**Navigation in Google Maps using Leap Motion and   
Google Maps API V3**

LICENSE THESIS

|  |  |  |
| --- | --- | --- |
|  | Graduate: | **Marius Cristian MĂNĂSTIREANU** |
|  |  |  |
|  | Supervisor: | **todo Adrian GROZA** |

**2014**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| DEAN, |  | | HEAD OF DEPARTMENT, | |
| **Prof. dr. eng. Liviu MICLEA** |  | **Prof. dr. eng. Rodica POTOLEA** | |

Graduate: **Marius Cristian MĂNĂSTIREANU**

**LICENSE THESIS TITLE**

1. **Project proposal:** *Short description of the license thesis and initial data*
2. **Project contents:** *(enumerate the main component parts) Presentation page, advisor's evaluation, title of chapter 1, title of chapter 2, …, title of chapter n, bibliography, appendices.*
3. **Place of documentation**: Technical University of Cluj-Napoca, Computer Science Department
4. **Consultants**:
5. **Date of issue of the proposal:** November 1, 2013
6. **Date of delivery:** June 28, 2014 (*the date when the document is* *submitted*)

|  |  |
| --- | --- |
| Graduate: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
| Supervisor: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Declaraţie pe proprie răspundere privind**

**autenticitatea lucrării de licenţă**

Subsemnatul(a)**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**, legitimat(ă) cu \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ seria \_\_\_\_\_\_\_ nr. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
CNP \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, autorul lucrării \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_elaborată în vederea susţinerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ a anului universitar \_\_\_\_\_\_\_\_\_\_, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

|  |  |  |
| --- | --- | --- |
| Data  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Nume, Prenume  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |
|  |  | Semnătura |

**De citit înainte** (această pagină se va elimina din versiunea finală):

1. Cele trei pagini anterioare (foaie de capăt, foaie sumar, declaraţie) se vor lista pe foi separate (nu faţă-verso), fiind incluse în lucrarea listată. Foaia de sumar (a doua) necesită semnătura absolventului, respectiv a coordonatorului. Pe declaraţie se trece data când se predă lucrarea la secretarii de comisie.
2. Pe foaia de capăt, se va trece corect titulatura cadrului didactic îndrumător, în engleză (consultaţi pagina de unde aţi descărcat acest document pentru lista cadrelor didactice cu titulaturile lor).
3. Documentul curent a fost creat în **MS Office 2007.** Dacă folosiţi alte versiuni e posibil sa fie mici diferenţe de formatare, care se corectează (textul conţine descrieri privind fonturi, dimensiuni etc.).
4. **Cuprinsul** începe pe pagina nouă, impară (dacă se face listare faţă-verso), prima pagina din capitolul **Introducere** tot aşa, fiind numerotată cu 1. Pentru actualizarea cuprinsului, click dreapta pe cuprins (zona cuprinsului va apare cu gri), Update field->Update entire table.
5. Vizualizaţi (recomandabil şi în timpul editării) acest document după ce activaţi vizualizarea simbolurilor ascunse de formatare (apăsaţi simbolul **** din *Home/Paragraph*).
6. Fiecare capitol începe pe pagină nouă, datorită simbolului ascuns Section Break (Next Page) care este deja introdus la capitolul precedent. Dacă ştergeţi din greşeală simbolul, se reintroduce (*Page Layout -> Breaks*).
7. Folosiţi stilurile predefinite (Headings, Figure, Table, Normal, etc.)
8. Marginile la pagini nu se modifică (Office 2003 default).
9. Respectaţi restul instrucţiunilor din fiecare capitol.

Table of Contents

[Chapter 1. Introduction 1](#_Toc396945675)

[1.1. Project context 1](#_Toc396945676)

[1.2. Motivation 2](#_Toc396945677)

[1.3. Paper structure 2](#_Toc396945678)

[Chapter 2. Project Objectives 4](#_Toc396945679)

[2.1. Project specifications 4](#_Toc396945680)

[2.2. Requirements 4](#_Toc396945681)

[2.3. Functional requirements 4](#_Toc396945682)

[2.4. Non-functional requirements 4](#_Toc396945683)

[2.5. Stakeholders? 4](#_Toc396945684)

[2.6. Project goal 4](#_Toc396945685)

[Chapter 3. Bibliographic Research 5](#_Toc396945686)

[3.1. Related work 5](#_Toc396945687)

[3.1.1. Leap Motion and Google Earth 5](#_Toc396945688)

[3.1.2. Leap Motion and Hyperlapse (by Teehan+Lax Labs) 5](#_Toc396945689)

[3.1.3. Conclusions 6](#_Toc396945690)

[3.2. Leap Motion controller 6](#_Toc396945691)

[3.2.1. Leap Motion hardware architecture 7](#_Toc396945692)

[3.2.2. Leap Motion system architecture 8](#_Toc396945693)

[3.3. Other gesture recognition sensors and devices 9](#_Toc396945694)

[3.3.1. Wii (by Nintendo) 9](#_Toc396945695)

[3.3.2. Kinect (by Microsoft) 9](#_Toc396945696)

[3.3.3. Xtion PRO Live (by Asus) 10](#_Toc396945697)

[3.3.4. PrimeSense Carmine 10](#_Toc396945698)

[3.3.5. RealSense (by Intel) 11](#_Toc396945699)

[3.3.6. PointTouch (by PointGrab) 11](#_Toc396945700)

[3.3.7. Elliptic Labs 12](#_Toc396945701)

[3.3.8. Myo (by Thalmic Labs) 12](#_Toc396945702)

[3.3.9. Nimble (by Intugine) 13](#_Toc396945703)

[3.3.10. Conclusions 13](#_Toc396945704)

[Chapter 4. Analysis and Theoretical Foundation 14](#_Toc396945705)

[Chapter 5. Detailed Design and Implementation 15](#_Toc396945706)

[Chapter 6. Testing and Validation 16](#_Toc396945707)

[6.1. Leap Motion precision and reliability 16](#_Toc396945708)

[6.1.1. Spatial positioning 16](#_Toc396945709)

[6.1.2. Gesture recognition 17](#_Toc396945710)

[6.2. Performance 21](#_Toc396945711)

[6.2.1. Library size 21](#_Toc396945712)

[6.2.2. Frames per Second 21](#_Toc396945713)

[Chapter 7. User’s manual 24](#_Toc396945714)

[Chapter 8. Conclusions 25](#_Toc396945715)

[Chapter 9. Bibliography 26](#_Toc396945716)

[Appendix 1 (only if needed) 28](#_Toc396945717)

# Introduction

In this chapter I will present you the project context of the license thesis together with the motivation for choosing this area of study, followed by a specification of the precise domain of the license thesis.

