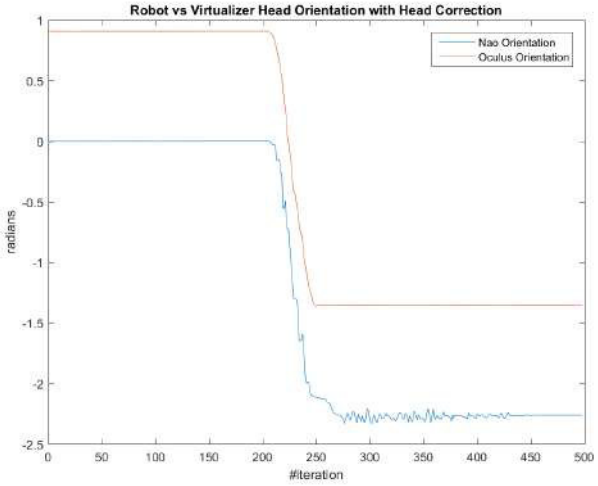
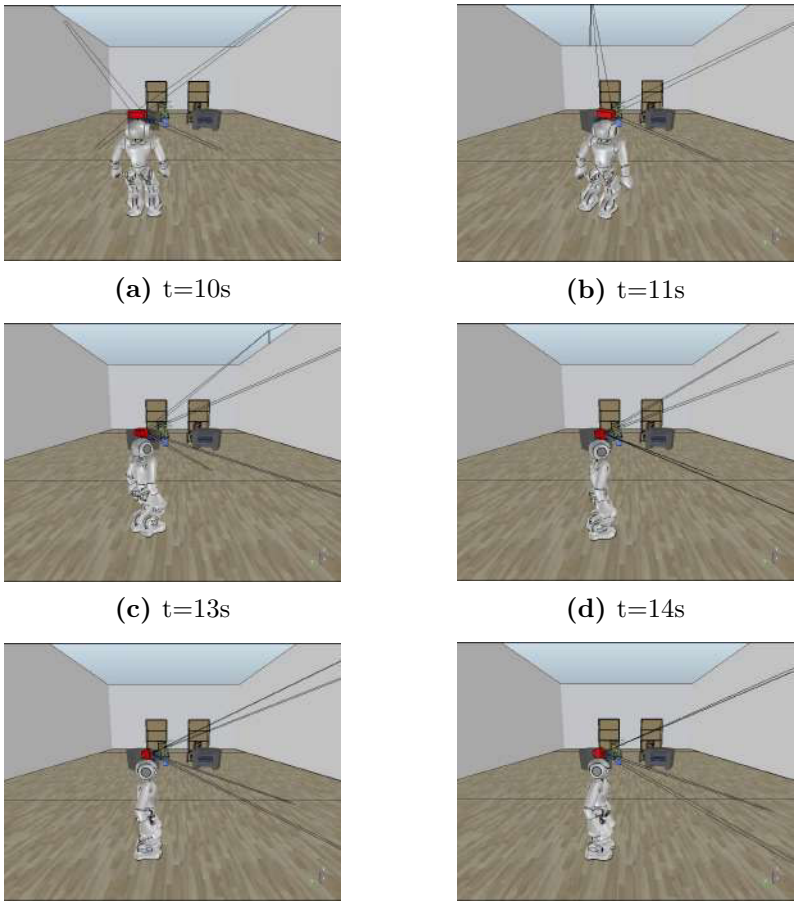


Figure 6.2. Signals from Nao's head and Oculus Rift during a rotation, with head correction: the delay is significantly lower, even though some oscillations are shown. The Rift does not start in a zero position because measurements are in the frame of the IR camera.



(a) t=10s **(b)** t=11s

(c) t=13s **(d)** t=14s

(e) t=16s **(f)** t=19s

Table 6.1. Frames from a video during a rotation with a simulated Nao

6.1.2 Rotation and walk

In this situation a more complex motion is considered, the user is rotating as before but at the same time he is walking. The result is that the robot will describe a small curve, until he stops. From the point of view of the plots a similar behaviour is expected, with a slight oscillation due to the motion of the hips and the head during the walk.

As predicted, it is possible to observe in fig.6.3 a small perturbation of the orientation due to an oscillation around the end point of the harness. As before the body is affected by a significant latency with an error that is asymptotically compensated. A better outcome is shown with the head, even though it suffers of slight fluctuations of user's head.

