ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Dottorato di ricerca in INFORMATICA
Ciclo XXVII

Settore Concorsuale di afferenza: 01/B1
Settore Scientifico disciplinare: INF/01

# A new 3D modelling paradigm for discrete model

Presentata da: FLAVIO BERTINI

Coordinatore Dottorato:
PAOLO CIACCIA

Relatore:
GIULIO CASCIOLA

Corelatore:
SERENA MORIGI

Esame finale anno 2015

*Nature, that framed us of four elements*
*Warring within our breasts for regiment,*
*Doth teach us all to have aspiring minds.*
*Our souls, whose faculties can comprehend*
*The wondrous architecture of the world*
*And measure every wandering planet's course,*
*Still climbing after knowledge infinite,*
*And always moving as the restless spheres,*
*Will us to wear ourselves and never rest,*
*Until we reach the ripest fruit of all,*
*That perfect bliss and sole felicity,*
*The sweet fruition of an earthly crown.*

- Christopher Marlowe -

# Abstract

*Until few years ago, 3D modelling was a topic confined into a professional environment. Nowadays technological innovations, the 3D printer among all, have attracted novice users to this application field. This sudden breakthrough was not supported by adequate software solutions. The 3D editing tools currently available do not assist the non-expert user during the various stages of generation, interaction and manipulation of 3D virtual models. This is mainly due to the current paradigm that is largely supported by two-dimensional input/output devices and strongly affected by obvious geometrical constraints.*

*We have identified three main phases that characterize the creation and management of 3D virtual models. We investigated these directions evaluating and simplifying the classic editing techniques in order to propose more natural and intuitive tools in a pure 3D modelling environment. In particular, we focused on freehand sketch-based modelling to create 3D virtual models, interaction and navigation in a 3D modelling environment and advanced editing tools for free-form deformation and objects composition.*

*To pursuing these goals we wondered how new gesture-based interaction technologies can be successfully employed in a 3D modelling environments, how we could improve the depth perception and the interaction in 3D environments and which operations could be developed to simplify the classical virtual models editing paradigm. Our main aims were to propose a set of solutions with which a common user can realize an idea in a 3D virtual model, drawing in the air just as he would on paper. Moreover, we tried to use gestures and mid-air movements to explore and interact in 3D virtual environment, and we studied simple and effective 3D form transformations. The work was carried out adopting the discrete representation of the models, thanks to its intuitiveness, but especially because it is full of open challenges.*

*The 3D paradigm studied and proposed is not only a conceptual model, but it has been formalized and implemented, and in several cases the outcome was validated with common users. Our aspiration is to contribute to create a "testing ground" where study and develop new algorithms for 3D modelling.*

# Acknowledgements

*Writing acknowledgements you run two main risks, write the chapter zero of the dissertation or forget to thank somebody. Fortunately, a "parametric" sentence fills all the memory's gaps. Therefore, my first thanks goes to all those people who have given me advice, support and ideas during these three years.*

*First of all, I want to thank my family, for better or worse I have to thank them for what I am now and for what they allowed me to do. A special thank to Daniela for having unconditionally supported and encouraged me at any time, often with simple smile.*

*Let me thank my advisor, Giulio Casciola, who allowed me to follow my interests, suggesting the direction and encouraging my ideas and my work. A special thanks to Serena Morigi who involved me in different and interesting projects.*

*A big thank to my supervisor, Danilo Montesi, he taught me the craft made of humility, passion and professionalism. I hope to have reciprocated the confidence that he has always placed in me and in my work. I feel truly indebted with Maurizio Gabbrielli, for supporting me already before the beginning of the PhD. It was a pleasure to share with him the passion for music whenever it was possible.*

*I express my gratitude also to Martin Hachet. During my visit to Bordeaux he welcomed me in a fantastic group and opened my eyes toward new interesting research directions. I would like to thank the whole* POTIOC *team.*

# Contents

# List of Figures

# List of Tables

# 1 Introduction

*You can tell whether a man is clever by his answers.*
*You can tell whether a man is wise by his questions.*

Naghib Mahfuz

Nowadays computers and devices derived from them make the user able to perform multiple and heterogeneous actions. The available operations vary in all fields of the individual's life, for instance education, health, productivity, entertainment and so on. These technologies allow the user to accomplish a plenty of tasks easier and more efficiently than in the past. Obviously, these devices have been undergone to a considerable number of upgrade and now they require lower skills to perform the same tasks.

It is unquestionable that over the years, personal computers have become inherently more and more complex but at the same time easier to use in the eyes of the common user. The graphic interface is one of the major innovations that has characterized the evolutionary process in the mid-eighties of the last century. It has even contributed to the mass diffusion of personal computers. Literally opening a window on the processes carried out by the machine, the interfaces system has made the personal computer able to lose the aura of mere box calculator and it radically changed the interaction modalities.

As a result, we have witnessed the development of new technologies for input/output activities that allow the user to interact with the personal computer exploiting the new graphical paradigm. Again, the end users have certainly earned benefit from these kinds of innovations that have improved the user experience compared to the hardware/software complexity. The use of mouse, trackball, graphic tablets has quickly become familiar. The main motivation is that these kind of devices can be easily considered as the natural evolution of the pen-sheet system to which the user has long been accustomed.

However, the graphical interfaces and the interaction devices nowadays commonly used have an obvious limitation that affects the geometrical aspect of the paradigm. They are two-dimensional visualization and interaction systems. These systems easily allow to point, select, scroll and so on text document, web pages, images, but they require unintuitive and not natural actions when they are used to interact in 3D environments. For example, a non-expert user has to do an interpretative effort to transform the images displayed and the two-dimensional movements into a three-dimensional space. In particular, the reference is made to the 3D modelling environments used to generate three-dimensional virtual models, commonly used in different fields: from the rapid industrial prototyping to the entertainment industry, which includes both the film industry and the video games production. In this class of systems, the third dimension assumes a fundamental importance. It is useful to better understand the model, both within the context and in relation to other objects that may be present. Furthermore, it is essential in order to consistently modify the model with respect to the proportions and shape that the user wants to create.

Thus, can we consider the absence of the third dimension in the common display and interaction system as a real need? Up to now, it has still been possible to develop 3D virtual models using the old two-dimensional paradigm. Designers and expert modelers certainly do not feel such a need. Also the old mainframe's technicians or the first personal computer users, who used text strings to impart commands through the terminal without graphical interfaces, have perceived the need of a graphic paradigm. It's also true that in the recent years we have witnessed to the development and the diffusion of new technologies which are strongly characterized by the presence of the third dimension: 3D printers, 3D televisions, helmets and glasses for the augmented reality. In addition, an objective consideration can be done: up from birth the human being uses the hands to know the world, to interact with objects and to create forms. Why don't we put him in a condition to use gestures and manual capability to manage virtual models, exactly as he has always been accustomed to do?

A snapshot on the currently available technology would allow us to say that, a professional designer is nowadays able to complete the entire prototyping chain using hardware and software instrumentation familiar to his working environment. In brief, the prototyping chain that can be identified as the industry standard *de facto*, consists of three well-known activities: ac-

quisition, editing and printing of the object.
Through scanning, the user can acquire the physical objects to reconstruct the corresponding virtual model. This phase is commonly performed using sophisticated equipment (e.g. coordinate measuring machine, measuring arm systems, white light scanner, laser scanner) and only recently devices designed for other use have been proposed for the same purpose [42]. The second phase is identified with the modelling and the modification of virtual models, this activity is carried out using software tools that are often characterized by high complexity of use. At last, using a 3D printers the user can quickly realize a prototype of what he has modeled, choosing between different materials. He has the assurance to obtain a relevant copy of the virtual model. The main drawback is that these activities are often bounded within professional environments. In terms of comparison, it is exactly what happened with the analog photography before the mass diffusion of digital cameras, simple image editing tools (that allow to adjust few essential parameters, like brightness, contrast or size) and low cost photo printers: you had to go to the professional studio to transform your shot in a photograph on paper.

