# Addressing the problem of Interaction in fully Immersive Virtual Environments: from raw sensor data to effective devices

Master Degree Thesis of Daniele Sportillo

Supervisior

Dott. Franco Tecchia

Co-supervisor

Dott. Marcello Carrozzino

*Università di Pisa, Scuola Superiore Sant'Anna*
Master of Science in Embedded Computing Systems

Academic Year 2014-2015

# Contents

# Preface

Virtual Reality is an immersive medium which gives the feeling of being entirely transported into a virtual three-dimensional world, and can provide a far more visceral experience than screen-based media.
From the user's point of view, the main properties of a Virtual Reality experience are *presence*, *immersion* and *interaction*:

- **Presence** is the mental feeling of being in a virtual space. It's the first level of magic for great VR experiences: the unmistakable feeling that you've been teleported somewhere new. Comfortable, sustained presence requires a combination of the proper VR hardware, the right content and an appropriate system. Presence feeling is strictly related to the user involvement.

- **Immersion** is the physical feeling of being in a virtual space, at a sensorial level, by means of interfaces. It's related to the perception of the virtual world as actually existing. The perception is created by surrounding the user in images, sounds and other stimuli that provide an engrossing total environment.

- **Interaction** is the user capacity to modify the environment and to receive from it feedback to his actions. It's related to the realism of the simulation. The interaction can be direct or mediated. In the first case the user interacts directly with the VE (CAVE-like systems), in the second case the user interacts with the VE by means of an avatar either in first person (HMD systems) or third person (Monitor).

## Purpose of the project

Immersion into Virtual Reality is a perception of being physically present in a non-physical world. The perception is created by surrounding the user of the VR system with images, sound or other stimuli that provide an engrossing total environment. The use of technological devices such as stereoscopic cameras, head-mounted displays, tracking systems and haptic interfaces allows for user experiences providing a physical feeling of being in a realistic world, and the term ‘immersion’ is a metaphoric use of the experience of submersion applied to representation, fiction or simulation.
One of the main peculiarity of fully immersive virtual reality is the enhancing of the simple passive viewing of a virtual environment with the ability to manipulate virtual objects inside it.
This Thesis project investigates such interfaces and metaphors for the interaction and the manipulation tasks. In particular, the research activity conducted has allowed the design of a thimble-like interface that can be used to recognize in real-time the human hand’s orientation and infer a simplified but effective model of the relative hand’s motion and gesture. Inside the virtual environment, users provided with the developed systems is therefore able to operate with natural hand gestures in order to interact with the scene; for example, they can perform positioning task by moving, rotating and resizing existent objects, or create new ones from scratch.
This approach is particularly suitable when there is the need for the user to operate in a natural way, performing smooth and precise movements. Possible applications of the system to the industry are the immersive design in which the user can perform Computer-Aided Design (CAD) totally immersed in a virtual environment, and the operators training, in which the user can be trained on a 3D model in assembling or disassembling complex mechanical machineries, following predefined sequences.

The thesis has been organized around the following project plan:

- Collection of the relevant State Of The Art
- Evaluation of design choices and alternatives for the interaction hardware
- Development of the necessary embedded firmware
- Integration of the resulting devices in a complex interaction test-bed
- Development of demonstrative applications implementing the device
- Implementation of advanced haptic feedback

## Structure of the document

Following the flow of the work this document has therefore been structured in the following chapters:

- *Chapter 1* gives a general overview of Virtual Reality technologies, introducing the problems addressed during this Thesis with a description of the tracking systems and the most common visualization systems which VR applications rely on.
- *Chapter 2* presents the results of the State Of The Art collection: the current solutions in the literature regarding interaction and manipulation in Immersive Virtual Environments are analyzed and a general overview of 3D interfaces for user interaction is presented.
- *Chapter 3* presents the proposed system, describing the requirements to satisfy, the design choices and the implementation of the devices in the existing system. *Section 3.6* analyzes the need of adequate feedback in Virtual Reality applications, presents a novel haptic interface and the way in which it has been integrated with the presented device.

- *Chapter 4* presents the applications that have been developed and which implements the proposed system, describing also the different ways in which the user's hands can be rendered in the virtual environment. It's also described the pilot study conducted for assessing the effectiveness of the training performed in a virtual environment with respect to the real counterpart, presenting the methodologies and the results of the study.
- *Chapter 5* presents the results of the whole Thesis project, analyzing the achieved objectives and a range of possible future developments which arise from the ongoing works.

# 1. Introduction

*If there is a sense of reality,*
*there must also be a sense of possibility.*

Robert Musil

The subject of this Thesis deals especially with Immersive Virtual Reality, which consists of an artificial environment where the user feels just as immersed as he usually feels in consensus reality. An Immersive Virtual Environment requires a strong *presence feeling* in order to make the interaction natural and to improve user's perception of it. The presence feeling is determined by the quality of sensorial information, the sensors mobility and the environment control.
To create a sense of full immersion, the 5 senses (sight, sound, touch, smell, taste) must perceive the digital environment to be physically real. Immersive technology can perceptually fool the senses through:

- Panoramic 3D displays or Head Mounted Displays which provide stereoscopic vision and in which the dynamic perspective is linked to the head movements. The Virtual Environments should be realized with realistic scale and properties.
- Surround sound acoustics or 3D audio effect
- Haptics and force feedback by means of wearable interfaces
- Smell replication

- Taste replication and artificial flavor

The *immersion* represents the boundaries within which the place illusion -the illusion of being in a place in spite of the consciousness of being not actually there- can take place.
Enabling the mind's continual suspension of disbelief requires particular attention to detail. If VR experiences ignore fundamental best practices, they can lead to simulator sickness - a combination of symptoms clustered around eyestrain, disorientation, and nausea. Historically, many of these problems have been attributed to sub-optimal VR hardware variables, such as system latency.

## 1.1 Visualization systems

An immersive VR system provides real-time viewer-centered head-tracked perspective with a large angle of view, interactive control, and binocular stereo display. One consideration is the selection of a technological platform to use for the presentation of VEs.
In this section is presented an overview of the two most common visualization systems used in immersive virtual reality applications: CAVE systems and systems based on Head Mounted Display. As stated in [47] HMDs offer many advantages in terms of cost and portability with respect to CAVE systems. From the other side, CAVEs provide broader field-of-view, higher resolution and a more natural experience for the user.

### 1.1.1 The CAVE system

A Cave Automatic Virtual Environment (CAVE) is a cubical, room-sized fully immersive visualization system. The acronym is also a reference to the allegory of the Cave in Plato's Republic in which a philosopher contemplates perception, reality and illusion. The walls of the room are made up of rear-projection screens on which high-resolution projectors display images. The user goes inside of the system wearing shutter/polarizer glasses to allow for

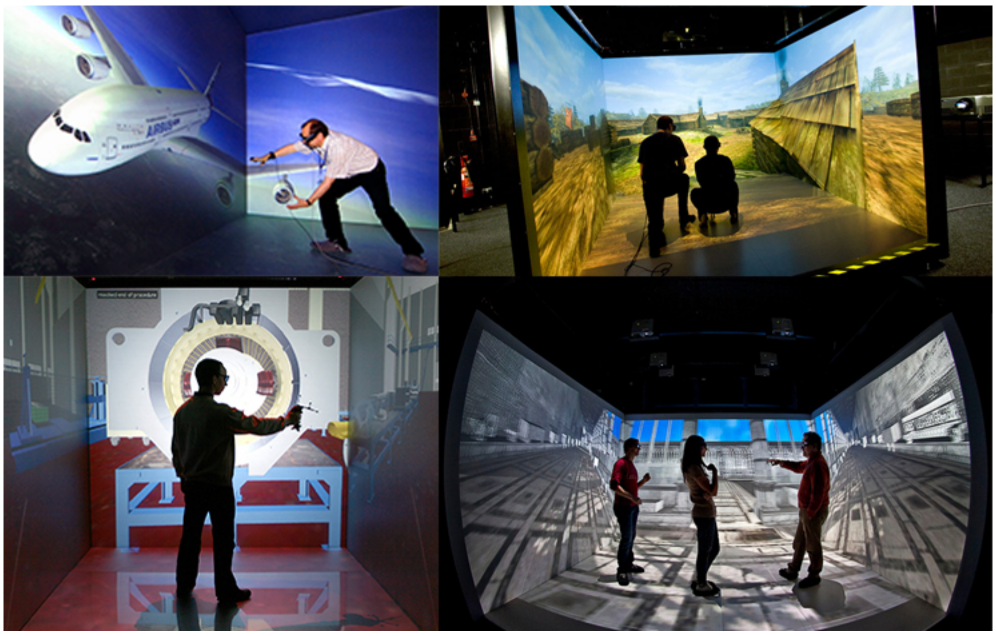


Figure 1.1: CAVE systems

stereoscopic viewing.
CAVE systems provide complete sense of presence in the virtual environment and the possibility of being used by multiple users at the same time. A part from glasses, users inside a CAVE do not need to wear heavy headgear (like in HMD systems).
The SSSA X-CAVE [39] is an immersive visualization system developed by PERCRO Laboratory of Scuola Superiore Sant'Anna (Fig. 1.1). It's a 4-meters 4-walls room in which each wall is divided into several tiles in order to ensure a good resolution. Front, right and left screens are subdivided into four back-projected tiles each, for a global resolution of about 2500x1500 pixels per screen, whilst the floor is subdivided into six front-projected tiles for a global resolution of about 2500x2250 pixels. The X-CAVE is managed by means of the XVR technology, exploiting its distributed rendering features on a cluster of five workstations.
The system is also provided with 7 cameras for optical position and orientation tracking.

### 1.1.2 Head Mounted Displays

A Head Mounted Display (HMD Fig. 1.2a) is a device worn by a user on the head which is provided with a monocular or a binocular display. Binocular HMDs can be used to achieve stereoscopic vision by showing different images for the two eyes; this allows also the user to perceive image depth.
There exists also a variant to this type of device (Optical HMD Fig. 1.2b) which displays images to the user's eyes and at the same time allows the user to see through it (Optical See-Through); OHMDs are mainly used in Augmented Reality or Mixed Reality applications [8].

(a) Traditional HMD

(b) Optical HMD

Figure 1.2: Head Mounted Displays

The **Rift** is a virtual reality head-mounted display developed by Oculus VR. The screen displays two images side by side, one for each eye. A set of lenses is placed on top of the screen, focusing and reshaping the picture for each eye, and creating a stereoscopic 3D image. Launched in a Kickstarter campaing, it represents a revolution in the field of HMD, mainly due to its low-cost ($ 350) and to its very high value for money.
The version of the Oculus Rift used during this Thesis is the DevelopmentKit 2 (DK2) which packs a 5-inch OLED display with a resolution of 960 x 1080 pixels per eye and a 100-degree field of view. The displays also use low persistence, meaning that they only light the pixels for 2 milliseconds of each frame. The headset has a refresh rate of up to 75 Hz, with an internal-tracking update rate of 1000 Hz and a positional-tracking update rate of 60 Hz.

Figure 1.3: A user wearing the Oculus Rift DK2

The position of the user's head is tracked with a system called 'Constellation', consisting of an external infrared camera, included in the package. Behind the face of the Rift are 40 high intensity IR LEDs, though you can't actually see them. Combined with the camera, this adds positional tracking. Positional tracking enables you have full motion freedom in the VR environment: you can sit down, stand up, lean forwards or even look around a corner. By knowing the position of the LEDs on the objects and their pattern, the system can determine the precise position of the object, down to sub-millimeter accuracy.
Oculus offers SDKs to integrate Rift in different software like Unity, Unreal Engine and UDK.

## 1.2 User tracking in VR

The tracking is one of the key technological challenges in Virtual Reality, and it is a crucial point for providing high fidelity immersion feeling and interaction capabilities to the users. Hence the importance of having a robust and accurate tracking system [32].
In this type of systems, the tracking devices interact with the processing

unit, providing the system with spacial information about the user. In systems that allow a user to move around within a physical space, trackers detect the position and the orientation of the user, the direction he is moving and his speed.
Tracking systems consist of a mix of hardware and software which is able to detect the absolute position and orientation of an object. Exploiting tracking data it is possible to measure and report all the 6 DOF of real-life. The accuracy of tracking is of primary importance: since virtual reality is about emulating and altering reality, if it is possible to accurately track how parts of the body (like the head or the hands) move in real life, then it allows for faithfully representing these inside a virtual environment.
Knowing head position and orientation allows to coherently update the viewpoint inside the VE. Knowing position and orientation of hands and fingers allows to detect if the user is interacting with virtual objects.
Trackers usually consists of:

- acquisition components (sensors) which record in real-time some relevant values
- processing components which receives data from sensors and calculate the desired values.

An ideal tracker should fulfill the following requirements [44]:

- Accuracy: it's related with the smallest change in position and orientation that can be detected by the tracker. In scene coordinates, position should be within 1mm, and orientation errors should be less than 0.1 degrees.
- Hi frequency of acquisition: the rate at which measurements are reported by the tracker to the host computer should be very high in order to detect any change.
- Low Latency: the delay between a change in position and orientation and the report of the change to the host should be small.

- Low encumber: users should be unrestricted in their mobility. It would be desirable to have no cables attached to the devices, and all mobile parts of a tracking system should be very lightweight. For full mobility, no stationary parts should be required at all.
- Robustness: Small motions should always lead to small changes in tracker output. Outliers have to be suppressed. Tracking operation should be continuous over time.
- 6-DoF: absolute position within the x, y and z coordinates of a space and the orientation (roll, pitch, yaw).
- No shadowing: there should be no space in which the tracker is not able to effectively measure.

When dealing with HMDs, in order to calculate the views on the display, the position and the orientation of the user's head are required [16].

- Head position is used to calculate the position of the viewpoint
- Head orientation is to calculate the view direction and the eye offset direction

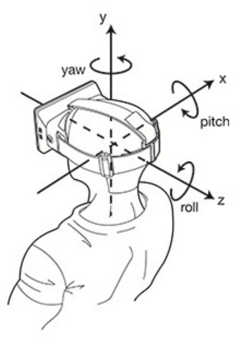


Figure 1.4: Orientation parameters for HMD

View position and direction are used to calculate the view volume and therefore to dynamically update the perspective. The eye offset direction is used to know on which axis stereo images must be separated.

### 1.2.1 Types of tracking systems

In VR, there exist several different kind of tracking systems: some of them are based on an emitter/receiver configuration, some other are self-contained systems. All these systems allow to track position and orientation of the user in different ways, with pros and cons for each solution.
In the following is presented an overview of the technologies which could be used for tracking purposes in VR.

#### Mechanical tracking

Mechanical trackers measure values through mechanical links placed in chains: they connect a known point to the tracked object and measuring joint angles is possible to retrieve position and orientation by means of Direct Kinematics. They are high-accurate and low-latency systems, but due to their cumbersomeness they are not so comfortable in the use and can limit the user's range of motion.

#### Magnetic tracking

Magnetic tracking relies on measuring the intensity of the magnetic field in various directions. There is typically a base station that generates AC, DC, or pulsed DC excitation. As the distance between the measurement point and base station increases, the strength of the magnetic field decreases. If the measurement point is rotated, the distribution of the magnetic field is changed across the various axes, allowing determination of the orientation as well.
Magnetic tracking accuracy can be good in controlled environments but it is subject to interference from conductive materials near emitter or sensor, from magnetic fields generated by other electronic devices and from ferromagnetic materials in the tracking volume.

### Acoustic tracking

Acoustic trackers use an emitter/receiver configuration. Usually they rely on microphones to the get time-of-flight measurements from ultrasonic sound sources. They can measure the time-of-travel of ultrasonic pulses to arrive to a set of receiver, or they can compare the phases of emitted acoustic waves with the phase of a reference wave.
When multiple emitter/receiver are present in a known position on a rigid object, the time difference between them can provide data about the orientation of that rigid object with respect to the transmitters.
The acoustic trackers are low-cost and lightweight systems, but they don't well perform in nosing and echoing environments or if there are obstruction between transmitters and receivers.

### Inertial tracking

Inertial tracking uses accelerometers and gyroscopes. Accelerometers measure linear acceleration. Since the derivative of position with respect to time is velocity and the derivative of velocity is acceleration, the output of the accelerometer can be integrated to find the velocity and then integrated again to find the position relative to some initial point.
Gyroscopes measure angular velocity: they are solid-state components based on MEMS technology, but have the same operating principle as the mechanical gyros. The angular velocity can be integrated to determine the angular position relative to some initial point.
Inertial trackers are very low-cost and provide high update rates and low-latency. On the other side, the integration and double-integration lead to significant drift, especially as it relates to position information and thus it is hard to rely on inertial tracking to determine position.
The problem of Inertial Navigation is presented more in-depth in the Section 2.6.

### Optical tracking

Optical trackers capture images by means of video cameras and extract information relying on vision algorithms.
Most optical trackers are marker trackers. Markers can be:

- **active** (typically IR lights that periodically flash): by synchronizing the time that they are on with the camera, it is easier to block out other IR lights in the tracking area. Alternatively it's possible to define a mask of permanent IR reflection point inside the scene.
- **passive** (typically retro-reflectors): they reflect the IR light back towards the light source. In this case the camera is equipped with an IR flash which is reflected off the markers.

Depending upon the position of the markers and the sensor, this type of tracker can be divided into two categories:

- inside-out systems in which the markers are in fixed places in the environment and the sensors are put on the target object;
- outside-in systems in which the markers are placed on the target object and the sensors are fixed.

Another variation of optical tracking is tracking **visible markers** which consist in predefined pattern images: the camera can recognize the existence of this marker and if multiple markers are placed in known positions, the position and orientation can be calculated.
The key is to be able to create markers that would be quickly identified by the cameras in a computationally-efficient manner as well as be able to create a sufficiently large number of distinct markers.
If the geometry of the object is known, it is possible to perform **markerless tracking** which continuously searches and compares the image with the known 3D model [48]. The markeless approach can be used also to to track complex human motions, by exploiting multiple camera views, as described in [28]

Since different objects can be tracked at the same time if they have different marker arrangements, this type of tracker is easy scalable. It also provides low-latency and high sample rate. On the other side it suffers from line of sight problem which can be partially mitigated by using multiple sensors and markers. Optical trackers require also a surrounding environment with low ambient infrared radiation in order to not experience performance losses.