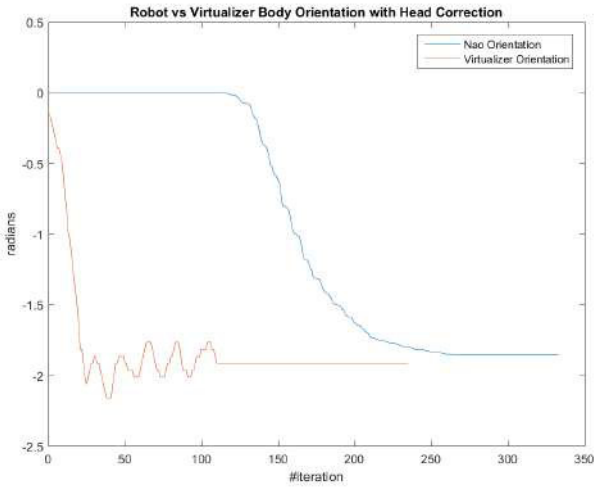


Figure 6.3. Signals from Nao's body and Virtualizer during a rotation and walk, with head correction: the signal from the Virtualizer shows some oscillations around the goal position due to the walking phase. After some initial delay the robot gets on the desired position, whereas those variations are sensitively reduced.

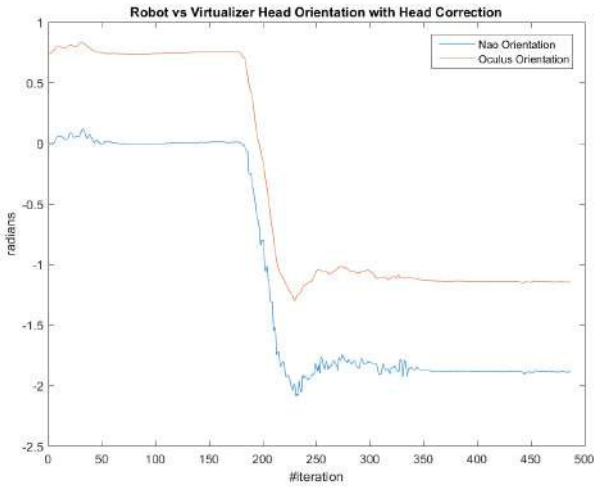
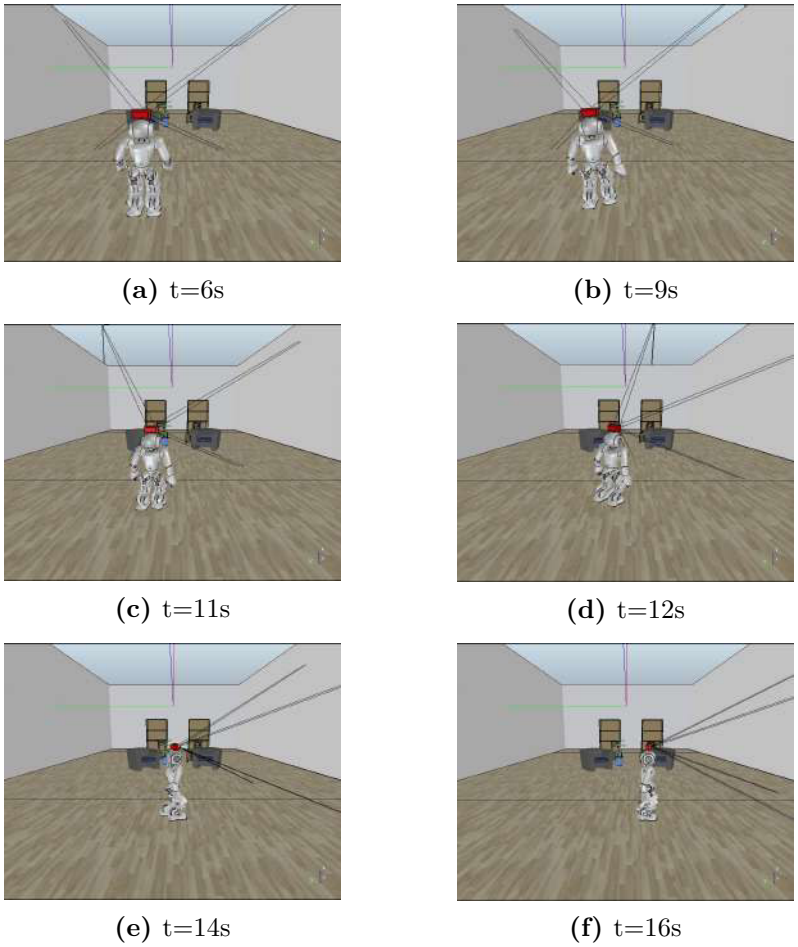


Figure 6.4. Signals from Nao's head and Oculus Rift during a rotation and walk, with head correction: as before the signal from the Nao's head shows a better responsiveness in following the Rift, even though it is affected by a bigger noise.