Would an user without specific experience be able to operate in each of the previous phases that characterize the generation and manipulation of a 3D virtual model? Are the hardware and software tools currently available sufficiently user-friendly to be used by a common user who wants to get closer to this technologies without professional aspirations? At the moment, the answers to these questions are quite negative. For example, the virtual model of an object can be modified only using unintuitive and complex tools, which often use, also for simple operations, key combinations to impart commands. However, several significant changes are taking place and they strongly affect the direction taken by these kind of technology. Returning to the example of the analog photography, that a rapid evolutionary process has lead to the spread of digital photography, a concrete prediction can be made. We can not exclude the possibility that in few years everyone will have available in their home a simple set of hardware and software solutions, thanks to which they will able to design, draw, manipulate and eventually print its virtual models. Obviously, the tools currently in use will suffer a simplification process in order to be used without any technical background. Furthermore, we can only guess some reasons that will bring the common users to approach this new type of activity: a father who decides to repair the child's toy by home 3D printing, producing just the broken part after it has been rebuilt with a simple acquisition system; a creative person who decides to decorate his house with few simple three-dimensional items created by his fancy; a gamer who decides to customize his avatar's appearance by adding or changing its details. We can also envisage new markets: nowadays when we go into a store, be it real or online, what we buy is a concrete physical object. Assuming we can buy a virtual model of the desired object, we can modify it afterwards, in order to adapt the shape to our needs, and easily print it at home. This new approach contains many of the costs related to the sale of physical objects, they would be reduced or even eliminated, optimizing the entire production process. Some recent innovations seem already go in this direction: websites that allow the manipulation of simple three-dimensional geometric shapes[1], hardware that enable the interaction through gestures and movements[2,3], online 3D printing services[4] and low-cost 3D printers.

In the recently years several technologies have been presented and the main introduced characteristic is to allow to interface with the computer in a really new way. They cover both the interaction actions carried out in the space near by the computer and the visualization mode of the contents, that is enriched with the depth perception. However, they are not fully integrated into the world of 3D modelling yet, especially hypothesizing a non-professional use as it has formerly been done. There are also a great number of 3D modelling software solutions characterized by functionality and different operative ways, but their use is still very complex and requires considerable experience in order to be used in full potential. Currently these issues hampers the diffusion of what we can identify as a new creative process. Non-professional users can not easily use tools that would allow them to produce and manage 3D virtual objects.

## 1.1 Research problem

Although the modelling tools offer many features and adopt different geometric approaches to create and edit the virtual model, typically only an experienced user is able to use them to their full potential. While they are unintuitive and quite complex in the hands of a common user.

In principle, a non-expert user who wants to create or manipulate the virtual model of an object, needs of a small number of functionalities that make it possible to create and modify the shape of the object as he will. First of all, the user would like to define a set of characteristic profiles of the object from which the system is able to reconstruct a well-defined shape. The target could be to generate more complex objects with respect to the basic shapes (e.g. cube, cylinder, cone, sphere), but still coherent with what the user has in

[1]http://www.123dapp.com
[2]http://www.wearfin.com
[3]http://www.thalmic.com
[4]http://i.materialise.com

mind. Instead, the manipulation requires that the user can modify the model both in a global mode and in punctual mode. In the first case, the operations allow to linearly modify the whole model structure, in the latter only some selected regions are affected by the changes. However, both these kind of transformations could be done using a simplified set of basic transformations (e.g. translation, rotation and scale). Finally, some simple functionalities could allow the user to quickly change the whole object shape, in the same way as he could do with the clay. These transformations could emulate the deformation principles related to the physics of materials. Obviously, all these features should be supported by simple and intuitive functionalities that allow to easily select and navigate inside the 3D environment. In addition, it would be desirable to integrate all these stuff into a 3D visualization environment. The new visualization mode could allow the user to reduce the stress that he usually makes to transform two-dimensional images, coming from the display, into a three-dimensional space.

In practice, the modelling tools currently available are very powerful if used by an experienced users, but are almost unusable for the common users. The shapes creation occurs using quite complex procedures and whether this does not happen the object are very simple (e.g. cube, cone, sphere, cylinder). Typically, a common user doesn't have high skills to modify these simple shapes into a complex virtual model. A simple example could be to transform a cone intersected by a sphere into a duck's head, it is not trivial work with current tools. It's not a simple task concretize an idea into a 3D shape, as it can be done drawing a silhouette with a pencil on a sheet. It's almost the same situation concerning the manipulation functions. For example, the common user does not have any deep knowledge about Bézier curves, spline surfaces, the concept of the control points or the Catmull-Clark subdivision surface algorithm. One more consideration regards the visualization mode, usually these modelling tools are used on personal computer equipped with a two-dimensional screen. They are not designed to offer high support for three-dimensional visualization and thus allow to more easily understand the proportions and relative positions among objects. Some simple visualization mode are usually offered like the possibility to change the camera position choosing between some predetermined perspectives (e.g. front, top and bottom), or a free navigation mode controlled with the mouse. However, all these methods require different interactions for have an overall view of the environment.

These set of issues make it difficult to think that currently 3D modelling software solutions may be used by every common user who want to grapple with the creation or the manipulation of virtual models. Instead, it is reasonable to think that many of the discussed problems can be dealt studying a new management paradigm for virtual models more oriented for the use by non-expert users. As a result, the work done during the doctorate was oriented to answer the following questions:

- Is it possible use some new interaction technology in a 3D modelling environment, although they are designed to interact through gestures? Can the depth perception be improved, providing functional environment that support the contents visualization? Is it possible obtain good results for these purposes using the hardware of a common personal computer, preserving the contact with reality?
- Which modelling operations can we design and introduce in this simplified paradigm, in order to allow anyone to easily create or edit a virtual model? What are the limits of the proposed solutions and the available technologies?

## 1.2 Proposed solution

In this dissertation we will present the study and the analysis that led us to propose new methodologies to operate in 3D modelling environment. Some solutions include and integrate acquisition, interaction and modification techniques that were already available, while others functionalities have been studied and adapted for the new use. Several new technologies have been still studied and deepened to verify if they could contribute positively to the purpose.

In particular, the paradigm takes in account all the phases that characterize the realization and the manipulation of a 3D virtual model:

- The concretization of an idea into a virtual model: a tool that allows you to draw in the air just as you would on paper.
- The navigation and the interaction in a 3D environment: how gestures and movements can be exploited to explore and operate on the virtual model, not only to trigger actions.
- The virtual model transformation: which operations we can integrate to modify the model, ensuring simple and effective functionalities.

In this list we exclude the 3D printing phase of the result, our purpose is to study new methodologies in order to produce a printable outcome.

The work was carried out using the discrete representation of the virtual model, called mesh, which vertices, edges and faces represents the discrete surface of the object. Afterwards, many of the presented solutions can easily be extended to the continuous representation, the mathematical surfaces of the virtual models. The choice of the discrete representation is mainly due to these reasons:

- The main purpose of the study was to evaluate a paradigm which can be intuitive in the eyes of the non-expert user. The discrete representation of the object surface is certainly more immediate and simpler than working with control points that indirectly modify the surface of the object. Even though the discrete model has more limitations than the continuous one.
- The modelling with discrete models is much more recent than the classical techniques, thus developing new solutions is an interesting challenge.

The paradigm studied is not only a conceptual model. It has been deepened and developed in each parts and it has been mathematically formalized where necessary. In that way we have the aspiration to provide a groundwork to experiment and develop new algorithms or adapt the existing ones, with the aim to improve the 3D modelling environments for non-expert users.

## 1.3 Structure of the dissertation

This dissertation is structured in three parts: sketch-based surface modelling, navigation and interaction in a 3D modelling environment and advanced modelling operations for discrete model. For each of them we dedicate a chapter structured as follow: an overview, the state of the art, the problem and the proposed solution accompanied with some final considerations.