### 1.2.2 Sensor Fusion

Sensor fusion is a process by which data from several different sensors are fused to compute something more than could be determined by any one sensor alone. The resulting information has less uncertainty than would be possible when these sources were used individually. Uncertainty reduction means more accuracy, completeness and information dependability. Thus, sensor fusion provides a more robust description of an environment or a process of interest.
The data sources for a fusion process are not specified to originate from identical sensors. It's possible to distinguish between

- *direct* fusion: it's the fusion of sensor data from a set of heterogeneous or homogeneous sensors, soft sensors, and history values of sensor data
- *indirect* fusion: uses information sources like a priori knowledge about the environment and human input

For example, fusing information coming from different ToF cameras and defining a 3D body model, it's possible to track the entire human body as described in [30]
A well-known fusion technique is the **Kalman filter** [26]: it's an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown and not directly measurable variables that tend to be more precise than those based on a single measurement alone [17]. It is recursive optimal estimator so that new measurements can be processed as they arrive and it doesn't need to store all previous measurements and reprocess all data each time step.

It consists of a set of mathematical equations that provides an efficient computational means to estimate the state of a process, in a way that minimizes the mean of the squared error.

## 1.3 Applications of Virtual Reality

Even if one of the wide adopted application for Virtual Reality is the gaming and the entertainment, Virtual Reality is not an end in itself and in the literature are present many other kind of possible applications some of which are more challenging or unusual than others.
For example Virtual Reality it's adopted by the **military** [34] for training purposes. This is particularly useful for training soldiers and to simulate hazardous situations or other dangerous settings where they have to learn how to react in an appropriate manner. The uses of VR in this field include flight or battlefield simulation medic training [46] and virtual boot camp. It has proven to be safer and less costly than traditional training methods.

Figure 1.5: VR for parachute simulation

**Healtcare** is one of the biggest adopters of virtual reality which encompasses surgery simulation [58], phobia treatment [59] and skills training [37]. A popular use of this technology is in robotic surgery [29]. This is where surgery is performed by means of a robotic device – controlled by a human surgeon, which reduces time and risk of complications. Virtual reality has also been used in the field of remote *telesurgery* [46] (Fig. 1.6) in where the operation is performed by the surgeon at a separate location to the patient. Since the surgeon needs to be able to gauge the amount of pressure to use when performing a delicate procedure, one of the key feature of this system is the force feedback [54]. Another technology deeply used in healthcare is

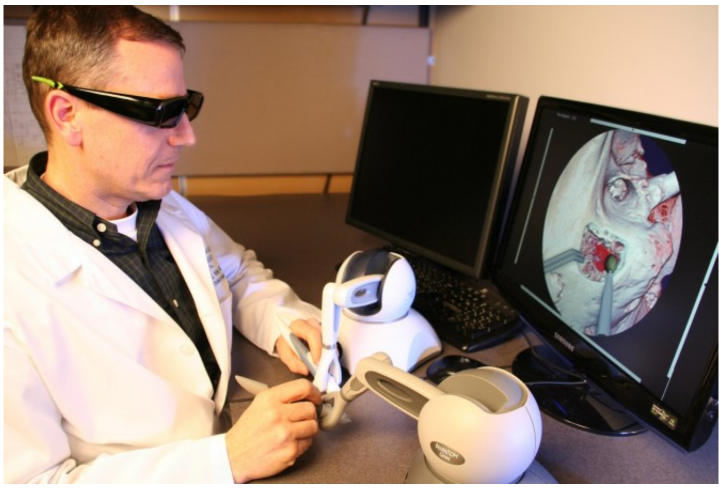


Figure 1.6: VR for telesurgery

Augmented reality that enables to project computer generated images onto the part of the body to be treated or to combine them with scanned real time images [11].
A new trend in Virtual Reality is certainly the use of such technology in the field of cultural heritage. **Virtual Heritage** [5] is one of the computer-based interactive technologies in virtual reality where it creates visual representation of monument, artifacts, building and culture to deliver openly to global audiences [43].
The aim of virtual heritage [41] is to restore ancient cultures as a real environment that user can immerse and understand a culture. By creating ancient culture simulation, virtual heritage applications become as a link between the user of the ancient culture and the modern user. The interaction between them is one way, where the virtual heritage applications are dead and user can learn about the culture by interacting with their environment.

Currently, virtual heritage has become increasingly important in the preservation, protection, and collection of cultural and natural history [6]. The world's resources of historical in many countries are being lost and destroyed [21]. With the establishing of new technology, Virtual Reality can be used as a solution for solving problematic issues concerning cultural heritage assets.

Figure 1.7: VR for cultural heritage

## 1.4 The XVR framework

XVR [51] is a flexible, general-purpose framework for the rapid development of Virtual Reality applications. The need of constructs and commands targeted to VR, has lead to the design of a dedicated scripting language which provides the developer the possibility to deal with 3D animation, positional sounds effect, audio and video streaming and user interaction.
XVR is actually divided into two main modules:

- the ActiveX Control module, which hosts the very basic components of the technology such as versioning check and plug-in interfaces;

- the XVR Virtual Machine (VM) module which contains the core of the technology such as 3d Graphics engine, the Multimedia engine and all the software modules managing the built-in XVR features.

The XVR-VM contains a set of byte-code instructions, a set of registers, a stack and an area for storing methods. The XVR Scripting Language (S3D) allows specifying the behaviour of the application, providing the basic language functionalities and the VR-related methods, available as functions or classes. The script is then compiled in a byte-code which is processed and executed by the XVR-VM.
In general an XVR application can be represented as a main loop which integrates several loops, each one running at its own frequency, such as graphics, physics, networking, tracking, and even haptics, at least for the hi-level control loop.

### 1.4.1 The S3D scripting language

An XVR program is always based on a set of 7 fundamental callbacks which get automatically executed upon.
These predefined functions constitute the basis of any project:

- **OnDownload()** is performed at the very beginning and triggers the download of the data files needed from the application.

- **OnInit()** function is the place where to put the initialization code for the app. All the commands are executed sequentially. All the other functions are not active until OnInit() completes its execution. It receives as a parameter the string defined inside the XVR related HTML code under the "UserParam" section. It gets called as soon as OnDownload() is finished.

- **OnFrame()** is the place for functions and methods that produce graphics output. This is the only function where the graphics context is visible. Placing graphics command outside this function would produce no results. It gets called at a frequency specified by SetFrameRate(). By default the frame rate is set to 100 Hz.

- **OnTimer()** runs independently (i.e at a different rate) by OnFrame() and it's where to put commands that must be independent from the rendering task. As the timer is hi-res, it is possible to setup timer parameters with SetTimeStep() so that this function can be called up to 1k times per second. The default time step is 10 ms, therefore OnTimer gets called by default at a frequency of 100 hz.

- **OnEvent()** is independent from both OnTimer() and OnFrame(). It gets called whenever the application receives an event message. Event messages can be external (i.e. Windows messages) or internal (i.e. generated anywhere in the XVR program). Events and messages are supported in XVR because they add flexibility to the programming environment for task where fixed timers are not the best option. If the application does not need them, this function can be ignored.

- **DownloadReady()** is called whenever a download triggered by the FileDownload() function is completed and receives as a parameter the ID (returned by FileDownload) of the downloaded file. This function can be used to asynchronously download resources not needed since the beginning of the XVR program.

- **OnExit()** is called when the application quits or when the user close the page the application is in. It is the right place to perform nice exits of external modules.

In addition to these basic functions, XVR offers several predefined classes, functions and data structures for different purposes. For example the *CVm-Camera* class provides functions for managing the scene and the camera properties. *CVmMesh* class, and *CVmObject* class, allow for managing 3D models by handling the geometric properties of the model and by dealing with reference systems and geometrical transformations. The appearance of the objects is managed by the *CVmMaterial* and *CVmTexture* classes.

# 2. Background and previous work

*If I have seen further, it is by standing on the shoulders of giants.*

Isaac Newton

In the literature, the problem of interaction in VR is widely addressed. Several different techniques, that make use of a variety of devices, have been also proposed for giving manipulation capabilities to the user inside the Virtual Environment.
In this chapter a survey of existing interaction techniques is presented, along with a general overview about the tracking devices that can be exploited for manipulation purposes.

## 2.1 Object positioning

A key manipulation task often analyzed in the literature regards the object positioning [27]. As described in [12], the object positioning task consists in at least two interaction metaphors:

- **grabbing** (or selection) that refers to the initial phase of the task, when the user specifies which object to grab and eventually denotes the center of rotation for the manipulation phase; by moving a cursor, typically attached to the user's hand, until it is within the selection region of the object, is possible to chose that object. Once chosen, the object can be actual selected using some pre-defined signal such as gesture, button press or voice command [38].

- **manipulation** that refers to the actual task, when the user moves the selected object within the environment, specifying both position and orientation. Manipulation tasks consists of:
  - *translation*: selected objects can move in a 1:1 correspondence with the user's hand or an amplification factor can be applied to the motion of the object to allow a greater range [38]
  - *rotation*: selected objects rotate of a rotation factor according to the orientation of the user's hand. In hand-centered manipulation the center of rotation is identified with the point in which the object is grabbed, but in some case it's desirable to have some remote center of rotation in order to allow the user to have a good perspective as he set the orientation of the object.

In [38], also the **scaling** operation is identified as fundamental form on interaction in a virtual world: according to Mine, scaling can be used both in the interactive construction of a virtual world to get components at their correct relative scale, and in the exploration of an environment to allows a user to view some small detail by scaling up the selected object.
As in the case of rotation, also the scaling operation requires the definition of key parameters:

- *center of scaling*: it's the point which all points of the object move toward when scaling down and all the points move away from when scaling up. In hand-centered scaling, the center of scaling is identified with the location of the hand, while in object-centered scaling is defined to be the center of the selected object.

- *scaling factor*: it can be hand-specified when movement of the hand determines it; in this case the scaling is uniform on all the axis. Alternatively can be specified providing handles (grabs) which the user can interact with to control the scaling of the selected object; this method is particularly well suited for the implementation of non-uniform scaling.

Different interaction metaphors are required when the object the user wants to manipulate is not in his arm range. In [12], Bowman and Hodges evaluate the techniques for grabbing and manipulating remote objects in immersive virtual environments. They split in two categories the techniques which attempt to solve this problem:

- **arm-extension** techniques in which the user's virtual arm is made to grow to the desired length, so that object manipulation can take place with the hand. These techniques make object manipulation simple since the user moves and rotates the object with natural hand and arm motions, but the grabbing is more difficult because the virtual hand must be positioned within the object, which may be small or distant. An example of arm-extension technique is the 'go-go' technique.
- **ray-casting** techniques which make use of virtual light ray to grab an object, with the ray's direction specified by the user's hand. The use of the light ray makes the grabbing task easy since the user is only required to point to the desired object, but manipulation is more difficult due to the fact that is not hand-centered. An example of ray-casting technique is the HOMER (Hand-centered Object Manipulation Extending Ray-casting) technique in which when the user grabs the object with the light ray, the virtual hand moves to the object position and the object is attached to the hand; when the object is dropped, the hand returns to its natural position.

In [18], Foxlin and Harrington investigate wearable solutions for the orientation and position of users' heads and hands tracking with respect to a world coordinate frame. They present a self-referenced head and hand tracker for wearable computers and portable VR which allows to control view parameters for Head Mounted Displays and manual interactions with the virtual world.
The technique they propose differs from the other because it's essentially

'sourceless': the system is based on the idea of combining a sourceless head orientation tracker with a head-worn tracking device that tracks a hand-mounted 3D beacon relative to the head.

## 2.2 VR approaches for Immersive Design

According to [57], the current application of Virtual Reality systems in the design process is limited mostly to design review. Weidlich et al., identify the reason for this limitation in the different data formats used for CAD and VR visualization, and propose the use of VR systems also during the early design stage in order to drastically reduce potential source of error especially during the outline and detailing phases.
The ways in which researchers have implemented the immersive modeling are:

- linking a VR system and a CAD core. An example is ARCADE [50], a cooperative, directly manipulative 3d modeling system which that takes a user-centered approach.
- using voxel models for geometry description. Examples are VCM [31], that allows virtual modeling by adding and removing and uses haptic feedback, and VADE [24], in which the VR environment used is an HDM.

## 2.3 VR for Training

The use of fully immersive Virtual Reality systems for training purposes has been extensively investigated in literature. A number of challenges have been highlighted, ranging from minimizing overall latency, interacting intuitively in the virtual environment, increasing user's perceptual awareness of the virtual world and providing the user with a strong sense of immersion and embodiment [49]. Examples can be found in the mining industry [55], in the aerospace industry [15], in the automotive industry [35], in logistics [10]

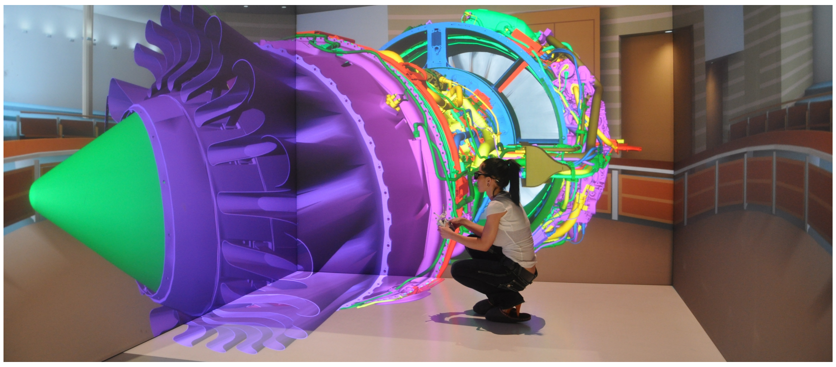


Figure 2.1: VR for Immersive Design

and, in general, in the sector of maintenance [36].
Opportunities provided by Mixed and Augmented Reality address issues similar to those addressed by VR, although with a slightly different perspective. In Mixed Reality, in fact, the real environment is not substituted by a virtual counterpart; rather it is integrated with virtual elements that enhance its information content. Therefore many safety issues effectively tackled by VR are commonly not addressed by MR systems which, in turn, may result more effective whenever the real context is fundamental.
In general, one of the most important consequences of living the training experience in a totally virtual context (as in VR), or keeping the vision on the real context (as in MR), is related to the body self visual perception. In VR, depending on the visualization device, users can still see their body (for instance in a CAVE) or not (if using a HMD; in this case a digital avatar must be shown in order to allow self perception). In MR the real context, including own body, is always present. This has of course an impact in training, especially in tasks where manipulation operations, or other types of direct interaction with the body, take place. Avatar representations, in fact, might not correspond exactly to the dimensions or the current posture of the user and might, although slightly, mislead the self perception and limit the effectiveness of the virtual training.

## 2.4 Techniques and devices for Interaction

An interaction technique [20] is the fusion of input and output, consisting of all hardware and software elements, that provides a way for the user to accomplish a low-level task [9].

- *Input* to computers consists of sensed information about the physical environment.
- *Output* from computers can comprise any emission or modification to the physical environment, such as a display (monitor, HMD, CAVE systems), speakers, or tactile and force feedback devices

There exist a pletora of technologies and devices for interaction in Virtual Environments. Some of them are presented in the following, based on how relevant to the developed system they appear.

### Standard input devices

Any sensed information about physical properties of people, places, or things can serve as input to computer systems. Familiar examples of input devices include the mouse, which senses movement across a surface, the keyboard, which detects a contact closure when the user presses a key and joysticks.
Joysticks are commonly available in two flavors: *displacement or isotonic* joysticks move about a pivot with motion in two or more axes, and *force sensing or isometric* joysticks employ resistive strain gauges which undergo a slight deformation when loaded.

### The Leap Motion Controller

The Leap Motion controller [1] is a small USB peripheral device designed to be placed on a physical desktop, facing upward. The device recognizes the human hands present in its field of view and provides, through its apposite software, information about the number of hands and for each of them some related features such as position and orientation in the space, velocity and so on. The controller has been designed to identify fingers with a precision of 0.01 mm.

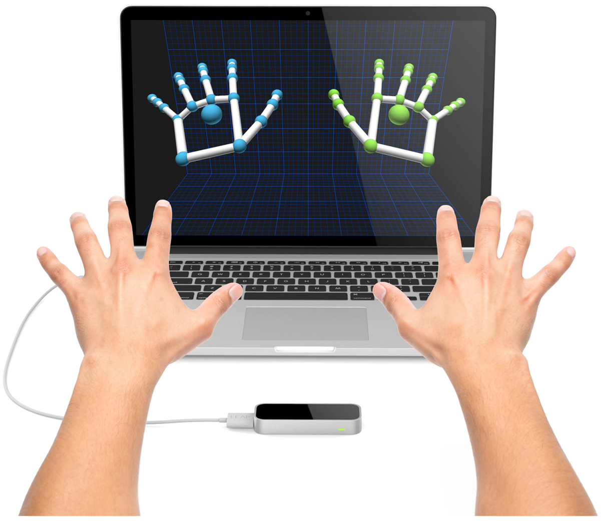


Figure 2.2: The Leap Motion Controller

The heart of the device consists of two cameras and three infrared LEDs. These track infrared light with a wavelength of 850 nanometers, which is outside the visible light spectrum.
The Leap Motion Controller's viewing range is limited to roughly 60 cm above the device. This range is limited by LED light propagation through space, since it becomes much harder to infer hand's position in 3D beyond a certain distance. LED light intensity is ultimately limited by the maximum current that can be drawn over the USB connection.
The device's USB controller reads the sensor data into its own local memory and performs any necessary resolution adjustments. This data is then streamed via USB to the Leap Motion tracking software.

**Software**

The Leap Motion Service is the software that processes the images. After compensating for background objects (such as heads) and ambient environmental lighting, the images are analyzed to reconstruct a 3D representation

of what the device sees.
Next, the tracking layer matches the data to extract tracking information such as fingers and tools. The tracking algorithms interpret the 3D data and infer the positions of occluded objects. Filtering techniques are applied to ensure smooth temporal coherence of the data. The Leap Motion Service then feeds the results – expressed as a series of frames, or snapshots, containing all of the tracking data – into a transport protocol.
Through this protocol, the service communicates with the Leap Motion Control Panel, as well as native and web client libraries, through a local socket connection (TCP for native, WebSocket for web). The client library organizes the data into an object-oriented API structure, manages frame history, and provides helper functions and classes. The API are available for the most common programming languages such as C, C#, Java, Javascript and Python. Plugins are available for the common Graphics Engine such as Unreal and Unity.
The software provides also the recognition of gesture such as circle, swipe and tap.