## Project context

When talking about future in technology often we have a vision, similar with the science fiction movies, of supercomputers controlled by sweeping and swaying the hands in the air. The reality is not that far actually; current trends in technology have pointed towards a world where humans and machines interact in such fashion: screens can now recognize faces and fingerprints, the glasses are not what used to be thanks to Google Glasses and there exists various devices that are able to interpret your body movement and transpose it in the virtual world. At this moment, the latest user interfaces are dominated by touch, but as we can see, things are changing fast, other technologies appear, and the future is pointing us in new directions.

This emerging trend that is mentioned here can be defined as gesture recognition. At this moment there exists applications in this domain which use the hands and fingers movement in order to control the machines; most of the application are present in the consumer electronic devices area, where humans can change the channel or adjust the volume of their smart TVs with just one simple move in the air, or they can swipe between pages of one recipe on their tablet while cooking and their hands are too busy and dirty to touch the device. Another field of interest where this kind of applications started to develop and emerge is the healthcare domain; there exists on the market software for surgeons that use the Microsoft’s Kinect controller in order to be able to manipulate digital images needed in the time of the surgeries while maintaining the environment clean and sterile.

As stated before, most of the applications at this time, are developed for the home consumer electronic devices and gaming industry, but soon it will extend to other fields of interest, such automotive, automation or healthcare area. The age of computer classical mice and their pads might soon come to an end. With gesture-control enabled by devices like the Leap Motion, Xbox Kinect and Myo fast emerging, and touchscreens taking over just about everything else, the spot next to the keyboard might soon be empty.

Gesture recognition is an emerging technology that has the potential to revolutionize the way humans interact with machines; not only in gaming and automotive applications but also in day-to-day activities on the home front. Standalone devices like the Leap Motion controller will spread the awareness of this new revolutionary human machine interaction technology.

Even if the future of this technology is a bit hard to predict, because as with any other technology, it is constantly changing and depends also on other factors, a market report written by a market research consulting company from the United States of America [1] states that the total gesture recognition and touch-less sensing market is expected to reach $22.04 billion by 2020 at a double digit CAGR[[1]](#footnote-1) from 2013 till 2020, having in mind that the market value in 2012 was approximately $2.2 billion (Figure 1.1). Therefore a growth of approximately 10 times is expected for the market of gesture recognition and touch-less sensing devices in a period of 8 years.

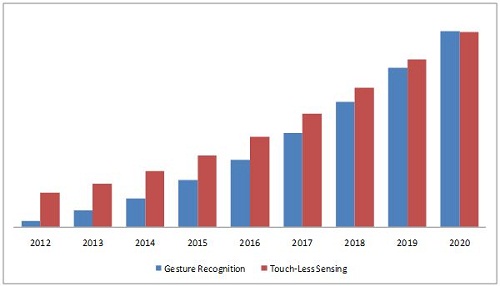


Figure 1.1 Gesture recognition and touch-less sensing market research (2012-2020)

## Motivation

What motives my decision to focus my research in the gesture recognition domain is the fact that this field of study was not explored much before, since this technologies are quite new on the market, leaving space to a lot of unknown factors.

The gesture recognition is an emerging technology that has the potential to revolutionize the way humans interact with the machines. The possibility to work with such a device that gives total freedom of move to the user and removes any other dependency (such as a mouse or a keyboard) is priceless.

The main speaker of a Los Angeles based web design conference focused on the future of interaction design, Christopher Noessel, mentioned during his presentation: “I think that the Leap Motion millimeter-wave gesture recognition is going to be super promising. Now that we can read fingers it’s going to get a lot more interesting as far as gesture recognition, and we’ll be able to maybe even sort of meet sci-fi at what sci-fi has been promising for that sort of thing” [2].

My work will be focused on the technology that the Leap Motion controller offers and together with the API provided by Google, for their products, Google Maps and Street View in order to develop an application that aims to enhance the user experience of browsing locations in Google Maps.

## Paper structure

The paper is structured under eight chapters and aims to inform the reader and to provide him a full understanding of the project context, objectives, and related work in this field of study, how the solution was designed, built and tested and finally how it can be used.

*Chapter 1* identifies the project context as well as the domain of study together with the motivation of this choice.

*Chapter 2 – todo*

*Chapter 3* presents the bibliographic research that I did in the gesture recognition domain, what are the related works in this field (both from the software and hardware point of view), what the Leap Motion controller can do, how it is built and how it works.

*Chapter 4 – todo*

*Chapter 5 – todo*

*Chapter 6* presents an analysis on Leap Motion’s accuracy, precision and reliability in both static and dynamic scenarios. An overview on how precise and accurate are the measurements in terms of special positioning and gesture recognition is presenter.

*Chapter 7 – todo*

*Chapter 8 – todo*

# Project Objectives

## Project specifications

## Requirements

## Functional requirements

## Non-functional requirements

## Stakeholders?

## Project goal

The project theme must be described in this chapter (as a research/design proposal, clearly formulated, with clear objectives 2-3 pages, and some explanatory figures).

Should represent about 10% of the paper. 6pag

# Bibliographic Research

In this chapter I will present my research related to what other similar solutions do exists related to my work at this moment on the market, I will present the way the Leap Motion Controller is build and how it works and finally I will present my research results in the gesture recognition devices and sensors field (trying to accentuate what other devices except the Leap Motion are present on the market).

## Related work

This section of the thesis briefly reviews related work in the areas directly associated with Leap Motion Controller and Google Maps API. Finally, conclusions will be drawn in order to accentuate the pluses and the contribution of this thesis in the fields mentioned above.

### Leap Motion and Google Earth

One of the most known application that is available in this field is the *Google Earth* plugin developed by Google themselves. The plugins allows the user to travel the world through a virtual glove and view satellite imagery, maps, terrain and 3D buildings [3]. Even though the solution provided by Google is quite unique and comes with a lot of features, the plugin comes with some downsides and some flaws.