The work presented in [4] has been deepened and further developed, with particular interest in the use of the *SmartPen* for the 3D free-hand sketching. As a result, the Chapter 2 is dedicated to the conversion of an idea into a 3D virtual model.

In the Chapter 3 we explore the navigation and the interaction in 3D environments, with particular interest in the study of features dedicated to interaction driven by movements and gestures in a 3D modelling environment. The matter is presented discussing the available options and the main critical points. At last, we present a solution describing the use of the LEAP[5] device as an advanced interaction device.

The modelling and manipulation operations are discussed in Chapter 4. In particular, we focused on two main topics: the first concerns the deformation

[5]https://www.leapmotion.com

of the discrete models. We present a model for detail preserving surface-based deformation of a body based on total curvature energy. The second topic regards the fusion of discrete models using Boolean operations. In that case, the target is to obtain a coherent and formally correct result that it can be used as input for further manipulated and refined operations.

The Chapter 5 is the final one, where we summarize the work done. Starting with an overall assessment, we will give an objective evaluation on the real effectiveness offered by the proposed solutions.

All the original contributions of the thesis are under evaluation or have been already published. In particular the work presented in Chapter 2 has been submitted in [69], while the work on the deformation presented in Chapters 4 has been published in [8].

# 2

# Sketch-Based Modelling

*The artist was not a special kind of man,*
*but every man a special kind of artist.*

Ananda K. Coomaraswami

In general, the simplest way to create a shape is to copy an existing one, for example using some simple device like pantograph. In that case the only thing to do is to follow the characteristic profiles of the original object. A little more complex task is the realization of a new shape. To achieve good results you should have a clear idea in mind and a good perception of proportions.
This is a very general argumentation that can be valid both in 2D, if you want to draw on paper, and in 3D, if your final goal is to create physical objects. The problem is amplified if you decide to develop a hardware/software platform able to allow you to sketch in the space. Several mathematical and topological issues are triggered and they must be solved.
We dealt the problem, verifying if the *SmartPen* and the Fast Interactive Reverse Engineering System (FIRES) [4], previously developed for reverse engineering tasks, could be further developed and used for a sketch-based modelling platform.

## 2.1 Overview

The massive diffusion of small sensors, able to provide heterogeneous information, has allowed the development of innovative interaction systems. Even though the principally efforts are oriented towards the development of interaction systems that regard the user and his environment, several interesting innovations affect the interaction between the user and the personal computer.

In the field of computer-aided design (CAD) the reverse engineering is one of the main examples of “create by copy” method. It has become an efficient process to create the virtual model of a physical object and it has benefited of these new sensor-based technologies. The measurement of the real object and the reconstruction, that transform the 3D point cloud in a 3D virtual object, usually constitute the pipeline of a reverse engineering system. Despite these two phases can cohabit inside the same reverse engineering system, typically this does not happen. The measurement step is performed using hardware equipment (e.g. coordinate measuring machine, measuring arm systems, white light scanner, laser scanner) that can be interfaced with any general purpose software for the reconstruction step. This is obviously an advantage, but the hardware equipment are sensible and quite expensive and the software tools are complex and suitable for a professional use. These characteristics hinder a spread diffusion of the classic reverse engineering platforms.

These needs have driven the development of the FIRES project [4] that was presented as a fast, simple and interactive acquisition and reconstruction of a virtual 3D model system. It works combining a simple and on-the-shelf hardware and an intuitive and easy 3D software interface. The measurement step is performed combining a stereo vision system made of two infrared cameras and a *SmartPen* equipped with infrared LEDs. The stereo camera infers the *SmartPen* position in the space and derives the pen tip coordinates. Hence dragging the *SmartPen* over the characteristic profiles of the real object, the system incrementally creates a 3D spline curves network. The reconstruction step uses the curves network to create a low-resolution mesh of the object and its refined subdivision-surface version. Although the whole system does not have the accuracy of a CAD system, its outcome, the 3D virtual model of the physical object, can be still used as input for any more powerful CAD tool.

However, all the classic reverse engineering systems require a real physical object to recreate a virtual model and it is a strong restriction to the user creativity. Both common user and professional designer have an early phase during with they try to transform an idea into a initial mockup by sketching. For this goal they can use classic techniques like pencil and paper and clay, otherwise in recent years, they can adopt more evolved software tools, that support sketching and object composition activities. The main issue is that there is not a magic solution. For instance, the users might have some difficulties to use primitive shape like cube, cylinder, sphere or similar; first of all because they are a limit to obtain complex shape, than because their transformation into a more complex shape is not a trivial process. A good alternative could be offered by a more powerful mathematical paradigm, like spline curves and surfaces. The main drawback in that case, is that a solid theoretical background is a mandatory requirement to use these solutions. A 3D sketching-based modelling could be an ideal solution that allow you to freely draw in the space as you are used to do with pencil on a sheet.

This is the main motivation that drove to evaluate if the *SmartPen* of the FIRES project could be also use for a 3D sketch-based modelling. A "create by sketch" system, that uses sketching tools, could be considered a generalization of a "create by copy" one. It implements the same steps (measurement and reconstruction) that lead to a virtual model. The measurement phase refers to the sketching activities, while the reconstruction operates on the defined strokes. Any optical equipment, like white light or laser scanners, can reconstruct only visible real objects, while the *SmartPen* seems to be a good candidate to sketch ideas directly in 3D. Despite some lacks like occlusion issues and evident difficulties with concave objects, the *SmartPen* system has bridged the gap between complex tools and common users. Hence we decided to explore in which direction further develop the system, to convert it into a new "create by sketch" system.

## 2.2 State of the art

Typically, a 3D reverse engineering system is implemented using a white light or laser scanner device. Other optical systems for reverse engineering can adopt cheaper hardware devices, in [42] they employ the Kinect that is generally used as gaming console. Seldom, we can find works that use other measuring systems to detect the features of the real object, for instance a coordinate measuring machine [21]. All classic reverse engineering systems require a physical object and in some cases, for example if they use rigid bound arms, they can reduce the degrees of freedom for the users. As a result they are unsuitable to be reused as sketching modelling systems.

If the Graphical User Interfaces (GUI) were a strong breakthrough in a quite distant past, in the last decade we have been witnessing to a second considerable revolution, the advance of the Tangible User Interfaces (TUI) [97]. TUI connect the digital and real worlds, showing a great potential coupled to a ease to use. They have collected an immediate appeal to a large range of users and are becoming a valid alternative to WIMP interface (windows, icons, menus, and pointers), mainly because they are not limited to two-dimensional applications. The TUIs have shown that the interaction can become three-dimensional, as a result the applications range from augmented reality [11], to multi-touch surfaces [45]. A sketching system can be considered a specific application of the TUI, especially if the user uses a special tool to draw strokes.