### Leap for VR

For using the controller in immersive VR applications that make use of HMDs, Leap Motion provides a frame to integrate the Leap controller with the Oculus Rift DK1 and DK2. The frame needs to be mount on the front face of the Oculus Rift and the controller is then placed inside it.

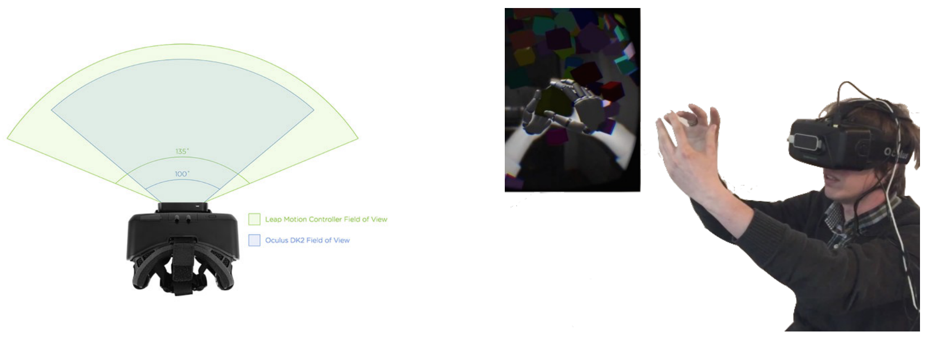


Figure 2.3: Oculus Rift DK2 and Leap Motion controller

This solution is particularly suitable for those applications which require the users interaction with the virtual environment using their own hands. No additional markers or devices are required to obtain the relative position and the orientation of the hands.
As shown in the Fig. 2.3 the Field of View of the Leap controller (135°) is higher than the Oculus Rift's one (100°): this means that the user will be able to freely operate without any blind spot. Moreover, the controller has no impact on DK positional tracking.
However, there are some critical points in the use of the Leap Motion controller for VR applications:

- the not so high accuracy does not allow precise object manipulation.
- it is not possible to have a visualization of the real hands, but only a virtual representation of them.

### Sensing gloves

A sensing glove is an input device for human–computer interaction worn like a glove. Various sensor technologies are used to capture physical data such as bending of fingers and often a motion tracker, such as a magnetic tracking device or inertial tracking device, is attached to capture the global position and orientation data of the glove.

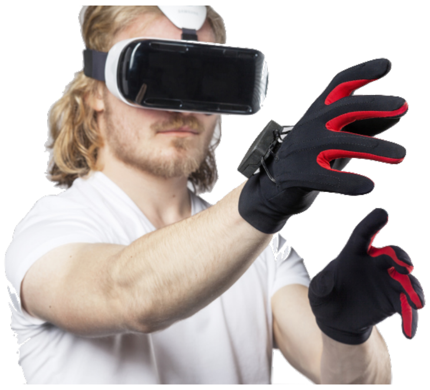


Figure 2.4: A user wearing an HMD and a pair of sensing gloves

## Time-of-Flight cameras

Time-of-Flight cameras are sensing systems able to capture RGB images along with per-pixel depth information.
They provide 3D imaging using a CMOS pixel array and an active modulated light source used to illuminate the scene. They project a known infrared pattern onto the scene and determines depth based on the pattern's deformation as captured by an infrared CMOS imager. By observing the reflected light and measuring the phase shift between the illumination and the reflection, 3D ToF cameras can tell the distance between from the sensor to objects in every single pixel.

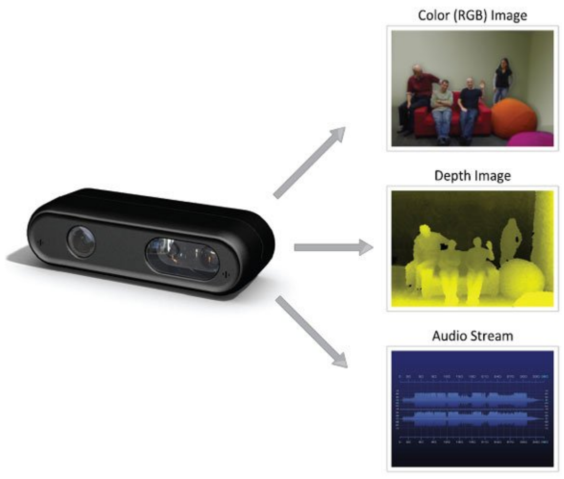


Figure 2.5: PrimeSense RGB camera with depth sensor

The compactness, ease of use, accuracy and high frame-rate make ToF cameras a suitable solution for a wide range of Virtual Reality applications [19] and in particular for tracking based on colored markers.

## Soli: a radar based wearable

Soli is a project by Google ATAP (Advanced Technology and Products) unveiled during the Google's I/O 2015 conference. Project Soli uses radar (RAdio Detection And Ranging) to detect micro-movements in hands and fingers. It transmits radio waves and picks up up reflected responses from any target it hits.

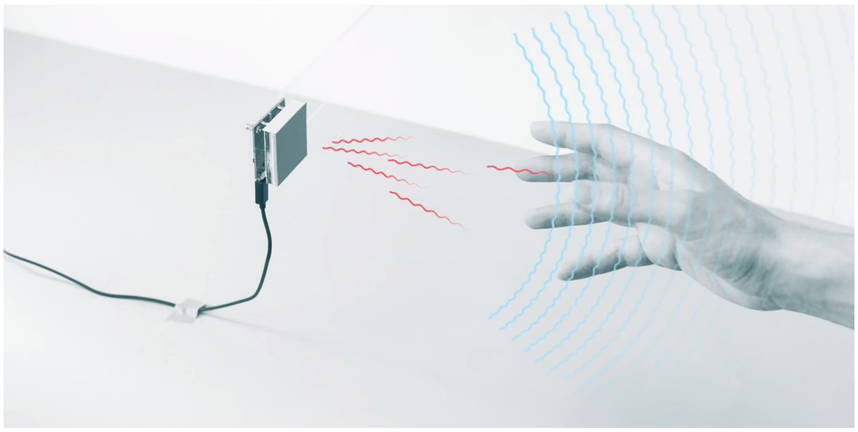


Figure 2.6: How Soli works

The system uses broad beam radar to measure doppler image, IQ and spectrogram. The chip recognizes movement, velocity and distance and can be programmed to change the input based on that distance.

## 2.5 Tracking Systems for natural Interaction

By using the systems above described it's possible to track the position and the orientation of any relevant part of the body of an user, and by defining adequate metaphors it's possible to provide users with the ability of naturally interact with the Virtual Environment [25].
The naturalness of interaction is a much discussed topic in the literature [33]: a system that allows the user to move in a natural way, allows also to reduce the time for learning and to habit to the system.
The process of recording the movement of objects or people is called **motion capture**. Often the purpose of motion capture is to record only the movements of the actor, not his or her visual appearance. The recorded data is thus mapped to a 3D model so that the model performs the same actions as the actor. This approach is deeply used in filmmaking and video game

development in order to recording actions of human actors, and using that information to animate digital character models in 2D or 3D computer animation.
In the Thesis project the motion capture has concerned only the user's hands: in particular we have been interested in the movements of the index finger with respect to the rest of the hand when the user grabs an object.

### 2.5.1 Color-Based Tracking

Using **colored markers** and a depth camera is possible to track the position of the markers with respect to the whole environment. This tracking system is particularly suitable when there is the need to distinguish two markers without any other information. For example, to perform positional hand tracking, users can simply wear color thimbles on their fingertip [53] as shown in Fig. 2.7, or wear a Color Glove on their hand [56] as shown in Fig. 2.8. The drawback in the use of this methodology is the limited FOV of the depth camera and the not very fast sample rate.

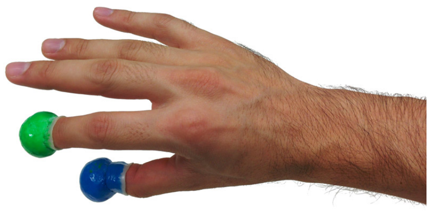


Figure 2.7: Finger tracking with color markers

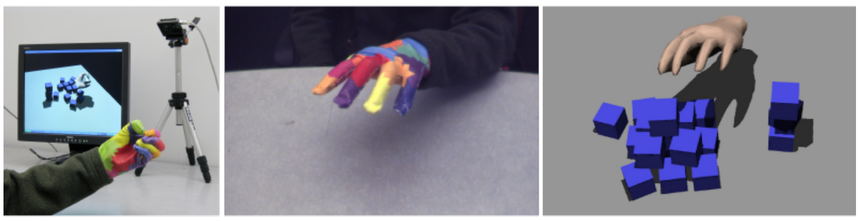


Figure 2.8: Real-Time Hand-Tracking with a Color Glove

### 2.5.2 Marker-based Tracking

Another way to track user's parts of the body is the use of motion capture system such as the one offers by OptiTrack (Fig. 2.9). The system consists of:

- infrared reflecting markers
- cameras equipped with a CMOS sensor capable of providing VGA images at 100 fps
- a software able to track up to 2000 markers in 6 DoF.

The X-CAVE system described in the Section 1.3.1 uses 7 *OptiTrack Flex 3* cameras for performing positional tracking.

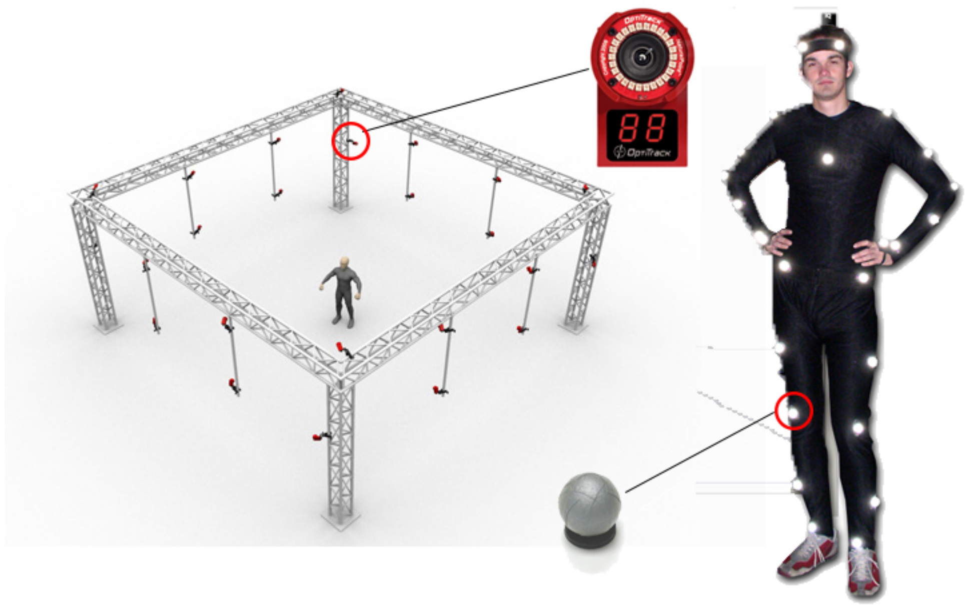


Figure 2.9: An OptiTrack system deployment

The OptiTrack system provides hardware-based synchronization between cameras; this allows them to expose frames at the exact same time.
Once cameras acquire a new frame, the OptiTrack software handles all additional processing required to track and combine the marker data coming

from all the cameras. Since there is also no guarantee that markers will be delivered in the same order from one frame to the next, additional algorithms are required to ensure markers sorting.

### 2.5.3 Smart Laser Scanner

Smart Laser Scanner [4] is a system capable of acquiring three dimensional coordinates in real time without the need of any image processing at all. Essentially, it is a smart rangefinder scanner that instead of continuously scanning over the full field of view, restricts its scanning are to a very narrow window precisely the size of the target.
Tracking of multiple targets is also possible without replicating any part of the system and without the need to wear special gloves nor markers.
Another interesting characteristic of the proposed 3D laser-based locator, is that it also can be used as an output device: indeed, the laser scanner can be used to write information back to the user, by projecting alphanumeric data onto any available surface, like the palm of the hand. This has been successfully demonstrated, without having to stop the tracking.

## 2.6 Inertial navigation with IMUs

Until recently the use of inertial sensors in domains such as human motion capture has been prohibited by their weight and size. Thanks to the performance improvements of small and lightweight Micro Electro-Mechanical Systems (MEMS) inertial sensors have made the application of inertial techniques to such problems possible. Nowadays, MEMS technology is of particular interest because it offers small and lightweight inertial sensors at low cost.
Inertial navigation [60] is a self-contained navigation technique in which measurements provided by accelerometers and gyroscopes are used to track the position and orientation of an object relative to a known starting point, orientation and velocity. The method used for the estimation of the measurement is known as dead-reckoning: it is the process of calculating one's current po-

sition by using a previously determined position, or fix, and advancing that position based upon known or estimated speeds over elapsed time. This is also applicable in the case of angular measurements, as with gyroscopes, in order to obtain the orientation.

### 2.6.1 MEMS Gyroscopes

MEMS gyroscopes make use of the Coriolis effect, which states that in a frame of reference rotating at angular velocity $\omega$, a mass $m$ moving with velocity $v$ experiences a force:

$$F_c = -2m(\omega \times v)$$

MEMS gyroscopes contain vibrating elements to measure the Coriolis effect; the simplest vibrating element geometry consists of a single mass which is driven to vibrate along a drive axis. When the gyroscope is rotated a secondary vibration is induced along the perpendicular sense axis due to the Coriolis force. The angular velocity can be calculated by measuring this secondary rotation.

The bias of a rate gyro is the average output from the gyroscope when it is not undergoing any rotation. The constant bias error of a rate gyro can be estimated by taking a long term average of the gyro's output whilst it is not undergoing any rotation. Once the bias is known it is trivial to compensate for it by simply subtracting the bias from the output.

### 2.6.2 MEMS Accelerometers

Accelerometers are electromechanical devices that measures acceleration forces either static, such as force of gravity, and dynamic, namely caused by moving the device.

There exist different ways to make an accelerometer using MEMS techniques. Some accelerometers use the piezoelectric effect which causes a voltage to be generated, but the most common way to do it's by sensing changes in capacitance.

$$C = \epsilon_r \epsilon_0 \frac{A}{d}$$

where

- $C$ is the capacitance, expressed in Farads,
- $A$ is the area of overlap of the two plates, expressed in square meter,
- $\epsilon_r$ is the relative static permittivity of the material between the plates,
- $\epsilon_0$ is the electric constant,
- $d$ is the separation between the plates, expressed in meters.

Typical MEMS accelerometers consist of a mass suspended by spring. The displacement $x$ of the mass is measured using the capacitance difference $\Delta C$ between a movable plate (on which the mass is attached) and two stationary outer plates. It can be proven that the displacement is approximately proportional to the capacitance difference:

$$x \approx d \frac{\Delta C}{C}$$

According to Hook's law, the spring exhibits a force $F$ proportional to the displacement $x$ ($F = kx$). From Newton's second law ($F = ma$) results that the acceleration is proportional to the mass displacement

$$a = \frac{k}{m} x$$

If the fixed plates are driven by square waves with voltage amplitude $V_0$, it can be proven [7] that the acceleration is also proportional to the voltage output.

### 2.6.3 Inertial Measurement Unit

An Inertial Measurement Unit (IMU) is an electronic device which consists of a 3-axis gyroscope, measuring angular velocity, and a 3-axis accelerometer, measuring linear accelerations. By processing signals from these devices it is possible to track the position and orientation of an object.

An IMU [40] works by detecting the current rate of acceleration using one or more accelerometers, and detects changes in rotational attributes like pitch, roll and yaw using one or more gyroscopes.
Recently, more and more manufacturers also include three magnetometers in IMUs. The magnetometers measure the strength and direction of the local magnetic field. Even if, due to nearby magnetic objects, they are affected by local disturbances in the earth's magnetic field, their data can be fused with the gyroscope data to allow better performance for dynamic orientation calculation in attitude and heading reference systems which base on IMUs [22].
Using multiple IMUs it's possible to perform extensive motion capture of some parts of the body or the entire body. With this approach the motion capture system remains self contained and it does not rely on any external infrastructure.

# 3. The proposed system

*Reality is not always probable, or likely.*

Jorge Luis Borges

After having investigated the technologies and methodologies for interaction, the requirements for the new device have been defined: the proposed system should allow natural interaction to the user, providing the most suitable manipulation metaphors to operate in an accurate and intuitive way in a Virtual Environment. The system would have to be also as portable as possible, unobtrusive for the user and with short setup and learning time.
In this chapter are presented the scenario in which the project fits, the existing system used for visualization (i.e. the devices used to present the Virtual Environment to the user), along with the design choices to satisfy the interaction requirements and the developed solutions that result from the implementation of that choices.

## 3.1 SSSA Mixed Reality system

The visualization system used in this project has been presented in [53]. Before the development of the presented Thesis project, this system was also used for providing interaction capabilities to the user; the interaction system allowed only the translation of the virtual objects present in the scene without any possibility of rotating them.

Regarding the visualization system, what made the approach novel, was that the user's hands were video captured in 3D, reconstructed in real-time and graphically embedded in a synthetic Virtual Environment rendered in an HMD worn by the user. The introduction of the photo-realistic capture of user's hands in a coherently rendered virtual scenario induces in the user a strong feeling of embodiment without the need of a virtual avatar as a proxy. Also the user's ability to grasp and manipulate virtual objects using their own hands provides an intuitive user interaction experience and improves the user's self perception and user's perception of the environment. We used the same approach in the current work, so users are able to use their hands manipulating virtual objects in the scene and are able to navigate in the scene using their own body movements.