From the beginning we can see that the plugin is way too sensitive to use. Trying to control the application out of the box can be a disaster since manipulating the space can be quite difficult. The earth can get into a state when is constantly spinning, the motion speed while using the plugin is varying too much because of the high sensibility and sometimes the movements are too large, fast or out of control. Another downside of Google Earth plugin is that you need to have installed Google Earth (which is a software that requires relatively high system settings, such as: 2GB+ free hard disk space, a network speed of 768 Kbits/sec and so on [4]).

### Leap Motion and Hyperlapse (by Teehan+Lax Labs)

The developers from Teehan+Lax Labs [5] have come up with a solution that is able to create hyper-lapse[[2]](#footnote-2) videos from Google Street View panoramas. Because of the fact that the original code of the hyperlapse.js is available on GitHub I investigated their solution: two locations must be defined, using Google Maps’ public Routing API a route is being computed between that two locations and all the panoramas available between this two locations are downloaded and cached into the browser memory and then stitched up together using GSVPano.js. After all the images are downloaded, using Three.js the movement between frames is being implemented.

The only big problem here is that this innovative library for Google Maps Street View, Hyperlapse.js, **breaks one of the Google Maps’ terms of agreement**, which is: “you must not use the Products in a manner that gives you or any other person access to mass downloads or bulk feeds of any Content” (more information can be found under 2(d) in the Google Maps/Earth Terms of Service [6]).

The app is entirely written in Javascript using LeapJS and a modified version of Hyperlapse.js [7] and is not public. There exists only an online video on YouTube platform demoing the application.

### Conclusions

As we can see from the presented related work that exists at this moment on the market (August 2014), there exists only two applications in this domain, which have involved both the Leap Motion controller and the Google Maps API, one of which is not public (and also breaks Google’s Terms of Service) and another one which is similar, but intended to use with another product – not Google Maps, but Google Earth, which requires some software dependencies such as the application itself.

Therefore, we can see there is a niche that can be explored. The presence of the flaws and lack of portability of the Google Earth solution and no plugins available for the actual Google Maps or Street View gives me the opportunity to develop an application that will have human interaction without the necessary use of the keyboard or mouse, through the support given by the Leap Motion controller.

## Leap Motion controller

The *Leap Motion* *controller* is a computer hardware 3D sensor device that supports hand and finger motions as input, but it does not require any hand contact leaving the hand navigate free into the air. The *Leap Motion Controller* senses how you naturally move your hands and lets you use your computer in a whole new revolutionary way. The *Leap Motion controller* (Figure 3.1) is sleek, light (45 grams) and has relatively small dimensions, being 80 mm long, 12.7 mm tall and having a width of 30 mm [8].

*The Leap Motion Controller* represents a revolutionary input device for gesture-based human-computer interaction. It allows for the precise and fluid tracking of multiple hands, fingers, and small objects in free space with sub-millimeter accuracy.

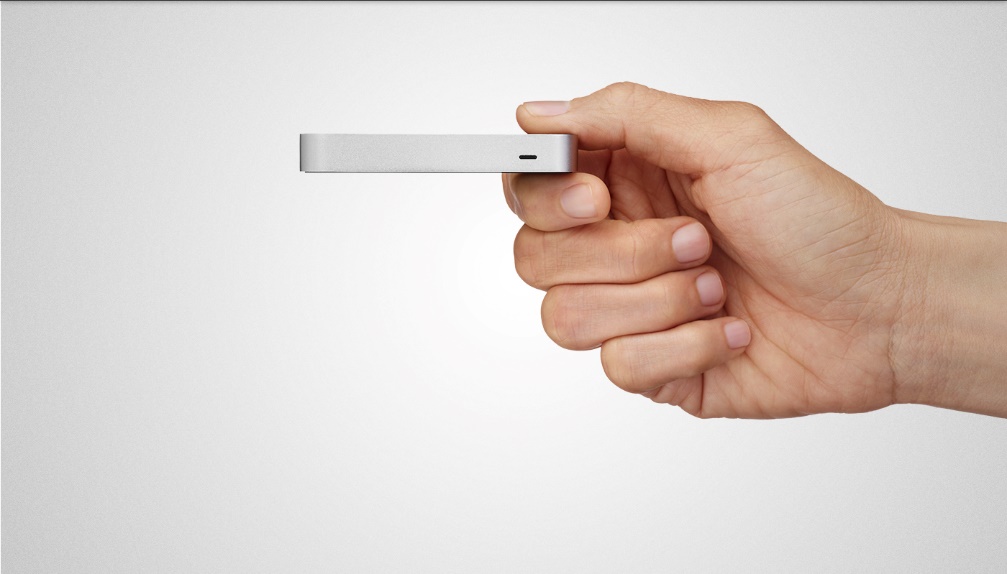


Figure 3.1 The *Leap Motion* *controller*

### Leap Motion hardware architecture

The *Leap Motion* *controller* is an input peripheral device with an USB connector. The device’s architecture is rather simple, it uses two monochromatic IR cameras and three infrared LEDs (see Figure 3.2). Therefore, the controller can be categorized as an optical tracking system based on the stereo vision principle. The Leap software analyzes the objects observed in the device’s field of view. It recognizes hands, fingers, and tools, reporting discrete positions, gestures, and motion.

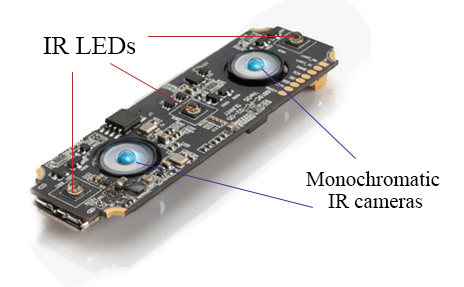


Figure 3.2 Leap Motion hardware architecture

The *Leap Motion controller* scans a region in the shape of an inverted pyramid centered at the device’s center and extending upwards – this space where the hands can be detected is also called the interaction box (see Figure 3.3). The effective range of the controller extends from approximately 25 to 600 millimeters above the device.