Recent researches introduce a new automatic and assisted modelling interface known as Sketch-Based Interfaces for Modelling (SBIM), that can be considered a subclass of the TUI. They support the user in the sketches digitization process [44,75]. Thus far, the sketching concept was limited in 2D and manually translated to a 3D virtual model, SBIM approach makes faster the generation of reasonable 3D shapes [46]. Despite we can not obviously expect the same accuracy of a pure CAD system, we are interested in these kind of applications. We can immediately make a distinction between 2D sketch-based modelling systems, that support the user in the 3D virtual model translation process of the 2D sketch, and pure 3D sketch-based modelling systems, with which the user draw directly into the 3D world.
One of the main problems is the interpretation of the 2D sketches, in [26] a method that minimizes the curvature among all the 3D candidates is proposed. While in [48] the 2D curve is projected onto a resulting 3D template. Others early works focused on methods for inferring plausible 3D free-form shapes from visible-contour sketches [49]. In [41,74] we can find the first complete works concerning a sketching interface for quickly and easily designing free-form models, with a very low learning curve. Later, in [93] a system based on analytic drawing is presented, it supports precise image-space construction of a linear 3D scaffold. Starting from the consideration that designers prefer sketch descriptive curves, that convey intrinsic shape properties, in [117] they describe a system to transform 2D sketches into 3D curve networks. There are several others systems that belong to the first category, systems that transform 2D sketches into 3D form. In [3, 63] a system for rapid ideation and visualization of 3D forms for design professionals is presented. It uses a 2D interface that makes easier the sketching process. An interesting evolution is proposed in [77] where the strokes are closed using surfaces and the user-drawn curve network serves as a control cage, from which a subdivision surface is generated.
Compared to previous works, we find many fewer examples of systems that allow to draw directly in the 3D environment. The first experiences are shown in [88,116]. In the first one 3-Draw system is presented. By virtue of two handheld sensors, designers who using 3-Draw to sketch their ideas in the air feel as if they are actually holding and working on objects. In the letter, a pure 3D sketching system for a virtual reality environment is presented. The user wears a headset and uses a wired stylus tracked by the system, that reconstructs surface stitching together pieces of spline patches with $C^0$ or $G^1$ continuity. Also in [73] the user uses the hands to sketch in the air and the system reconstruct the underlying mesh, they adopt innovative solutions to stitch the patches and to refine the connection surface. An alternative solution is pro-

posed in [36], ShapeTape is presented as a specialized high degree-of-freedom curve input device that allows to directly control the shape and position of a virtual curve. There are others interesting approaches, in [92] the effort is oriented to support the needs and interests of artists and the hand is used to mark 3D space in a semi-immersive virtual environment. While in [70, 122] hand sketching and gestural interface for the 3D modelling and the fast conceptualization are presented.
SBIM often addresses problems not closely related to the modelling, in [118] they propose a framework that automatically turns a freehand sketch drawing inferring multiple scene objects to semantically valid. While in [100] the user places specific geometric primitives on the relevant parts of the sketch, then the system first fits the primitives projection to the sketch lines, and then infers geosemantic constraints that link the different parts.

The construction of a right 3D curves network derived by the user-drawn sketches is only the first step that characterize a 3D sketch-based modelling system. The next step is also known as the surfacing problem, in practice the system must be able to close the curves network with a smooth surface or a surface mesh. This task can be accomplished following two main methods: the first one requires the use of disjoint surface patches, the latter uses mesh or subdivision surfaces, the discrete representation of the virtual model.
In [60] they propose an adaptive surfacing algorithm that uses $G^1$ smoothly stitching bi-quintic Bézier patches. While in [1], starting from a three-space curves that suggest the contours of shapes the system automatically generates intuitively appealing piecewise-smooth surfaces, using a linear algebra representation. In addition they provide an intelligent user interface for modifying the surface patches. SmoothSketch is the solution proposed in [49] that use surface for inferring plausible 3D free-form shapes. SmoothSketch allows sketches with cusps and T-junctions, a feature that distinguishes it from other approach. All these methodologies create a patched surface and it could become a problem if the resulting model must be further used: all the patches have to be stitched together.
There are many more works that use mesh or subdivision surface to fitting a curves network. In [59] subdivision algorithm is used to interpolate a nets of curves, obtaining a $G^2$ surfaces in almost each points. Even in [94] they obtain a smoothing arbitrary triangle mesh solving a nonlinear fourth order partial differential equation (PDE) and the resulting mesh satisfies $G^1$ boundary conditions. The lofting technique used in [90] allows to create a $C^1$ surface in the most common cases. An interesting solution is presented in [74], a functional optimization is used to manage arbitrary topology curves, even not connected to each other. In [9, 109] they propose an alternative approach, the system constructs by interpolation a quad mesh, iteratively filling the closed curves.

The solution proposed in [89] exploits aspect of the input, as topology and geometry, to triangulate highly nonplanar and concave curve cycles. An interesting and innovative solution is presented in [71], whose main characteristic is that the system refines the reconstructed mesh from time to time that the user adds a curve. Moreover the user-drawn strokes are drawn directly in the 3D space. Typically, all the mesh-based solutions do not allow a local control on the fine shape details. Maybe an hybrid solution, such as that proposed in [4] could be an effective choice.

The sketch-based modelling systems are a really interesting topic, we investigate a portable and low cost platform for reverse engineering and interactive design of sketched models using 3D curve networks, that integrates both acquisition and reconstruction phases. The main challenges regard the concave objects' acquisitions, if the system is used as reverse engineering, and the real-time and the precision of the reconstruction step, if it is used for sketching.

## 2.3 The FIRES system

The *SmartPen* was developed within the FIRES project, thus to evaluate if they could be even used as sketch-based modelling system, we needed to study the original setting and its evolution, highlighting and solving the shortcomings, if any.

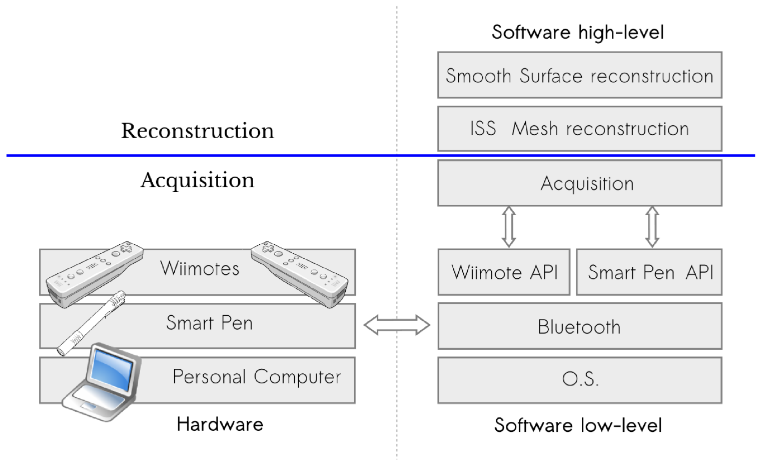


Figure 2.1: Hardware and software layers in FIRES project [4]. Below the blu line the hardware/software components for the acquisition, above the software component for the reconstruction.

Since the first version [4], the FIRES project is made up of a hardware/software layer, for the acquisition, and a software layer, for the reconstruction (Figure 2.1). The main innovative feature is that the system integrates the measurement and reconstruction steps and allows you to have a real-time feedback on the ongoing work. In practice, the user uses the *SmartPen* device to follow the characteristics profiles on the object to acquire, the acquisition step, and the system reconstructs the virtual model of the object using the user-drawn curves, the reconstruction step.

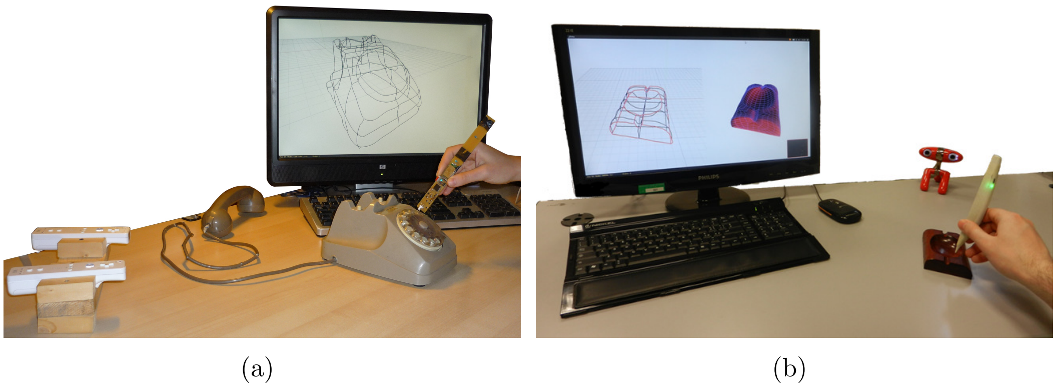


Figure 2.2: Experimental equipment during a typical FIRES work session, first version (a) and second version (b).