### 3.1.1 Hardware configuration

The setup used for the optical tracking consists of 8 OptiTrack Flex:V100 cameras,each one equipped with a CMOS sensor capable of providing VGA images at 100 fps. The tracking software is OptiTrack Tracking Tools version 2.5.3. This system has been used to track the user head position, while the orientation of the user head is estimated by means of the IMU built-in in the Oculus DK2; the inertial sensors used in the HMD is reported as a custom 9-axis tracker (gyroscope, accelerometer and magnetometer) with a 1000Hz update rate and 2ms latency based on the Adjacent Reality Tracker. It can provide 3DOF rotational tracking with yaw drift correction.
The RGBD camera is a Primesense Carmine 1.09 short range 3D camera. The camera has a FOV of 57.5x45 degrees and returns a depth map of (approx.) 640 x 480 depth samples at the rate of 30Hz as well as a RGB map of the same resolution and frame rate. Being the short-range version, it can see objects as close as 35 centimeters.
The software modules of the system run currently on two workstations: the first one is allocated to the optical tracking software, and it is equipped with a Core i7 3770 CPU (4 core with HT @3.4Ghz), 24GB of Ram and a FirePro V7900 GPU. The second workstation is used for the real-time

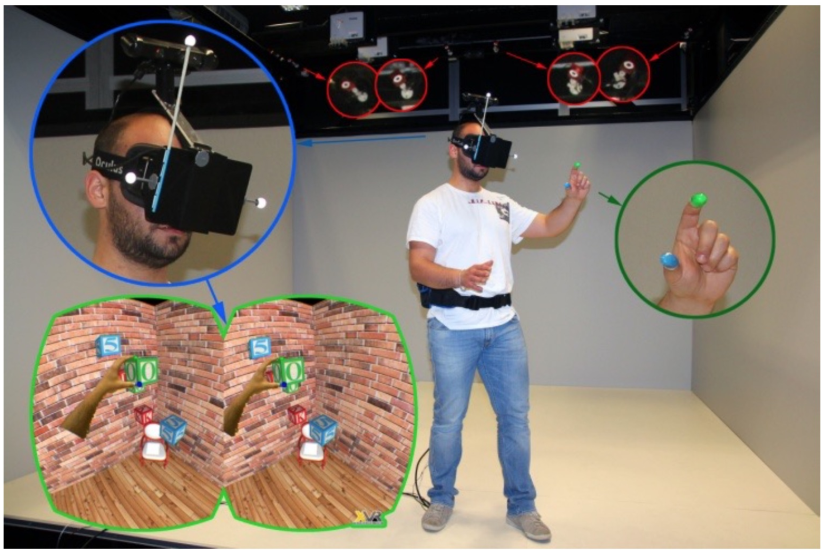


Figure 3.1: The system physical layout: OptiTrack cameras in red, finger thimbles in dark green, HMD with markers in blue and an example of what user sees in light green.

rendering, the Primesense data handling and the general management of the VR application, and it is equipped with a Core i7 960 (4 core with HT@3.2Ghz), 24GB of Ram and a Nvidia 680GTX GPU with 1,5GB of memory. Both systems are running Windows 7 64bit.
The graphical workstation is connected to the HMD by means of a 10 meters video cable, a 10meters USB active extension cableand a power supply cable for the HMD As shown in the Fig. 3.1, the user wears an HMD and is free to walk around the scene as well as to use his own hands to manipulate virtual objects in the scene by means of a RGBD camera mounted on top of the HMD. This allows the system to get in real-time a textured geometric mesh of the visible parts of the hands and body (as seen from her/his own perspective) that can be rendered like any other polygonal model in the scene.
The Fig. 3.2 shows the visualization system. It is composed by the following items:

- an optical marker, used for positional head tracking
- an Oculus Rift DK2 HMD connected to an Intel workstation for visual

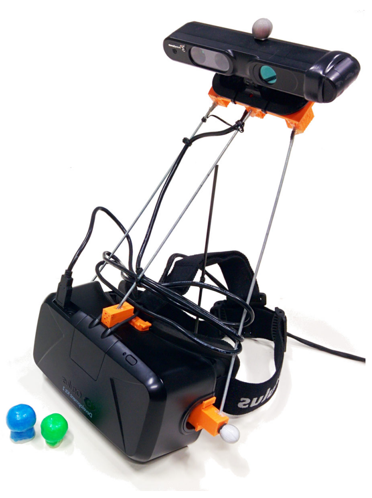


Figure 3.2: The visualization system: Oculus Rift DK2 and PrimeSense Carmine 1.09

feedback

- a 3D camera mounted on top of the HMD support, and integral to it, which is used both for the real-time 3D capturing of the user hands correctly co-located in the virtual environment (and all the other parts of the body framed by the camera) and for the tracking of the hands.

### 3.1.2 Finger tracking and interaction metaphors

In this scenario, fingers tracking was performed by taking advantage only of the RGBD data in order to detect the colored thimbles. Simple color filtering was then used in order to identify which pixels of the RGB image match the marker colors. The algorithm also used the depth map data in order to efficiently pre-cull away those pixels that are too far to be part of the user hands.
A simple collision detection algorithm was applied to this data in order to enable grabbing and dragging interactive virtual objects. The implemented interaction was therefore almost completely natural, with the only added metaphor simulating a simplified grabbing. In fact, no actual physically-based contact is retrieved: when the two fingers (whose position was visually

highlighted by making two spheres, with the same colors, appear on top of them - fingers are anyway visible as they are acquired in 3D and streamed by the 3D camera) touched each other within or in close proximity of an interactive object, the two spheres became one single red sphere, meaning that the object could be grasped and moved (the user could interact with just a single object at a time).

### 3.1.3 Existing limits and motivations for a new device

The system presented in [53] allows for augmenting the virtual space with the real-time 3D rendering of the hands and body of the users, giving them the perception of being physically present in the virtual environment. The authors proposed also an interaction approach based on the tracking of colored thimbles, which allows the user to manipulate virtual objects using their own hands. However, some weaknesses to this approach can be found:

- *The interaction system allows only the translation of the virtual objects present in the scene.* Since also the rotation and scaling are deemed fundamental in object positioning tasks, the development of a new device capable of providing a more enhanced interaction metaphor has been necessary.

- *For each finger, the previous interaction system requires a thimble of a different color.* Another requirement for the new device has been to reduce the number of needed marker required for the tracking purpose.

- *Using the previous system in situations that concern small mechanical elements, the accuracy of the manipulation is not so high, because of the intrinsic inaccuracy of the RGBD camera.* The Thesis project has addressed also this type of problem, trying to improve this this aspect.

## 3.2 Device presentation

The developed device is a thimble-like interface that can be used to recognize in real-time the human hand's orientation and infer a simplified but effective model of the relative hand's motion and gesture. It's provided with:

- markers for positional tracking (both color-based and retro-reflective)
- IMUs for hand and finger orientation recognition
- force sensor for pinches and grabs detection
- vibration motor for simple haptic feedback

### 3.2.1 The ODROID-W module

The module used for the development is the ODROID-W, a miniature computing module fully compatible with all software available for the Raspberry-Pi.

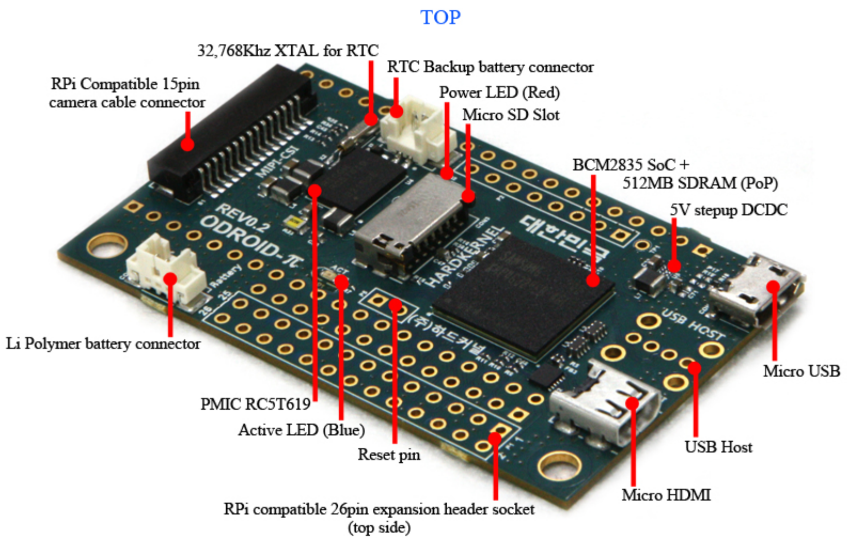

TOP
32,768Khz XTAL for RTC
RTC Backup battery connector
RPi Compatible 15pin camera cable connector
Power LED (Red)
Micro SD Slot
BCM2835 SoC + 512MB SDRAM (PoP)
5V stepup DCDC
Li Polymer battery connector
Micro USB
PMIC RC5T619
Active LED (Blue)
Reset pin
USB Host
Micro HDMI
RPi compatible 26pin expansion header socket (top side)

Figure 3.3: ODROID-W

This module integrates a BCM2835 ARM11 700Mhz processor produced by Broadcom, 512MB SDRAM and like other Raspberry Pi on the board are available:

- 32 GPIOs
- 2 I2C buses
- 1 SPI bus

The ODROID-W provides also a Ricoh RC5T619 Power Management Integrated Circuit (PMIC) that includes RTC, ADC, UPS and Battery gauge. Since these peripherals are not present in the standard Raspberry-Pi boards, the needed drivers are offered by a patch for the official distribution of the OS Raspbian. The last official distribution released is based on the RPi kernel 3.12.32.

### 3.2.2 The Inertial Measurement Unit: MPU-9150

The MPU-9150 is a System in Package (SiP) that combines two chips: the MPU-6050, which contains a 3-axis gyroscope, 3-axis accelerometer and an onboard Digital Motion Processor capable of processing complex MotionFusion algorithms; and the AK8975, a 3-axis digital compass. The part's integrated 6-axis MotionFusion algorithms access all internal sensors to gather a full set of sensor data.

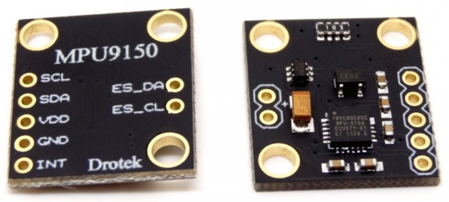


Figure 3.4: MPU-9150

The MPU-9150 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs, three 16-bit ADCs for digitizing the accelerometer outputs and three 13-bit ADCs for digitizing the magnetometer outputs.

### Communication Interface

The MPU-9150 communicates to a system processor using an I2C serial interface. The MPU-9150 always acts as a slave when communicating to the system processor.
The MPU-9150 has an auxiliary I2C bus for communicating to off-chip sensors. This bus has two operating modes:

- I2C Master Mode: The MPU-9150 acts as a master to any external sensors connected to the auxiliary I2C bus
- Pass-Through Mode: The MPU-9150 directly connects the primary and auxiliary I2C buses together, allowing the system processor to directly communicate with any external sensors.

### Gyroscope Features

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full- scale range of ±250°, ±500°, ±1000°, and ±2000°/sec
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Digitally-programmable low-pass filter

### Accelerometer Features

- Digital-output 3-Axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g
- Orientation and tap detection
- User-programmable interrupts

**Magnetometer Features**

- 3-axis silicon monolithic Hall-effect magnetic sensor with magnetic concentrator.
- Full scale measurement range is ±1200 $\mu$T

### 3.2.3 The force sensor

The force sensor (Fig. 3.5) is a force-sensing resistor with a round, 0.4 mm diameter, sensing area. The FSR will vary its resistance depending on how much pressure is being applied to the sensing area. The harder the force, the lower the resistance. The harder the force, the lower the resistance. When no pressure is being applied to the FSR, its resistance will be larger than 1M$\Omega$, with full pressure applied the resistance will be 2.5k$\Omega$. This FSR can sense applied force anywhere in the range 0.1N - 10N

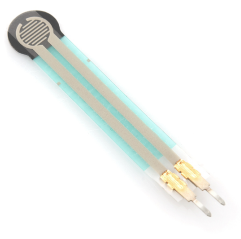


Figure 3.5: Force sensor

### 3.2.4 Simple Haptic Feedback

The feedback is provided by means of a 3V vibration motor (Fig.3.6). The simplest and most popular type of vibration motor is known as an ERM (Eccentric Rotating Mass). It's a standard DC motor with an off-center load attached to the shaft. When the motor is powered and rotates, the unbalanced mass creates a centripetal force and if the motor is attached to an object, that force acts on the object and causes it to displace. As DC motors turn very fast, a full circle of displacement can happen over 100 times a second. This fast and repeated displacement is what is felt as vibrations. The motor is driven using a PWM signal in order to change the voltage applied to it according to the haptic feedback that is required.

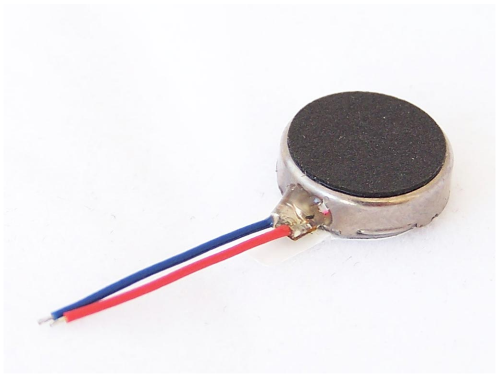


Figure 3.6: Vibration motor

### 3.2.5 Network connectivity

A miniature WiFi module has been used to provide wireless connectivity to the device. The module has been connected to the USB port offered by the ODROID-W. The needed drivers were already available in the modified Raspian distribution. A local network has been thus deployed using an Access-Point 2.4 GHz band connection.

## 3.3 Device design

As described above, the ODROID-W provides 2 I2C buses. Each of the 2 buses can be redirected to several physical pins by changing the mode of these pins:

- I2C-0
  - GPIO_0 (SDA) and GPIO_1 (SCL) in ALT0 mode

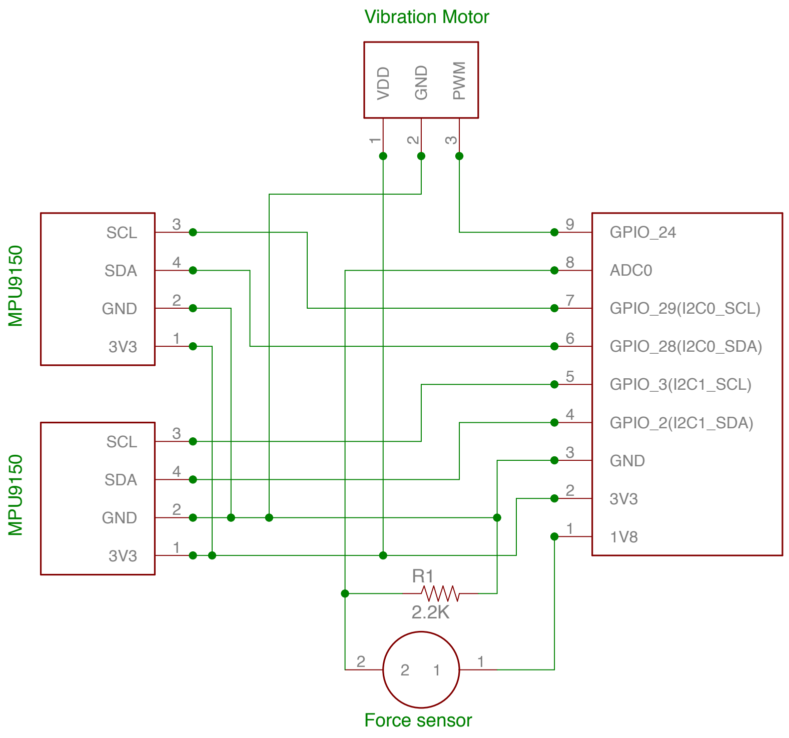


Figure 3.7: High-level device schematics

  - GPIO_28 (SDA) and GPIO_29 (SCL) in ALT0 mode
  - GPIO_44 (SDA) and GPIO_45 (SCL) in ALT1 mode

- I2C-1

  - GPIO_2 (SDA) and GPIO_3 (SCL) in ALT0 mode
  - GPIO_44 (SDA) and GPIO_45 (SCL) in ALT2 mode

Only one of these pairs can be activated at any moment per bus.
The I2C-0 bus is used by default by the GPU to talk to the camera module, while the I2C-1 bus can be used by the user however he wants: the camera is physically connected to GPIO 0/1. These pins are setup as inputs by default and the GPU will change them to ALT0 whenever it needs to talk to the camera and switches them back to INPUT immediately after.

The Ricoh RC5T619 includes among other things 2 ADCs, one of which has been used to read the force sensor. Also the PMIC uses I2C-0 to talk to the CPU.
In the project the buses have been used to connect the 2 IMUs: one IMU is connected to the pins GPIO_2 and GPIO_3 and uses the I2C-1 bus, while the other is connected to the pins GPIO_28 and GPIO_29 and uses the bus I2C-0. This bus is shared between this IMU and the PMIC.
Every time a reading from the ADC is required, the mode of the pins GPIO_0 and GPIO_1 must be set to ALT0 and the mode of the pins GPIO_28 and GPIO_29 must be set to INPUT.
After the reading the mode of the pins GPIO_0 and GPIO_1 must be set to INPUT and the mode of the pins GPIO_28 and GPIO_29 must be set to ALT0 in order to enable the bus for the IMU.

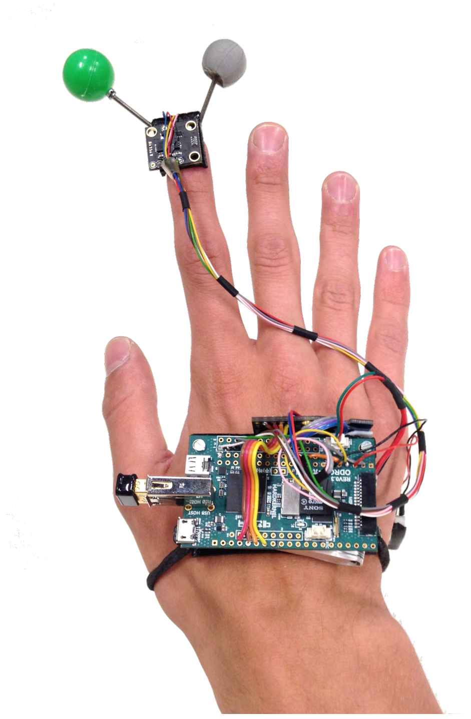


Figure 3.8: Top-view of the device

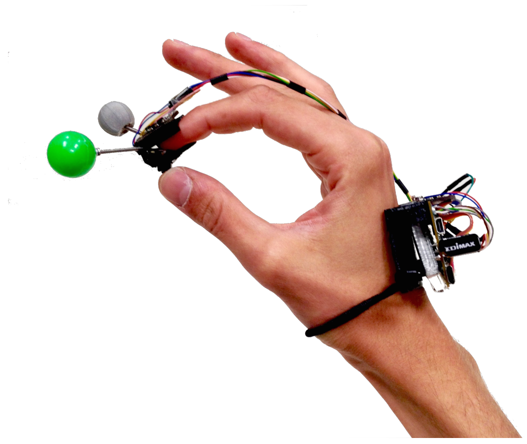


Figure 3.9: Pinching operation

## 3.4 Software development

The ODROID-W module can be easily programmed writing programs or scripts in the most common languages (such as C, C++, Python) and compiling it like any other Unix machine. The module can be simply connected to an external monitor through the mini-HDMI port and the user can input using USB keyboard and mouse. However this would imply physical cable connections and disconnections every time the device needed to be tested or reprogrammed.