Figure 3.3 *Leap Motion’*s field of view

### Leap Motion system architecture

The Leap Motion software runs as a service. The software connects the input device (Leap Motion controller) over the USB hub with the computer. The Leap Motion service receives motion tracking data from the hardware device. Over that, the SDK provides two varieties of API for getting the Leap Motion data:

* A native interface (a dynamic library that can be used to create new Leap-enabled applications – the library can be linked in C++, ObjectiveC, Java, C# or Python)
* A WebSocket interface (together with the JavaScript client library LeapJS allows the creation of Leap-enabled web applications)

Because of the nature of my application proposal, I will focus on the WebSocket interface and try to explain how it works.

The Leap Motion service runs a WebSocket server on the localhost domain at the port 6437. The WebSocket interface provides motion data (captured from the Leap Motion controller) in the form of JSON messages. A JavaScript client library, as stated before, is available and can be used in order to consume the JSON messages and present the tracking data as regular JavaScript objects.

This interface is intended to be used in web applications, but it can be accessed through any kind of application that can establish a WebSocket connection.

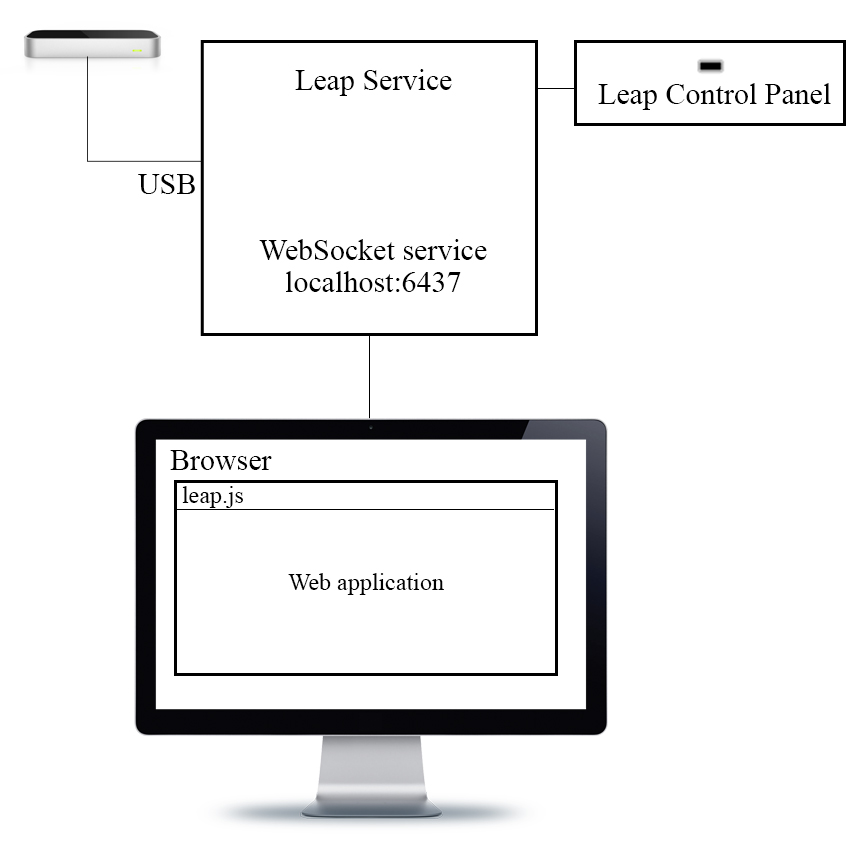


Figure 3.4 Leap Motion WebSocket interface

The Leap Motion service provides a WebSocket server listener at localhost:6437 address. The server sends tracking data as JSONs messages while it receives data from the hardware device. The leap.js client JavaScript library it is designed in such way to establishes the connection to the WebSocket server and to consume the JSON messages; the library should be included and used in web applications [9].

## Other gesture recognition sensors and devices

The gesture recognition is an emerging technology that has the potential to revolutionize the way humans interact with the machines. Together with the evolution of touchless sensing industry, the demand of the market in the past years for gesture recognition devices and sensors has increased and therefore we can find a lot of devices on the market in different shapes and sizes.

In this sub-chapter I will enumerate and briefly describe what other important 3D sensors devices are available on the market, beside the Leap Motion controller.

### Wii (by Nintendo)

In the domain of gesture recognition, one of the first accurate and commercially viable solutions was the Nintendo Wii controller. The Wii controller, released in 2006, looks like a TV remote and was designed as a game controller. The primary control of the movement is the controller itself, containing solid-state accelerometers that let it sense full 3D gestures patters, such as:

* Tilting and rotation up, down, left and right
* Rotating along the main axis (as with a screwdriver)
* Acceleration up, down, left and right
* Acceleration towards the screen and away [10]

Because of the fact that the WiiMote can operate as a separate device, it was used in many applications that worked with a computer.

Compared with the Leap Motion Controller, the WiiMote can detect hand gestures in the air field, but it has no information about the fingers position, alignment and gestures.

### Kinect (by Microsoft)

The Kinect (see Figure 3.5), developed by Microsoft and released in 2010, comes as an add-on sensor for the Xbox 360 gaming console. The controller has visual and auditory (voice recognition) inputs and includes a 3D depth-sensing camera which gives the opportunity to the developers to incorporate, acquire and recognize full body gestures of multiple users at a time. Moreover, having an open SDK allows the developers to integrate this controller in other fields also, allowing it to detect other objects than human bodies.



Figure 3.5 Microsoft Kinect

Gathering massive amounts of data from motion-capture in real-life scenarios, the Kinect developers processed that data using a machine-learning algorithms and they were able to map the data to models representing people of different ages, body types, genders and clothing. With select data, developers were able to teach the system to classify the skeletal movements of each model, emphasizing the joints and distances between those joints [11].

However, in comparison with the Wii controller, the Microsoft Kinect allows the full body gesture recognition, instead of only hand movement recognition, but compared with the Leap Motion controller it has no precision at the level of fingers.

### Xtion PRO Live (by Asus)

The Xtion PRO Live is an alternative to Microsoft’s Kinect. It is capable of registering:

* gesture recognition by tracking people’s hand motions, having some predefined poses such as: push, click, circle, wave,
* (whole) body recognition,
* RGB detection: the controller is able to capture full RGB colored images allowing developers to design security systems, digital signage and many more,
* audio signals, allowing for voice recognition, video conference (together with the RGB detection) and so on [12].

In comparison with Microsoft’s product, Kinect, the Xtion Live has no motor, therefore the user should manually position the sensor, the controller is also a less popular device and therefore a lack of applications available on the market can be noticed.