(a) t=6s **(b)** t=9s

(c) t=11s **(d)** t=12s

(e) t=14s **(f)** t=16s

Table 6.2. Frames from a video during a walk and rotation with a simulated Nao

6.2 Real robot

Being in real situation, we cannot expect the precision of the simulator regarding the control laws. This is because there is no more an exact and exterior orientation estimation system. In this case indeed the estimation is done through an internal system, thus the robot won't never reach exactly the desired position. This error will be also worsen by communication delay. Fortunately this error is perceivable mostly on robot's position, while the estimation of the yaw of the head keeps its preciseness.

Another problem connected to a real situation is also the bad wireless communication. Indeed switching from a wired communication to a wireless one, the frame rate drops violently. Results are slightly better if resolution is lowered, but at the same time the immersion sensation is pretty unnatural. For the purposes of getting a satisfying result in terms of good decoupling, it is necessary a better vision system. However for a naive user that is not used to it, the sensation is pretty striking as well. In section 7 will be given a possible solution to obtain a better streaming system.

Figure 6.5. A photo taken from a test with a real robot: at the start of the simulation it is advisable not to move the head in order to avoid any malfunctioning of robot's one

6.2.1 Simple Rotation

The following pictures show that the behaviour of the robot is analogous to the previous cases, even though some oscillations are registered during the estimation of robot's orientation. This is due obviously to the absence of an external odometry estimation system, like V-REP. Indeed in this case the orientation is estimated through internal sensors, that are imprecise and noisy.

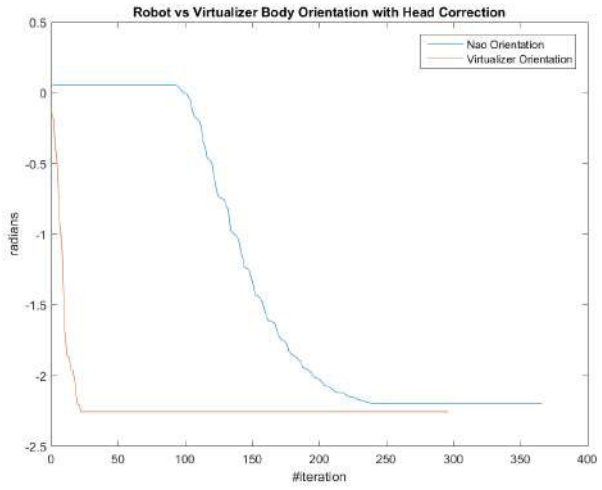


Figure 6.6. Signals from a real Nao's body and Virtualizer during a rotation, with head correction

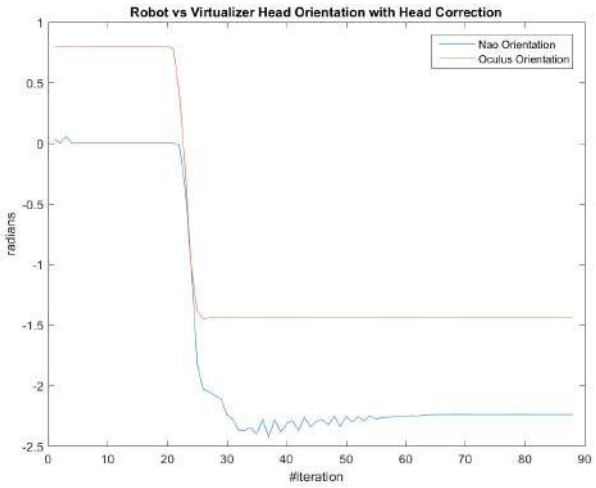


Figure 6.7. Signals from a real Nao's head and Oculus Rift during a rotation, with head correction

In order to give a visual feeling of how significant is the latency, the following table shows a sequence of screenshots taken from a video done in the laboratory. As clear the user is at its goal position already at the second frame, while the robot barely started its motion.