The hardware component of the acquisition layer consists of two infrared cameras, available on the Nintendo Wii remote controllers[1], and four aligned infrared light emitter, mounted on the pen's prototype (Figure 2.4(b)). While, on the software side, a specific component of the system continuously tracks the 3D position of the *SmartPen* using a stereo vision triangulation. In this way, the user can intuitively and interactively draw the characteristics curve-profiles of the object and the system automatically and incrementally constructs the corresponding curves network. Whole this process is called Interactive Surface Sketching (ISS) and allows the user to add, modify or delete curves during the acquisition process and to benefit of several modelling tools to enrich the topological information of the virtual model.
The software component that manage the reconstruction uses the polyline mesh data struct produced by ISS: a mesh data struct with vertices, edges and faces, enriched with the polylines corresponding to the acquired curves. The real-time outcome of the reconstruction step are meshes and subdivision surfaces that represent the final yield of the reverse engineering activity. To

[1]http://en.wikipedia.org/wiki/Wii_Remote

achieve a good result, the reconstruction is composed of three different steps: triquadrification, basic refinement and subdivision surfaces. The triquadrification step transforms the polyline mesh, that without constraints can contain n-sided, non-planar, and non-convex faces, into a tri-quad mesh. The tri-quad mesh is further refined by the basic refinement step, it uses bilinearly blended Coons patches on the coarse tri-quad mesh to obtain base mesh that interpolates the polylines. Finally, the base mesh is the input of the subdivision surface step to produce a smooth surface.
Low-cost technology was a remarkable feature that already characterized this first version of the system, nevertheless its simple setup (Figure 2.2(a)) allowed to obtain good results with an optimal compromise between accuracy and cost. Moreover, it was already clear that one of the main features, that distinguished the FIRES system from the others optical light-based reverse engineering system, was the capability to continuously track the *SmartPen*, even if it was not touching the physical object. As a result using several *ad hoc* algorithms, as the "hole creation tool", the "skinning tool" or the "symmetry tool", it was already possible to add virtual parts to an existing object, for instance an handle, as shown in Figure 2.3.

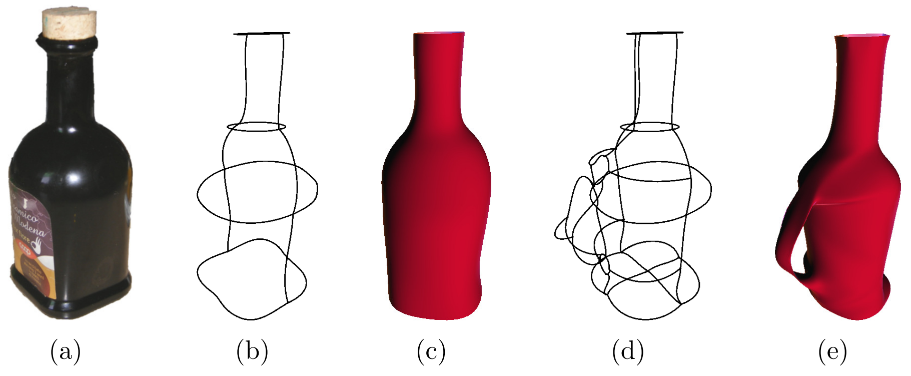


Figure 2.3: A first example of a virtual part added to an acquired object [4]. The physical object (a), the curves network acquired (b) and the relative reconstruction virtual model (c). The same acquisition enriched with a handle (d) and (e).

Despite the first good results, both acquisition and reconstruction phases have been improved in a second version of the FIRES system, presented in [87]. Some defects, detected during several acquisition sessions, could be corrected in order to enhance the accuracy of the system and the quality of the final results. We could summarized as follow the areas of improvement:

- the limited resolution of the wiimote cameras reduced the precision of the acquisition step;

- the stereo vision was strongly affected by the human accuracy;
- the basic refinement was not sufficiently to obtain a regular input for the next subdivision surface step.

The second version of the FIRES system has brought several changes (Figure 2.2(b)). First of all, the two Nintendo Wii remote controllers were replaced with a higher resolution stereoscopic webcam: Minoru 3D Webcam[2] (Figure 2.4(a)). In that case, the new stereo vision system has brought a twofold advantage: it improves the resolution and reduces any tilt and jerk errors that typically affect two disjoint devices like the wiimote. Then the *SmartPen* was equipped with different smart sensors like three-axis accelerometers, magnetometers and gyroscope to improve the 3D tracking. Furthermore, the disposition over the *SmartPen* of the infrared LEDs emitter was rearranged: the tip LED was eliminated and all the four LEDs were moved on the body of the pen (Figure 2.4(c)). This solution gave rise to a greater freedom of movement of the *SmartPen*, reducing the occlusion problems of the LED on the tip.

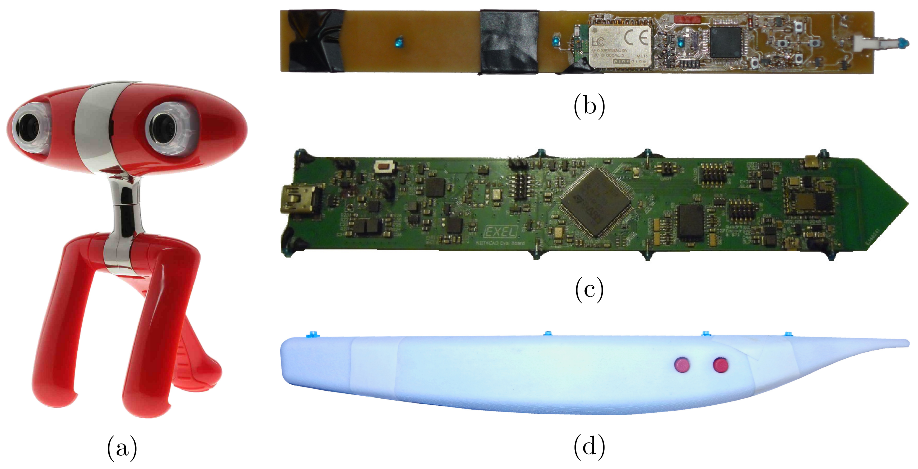

(a) (b) (c) (d)

Figure 2.4: On the left the new Minoru stereoscopic webcam (a). On the right, the first version of the *SmartPen* with one LED on the tip (b); the second prototype version with a different LEDs disposition and equipped with inertial and magnetic sensors (c) and its final concept (d) designed by Alessi[3].

[2]http://www.minoru3d.com
[3]http://www.alessi.com

The hardware/software layer, shown in Figure 2.1, has remained the same, but several alternative algorithms have been introduced. For instance, the video stream is filtered with the sensor raw data derived by the inertial and magnetic sensors using the Kalman Filter [24], a well-known recursive linear estimator suitable for modelling dynamic systems with inherent noise. This solution produces a better and more stable signal to perform the sampling and then to regularize the curves network.
If in the previous version of the system, the second step of the reconstruction was based on bilinearly blended Coons patches, the new reconstruction module adopts a surface diffusion flow method. While the subdivision surface step is performed using the Catmull-Clark scheme, whose interpolation rules allow to preserve the sharp features. Indeed, the user has got the possibility to indicate which curves represent a sharp edge of the physical object. The novel surface reconstruction procedure, based on functional optimization, allows to obtain more realistic results thanks to the capability of preserving the sharp features of the object (Figure 2.5).