### PrimeSense Carmine

PrimeSense Carmine is an alternative for the Xtion Live (by Asus), but manufactured under a different brand name. The only difference at some point was that the PrimeSense Carmine had a better USB 3.0 compatibility, but the problem was solved with a firmware update for the Asus device, making both devices equivalent [13].

### RealSense (by Intel)

Similar with the Microsoft’s Kinect, Asus’ Xtion Live and PrimeSense Carmine the RealSense technology developed by Intel is based on the depth sensors. The camera provides to the PCs and tablets a 3D vision, while the voice technologies incorporated can send audio information to the affiliated devices.

The device can:

* analyze facial images (by tracking multiple faces, identifying the eyes, mouth and nose),
* detect and track hands and fingers (up to 10 simultaneous fingers, and 8 gestures predefined, such as thumbs up, dynamic waving of the hand and so on, but it also has the possibility to process raw depth data, similar with the Leap Motion controller and SDK),
* recognize speech
* subtract the background (from the user’s body, for instance),
* track objects within the camera range and draw CG images on real world scenarios [14].

### PointTouch (by PointGrab)

PointTouch, by PointGrab, is designed for consumer electronic devices (such as personal computers, tablets, TVs and others). The device will create a virtual touch space between the devices and the user, so they can point directly at anything on the screen in order to access it. The interesting part of the technology is that PointTouch does not rely only on the hand position, but it computes the line of sight based on a highly accurate 3D finger and face position instead (see Picture 3.6) [15].



Figure 3.6 PointTouch eye-finger-device system

### Elliptic Labs

The engineers from Elliptic Labs developed a unique, innovative product that is able to deliver touchless gestures for multi-platform human interaction by using ultra sounds.

The technology behind it works in a similar way the bats are adapting to the environment. Ultrasounds signals are sent through the air from the speakers integrated in the devices, then they bounce against the user’s hands and fingers back to the device where are recorded by the microphones provided by the sensor.

The SDK provided by Elliptic Labs contains a few predefined hand and finger gestures, such as selection (screen tap), horizontal and vertical scroll (using swipes), drop down menus (using vertical swipes), rotate (using circle gestures) and trajectory control (using swipes).

Because of the fact that the technology involved in the gesture recognition process uses sound waves, it gives the opportunity to use the full 3D space available at 180° in front of the device (sensor) without any other restrictions. Using ultra sounds, the sensor’s size is really small, enabling easier integration in any devices (it can fit in your monitor or tablet frame). Another plus of this technology is that it uses a small fraction of power in comparison with other 3D recognition technologies like depth sensors, IR cameras and so on [16].

### Myo (by Thalmic Labs)

The Myo, developed by Thalmic Labs, is a revolutionary armband device that lets you use the electrical activity in your muscles to wirelessly control your computer, phone, and other favorite digital technologies (see Figure 3.7).



Figure 3.7 The Myo armband

Using EMG sensors[[3]](#footnote-3), the Myo armband measures the electrical activity from the user’s muscles in order to detect what gestures he is making. Having a relatively low price (150$ - August 2014), the Myo armband can revolutionize the way people can interact with the digital world, leaving them full movement capability (the device’s area of interaction is only restricted by the range of the Bluetooth emitter, which can be up to 100 meters), but also can revolutionize the life of impaired people helping, not necessarily to reproduce the movement of the hand, but to transmit data to a computer, smartphone, or any other device that has bluetooth sensors, and help them interact with the digital world in a much more simpler way [17].

Unfortunately, the device, at this moment (August 2014), is only available for pre-ordering, the shipping beginning from September 2014.

### Nimble (by Intugine)

Nimble is a device, developed in India, which uses a ring and a sensor combination (see Figure 3.8) in order to detect gestures. Compared with the Leap Motion controller, the device requires the ring to be present on your hand, while using Leap Motion controller your hand is completely free, but limited to the interaction box available; the Nimble sensor can detect interaction up to 4.5 meters away from the device.



Figure 3.8 Intugine Nimble sensor and ring

The intent is to integrate the Nimble device not only with the computers, but also with smart TVs, home automation devices and game consoles. We can say that Nimble effectively can replace not only the mouse and the keyboard but also the remote controllers or gaming consoles.

At this moment, the Nimble controller is only available for pre-ordering and it will be released in the last quarter of 2014, therefore is not accessible at this moment on the market [18].

### Conclusions

The market and field of gesture recognition devices, sensors and software is increasing at this moment, as we speak. The gesture recognition is an emerging technology that has the potential to revolutionize the way humans interact with the machines (computer, tablets, smartphones, TVs and so on). As we can see from the previously presented devices, at this moment there are quite a few options in this field of technology, each of them providing similar solutions (what), but coming up with different approaches (how).

# Analysis and Theoretical Foundation

Together with the next chapter takes about 60% of the whole paper. ~15pag

The purpose of this chapter is to explain the operating principles of the implemented application.

Here you write about your solution from a theory standpoint – i.e. you explain it and demonstrate its theoretical properties/value, e.g.:

* used or proposed algorithms,
* used protocols,
* abstract models,
* logic explanations/arguments concerning the chosen solution,
* logic and functional structure of the application, etc.

YOU DO NOT write about implementation.

YOU DO NOT copy/paste info on technologies from various sources and others alike, which do not pertain to your project (no fillers, please!).

# Detailed Design and Implementation

Together with the previous chapter takes about 60% of the paper. ~15pag

The purpose of this chapter is to document the developed application such a way that it can be maintained and developed later. A reader should be able (from what you have written here) to identify the main functions of the application.

The chapter should contain (but not limited to):

* a general application sketch/scheme,
* a description of every component implemented, at module level,
* class diagrams, important classes and methods from key classes.

# Testing and Validation

In this chapter I will present an analysis about Leap Motion’s precision and reliability speaking in terms of spatial positioning (both static setups and dynamic setups) and gesture recognition accuracy, as well as an analysis about the performance of the library (I will discuss about the size of the project and frames per second).

## Leap Motion precision and reliability

### Spatial positioning

The study’s goals on which this subchapter is based was to analyze the precision and reliability of the *Leap Motion Controller* in static and dynamic conditions and to determine its suitability as an economically attractive finger/hand and object tracking sensor [19].