(a) t=23

(b) t=31

(c) t=35

(d) t=39

Table 6.3. Frames from a video during a rotation with a real Nao: the caption below every picture is the time instant where the frame was taken

6.2.2 Rotate and Walk

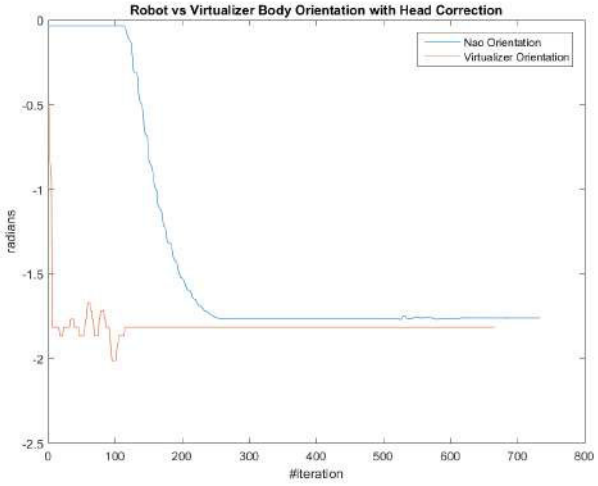


Figure 6.8. Signals from a real Nao's body and Virtualizer during a rotation and walk, with head correction

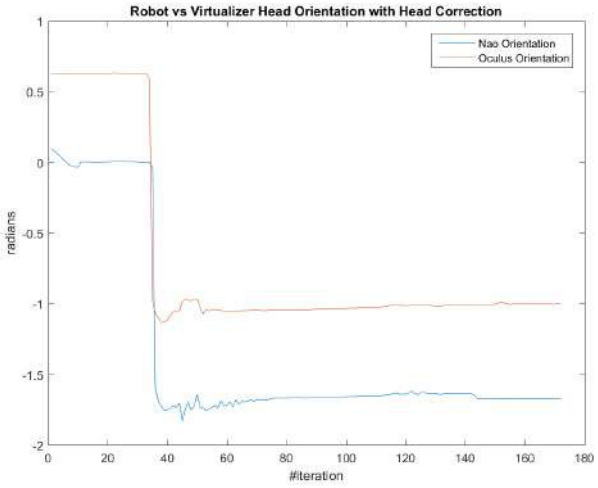


Figure 6.9. Signals from a real Nao's head and Oculus Rift during a rotation and walk, with head correction

As before table 6.4 shows some frames taken from a video: even though in 6.4b the desired position is already reached, the robot will rotate to the correct position only in the last one. Thus the delay is still remarkable, however a full navigation is still possible.

(a) t=0

(b) t=2

(c) t=5

(d) t=7

Table 6.4. Frames from a video during a walk and rotation with a real Nao: the caption below every picture is the time instant where the frame was taken

Chapter 7

Conclusions and Future Works

This document presented an application of Virtual Reality and Telepresence to Robotics, both in a simulated world and a real one. The emulation of the real world in a fictitious environment is catching larger and larger attention of the public and the producers, thus new devices are taking hold with excellent performances. This is possible because recent technologies show the best performances with reduced dimensions. This work than handled the problem of taking these new devices and apply them to new situations, like humanoid robotics. It is indeed possible to enter inside a robot, moving and watching as a robot.

Regarding future implementations, this work presents a large scope of improvements. Indeed what concerns the locomotion module, it may be extended introducing also grasping of objects. This may be implemented using a hand-tracking algorithm when the robot is not moving, through for example a Kinect. In this way it possible to mirror user's movements to robot, with the purpose of grasping an object that is placed in front of it. The same camera can be also track feet, in order to go up and down stairs.

It is also possible to improve velocity mapping between user and robot introducing a scaling factor. Indeed this constant cannot be fixed for every user, instead it shall depend on the singular user. Indeed he will have a greater or lesser predisposition to walk on the Virtualizer, as much as the velocities that can be reached will be different. This quantity, together with other considered gains, may be computed by a neural network, learning the most appropriate value from the physiognomy of each person.

Finally in merit to the vision module, it is possible to mount on the robot a stereo camera in order to bring stereoscopy as done in simulation. However one of the biggest drawback of this robot is that the USB port on its head does not allow a good and efficient sending of frames. As a result we may experience a laggy result, not satisfying for immersion. One possible solution is to mount also a small router as a backpack, in such a way that the internal antenna of the robot is responsible to send and receive motion data. This router indeed will send only frames from the stereo camera, in order to lighten the mole of data transmitted.