In order to achieve a more comfortable development, an SSH server has been setup. In this way the module can be easily programmed over-the-air using any SSH terminal from a machine connected to the same network.

In the project all the embedded applications have been written in C++.

### 3.4.1 Embedded firmware

The aim of the application that runs on the ODROID is to provide the instant orientation of the user's hands and the force exercised between thumb and index fingers. To achieve this result the application needs to read the data coming from the IMU (linear acceleration, angular velocity and compass heading) and fuse them in order to retrieve the orientation. For this purpose

RTIMULib [45] library has been used: RTIMULib is a library that connects a 9-dof, 10-dof or 11-dof IMU to an embedded Linux system and obtain Kalman-filtered quaternion or Euler angle pose data.
In the initialization stage, the application loads the configuration files relative to the IMUs and creates the RTIMU instances from them. In absence of configuration file, the instances are anyway created with the default settings and with no calibration data.
The configuration file for our applications has the following characteristics:

- IMU type: InvenSense MPU-9150
- Fusion type: RTQF
- I2CBus: 1/0
- Compass calibration settings
- Accelerometer calibration settings
- Gyroscope bias data
- Gyroscope and accelerometer sample rate: 50Hz
- Compass sample rate: 25Hz
- Gyroscope and Accelerometer low pass filter: 20Hz for Gyroscope, 21Hz for the accelerometer
- Gyroscope full scale range: +/- 1000 degrees per second
- Accelerometer full scale range: +/- 8g

Inside the loop the application waits for one of the following command via UDP, listening on the port 45002:

- **Start command**: must include the ID of the IMU (0 for the IMU placed on the hand, 1 for the IMU placed on the finger) and the destination port of the message. As soon as the application receives this command, it starts reading from the sensors and sending the data to the destination which is the address from which the Start command arrives. The *start* command is required at least every 10 seconds and it's interpreted as a *keep alive* message; the absence of *start* commands for more that 10 seconds it's interpreted as a *stop* command. The format of the Start command is the following

Start,<IMU_ID>,<PORT>

- **Stop command**: it terminates the readings from the sensor and the packet sending and put the application into the initial waiting state for energy saving. The format of the Stop command is the following

  Stop,<IMU_ID>

During its activity, the application waits also for **Vibration commands**: they tell the device that an activation of the vibration motor is required. The format of the Vibration command is the following

Vibration,<Duration>,<PWM>

The duration field contains the time in millisecond for which the motor will be activated; the PWM value allows to regulate the intensity of the vibration. Even if the MPU-9150 provides up to 1000 accelerometer and gyroscope samples per second, for the application a much smaller rate is required. In order to have accurate measurements and to save energy a 50Hz-rate has been chosen for the readings from accelerometer and gyroscope and a 25Hz-rate for the readings from magnetometer. The raw data coming from the IMU is then processed using the RTQF fusion algorithm which ouputs orientation data.

During the loop, at each iteration the application reads also the value coming from the ADC to which the force sensor is connected. All this data is packetized into a structure which contains 7 floats:

| | | | | | | |
|---|---|---|---|---|---|---|
| H Rot X | H Rot Y | H Rot Z | F Rot X | F Rot Y | F Rot Z | Force value |

and the 224-byte packet is sent over UDP.

### 3.4.2 RTQF: the fusion algorithm

RTQF [3] is the default fusion algorithm used in all versions of RTIMULib. It's a much simplified version of a Kalman filter that does just enough for effective fusion of data from gyros, accelerometers and magnetometers. Essentially how RTQF works is that the gyro rate outputs are used, along with the time between samples, to linearly extrapolate the previous orientation to the predicted current orientation of the IMU. In order to stop this getting totally out of line with reality, the accelerometers and magnetometers provide an absolute reference (pitch and roll for the accelerometers, yaw for the magnetometers). The accelerometer and magnetometer outputs can be quite noisy and subject to all kinds of perturbations so the trick is to combine everything in a way that produces nice results in real life. RTQF calculates two quaternions at every step – one is the predicted quaternion from the gyro rates while the other is the ground frame-referenced measured quaternion from the accelerometers and magnetometers. The predicted quaternion is the most stable but subject to drift. The measured quaternion is less stable but has no drift.

**Slerp** stands for Spherical Linear Interpolation and is a technique for finding an intermediate quaternion between two other quaternions. Quaternion Slerp uses a parameter, the Slerp power, that is used to control where between the two original quaternions the new one is generated. So, in this case, the Slerp power controls where the resulting quaternion is between the predicted quaternion and the measured quaternion.

The Slerp power can range between 0 and 1. If it is 0, the measured quaternion is ignored and only the gyros are used effectively. If the Slerp power is 1, the predicted quaternion is ignored and only the measured state from the accelerometers and magnetometers is used. RTIMULib uses a default value of 0.02. This means that mostly the gyros are used although the predicted state is corrected by 1/50 of the difference between the predicted state and measured state.

**Calibration**

In order to work correctly the IMUs need to be calibrated with the tool provided by RTIMULib. The end result is a file in the working directory called RTIMULib.ini that contains all of the generated settings.
RTIMULib supports the following IMU calibration options:

- Compass min/max calibration. This is a simple scheme that obtains the minimum and maximum readings from the magnetometers and then scales the readings based on these values.
- Compass ellipsoid fitting. This is a secondary (optional) stage to the min/max calibration that further refines the response by fitting an ellipsoid to the magnetometer readings and shrinking these coordinates so that they fit on the surface of a sphere located at the origin. This option requires GNU Octave on the host system.
- Accelerometer calibration. This uses a simple min/max system to scale the accelerometer outputs to between -1g and +1g when not being artificially accelerated.

## 3.5 Integration in the existing system

A pair of presented devices has been used in the configuration with 2 IMUs to track the orientation of the user's hands and index fingers. An additional device, in the configuration with only 1 IMU, has been placed inside of a helmet in order to track the orientation of the head: the helmet is also provided with a marker for positional tracking and need to be worn by the user in a CAVE system. The tracking data is used to update the viewpoint and the images on the walls.
Data (euler angles and force value) from the devices are sent via UDP to a middleware data collection application. This middleware collects all the data from the tracking devices (including OptiTrack and RGBD camera), formats the data in a string according to the XVR framework, and forwards it via UDP to the graphical application on the XVR default port 45000.

VR_SENSOR_*ID*, pos_x, pos_y, pos_z, rot_x, rot_y, rot_z, force_value

The position fields are filled with the data coming from the tracking system, while the rotation and the force value fields with the data coming directly from the IMU.

## 3.6 Introducing advanced Haptic Feedback

Haptic devices can provide a sense of touch and force feedback to the users [13]: for these reasons they are particularly suitable for those applications which contemplate the exploration of the virtual environment or the interaction with the object present in it [42].
The device described in the Chapter 3 is only able to provide a very simple feedback to the user by means of the vibration motor. The intensity of the vibration is modulated according to the haptic feedback that is required.
In order to provide advanced feedback, that vibration motor has been substituted with a more effective device that is able to modulate skin stretch at the fingertip.
This Thesis has not deal with the design and the realization of this device, but with its integration in the proposed system.

### 3.6.1 Hardware description

The advanced haptic device is a wearable fingertip [14] with 3 DoF asymmetric 3-RSR kinematics. Rendering of skin stretch in 3 degrees of freedom (DoF), with contact - no contact capabilities, has been implemented through rigid parallel kinematics.
A differential method for solving the non-trivial inverse kinematics has been also proposed and implemented in real time for controlling the position of the skin tactor.
The result of the design choices has been a small and lightweight device particularly suitable for rendering haptic feedback during manipulation in virtual environments. The device is presented in a dual finger configuration

(index and thumb) in order to allow two-fingers fine manipulation of objects.

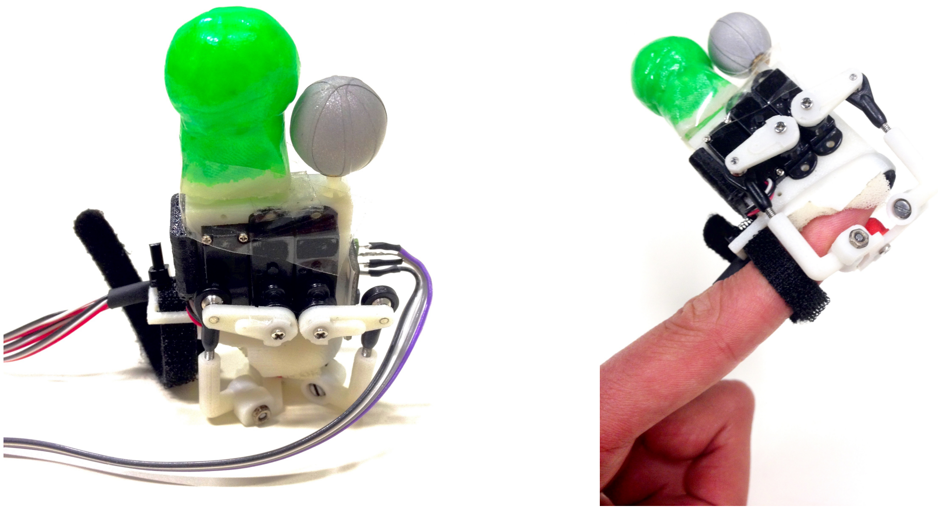


Figure 3.10: The haptic device

To actuate the device 3 lightweight (4g) and compact size (20x23x8 mm) electromagnetic servo motors have been used, driven through a Pololu Micro Maestro control board.
Thanks to the parallel structure, all the actuators have been mounted on the base fixed to the back of the fingertip. Thus, the encumbrance within the hand workspace has been minimized to the only end-effector with the skin tactor.

### 3.6.2 Device integration

In order to integrate the haptic device into the existing system:

- the device itself has been provided with markers (for positional tracking) and an IMU (for orientation tracking)
- the graphical application has been modified in order to handle the information coming from the device and to faithfully actuate the motors.

Since the purpose of this work is to investigate the interaction with virtual objects, both finger configurations, for the thumb and for the index finger have been used.
On top of each device have been placed an infrared marker and a colored marker in order to track the position of the two fingers using the data coming either from the OptiTrack and the 3D camera mounted on top of the Oculus. The actual position of the fingers has been obtained using only the data coming from the Optitrack system, but the use of the colored markers came from the need to distinguish the thumb (blue) from the index (green) finger.
The orientation tracking of the fingers takes place by means of an IMU mounted on the device. Since the servo motors generate an electromagnetic field, the magnetometer used in the IMU can be subjected to malfunction if located close to the motors. Thus, several trials have been done in order to obtain a location for the IMU where the influence of the motors was negligible. Each trial needed a calibration specific for that location and the best position resulted to be the one shown in Fig. 3.11.

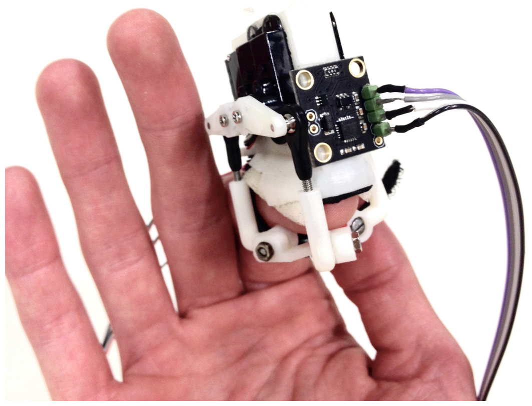


Figure 3.11: The haptic device and the IMU

Providing the orientation of the two fingers, the IMUs allow to establish the actual position of the fingers in the space with respect to the marker's positions with the following formula:

$$pos_f = pos_m + d_{fm} \cdot R_f$$

where:

- $pos_m$ is the marker position obtained from the OptiTrack
- $d_{fm}$ is the distance (fixed) between the marker and the finger
- $R_f$ is the Rotation Matrix of the finger obtained from the IMU

# 4. System Integration and Testing

*If we are to make reality endurable,*
*we must all nourish a fantasy or two.*

Marcel Proust

Upon completion of the development of the interaction device, the implementation and the evaluation of the previously presented solution have been addressed.
To handle most of the basic VR requirements (loading the 3D model of the environment, performing stereoscopic rendering, gather sensors data) we use the flexible and efficient XVR framework [51], that allows us to have a fine-grained control on the basic aspects of visualization and interaction.
In this chapter are presented two different applications which make use of the system. For the applications, the virtual environment has been represented with a 4 by 4 meters sized virtual room. The user, located inside the room can freely move inside it, wearing on both hands the developed device.

## 4.1 Hand Model Rendering

The first graphical aspect addressed for the application development has been the *look and feel* of the user hands. The simplest way for representing the movements of the hand is to show an object (such as a sphere) placed according to the positional information received from the tracking system.
To provide an higher feel of presence to the user, two more complex render strategies have been taken into account as described in the following.
Since the aim of the project is to provide users with the ability of performing manipulation tasks the markers for the positional tracking are placed on the index fingertip as the Fig. 4.1 shows.

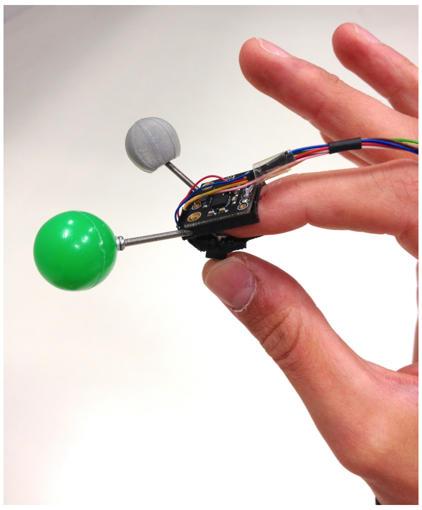


Figure 4.1: Location of the marker for positional tracking

### Real Hand

The system described in the Section 3.1 is used to show a real-time textured geometric mesh of the visible parts of the hands and body as seen from the user perspective. For the more specific tasks of real-time reconstruction and visualization of the data captured by the camera the custom rendering XVR plugin based on the hardware-accelerated approach and described in [52] has been used. All the external modules have been developed in C++.

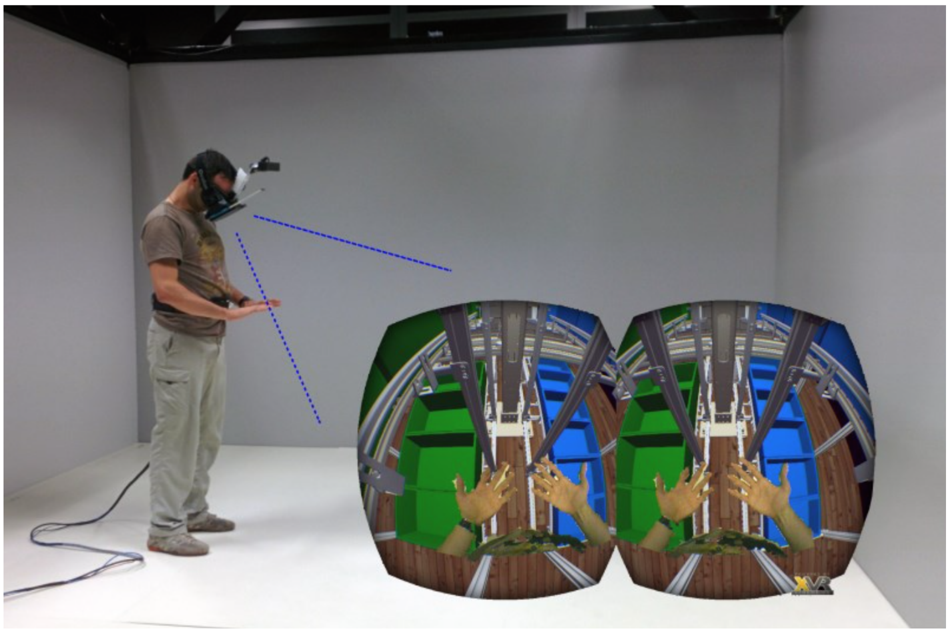


Figure 4.2: Real Hands rendering

The position tracking of the hand is performed fusing the data coming either from the 3D camera mounted on top of the Oculus and the OptiTrack system. Color markers have been also used to distinguish the two hands.
The user sees his own real hands operating in the virtual environment, so high natural and intuitive interaction is provided. On the other side the user seeing the devices on his hands could be distracted or annoyed.

## Virtual Hand

According to [23] when one's own hand is placed out of view and a visible fake hand is repeatedly stroked and tapped in synchrony with the unseen hand, subjects report a strong sense in which the fake hand is experienced as part of their own body.
The render of virtual hands is achieved by showing a virtual representation of human hands in the same position of where the real actually are. The hands are oriented and animated according to the information coming from the IMUs:

- the IMU placed on top of the index provides the orientation of the whole hand
- the IMU placed on the back of the hand provides the relative orientation of the hand with respect to the index finger orientation.

The system does not provide any information about the rotation of the other finger but the index. However has been observed that when performing a pinch

- the middle, the ring and the little fingers follow the movement of the index in a proportional way;
- the thumb gets close the index during the pinch and *gets away* during the release.

In order to make easy the animation and the management of the avatar hands, an high-level class *HandCharacter* has been created; this class allows to manipulate the complex hierarchies of objects of the hand, providing some methods for rotating the index, the thumb and the other fingers and handling the rotation of the single phalanxes of each finger.