Two types of measurements were performed within the experiment, under two experimental conditions:

* Static conditions: acquisition of a limited number of static points in space (3,000 measurements were taken of 37 positions)
* Dynamic conditions: tracking of moving objects with constant inter-object distance within the calibrated space (119,360 measurements were taken in an attempt to cover the estimated useful sensory space of the controller).

The results of the study, in the *static scenario*, are presented in the 6.1 Table. As we can see, the standard deviation is less than 0.5 mm at all times, therefore we can say that the controller is quite precise and accurate in determining the static positioning in space.

Table 6.1 Results of static study

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Standard deviation | x Axis | y Axis | z Axis | Spatial position |
|  | position | (stdx) | (stdy) | (stdz) | (std) |
| Minimal  std | std(mm) | **0.0081** | 0.0093 | 0.015 | 0.013 |
| location(x, y, z) (cm) | (0, 30, 0) | (-10,-10,-5) | (0, 20, -5) | (0, 15, 0) |
| Maximal  std | std(mm) | 0.39 | **0.49** | 0.37 | 0.38 |
| location(x, y, z) (cm) | (-20, 20,0) | (-20, 30, 0) | (-20, 30, 0) | (-20, 30,0) |

The lowest standard deviation (0.0081 mm) was measured on the x axis at 30 cm above the controller, while the highest standard deviation (0.49 mm) was measured on the y axis at the leftmost and topmost positions [19]. In terms of spatial positioning, the results show a minimal standard deviation of 0.013mm (when the object is positioned right above the controller at 15mm height), while the maximum standard deviation of 0.38 mm was measured at 20cm on the left hand side of the controller and 30cm height.

The set of measurements in the *dynamic scenario* revealed that the accuracy of the controller drops when the objects move away from the sensor, therefore resulting in an inconsistent performance of the controller.

Previous similar tests were done using a robotic arm, both in static and dynamic scenarios, in order to measure the precision of the sensor [20]. The study from 2013 was focused on an early version of the Leap Motion Controller taking into account the so-called tremor (defined as an involuntary and approximately rhythmic movement of muscles). Depending on the human age, the tremor value can be in the range of 0.4mm ± 0.2mm (for young adults) to 1.1mm ± 0.6 mm (for older adults) [21]. The study from 2013 has revealed even bigger standard deviations, therefore this proves that the Leap software has improved from earlier phases by now, and even though this deviation occurs, the study has shown that comparable controllers in the same price range (e.g., the Microsoft Kinect) were not able to achieve the accuracy that the Leap Motion has [20].

From the studies presented above, we can draw the conclusion that the accuracy of the Leap Motion controller decreases when the objects are moving away from the sensor and when moving to the far left or right of the controller. However, having a relatively low standard deviation (less than 0.5 mm at all times [19]), the controller can be considered, in my opinion, a very good tool, with a high quality/price ratio, for personal use, but not as a professional tracking system.

### Gesture recognition

As we can see, the Leap Motion Controller has very good precision and accuracy, but when it comes to Gesture recognition, the Leap software has some flaws.

The Leap Motion software recognizes certain movement patterns as gestures which could indicate a user intent or command. The following movement patters are recognized by the Leap Motion software:

* *Circle* (a single finger tracing a circle),
* *Swipe* (A long, linear movement of a finger),
* *Key Tap* (A tapping movement by a finger as if tapping a keyboard key) and
* *Screen Tap* (a tapping movement by the finger as if tapping a vertical computer screen) [22].

The software lacks accuracy in detecting the right movement at the right time, but does not lack of accuracy and precision in terms of hand positioning. To illustrate this issue manual tests were done. I, as a user, I intended to perform 50 gestures of each type mentioned above, and I noted down the results that the Leap Motion has given and have built charts from it.

**NOTE**: The results can vary from one user to another. The Leap software does not have a learning algorithm implemented, but the user can learn the patterns that the Leap software recognizes and try to mimic them.

#### Circle gesture

A circle gesture is defined as a single finger tracing a circle. For this test, I intended to perform 50 circle gestures (both clockwise and counter-clockwise) positioning my hand in different parts of the interaction box above the controller. Out of 50 gestures performed, 45 were detected as *circles* and 5 were detected as *key taps* (see Figure 6.1).

Figure 6.1 Circle gesture

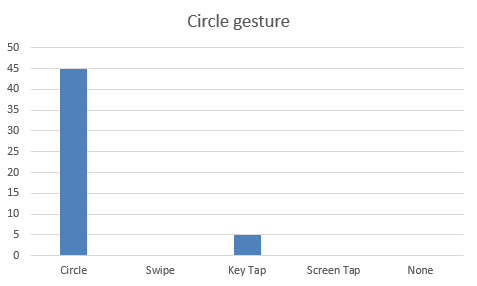


Figure 6.1 Circle gesture statistics

#### Swipe gesture

A swipe is defined as a long linear movement of a finger or hand. This gesture is highly used in mobile application industry (e.g. unlocking the screen, navigating through screens or photos, etc.).

Out of 50 gestures performed, only 28 were detected as swipe gestures, 10 were not detected at all, and the rest were detected as either key taps or circle gestures (see Figure 6.2).

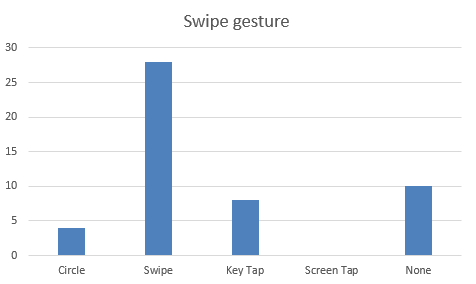


Figure 6.2 Swipe gesture statistics

#### Key Tap gesture

The Key Tap gesture is defined as a movement representing a finger tapping downwards and back upwards as if tapping a keyboard key. As we can see from the image below (Figure 6.3), the Key Tap gesture is the gesture which is the most easily detected by the Leap software, out of 50 gestures performed, 48 were registered as key taps and two gestures were not recognized.

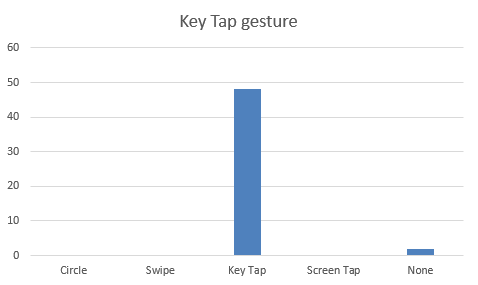


Figure 6.3 Key Tap gesture statistics

#### Screen Tap gesture

A tapping movement by the finger as if tapping a vertical computer screen (moving forward on the z axis and then backwards) describes a *Screen Tap* gesture.