The second version of the FIRES project has reached an high quality level and it offers innovative solutions in the reverse engineering field. Nevertheless, during the study phase of the system, we have identified some critical points that we can summarize as follow:

1) the Kalman filter was not properly calibrated and the set of sensors could be further exploited;

2) the LEDs layout often leads to an occlusion of the external LEDs of the series. Despite the tip LED was moved on the body of the *SmartPen* (Figure 2.4(d)), it often occluded by the object, while the LED on the top falls out the field of view of the cameras;

3) the linear LEDs layout does not allow to determine any rotations of the *SmartPen*. In case of further evolutions of the system, it could be useful to determine the orthogonal vector to the *SmartPen* vertical axe;

4) it was noticed of a drift problem that affects the *SmartPen* position estimation. It's probably due to a heating of the 3D stereo camera and involves a misalignment of the acquired curves;

5) despite the two camera are built-in the same physical device they showed a synchronization problem. Perhaps, it is due to the low cost hardware solution;

6) the two cameras are too close to each other, this reduce the field of work and increase the occlusion problems;

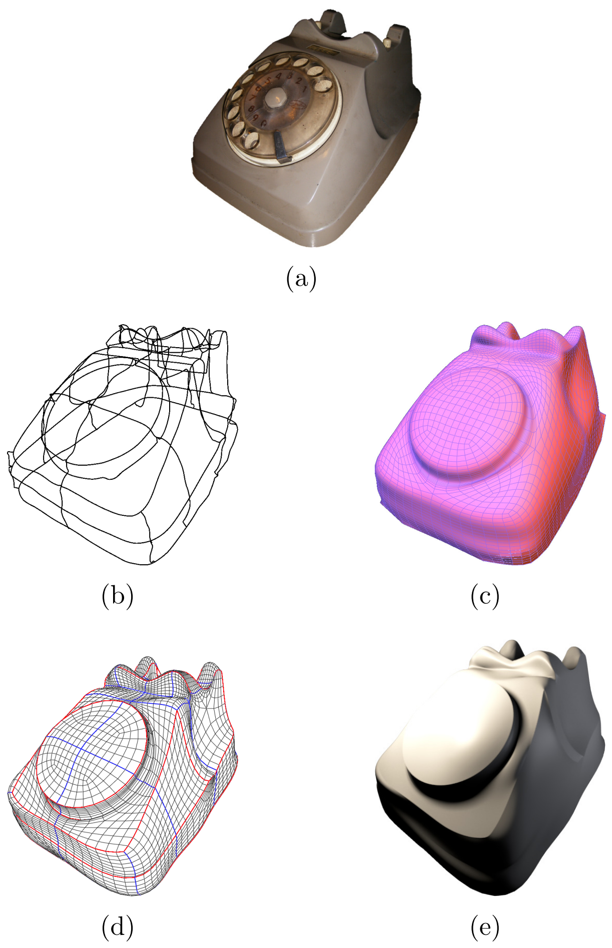


Figure 2.5: The same physical object (a) reconstructed with the first and the second FIRES version (c) and (e). In (b) you can see the simple curves network acquired with the first version of the system. While in (d) the new curves network, superimposed to the subdivision surface, is enriched with sharp edges information (red profiles).

7) the wide profile of the *SmartPen* tip is an obstacle if you want to acquire objects with narrow section profiles;

8) the rigid structure of the *SmartPen* and the short tip do not allow to follow concave profile of the object.

The evaluation of the *SmartPen* as sketch-based modelling tool was carried out contextually with the resolution of some of the listed problems.
The issues that concern the LEDs layout would require a re-engineering of

the pen that it is out of the scope of our work, but we highlighted them to drive future developments. Maybe in that case, a good solution could be create a triangular structure with three LEDs, located in the middle of the *SmartPen*'s body. It would allow to detect rotation and it would be less prone to occlusion.
The drift and synchronization problems are closely related to the quality of the chosen hardware. Trying to maintain unchanged the goal of the project, we adopted technical artifices, like delay the start of the acquisition in order to stabilize the system or empty the buffer of the cameras driver. A better solution might be to build a stereo vision system with higher quality camera, a hardware synchronization and a wider space between the objectives.
We proposed interesting solutions for the concave problem and the acquisition of profiles with narrow section, adopting several pen tip prototypes to simulate various situations of use [69].

### 2.3.1 Sensor data and Kalman Filter

The inertial (accelerometer and gyroscope) and magnetic sensors mounted on the *SmartPen* are also use to track the orientation of the device with respect to a fixed frame. We chose to use a Complementary Filter [65] thanks to its good trade-off between accuracy and lower computational cost on embedded platform. In practice, we adopted the approach implemented in [64], where two different orientation estimates are fused together to obtain a better result.

The *SmartPen* orientation was thus obtained using the gyroscope data stream, that are more accurate for higher dynamic frequencies, and the accelerometer and magnetometer data stream, that are more accurate in static condition. The Complementary Filter quaternion output gives the *SmartPen* sensor's orientation as the rotation between the *SmartPen*'s reference frame and the geomagnetic reference frame.

An interesting comparison between the orientation estimated by the video triangulation and by the sensor stream data is shown in Figure 2.6. The second one (black line) is typically less noisy than the video triangulation, which roll angle (*SmartPen* rotations towards the cameras) is affected by the distance from the cameras. Moreover, the yaw angle lacks in the video orientation, due to the linear LEDs layout as recalled in the previous section.

The implementation of the Complementary Filter, embedded on the *SmartPen*, has been optimized for the limited resource available on the device (32-bit precision and no floating point unit). Then we performed a qualitative test to evaluate the result. The embedded algorithm shows a mean difference of $0.01°$, with peak differences always below $1°$, if compared with a standard

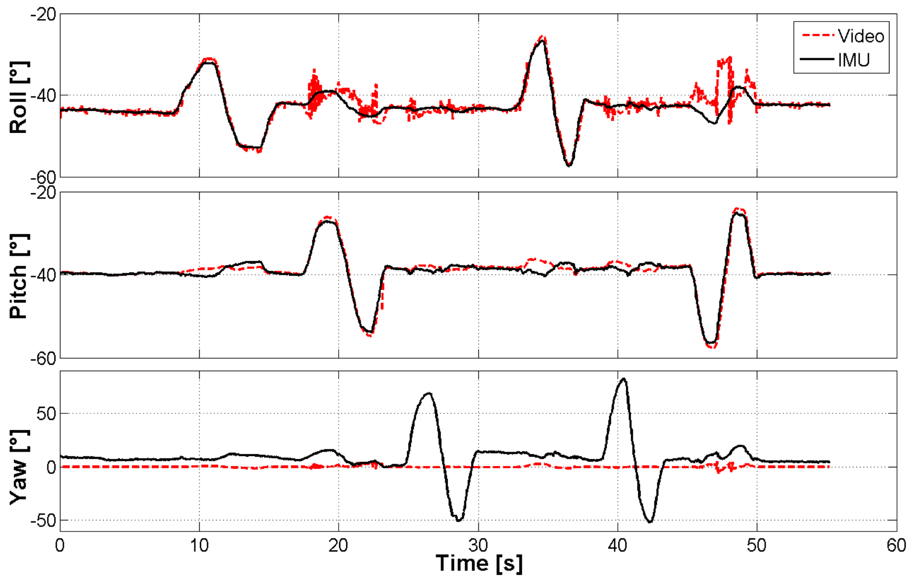


Figure 2.6: Comparison between the *SmartPen*'s orientation estimated by the video triangulation (red line) and by the Complementary Filter (black line).

implementation on 64-bit platform with double precision unit.
The sensors data stream have been useful in the final step of the acquisition process to reduce the noise of the *SmartPen* 3D position, estimated from the stereo vision system. The video signal has been filtered with the output of the Complementary Filter operation in order to stabilize the *SmartPen* 3D position.

The Kalman Filter is a recursive linear estimator appropriate for discrete-time controlled process with inherent noise [24]. It is suitable to combine data generated by multiple measurements, in order to improve the quality with respect to what we can achieve using a single measurement by itself. The Kalman Filter tries to estimate the state $x \in \mathbb{R}^n$ of a discrete-time controlled process using the measurement $z \in \mathbb{R}^m$ and the respective linear equations are:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{2.1}$$

$$z_k = Hx_k + v_k \tag{2.2}$$

where the matrix $A$ represents the state of the system at the previous time step $k-1$, the matrix $B$ drives the optional control input $u \in \mathbb{R}$ to the state $x$ and $H$ relates the state to the measurement $z_k$. While, the independent random variables $w_k$ and $v_k$ represent, respectively, the process noise and the measurement noise, with normal probability distributions:

$$P(w) \approx N(0, Q) \tag{2.3}$$

$$P(v) \approx N(0, R) \tag{2.4}$$

where $Q$ and $R$ represent, respectively, the process noise covariance matrix and the measurement noise covariance matrix.