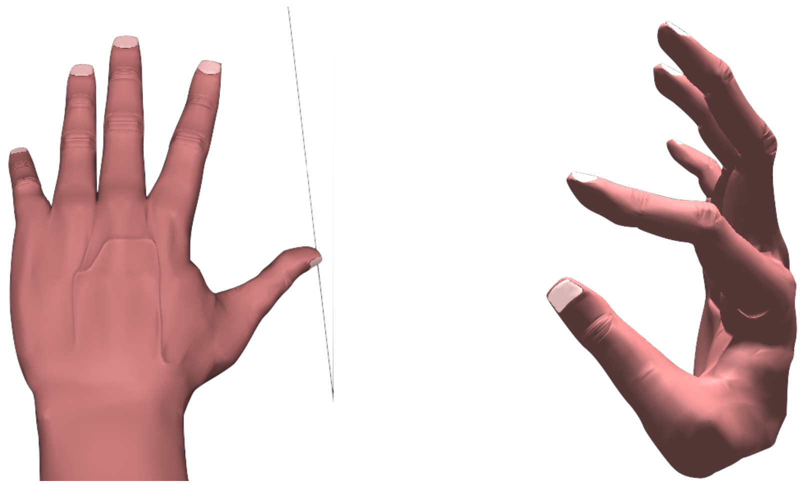
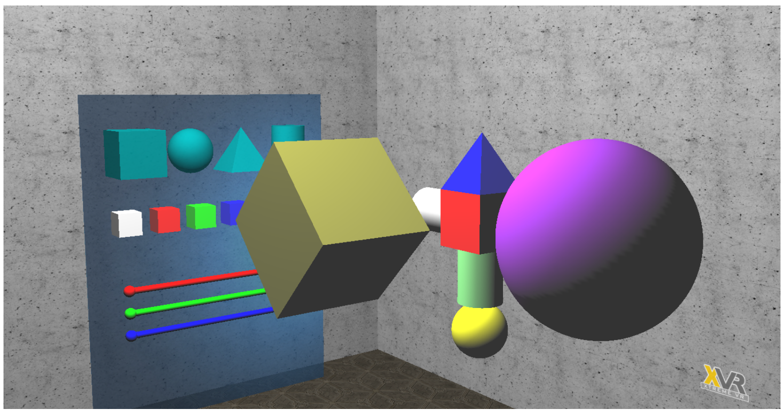


Figure 4.3: The virtual render of the hand

## 4.2 World Builder



This application allows the users to populate a scene with objects of elementary shape such as cubes, spheres, pyramids and cylinders. The application provides different type of hands render according to the needed and the requirements.
Initially the user is in an empty room and in order to add objects inside the scene has to interact with the *palette*.

### 4.2.1 The palette tool

The palette is the tool which allows users to customize the scene with objects. In particular it provides the possibility to put an object inside the scene and to color it. The palette is enabled with the solicitation of the force sensor placed on the left-index phalanx. The Figure 4.4 shows a model of a palette divided two sections: shapes and colors. The *shapes* section shows the procedural objects which can be easily imported to the scene with a simple grab. The imported objects take default dimension (0.2m) and gray color. To modify the latter, the *colors* section can be used. It shows a series of fixed colors which can be applied to the object by grabbing them to the desired object. Three sliders with the primary colors red, green and blue are also provided in order to customize the final color.
When the palette is activated, it's placed in the left part of the current

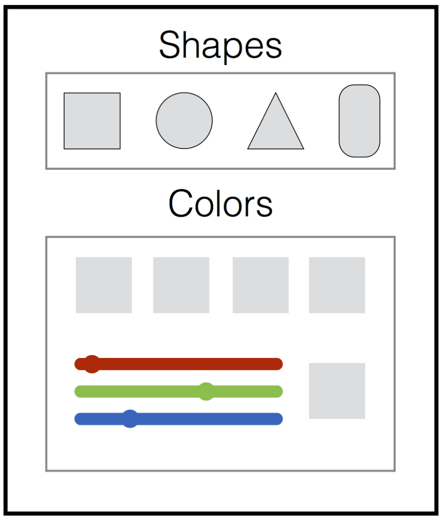


Figure 4.4: Design of fixed palette

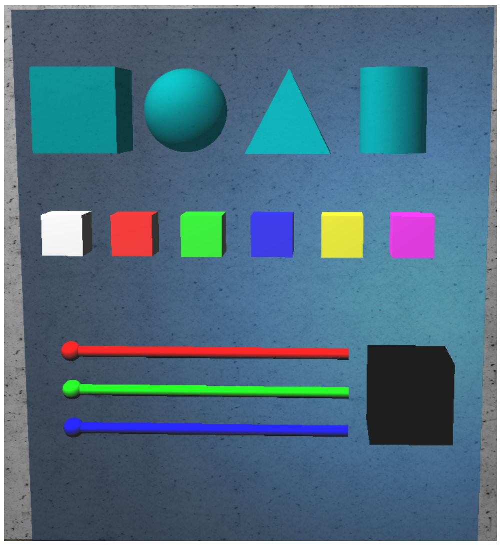


Figure 4.5: The VR palette

viewpoint of the user and remains there until another activation occurs.
A mobile version of the palette is also available: when active, it's shown on the user's left-hand and follows the orientation of the hand. In this case in the side facing the palm of the hand are shown the same object, while in the side facing the back of the hand are shown the colors and the sliders. The mobile palette is shown until the user pinches.

### 4.2.2 Object manipulation metaphors

To place an object inside the scene the user has to grab it from the palette. A grasping condition occurs when the user pinches an object by exerting a force between the index and the thumb -namely when the force sensor detect a force above a certain threshold- within or in close proximity of an object. Once an object is placed into the scene, the user can interact with it using his own hands in order to customize its position, orientation and dimension.

#### Translation

The translation of an object is enabled with a weak pinch on one of its surface and it lasts until the user release the pinch. This state transition is announced with a sound, a vibration feedback and for the duration of the translation the involved object takes a red color. Inside this state, using the tracking information, the position of the object will be update according to the hand one and it allows the user to place an object into a desired position inside the scene.

### Rotation

The rotation of an object is enabled with a strong pinch on one its surface and it lasts until the user release the pinch. Also in this case, the state transition is announced with a sound, a vibration feedback and a green color of the involved object. Inside this state, using the IMU and the tracking information, both the orientation and the position of the object will be update according to the hand ones. The rotation of the object is pivoted to the pinching point and while rotating, the object can be also translated.
To let the object rotate around its pivot point the following procedure 4.6 is executed:

1. The pinching point (pivot point) coordinates are computed with respect to both the world frame and the local frame of the object.

2. The object is translated into the pinching point.

3. The object is rotated with respect to the hand rotation around its center.

4. The object is translated back of a quantity equal to the distance between its previous position and the pinching point position with respect to the local frame.

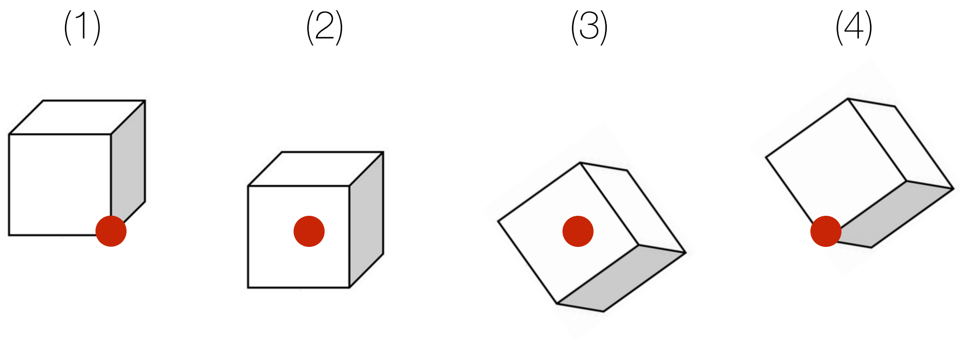


Figure 4.6: Rotation of a cube around a pivot point

### Scaling

The objects inside the scene can also be scaled, either on a single axis and uniform on the three axis:

- To scale an object on a specific axis -namely to extrude one of its surface- a pinch on one of the six red dots is needed. Each dot allows the extrusion of the relative surface which belongs. The *center of scaling* in this case correspond with the center of the object. The *scaling factor* on the scaled axis is proportional to the hand displacement in the sense that the pinched point behaves like attached to the hand, while on the other axis the axis factor remains constant. Since the scaling on a certain axis produces a scale in both direction, the object undergoes a scale corresponding of the half of the hand displacement and a translation equal to the half of the hand displacement on the same axis.

- The homogeneous scaling of an object is enabled with a weak pinch with both hands on the object surfaces. As above, the*scaling factor* is proportional to the hand displacement and the *center of scaling* is the center of the object. In this case the translation undergone by the object is directed to the perpendicular of the hands' displacement direction in order to maintain the proportions between the pinched points. This provide high-intuitive interaction.

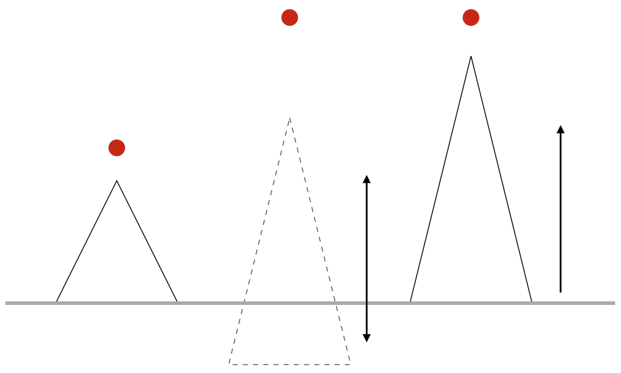


Figure 4.7: Scaling of a pyramid on the vertical axis. The object is first scaled and then translated

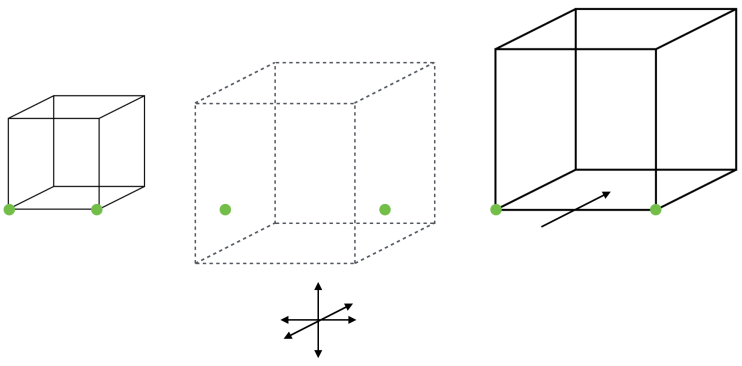
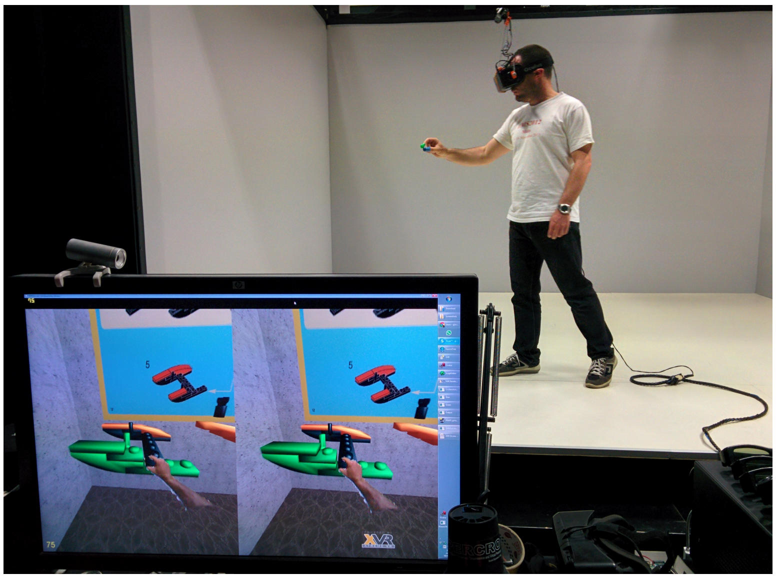


Figure 4.8: Homogeneous scaling of a cube. The object is first scaled and then translated

In each frame the following steps are executed:

1. Get head position and orientation from the OptiTrack
2. Get hands position (from OptiTrack and RGBD camera) and orientation (from IMU)
3. Compute the new viewpoint and the hands animation
4. Check whether or not the user is pinching with the finger
5. If yes, check the eventual objects involved in the pinching
6. Update object (position, orientation and dimension) according to the pinching mode (Translation, Rotation)
7. Provide the adequate feedback
8. Update the scene and the state variables

## 4.3 Operator Training Simulator



Global industrial manufacturing capacities constitute a large part of the world wealth and economy. A key component of any manufacturing business is training: training a specialized workforce as well as training the customers about the produced machineries requires huge amounts of time, resources and logistic facilities. Training has spill-over benefits for the industry (by providing a pool of skilled workers) and for the society (the improved employment outcomes and flow-on effects such as improved health and lower social welfare costs). Currently, in the field of industrial manufacturing training is a hugely expensive activity traditionally burdened by a number of issues such as the cost of realizing a training environment, the cost of using machineries beyond the working hours, security risks when a trainee uses an equipment and more. These considerations have in time lead to the suggestion that the use of Virtual Reality could introduce significative benefits in the training processes, by removing the need of physical mock-ups in the training process or at least in some of the procedures.
The application presented is intended to be used by industrial companies who needs to train their operators on the tasks of assembly, disassembly or maintain large mechanical machines. What motivates the use of Virtual Reality

is that a real copy of the machine could be cumbersome and expensive, and very likely it might result impossible to work together on the same machine at the same time. Moreover usually an expert assistant is required during the training phase in order to assist the operator. The proposed application provides the needed metaphors to interact and manipulate a 3D model of the machine in absolute autonomy with the purpose of following out a task. The application provides a controlled and safe training environment, in which damages to the real machine are reduced or avoided; hence, inexperienced users can take advantage of virtual training before actually facing the real machine.
The application presents a scene consisting of a room, within which the user, wearing the visualization system described above, can walk around. At the center of the room is placed the 3D model of the machine, while on the wall are present the instruction tables that help the operator during the task.
The application consists of two fundamental modes of operation: the authoring mode and the training mode.

### 4.3.1 Authoring mode

This modality is intended to be performed only once by an expert who owns already deep-knowledge of the machine, of the procedures that can be performed on it, such as maintenance, and of each of the steps that needs to be followed to disassemble it. The expert can disassemble the machine, piece after piece. During this phase, the expert defines steps and sequences. In our mind a **sequence** consists of an ordered list of steps, and each **step** consists of an unordered list of pieces to move. According to this *notation*, inside a step the pieces can be moved without a specific order, but inside a sequence the steps must be *sequentially* performed. The expert can also mark a step as a **group** of pieces: this implies that the step will consist only in the translation of the entire group into its target position. The expert thus disassembles the group piece-by-piece, with the possibility to further define nested groups. The concept of group is of primary importance especially in case of complex hierarchical machines: it allows to assemble/disassemble portions of the ma-

chine in a location different from the final one, helping the operator to have a more organized and schematic view of the entire machine. If during the disassembly the expert makes a mistake, he can navigate through the steps, undo the changes and start over.
While authoring, the expert places each item in a specific location. This location is marked as starting position for the specific piece during an assembly session: in other words, when the trainee will assemble the machine, at the beginning of each step he will find the pieces exactly in the position the expert left it. If inside a step, two or more pieces are left in the same position (and have the same dimensions) they are marked as **equivalent**. In this case, only one of the equivalent pieces is shown with a label indicating the number of equivalent pieces of which consists. This implies that in assembly mode, the operator can place an equivalent piece in the target position of any other equivalent (e.g. if the expert marks some screws as equivalent, the operator will be able to place screws in any well fitting location). During this phase the expert can also take snapshots (Appendix A.2) of the state of the machine from its own point of view. These snapshots are stored and can be used and modified in order to define instructions tables that can be presented to the operators during the training phase.
At the end of the authoring session an ordered list containing all the sequences, the steps and the equivalences defined by the expert is saved on file (using the tool described in Appendix A.1), as shown in Fig. 4.9.

### 4.3.2 Training mode

This mode is intended to be performed by operators for training. The trainee can either work on the machine in a 'free' mode, or can perform a training session on the machine assembly or disassembly. In the first case the operator can interact with the machine model, without any constraint in terms of sequences and steps, in order to discover how it is made. When instead the operator performs a training session, he's constrained to the specific sequence previously executed by the expert in the authoring mode.
On the walls of the room, the operator can find the instruction tables relative

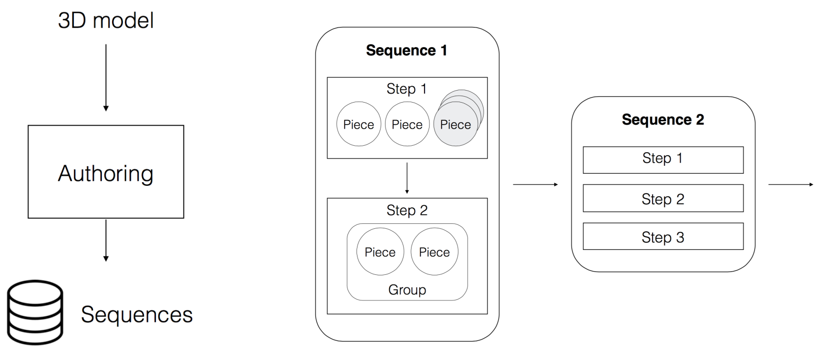


Figure 4.9: The authoring stage produces the sequences structure starting from the 3d model

to the step he's performing, and information about the progress of his task: number of remaining pieces, number of steps completed, number of sequences completed and time passed.
During the training several hints could be activated, depending on the difficult of the task, to help the operators: It is possible to define help layers that can be dynamically presented to the users that encounter difficulties in performing certain operations. Furthermore the operators can perform the same task several times - with a decreasing level of help - until they acquire the needed familiarity with the machine. The application's flow chart is shown in the Fig. 4.10.