I have to mention that it took me quite some time to learn the gesture, at the very beginning of the implementation the project, I had problems reproducing the gesture, most of the times the Leap software not recognizing gesture interaction while trying to perform a screen tap.

Therefore, in the Figure 6.4, we can notice that out of 50 gestures performed, 35 were detected as screen taps, 10 gestures were not detected, three of them were classified as being circle gestures and one as a key tap.

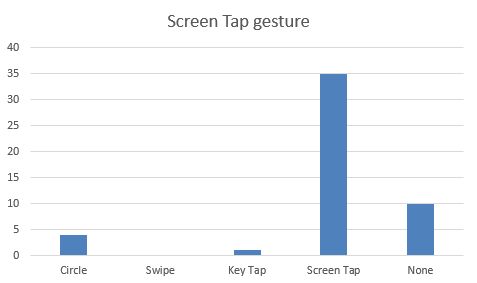


Figure 6.4 Screen Tap statistics

In conclusion, after I have performed the manual tests, summing it all up, in 78% of the cases the gesture was correctly detected and classified, 11% of times the gesture was misclassified and also 11% of the times the gesture was not detected at all (see Figure 6.5)

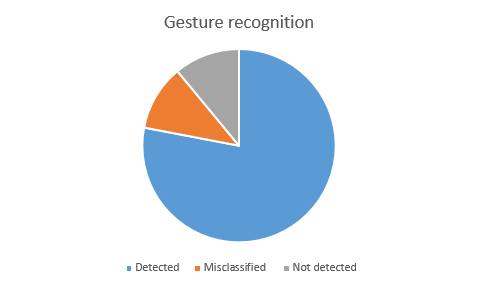


Figure 6.5 Gesture recognition statistics

I would say that the Leap Motion Controller is a very good option for detecting the spatial position in terms of 3D in the interaction box that is available above the controller itself, but not quite such a good option, at the moment when the document is written, for detecting movement patterns. Even if 78% is a quite high number, there still are approximately 2 times out of 10 when the controller, and implicitly the product developed will not act as it should. The lack of gesture recognition (11%) is not quite concerning from my point of view, but detecting other gesture than performed in 11% of the times is quite concerning because there could be other implementations available for different gestures and then this will result in an unexpected and unwanted behavior of the product developed.

The Leap Motion Controller’s lack of artificial intelligence involvement in the software (i.e., not being able to adapt with the user interaction and to learn its behavior) could force the developers to give up at all the gesture implementation and integration in the product that is being developed in order to avoid performance issues and unwanted behavior that might occur because of the lack of accuracy in detecting the right moves at the right time.

## Performance

### Library size

First of all I would like to discuss about the size of the project. The application was designed in such way to be included as a library over the Google Maps.

All the JavaScript files (my files together with the leap.js library) together with the CSS files sum up to 379 KB (-- update this if additional functionality will be added--). At the first sight you would say that this is an acceptable size, but in the case of web development the best practices recommend to minify the sources (JavaScript files, CSS files, HTML files) in order to obtain the smallest size possible. The smaller the size, the faster the load will be.

UglifyJS2[[4]](#footnote-4) was used to minify the sources by removing the spaces, comments, unused code, unreachable code, converting the if else statements into one line statements and so on; the final size of all the libraries together is 87 KB(-- update this if additional functionality will be added--).

Using a simple online download calculator we can see that at an average speed of 20 Mbps (which matches the average internet speed around the globe) the files will be downloaded under 1 second.

### Frames per Second

When using such an application the intent is to have smooth animations in the UI that are appealing and could be easily followed without disturbing the user’s attention. The transactions from one position to another on the map, as well as rotating the camera in Street View should be smooth and without any lag.

Because I could not find any automated tools that can provide an average FPS (frames per second) over a predefined period of time I used Google Chrome’s FPS tool and measured the activity over one minute of interaction with the application in both Google Maps and Street View, first without the library included (therefore using mouse and keyboard interaction), then with the library included (using the Leap Motion controller). During the monitoring phase of the FPS variance 20 samples were taken and graphs were plotted.

First test was done while navigating in Google Maps, as we can see from Figure 6.6, the values were quite close. We can see plotted with the blue line the FPS samples taken while using the mouse and keyboard as input devices, while with orange we can see the samples taken while interacting with the Leap Motion controller.

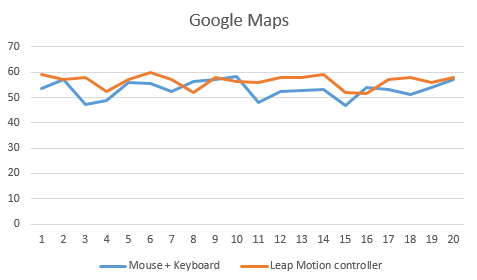


Figure 6.6 Google Maps FPS variance

Even if the values are quite similar and close to each other, we can notice a slightly improvement with the Leap Motion implementation. This can be justified by the fact that I used another implantation than Google uses for moving the map. Moreover, the WebSocket service through which the Leap Motion controller communicates with the browser sends information, in form of *frame* JSON messages, as often as every 10ms. The browser only repaints the page every 16ms (60Hz) at best. The LeapJS controller generates animation frames based of the browser’s *animation.loop*, therefore the “request” to render the page will not occur between other requests, generating non smooth transactions between frames, instead will “request” to render the page at the same time when the browser should.

The second test was done while navigating in Street View (see Figure 6.7). As in the previous graphic we can notice that the blue line represents the FPS variance while using the mouse and the keyboard, while with orange the values from the Leap Motion controller are plotted. In this case we can notice that the difference between the two cases presented is quite big.

Even if, from my point of view, rotating the camera through swipe gestures is not that accurate, smooth and therefore fun to use than while using the mouse, we can clearly see from this test that this technique implemented in my work that rotates the camera using the Leap Motion controller provides higher frames per second than the solution given by Google.