In our implementation of the Kalman Filter we excluded the $B$ optional term in (2.1), while the measurement vector $z_k$ in (2.2) is composed by $\{p_v, o_s\}$, respectively the position estimated from the stereo vision system and the orientation estimated from the inertial and magnetic sensors. The vector $z_k$ is used to correct the prediction of the system and, at each time step $k$, the equation (2.1) allows to estimate the filtered state $x_k$, composed by the *SmartPen*'s position, velocity and orientation.

Combine the estimate of the stereo video system with the estimate coming from the inertial and magnetic sensors into a modular vector $z_k$, proved to be a winning choice that simplifies the tuning phase. It allow to manage the different working frequency between the stereo vision system ($30fps$) and the sensors unit (more then $50Hz$). At each time step $k$ the system constructs the vector $z_k$ with the most updated information between the two sources and corrects the measurement and the relative noise using the matrix $H_k$. Typically, the stereo video system defines a position and the inertial and magnetic sensors refine it for at least three times before the 3D position is updated again.

To obtain an optimal fusion between sources, each measurement should be weighted by the relative portion of the matrix $R$. However, if in our implementation the definition of the measurement noise covariance matrix $R$ is quite possible because we are able to measure the process and define it state, we do not have ability to define the process noise covariance matrix $Q$. Hence we decided for a off line tuning of the parameters of the filter and the $w_k$ and $v_k$ represent the zero mean Gaussian white noise.

We used two different tests, static and dynamic, to evaluate the improvement brought by the Kalman Filter. In Figure 2.7 we show the static noise estimation of the *SmartPen* at different distances, comparing the pure value obtained by the stereo video system and the filtered one using the inertial and magnetic sensors. The Kalman Filter significantly reduces the error on the z-axis, the most affected by the distance. While in Figure 2.8 we show the dynamic noise estimation at two different distance during the acquisition of a simple square.
The filter effect becomes fundamental with the increase of the distance and nullifies any flickering of the user's hand. Moreover, the Kalman Filter has

proved its contribution to stabilize the tracking signal, that it is very important when the user does not have any support points and decides to freely draw in the air, as in the sketch-based modelling.

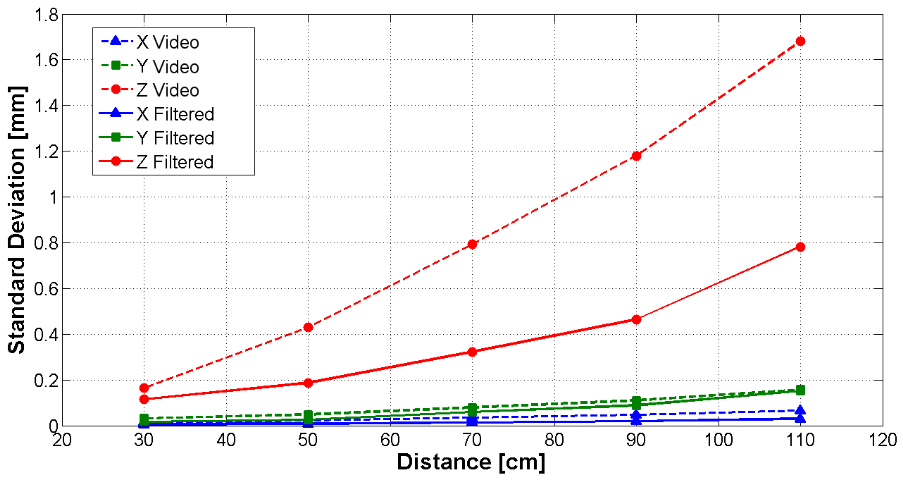


Figure 2.7: The mean standard deviation of the static noise estimation at different distances. The values obtained by the stereo vision system (dashes line) and the filtered one (solid line).

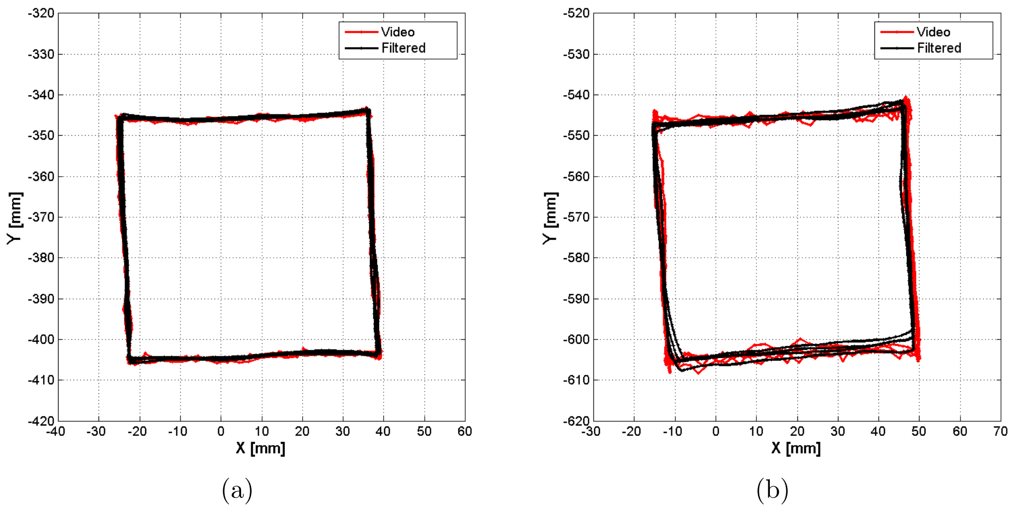


Figure 2.8: The dynamic noise estimation at 40 *cm* (a) and 60 *cm* (b). We tracked four times a 6 *cm* edge square and plotted the result of the stereo vision system (red line) and the filtered one (black).

### 2.3.2 Narrow profiles and concave objects

In this section we present the solution proposed to solve two distinct issues, like objects with narrow and deep profiles and concave objects. The idea was initially developed to address to the first one, but the correct generalization of the method allowed us to make the system more powerful.

In the previous section we seen that at the end of the acquisition process the video stream data are fused with the sensors stream data to improve the estimation of the *SmartPen* movements. There are four steps prior to the application of the Kalman Filter, which allow to obtain a 3D estimation of the *SmartPen* position, starting from a pair of 2D frames collected by the stereo cameras. The first step transforms the 2D images collected by the cameras into binary images using a threshold value, this allow to highlight only the visible infrared LEDs. The second step uses a blob detection algorithm to identify the LEDs and to define their 2D position in the frame. Then the 2D coordinates are triangulated to reconstruct the 3D position of the LEDs, in the triangulation step. The fourth step uses the 3D LEDs position to estimate the *SmartPen* 3D position.
This last phase was studied *ad hoc* to address partial occlusions. The redundant number of LEDs and their mutual distance, visible in Figure 2.9, have been designed in order to guarantee the estimate of the position even if only two pair of LEDs are visible.