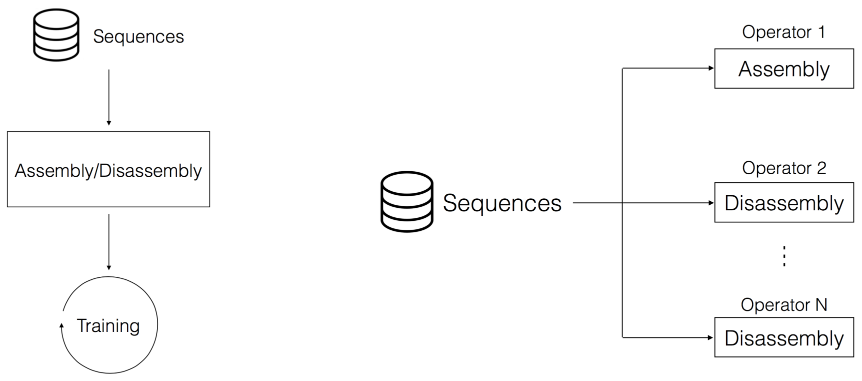


Figure 4.10: The training system flow chart.

#### Disassembly mode

In this modality the instruction sequences are loaded from file in the same order they have been saved. At the beginning of the task, the machine is completely mounted and in order to move forward on the task, the operator must complete the needed steps by removing the right pieces from their starting position. Since it is a disassembly task, the operator is not required to put the items into a target position; in order to clean up the scene, at the end of each step, the moved pieces are translated into the position the expert left them. The operator is able to move only the pieces that must be actually moved inside that step. When the operator grabs a piece that can be moved it becomes green, while if the piece cannot be moved it becomes red. Each action results also in an acoustic feedback that alerts the users that the action has been actually performed by the system.

#### Assembly mode

In this modality the instruction sequences are loaded from the file in a backward order. At the beginning, the first piece is already placed into its target position. The remaining pieces relative to the step/sequence are showed in their start position, namely where the expert left them during the authoring stage. Equivalent pieces are showed together with a label indicating the number of multiple items. The task of the operator is to put all the pieces into their right target position. If the operator leaves the pieces in a closest range of the target, they are automatically snapped to the correct position. Also in this case each action results in an acoustic feedback that alerts the users that the action has been actually performed by the system.

## 4.4 Integration of the advanced haptic device

In order to integrate the advanced haptic device described in Section 3.6 the condition for the object grasping, previously based on the force exerted between thumb and index fingers and measured by means of the force sensor, has been substituted with a more realistic one based on the contact between

the fingers and the objects.
If an object is touched with only one finger, it is moved in the same direction of the finger movement. If instead an object is touched with both fingers it is grasped and it's moved according to the movement of the mid-point between the fingers position.
The contact-no contact condition is based on the closeness of the fingers to the surface of the object. When a finger enters is contact with an object two quantities are computed:

- the normal vector between the surface and the contact point: it's used to calculate the orientation of the force on that finger.
- how much the finger virtually enters the object: it's used to calculate the value of the force that will be exerted.

For the collision detection between the fingers and the objects the native XVR function *isColliding* has been used.

```
bool IsColliding(
   vector[3] FromPoint,
   vector[3] ToPoint,
   vector[3] &ContactPoint,
   vector[3] &ContactNormal
);
```

The function tests the intersection between the current object and the segment (FromPoint, ToPoint). In case of collision, the intersection point is returned in ContactPoint and a vector representing the normal to the contact point is returned in ContactNormal.
The returned ContactNormal is expressed in the object coordinate system. So in order to refer it to the finger the coordinates of the normal needs to be converted to the finger reference system.
The position of the end-effector with respect to the fingertip gives the feeling of the exerted force.
The inverse kinematics of the device has been solved offline for each point

the end-effector can reach with a resolution of 0.1mm. The results have been stored in a look-up table which contains the values relative to the angular positions of all the actuators for each point. These value are sent to the device using serial communication (USB).

## 4.5 Preliminary user studies

In our pilot study, the aim of the subjects has been to perform an assembly task of a 53-pieces model of the LEGO® Creator Sea Plane [2]. The final task has been preceded by a training phase on the same real model, or on an equivalent 3D model. The 3D model has been exported from 3ds MAX®

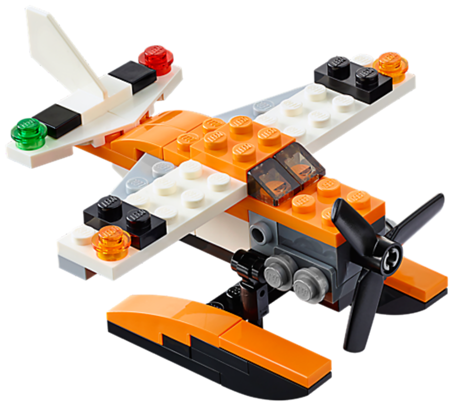


Figure 4.11: The real model

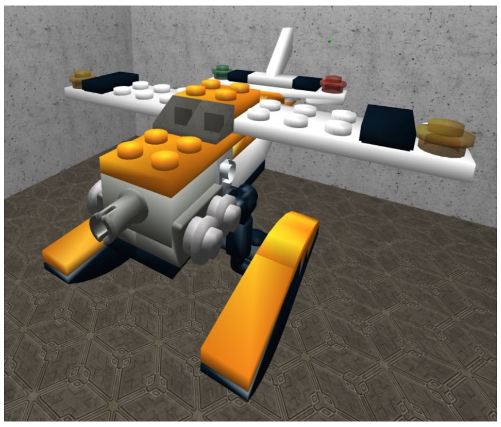


Figure 4.12: The 3D model

using the specific XVR plugin. The authoring stage has been performed by one of the experimenter strictly following the original instructions provided with the model, dividing the model into 3 groups (the flying boat, the tail and the pontoons) and identifying 5 sequences.

### 4.5.1 Description of the pilot study

The test group consisted of 8 subjects, divided into two sub-groups. Both sub-groups have performed a 30-minutes training, the first one using the real model, and the second one using the virtual model with the presented system.

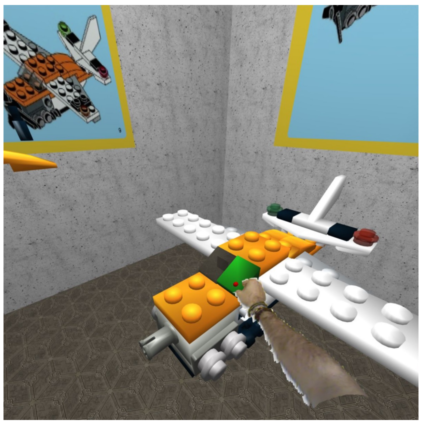


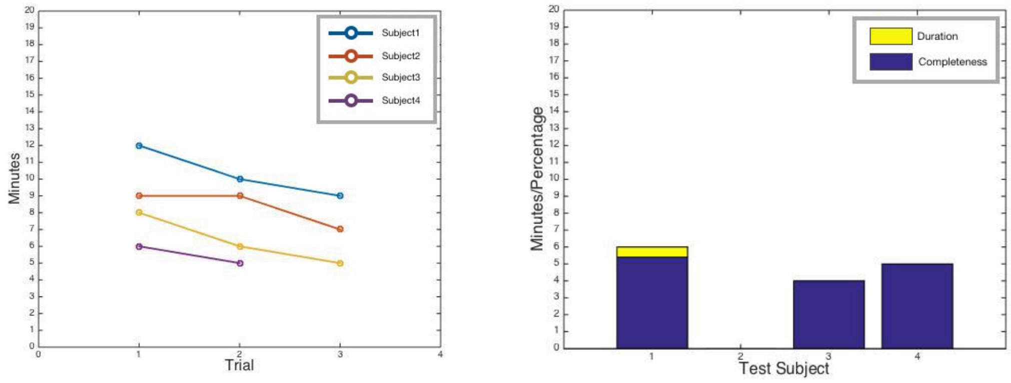
Figure 4.13: Virtual training

Figure 4.14: Assembly of the real model

During the training, the subjects were provided with the same instruction tables needed to accomplish the requested task and they could use the time at their disposal to perform the task several times, or just to study the model and the relative instruction tables.
After the training each subject has performed the actual task: an assembly of the real model without time constraints and instruction tables. The performing time has been registered and the results evaluated according to the number of pieces correctly placed.

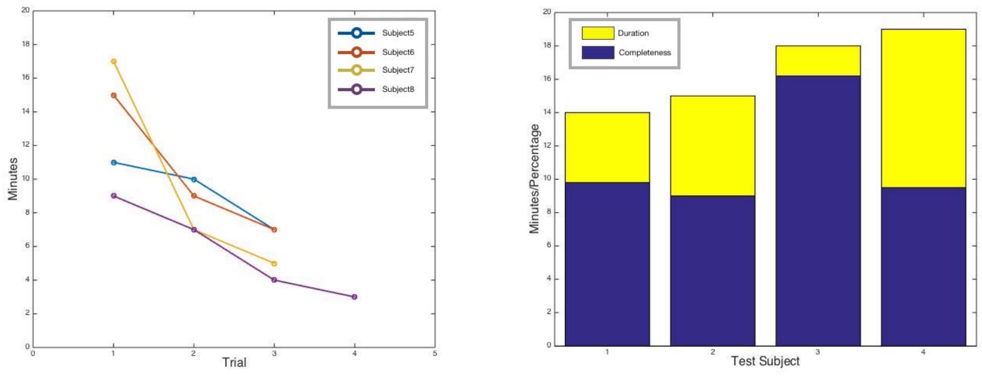
### 4.5.2 Results

The Fig. 4.15 and 4.16 show the results of the pilot study. In particular the Fig. 4.15a and 4.16a show the completion times of assembly sequences performed respectively during the real training and the virtual training. The Fig. 4.15b and 4.16b show the time needed to assemble the real model after the training (in yellow), and the percentage of completion of the model (in blue).



(a) Duration of training sequences (b) Real assembly

Figure 4.15: Training on the real model



(a) Duration of training sequences (b) Real assembly

Figure 4.16: Training on the virtual model

The charts show that the time needed to complete an assembly sequence significantly decreases in the virtual training with respect to the number of trials. The first trial in virtual training requires more time because of the needed familiarization with the environment. Even if the assembly of the real model after the virtual training has required longer time, all the subjects have been able to complete the assembly of the model for almost the 50%, with a peak of 90%.

# 5. Conclusions and future works

*There are two possible outcomes: if the result confirms the hypothesis, then you've made a measurement. If the result is contrary to the hypothesis, then you've made a discovery.*

Enrico Fermi

This Thesis project has been investigating interfaces and metaphors for interaction in Fully Immersive Virtual Environments with the aim of providing users with the ability of performing manipulation tasks.
The research activity has begun with the collection of the relevant State of The Art and the analysis of the solutions currently suggested in the literature to let users the manipulation of virtual objects.
Then, always keeping in mind a set of usability criteria to satisfy (reliability, weightlessness, cumbersomeness), an original device has been designed. The set-up of a practical and effective inter-networked test environment has been important during the design of the necessary components and embedded firmware that would allow verification and tests on the fly. This has helped to significantly speed-up the development phase since the device could be programmed without been physically connected to the complex VR equipment.
The result of this work has been thus integrated in the existing system with

the purpose of validating it in a complex interaction testbed and two demonstrative applications have been implemented for this specific task.
The second part of the project has been concerned with the implementation of advanced haptic feedback in the system, and this has been obtained by means of a novel haptic device.

## 5.1 Achievements

The various research activities have allowed the design of a thimble-like interface that can be used to recognize in real-time the human hand's orientation and infer a simplified but effective model of the relative hand's motion and gesture. Inside the virtual environment, users provided with the developed system are therefore able to operate with natural hand gestures in order to interact with the scene; for example, they could perform positioning task by moving, rotating and resizing existent objects, or create new ones from scratch.
More in particular, the existing SSSA system, which consists of a powerful optical tracking system and an HMD with a 3D camera mounted on top of it, has been thus exploited to develop the demonstrative applications that involves the use of the user's hands for performing manipulation tasks:

- **World Builder** allows users to populate a room with objects of elementary shape chosen from a palette. The objects can be manipulated in position, orientation and dimension and their color can be easily modulated.

- **Operator Training Simulator** allows the training of the users in assembling or disassembling complex mechanical machineries by following predefined step sequences.

A preliminary user study, aimed at assessing the effectiveness of the training performed in a virtual environment has then been conducted: the users have been asked to assembly a model of a seaplane first using a Mixed Reality system and then in the real counterpart scenario.

The collected results suggest that the virtual training could be so effective as the real one, but more extensive user studies need to be performed in order to further assess the benefits of the virtual approach.

## 5.2 Future directions

At the current status the project already provides the necessary tools and methodologies for hand's orientation tracking, allowing users to manipulate virtual object by means of interaction metaphors.
However the modular architecture enables several possibilities for future development that would allow addressing some exciting challenges that remains still open. More in detail:

### 5.2.1 Combining Optical and Inertial Tracking

The OptiTrack system used in this Thesis is able to provide data coming from the cameras at 100 FPS. Especially when dealing with haptic interfaces, there is the need to have positional data at higher frequency in order to determine contact-no contact situation and to provide the adequate feedback with no-delay. Numerical stability is also a well known problem in haptic rendering that is highly alleviated by a high-frequency rendering loop.
In the presented device, the IMU MPU-9150 is only used to obtain fused data about the orientation of the user hand from accelerometer, gyroscope and magnetometer; the accelerometer can provide raw data up to 1000 Hz and we think that it can be used to obtained the linear acceleration of the user's finger at high frequency, in order to compensate the OptiTrack measures and have fine-grained positional tracking.
Some work has already been done, especially for what concerns the data reading, the gravity compensation and the correspondence between the different coordinate systems involved.
Obviously the challenges to address are several:

- the **double integration** to obtain position from acceleration introduces unavoidable measurement errors due to tiny vibrations, imperfections in the manufacturing and so on. Accumulating these errors over time results in a *Random Walk*. The OptiTrack measurements 'resets' the error at fixed rate, but the introduced error must be correctly evaluated also in the case of short-term double integration.

- the **latency** between the measurement of the position with the OptiTrack and its communication to the device. This point is crucial for the estimation of the position using the accelerometer because the device uses the last received position as starting one: noon-up-to-date data could produce a significant error in positional reckoning. The introduction of measurement *timestamps* could alleviate the consequences of this problem.
  Another issue to face regarding the communication is the **unreliability** of the wireless link. If several packets get lost the error introduced could be inappropriate for some applications, so space is present here for some more sophisticate communication protocol.

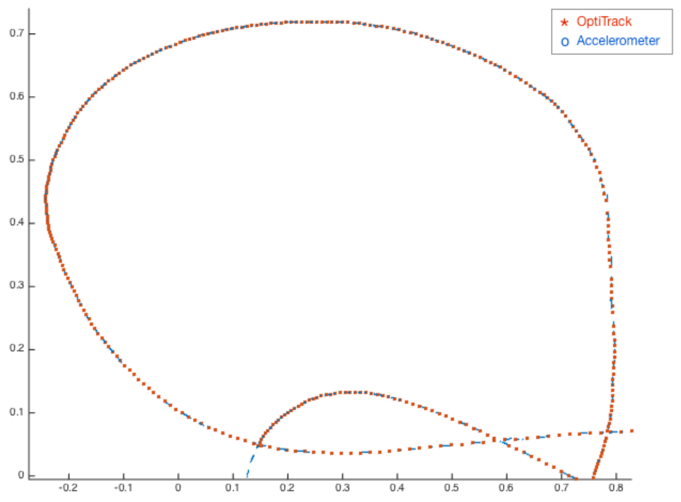


Figure 5.1: A scatter plot of a trajectory

### 5.2.2 Using active IR leds for Optical Tracking

The markers used for the Optical tracking are passive retro-reflective markers: even if they have small dimension they are bigger and more visible than an active IR led. Moreover there is no chance to distinguish a marker from another but to use algorithms that associate the same marker to the closer position measured. In the project the markers relative to the two hands have been distinguished with the use of colored marker and a RGBD camera.
The use of active IR leds for optical position tracking would allow the definition of different illumination patterns for each led. To do so, the device needs to recognize the illumination rate of the OptiTrack and to synchronize with it. The application should be only able to recognize the pattern and to associate to it the correct position.

# Appendices

# A. Support tools developed

## A.1 XVR Object SerDes

The XVR Object SerDes is an XVR class which provides methods for serializing and deserializing XVR objects. The class works also in case of nested objects (i.e. objects which have other objects as attributes). This is useful for storing XVR objects without losing their type and structure.
A function *ClassName*_Factory() is required in order to serialize instances of that class. The function must return an array containing an instance of the class and an array containing the attributes to serialize.

```
function ExampleObj_Factory() {
  return {ExampleObj(), {"name", "id", "arr"}};
}
```

The library offers two methods: Serialize and Unserialize.

### Serialize

**string** Serialize(obj)

- **obj**: The object to be serialized. **serialize()** handles all types.
- **return value**: A string containing a byte-stream representation of **obj** that can be stored anywhere.

The serializer generates a storable representation of an object. The serialization stage, given an instance of a class and a list of attributes to serialize,

returns a dump of the instance in a string. Primitive objects (i.e. string, int, bool, vector) are encoded using the XVR function *VarStringEncode(var)* which encodes the content of the variable *var* in a string.
Non-primitive objects (i.e. array, class) are recursively serialized according to their type.

Listing A.1: Example of serialization

```
var object_to_be_serialized = TestObj();
var child_obj = ChildObj();
object_to_be_serialized.id = 1;
object_to_be_serialized.name = "Father";
object_to_be_serialized.arr = {true, 2, "Lorem"};

child_obj.name = "Child";

object_to_be_serialized.child = child_obj;

var serializer = Serializer();
var serialized_obj = serializer.Serialize(object_to_be_serialized);
```

## Unserialize

**class** Unserialize(serialized_obj)

- **serialized_obj**: the serialized string.
- **return value**: the converted obj is returned,

The unserializer takes a serialized variable and converts it back into an XVR object. Also in this case, primitive objects are decoded using the XVR function *VarStringDecode(var)*, while non-primitive objects are recursively deserialized according to their type.

Listing A.2: Example of deserialization

```
var serializer = Serializer();
var new_obj = TestObj();

new_obj = serializator.Deserialize(serialized_obj);
```

## A.2 XVR ScreenTool

XVR ScreenTool is a simple tool that provide an intuitive way to take screenshots from the inside of an XVR application.
The action can be triggered by any XVR event, such as key pressure, timer expiration, collision detection and so on. The image is then saved in PNG format.
This tool is particularly suitable for those situations in which the user is far from the keyboard or is wearing an HMD and the traditional way to take screenshot is not comfortable.