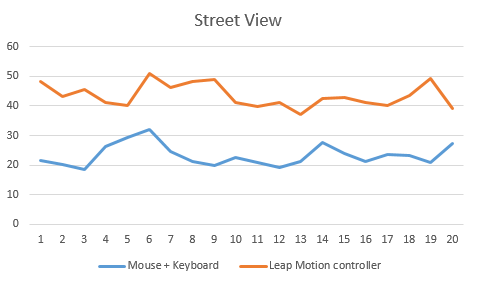


Figure 6.7 Street View FPS variance

Wrapping it up, we can clearly see from this two FPS tests that using the implementation provided in this project, we can obtain a slightly higher performance, from the frames per second points of view.

# User’s manual

**Todo: min requirements leapmotion.com/setup**

In the installation description section your should detail the hardware and software resources needed for installing and running the application, and a step by step description of how your application can be deployed/installed. An administrator should be able to perform the installation/deployment based on your instructions.

In the user manual section you describe how to use the application from the point of view of a user with no inside technical information; this should be adorned with screen shots and a stepwise explanation of the interaction. Based on user's manual, a person should be able to use your product

# Conclusions

About. 5% of the whole. 2-3pag

In this chapter you present:

* A summary of your contributions/achievements,
* A critical analysis of the results achieved,
* A description of the possibilities of improvements/further development.

# Bibliography

|  |  |
| --- | --- |
| [1] | Markets and Markets, "Gesture recognition & Touch-less sensing market (2013-2020)," Markets and Markets, Dallas, U.S., 2014. |
| [2] | L. Swenson, "The Future of Interaction Design With Christopher Noessel and Maggie Hendrie," 11 July 2013. [Online]. Available: http://www.mediacontour.com/the-future-of-interaction-design-with-christopher-noessel-and-maggie-hendrie/. [Accessed August 2014]. |
| [3] | Google, "Google Earth plugin for Leap Motion," Google, [Online]. Available: https://airspace.leapmotion.com/apps/google-earth/weblink. [Accessed August 2014]. |
| [4] | Google, "Google Earth System requirements," Google, [Online]. Available: https://support.google.com/earth/answer/20701?hl=en. [Accessed August 2014]. |
| [5] | Teehan+Lax Labs, "Teehan+Lax - Defining Experience," Teehan+Lax Labs, [Online]. Available: http://www.teehanlax.com/. [Accessed August 2014]. |
| [6] | Google, "Google Maps/Earth Terms of Service," Google, [Online]. Available: http://maps.google.com/help/terms\_maps.html. [Accessed August 2014]. |
| [7] | Teehan+Lax Labs, "Driving Google Street View with Leap Motion," Teehan+Lax Labs, [Online]. Available: http://youtu.be/1-lXnyFm\_Wc. [Accessed August 2014]. |
| [8] | Leap Motion, "Leap Motion: Technical Specifications," Leap Motion, 2014. [Online]. Available: https://www.leapmotion.com/product. |
| [9] | Leap Motion, "Leap Motion System architecture," Leap Motion, [Online]. Available: https://developer.leapmotion.com/documentation/skeletal/javascript/devguide/Leap\_Architecture.html. [Accessed August 2014]. |
| [10] | How Stuff Works, "How the Wii Works," How Stuff Works, [Online]. Available: http://electronics.howstuffworks.com/wii.htm. [Accessed August 2014]. |
| [11] | How Stuff Works, "How Microsoft Kinect Works," How Stuff Works, [Online]. Available: http://electronics.howstuffworks.com/microsoft-kinect.htm. [Accessed August 2014]. |
| [12] | Asus, "Asus Xtion PRO Live," Asus, [Online]. Available: http://www.asus.com/Multimedia/Xtion\_PRO\_LIVE/. [Accessed August 2014]. |
| [13] | iPiSoft, "Depth Sensors Comparison," iPiSoft, [Online]. Available: http://wiki.ipisoft.com/Depth\_Sensors\_Comparison. [Accessed August 2014]. |
| [14] | Intel, "Intel RealSense," Intel, [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html. [Accessed August 2014]. |
| [15] | PointGrab, "PointGrab," PointGrab, [Online]. Available: http://www.pointgrab.com/. [Accessed August 2014]. |
| [16] | Elliptic Labs, "Elliptic Labs," Elliptic Labs, [Online]. Available: http://www.ellipticlabs.com/. [Accessed August 2014]. |
| [17] | Thalmic Labs, "Myo armband," Thalmic Labs, [Online]. Available: https://www.thalmic.com/en/myo/. [Accessed August 2014]. |
| [18] | Intugine, "Intugine Nimble," Intugine, [Online]. Available: http://intugine.com/. [Accessed August 2014]. |
| [19] | J. Guna, G. Jakus, M. Pogačnik, S. Tomažič and J. Sodnik, "An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking," *Sensors,* no. 14, pp. 3702-3720, 2014. |
| [20] | F. Weichert, D. Bachmann, B. Rudak and D. Fisseler, "Analysis of the Accuracy and Robustness of the Leap Motion Controller," *Sensors,* no. 13, pp. 6380-6393, 2013. |
| [21] | M. M. Sturman, D. E. Vaillancourt and D. M. Corcos, "Effects of Aging on the Regularity of Physiological Tremor," *Journal of Neurophysiology,* vol. 93, pp. 3064-3074, 2005. |
| [22] | Leap Motion, "Leap Motion JavaScript API," Leap Motion, [Online]. Available: https://developer.leapmotion.com/documentation/skeletal/javascript/. [Accessed August 2014]. |
| [23] | Wikipedia, "Hyperlapse," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Hyperlapse. [Accessed August 2014]. |

# Appendix 1 (only if needed)

…

Relevant code sections

…

Other relevant info (proofs etc.)

…

Published papers (if any)

etc.

1. Compound annual growth rate (CAGR) is a business and investing specific term for the geometric progression ratio that provides a constant rate of return over the time period. [↑](#footnote-ref-1)
2. A hyper-lapse is an exposure technique in time-lapse photography, in which the position of the camera is being changed between each frame in order to create a tracking shot [23]. [↑](#footnote-ref-2)
3. Electromyography (EMG) is a technique for evaluating and recording the electrical activity produced by skeletal muscles. [↑](#footnote-ref-3)
4. UglifyJS2: <https://github.com/mishoo/UglifyJS2> [↑](#footnote-ref-4)