# B. Code Listing

Listing B.1: Reading from IMU

```
while (imu0->IMURead() && imu1->IMURead()) {
        /* Retrieve data from IMUs */
        RTIMU_DATA imuData0 = imu0->getIMUData();
        RTIMU_DATA imuData1 = imu1->getIMUData();

        /* Retrieve data from ADC */
        float force_value = getForceValue();

        /* Fill data packets */
        data_IMU[0][0] = imuData0.fusionPose.x() * RTMATH_RAD_TO_DEGREE;
        data_IMU[0][1] = imuData0.fusionPose.y() * RTMATH_RAD_TO_DEGREE;
        data_IMU[0][2] = imuData0.fusionPose.z() * RTMATH_RAD_TO_DEGREE;
        data_IMU[0][3] = 0;
        data_IMU[1][0] = imuData1.fusionPose.x() * RTMATH_RAD_TO_DEGREE;
        data_IMU[1][1] = imuData1.fusionPose.y() * RTMATH_RAD_TO_DEGREE;
        data_IMU[1][2] = imuData1.fusionPose.z() * RTMATH_RAD_TO_DEGREE;
        data_IMU[1][3] = adc_value;

        /* Send data packets */
        for(int i = 0; i<NUM_OF_IMU; i++)
        {
                sendto(sock, data_IMU[i], 4*sizeof(float), 0,
                (struct sockaddr *) &IMUSocket[i], sizeof(IMUSocket[i]));
        }
}
```

Listing B.2: ADC Reading

```
float getForceValue()
{
        /* Disable I2C for IMU (pin 28|29) */
        INP_GPIO(28);
        INP_GPIO(29);
        /* Enable I2C for ADC (pin 0|1) */
        SET_GPIO_ALT(0,0);
        SET_GPIO_ALT(1,0);

        FILE *f0 = fopen("/sys/bus/iio/devices/iio:device0/in_voltage1_raw", "r");
        fscanf(f0, "%i", &adc_input);
        fclose(f0);

        float adc_value = 1 - (max(0, min(1, ((adc_input - min_val) /
                (max_val - min_val)))));

        /* Disable I2C for ADC (pin 0|1) */
        INP_GPIO(0);
        INP_GPIO(1);
        /* Enable I2C for IMU (pin 28|29) */
        SET_GPIO_ALT(28,0);
        SET_GPIO_ALT(29,0);
        return adc_value;
}
```

Listing B.3: Accelerometer compensation

```
RTIMU_DATA imuData1 = imu1->getIMUData();
uint64_t temp_now = RTMath::currentUSecsSinceEpoch();
float deltaTime = temp_now - now;
deltaTime /= 1000000.0f;
float accel[3], gravity[3], data_IMU1[3];

```

```
/* Get Accelerometer values */
RTVector3 accelData = imu1->getAccel();
accel[0] = accelData.x();
accel[1] = accelData.y();
accel[2] = accelData.z();

/* Get Orientation (RPY) values */
data_IMU1[0] = imuData1.fusionPose.x() * RTMATH_RAD_TO_DEGREE;
data_IMU1[1] = imuData1.fusionPose.y() * RTMATH_RAD_TO_DEGREE;
data_IMU1[2] = imuData1.fusionPose.z() * RTMATH_RAD_TO_DEGREE;


/* Compute gravity vector */
gravity[0] = 2 * (imuData1.fusionQPose.x() * imuData1.fusionQPose.z() -
        imuData1.fusionQPose.scalar() * imuData1.fusionQPose.y());
gravity[1] = 2 * (imuData1.fusionQPose.scalar() * imuData1.fusionQPose.x() +
        imuData1.fusionQPose.y() * imuData1.fusionQPose.z());
gravity[2] = imuData1.fusionQPose.scalar() * imuData1.fusionQPose.scalar() -
        imuData1.fusionQPose.x() * imuData1.fusionQPose.x() - imuData1.fusionQPose.y() *
        imuData1.fusionQPose.y() + imuData1.fusionQPose.z() * imuData1.fusionQPose.z();

/* Remove gravity vector from accelerometer measure */
accel[0] = accel[0] - gravity[0];
accel[1] = accel[1] - gravity[1];
accel[2] = accel[2] - gravity[2];

/* Change the coordinate system according to the global reference system */
glm::mat4 rotationMatrix = glm::yawPitchRoll(data_IMU1[2], data_IMU1[1], data_IMU1[0]);
glm::vec4 accel_base = rotationMatrix * glm::vec4(accel[0], accel[1], accel[2], 1.0f);

/* Check if there are new measures from OptiTrack */
if(!OptitrackUpdated())
{
        /* If not, predict velocity and position */
        accel[0] = accel_base[0] * 9.81;
        accel[1] = accel_base[1] * 9.81;
        accel[2] = accel_base[2] * 9.81;

        static float prev_accel[3] = {accel[0], accel[1], accel[2]};

        vel[0] += (prev_accel[0] + accel[0]) * 0.5f * deltaTime;
        vel[1] += (prev_accel[1] + accel[1]) * 0.5f * deltaTime;
        vel[2] += (prev_accel[2] + accel[2]) * 0.5f * deltaTime;

        static float prev_vel[3] = {vel[0], vel[1], vel[2]};

        pos[0] += (prev_vel[0] + vel[0]) * 0.5f * deltaTime;
        pos[1] += (prev_vel[1] + vel[1]) * 0.5f * deltaTime;
        pos[2] += (prev_vel[2] + vel[2]) * 0.5f * deltaTime;

        prev_accel[0] = accel[0];
        prev_accel[1] = accel[1];
        prev_accel[2] = accel[2];

        prev_vel[0] = vel[0];
        prev_vel[1] = vel[1];
        prev_vel[2] = vel[2];
}
```

# Bibliography

[1] How does the leap motion controller work? `http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/`. Accessed: 2015-06-21.

[2] Lego® creator sea plane. `http://shop.lego.com/en-US/Sea-Plane-31028`. Accessed: 2015-05-08.

[3] Quaternion slerp – how rtqf performs sensor fusion in the latest rtimulib versions. `https://richardstechnotes.wordpress.com/2015/03/29/quaternion-slerp-how-rtqf-performs-sensor-fusion-in-the/-latest-rtimulib-versions/`. Accessed: 2015-06-21.

[4] Smart laser scanner. `http://www.k2.t.u-tokyo.ac.jp/perception/SmartLaserTracking/index-e.html/`. Accessed: 2015-06-21.

[5] Alonzo C Addison. Emerging trends in virtual heritage. *Multimedia, Ieee*, 7(2):22–25, 2000.

[6] Alonzo C Addison. Virtual heritage: technology in the service of culture. In *Proceedings of the 2001 conference on Virtual reality, archeology, and cultural heritage*, pages 343–354. ACM, 2001.

[7] Matej Andrejašic. Mems accelerometers. In *University of Ljubljana. Faculty for mathematics and physics, Department of physics, Seminar*, 2008.

[8] Ronald T Azuma et al. A survey of augmented reality. *Presence*, 6(4):355–385, 1997.

[9] Steve Benford and Lennart Fahlén. A spatial model of interaction in large virtual environments. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work 13–17 September 1993, Milan, Italy ECSCW'93*, pages 109–124. Springer, 1993.

[10] Massimo Bergamasco, S Perotti, Carlo Alberto Avizzano, Marcello Angerilli, Marcello Carrozzino, and Emanuele Ruffaldi. Fork-lift truck simulator for training in industrial environment. In *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, volume 1, pages 5–pp. IEEE, 2005.

[11] Wolfgang Birkfellner, Michael Figl, Klaus Huber, Franz Watzinger, Felix Wanschitz, Johann Hummel, Rudolf Hanel, Wolfgang Greimel, Peter Homolka, Rolf Ewers, et al. A head-mounted operating binocular for augmented reality visualization in medicine-design and initial evaluation. *Medical Imaging, IEEE Transactions on*, 21(8):991–997, 2002.

[12] Doug A Bowman and Larry F Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 35–ff. ACM, 1997.

[13] Grigore C Burdea, Grigore C Burdea, and Cristian Burdea. *Force and touch feedback for virtual reality.* Wiley New York, 1996.

[14] Ilaria Bortone Daniele Leonardis, Massimiliano Solazzi and Antonio Frisoli. A wearable fingertip haptic device with 3 dof asymmetric 3-rsr kinematics.

[15] Antonino Gomes De Sa and Gabriel Zachmann. Virtual reality as a tool for verification of assembly and maintenance processes. *Computers & Graphics*, 23(3):389–403, 1999.

[16] Andrew T Duchowski, Vinay Shivashankaraiah, Tim Rawls, Anand K Gramopadhye, Brian J Melloy, and Barbara Kanki. Binocular eye tracking in virtual reality for inspection training. In *Proceedings of the 2000 symposium on Eye tracking research & applications*, pages 89–96. ACM, 2000.

[17] Eric Foxlin. Inertial head-tracker sensor fusion by a complementary separate-bias kalman filter. In *Virtual Reality Annual International Symposium, 1996., Proceedings of the IEEE 1996*, pages 185–194. IEEE, 1996.

[18] Eric Foxlin and Michael Harrington. Weartrack: A self-referenced head and hand tracker for wearable computers and portable vr. In *Wearable Computers, The Fourth International Symposium on*, pages 155–162. IEEE, 2000.

[19] Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real time motion capture using a single time-of-flight camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 755–762. IEEE, 2010.

[20] Ken Hinckley, R Jacob, and Colin Ware. Input/output devices and interaction techniques, 2004.

[21] Hidekazu Hirayu, Takeo Ojika, and Ryugo Kijima. Constructing the historic villages of shirakawa-go in virtual reality. *MultiMedia, IEEE*, 7(2):61–64, 2000.

[22] Patrick Y Hwang. Inertial measurement unit with magnetometer for detecting stationarity, December 17 2002. US Patent 6,496,779.

[23] Wijnand IJsselsteijn, Yvonne de Kort, Antal Haans, et al. Is this my hand i see before me? the rubber hand illusion in reality, virtual reality, and mixed reality. *Presence*, 15(4):455–464, 2006.

[24] Sankar Jayaram, Uma Jayaram, Yong Wang, Hrishikesh Tirumali, Kevin Lyons, and Peter Hart. Vade: a virtual assembly design environment. *Computer Graphics and Applications, IEEE*, 19(6):44–50, 1999.

[25] Dongsik Jo, Yongwan Kim, Eunji Cho, Daehwan Kim, Ki-Hong Kim, and Gil-Haeng Lee. Tracking and interaction based on hybrid sensing for virtual environments. *ETRI Journal*, 35(2):356–359, 2013.

[26] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

[27] Hirokazu Kato, Mark Billinghurst, Ivan Poupyrev, Kenji Imamoto, and Keihachiro Tachibana. Virtual object manipulation on a table-top ar environment. In *Augmented Reality, 2000.(ISAR 2000). Proceedings. IEEE and ACM International Symposium on*, pages 111–119. Ieee, 2000.

[28] Roland Kehl and Luc Van Gool. Markerless tracking of complex human motions from multiple views. *Computer Vision and Image Understanding*, 104(2):190–209, 2006.

[29] Patrick A Kenney, Matthew F Wszolek, Justin J Gould, John A Libertino, and Alireza Moinzadeh. Face, content, and construct validity of dv-trainer, a novel virtual reality simulator for robotic surgery. *Urology*, 73(6):1288–1292, 2009.

[30] Steffen Knoop, Stefan Vacek, and Rüdiger Dillmann. Sensor fusion for 3d human body tracking with an articulated 3d body model. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1686–1691. IEEE, 2006.

[31] Frank-Lothar Krause, Martin Göbel, G Wesche, and T Biahmou. A three-stage conceptual design process using virtual environments. 2004.

[32] Nurakhmed Nurislamovich Latypov and Nurulla Nurislamovich Latypov. Method for tracking and displaying user's spatial position and

orientation, a method for representing virtual reality for a user, and systems of embodiment of such methods, December 21 1999. US Patent 6,005,548.

[33] Bastian Leibe, Thad Starner, William Ribarsky, Zachary Wartell, David Krum, Brad Singletary, and Larry Hodges. The perceptive workbench: Toward spontaneous and natural interaction in semi-immersive virtual environments. In *Virtual Reality, 2000. Proceedings. IEEE*, pages 13–20. IEEE, 2000.

[34] Ajey Lele. Virtual reality and its military utility. *Journal of Ambient Intelligence and Humanized Computing*, 4(1):17–26, 2013.

[35] Jing-Rong Li, Li Pheng Khoo, and Shu Beng Tor. Desktop virtual reality for maintenance training: an object oriented prototype system (v-realism). *Computers in Industry*, 52(2):109–125, 2003.

[36] D Magee, Y Zhu, Rish Ratnalingam, P Gardner, and David Kessel. An augmented reality simulator for ultrasound guided needle placement training. *Medical & biological engineering & computing*, 45(10):957–967, 2007.

[37] Fabrizia Mantovani, Gianluca Castelnuovo, Andrea Gaggioli, and Giuseppe Riva. Virtual reality training for health-care professionals. *CyberPsychology & Behavior*, 6(4):389–395, 2003.

[38] Mark Mine et al. Virtual environment interaction techniques. *UNC Chapel Hill computer science technical report TR95-018*, pages 507248–2, 1995.

[39] Anders K Møller, Pablo F Hoffmann, Marcello Carrozzino, Claudia Faita, Giovanni Avveduto, Franco Tecchia, Flemming Christensen, Dorte Hammershøi, and TeCIP-Scuola Superiore S Anna. Joint evaluation of communication quality and user experience in an audio-visual virtual reality meeting. In *4th International Workshop on Perceptual Quality of Systems*, pages 151–157, 2013.

[40] Melvin M Morrison. Inertial measurement unit, December 8 1987. US Patent 4,711,125.

[41] Zakiah Noh and Mohd Shahrizal Sunar. A review on 3d reconstruction for augmented reality in virtual heritage system.

[42] Renaud Ott, Daniel Thalmann, and Frédéric Vexo. Haptic feedback in mixed-reality environment. *The Visual Computer*, 23(9-11):843–849, 2007.

[43] Pieter Pauwels, Ruben Verstraeten, Ronald De Meyer, and Jan Van Campenhout. Architectural information modelling for virtual heritage application. In *Digital Heritage–Proceedings of the 14th International Conference on Virtual Systems and Multimedia*, pages 18–23, 2008.

[44] Miguel Ribo, Axel Pinz, and Anton L Fuhrmann. A new optical tracking system for virtual and augmented reality applications. In *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*, volume 3, pages 1932–1936. IEEE, 2001.

[45] Richards-Tech. Rtimulib - a versatile c++ and python 9-dof, 10-dof and 11-dof imu library. `https://github.com/richards-tech/RTIMULib`, 2015. [Online; accessed 2015-06-21].

[46] Richard M Satava. Virtual reality and telepresence for military medicine. *Computers in biology and medicine*, 25(2):229–236, 1995.

[47] M Shapiro. Comparing user experience in a panoramic hmd vs. projection wall virtual reality system. Technical report, tech. rep., Sensics, Inc, 2006.

[48] Gilles Simon, Andrew W Fitzgibbon, and Andrew Zisserman. Markerless tracking using planar structures in the scene. In *Augmented Reality, 2000.(ISAR 2000). Proceedings. IEEE and ACM International Symposium on*, pages 120–128. IEEE, 2000.

[49] M. Slater, A. Frisoli, F. Tecchia, C. Guger, B. Lotto, A. Steed, G. Pfurtscheller, R. Leeb, M. Reiner, M.V. Sanchez-Vives, P. Verschure, and U. Bernardet. Understanding and realizing presence in the presenccia project. *IEEE Computer Graphics and Applications*, 27(4):90–93, 2007. cited By 7.

[50] André Stork. *Effiziente 3D-Interaktions-und Visualisierungstechniken für benutzer-zentrierte Modellierungssysteme*. PhD thesis, TU Darmstadt, 2001.

[51] Franco Tecchia. A flexible framework for wide-spectrum vr development. *Presence*, 19(4):302–312, 2010.

[52] Franco Tecchia, Leila Alem, and Weidong Huang. 3d helping hands: a gesture based mr system for remote collaboration. In *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*, pages 323–328. ACM, 2012.

[53] Franco Tecchia, Giovanni Avveduto, Raffaello Brondi, Marcello Carrozzino, Massimo Bergamasco, and Leila Alem. I'm in vr!: using your own hands in a fully immersive mr system. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, pages 73–76. ACM, 2014.

[54] OAJ Van der Meijden and MP Schijven. The value of haptic feedback in conventional and robot-assisted minimal invasive surgery and virtual reality training: a current review. *Surgical endoscopy*, 23(6):1180–1190, 2009.

[55] Etienne Van Wyk and Ruth De Villiers. Virtual reality training applications for the mining industry. In *Proceedings of the 6th international conference on computer graphics, virtual reality, visualisation and interaction in Africa*, pages 53–63. ACM, 2009.

[56] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics*, 28(3), 2009.

[57] D Weidlich, L Cser, T Polzin, D Cristiano, and H Zickner. Virtual reality approaches for immersive design. *CIRP Annals-Manufacturing Technology*, 56(1):139–142, 2007.

[58] JD Westwood, HM Hoffman, D Stredney, and SJ Weghorst. Validation of virtual reality to teach and assess psychomotor skills in laparoscopic surgery: results from randomised controlled studies using the mist vr laparoscopic simulator. *Medicine Meets Virtual Reality: art, science, technology: healthcare and evolution*, page 124, 1998.

[59] Brenda K Wiederhold and Mark D Wiederhold. *Virtual reality therapy for anxiety disorders: Advances in evaluation and treatment.* American Psychological Association, 2005.

[60] Oliver J Woodman. An introduction to inertial navigation. *University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696*, 14:15, 2007.

# List of Figures

One of the main peculiarity of fully immersive virtual reality is the enhancing of the simple passive viewing of a virtual environment with the ability to manipulate virtual objects inside it. This Thesis project investigates such interfaces and metaphors for the interaction and the manipulation tasks. In particular, the research activity conducted allowed the design of a thimble-like interface that can be used to recognize in real-time the human hand's orientation and infer a simplified but effective model of the relative motion and gesture. Inside the virtual environment, users provided with the developed systems will be therefore able to operate with natural hand gestures in order to interact with the scene; for example, they could perform positioning task by moving, rotating and resizing existent objects, or create new ones from scratch.

–

Pisa, July 2015