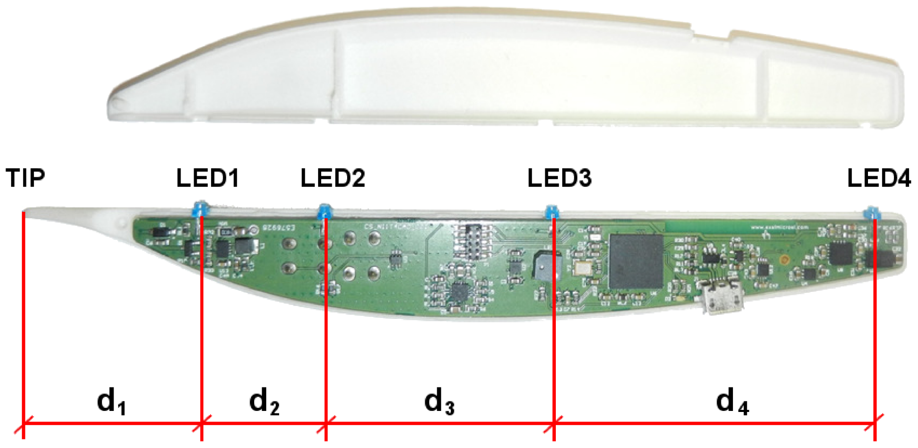


Figure 2.9: The disassembled *SmartPen* with the mutual distances between each pair of contiguous LEDs: $d_1 = 32\ mm$, $d_2 = 22.5\ mm$, $d_3 = 42.5\ mm$ and $d_4 = 62.5\ mm$.

In brief, the output of the triangulation step is the quadruplet $(\delta_1, \delta_2, \delta_3, \delta_4)$, where $\delta_i$ is a boolean value that indicates if the LED$_i$ is visible or not, and a set of $\bar{p}_i$ 3D coordinates, for each visible LEDs belonging the quadruplet. Knowing the physical distances and the estimated ones, it is possible calculate the centroid of the *SmartPen* for each configuration of visible LEDs, using the cumulative distances. The centroid $C \in \mathbb{R}$ of the visible LEDs is computed, using the theoretical distances, as follow:

$$C = \frac{\sum_{i=1}^{4} l_i \delta_i}{\sum_{i=1}^{4} \delta_i} \tag{2.5}$$

where $l_i$ is the cumulative distances for the LED$_i$ from the LED1. The distances between the LEDs are all different (Figure 2.9) hence, for each configuration of visible LEDs, we can computed the centroid $C' \in \mathbb{R}^3$, using the estimated distances, as follow:

$$\bar{C}' = \frac{\sum_{i=1}^{4} \bar{p}_i \delta_i}{\sum_{i=1}^{4} \delta_i} \tag{2.6}$$

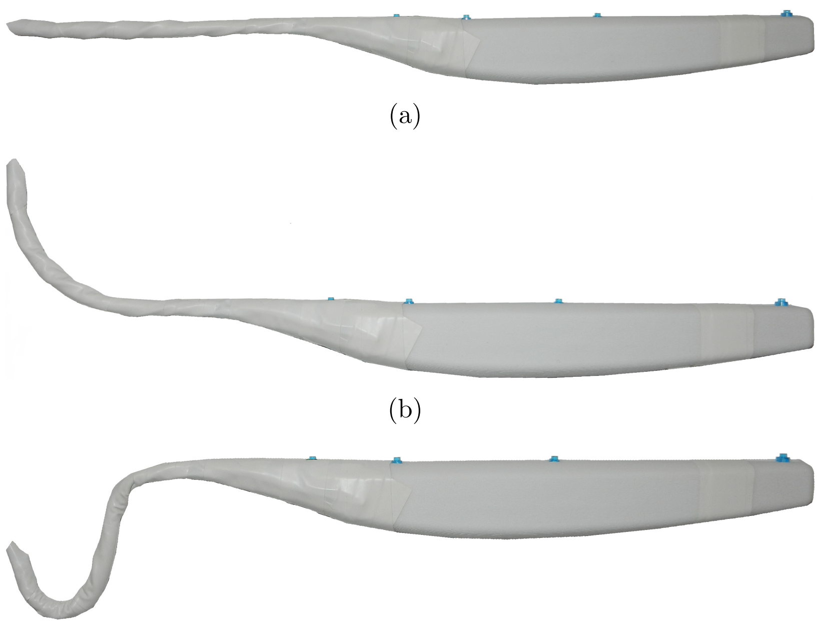
and the distance $T_\delta \in \mathbb{R}$ between the first visible LED and the *SmartPen*'s tip, both are at a unique and known position. The least squares method is used to estimate the normalized LEDs direction $\bar{v}_{pen}$. Then the 3D position of the *SmartPen*'s tip, $\bar{p}_{tip}$, is computed as follow:

$$\bar{p}_{tip} = \bar{C}' + \bar{v}_{pen}(C + T_\delta) \tag{2.7}$$

This is a good solution that address different problems, like occlusions, matching errors, triangulation errors or limited field of view. It works fine with small object, but if two visible LEDs seem to be sufficient, the most occlusion problems often occur with big or concave objects, like a quite deep jar. The *SmartPen* is too short and the tip is too wide to reach all deep and narrow profiles.

Since the distance $d_1$ (Figure 2.9) could be setted at the beginning of each acquisition session, and consequently the $T_\delta$ can coherently vary, we have proposed a first naïf solution that extend the tip position, as shown in Figure 2.10(a). It allows to reach narrow and deep profiles on the object and helps to reduce reflection problems, while the method to estimate the *SmartPen*'s tip position (2.7) remains unchanged. Unfortunately, since the range of the pitch and roll *SmartPen*'s angles is limited to correctly identify and match the pair of visible LEDs, the extended tip does not even allow to tilt the device in order to acquire concave portions of the object.

The second solution is a generalization of the previous one, and it arises from the consideration that the sensors' unit provides also the yaw angle. We introduced a bendable tip, as show in Figures 2.10(b) and 2.10(c). It requires an update of the estimation procedure of the 3D position of the *SmartPen*'s tip and a new calibration phase at the beginning of the acquisition session. In practice, the initial setting of the system is enriched with a new displacement distance, $d_o$ in Figure 2.11. It defines the offset of the tip with respect to the vertical axis of the *SmartPen*. After the $\bar{p}_{tip}$ estimation (2.7) and during the data fusion step (2.2), $d_o$ is employed to correctly translated the $\bar{p}_{tip}$, orthogonally to the $\bar{v}_{pen}$ direction, using the yaw angle provided by the magnetic sensor.


(a)

(b)

(c)

Figure 2.10: The extended *SmartPen*'s tip (a), to reduce occlusion problem and to reach narrow profiles. Two different bendable tip configurations (b) and (c), to reach concave surfaces.

Since there are no LEDs on the terminal of the new tip, the stereo vision system does not have any cues to detect the displacement distance of the tip from the vertical axis of the *SmartPen*. Moreover, each time the system starts, the stereo vision system partially knows the coordinates system defined by the *SmartPen*: it knows only the planes defined by the pitch and the roll angle. Hence, the alignment of the bent tip can be done only exploiting the yaw angle of the magnetic sensor.

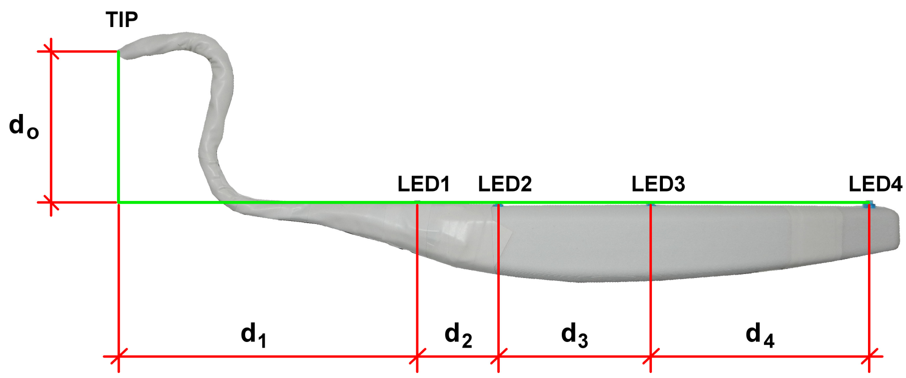


Figure 2.11: The mutual distance between each pair of contiguous LEDs and the new displacement distance of the bendable tip.

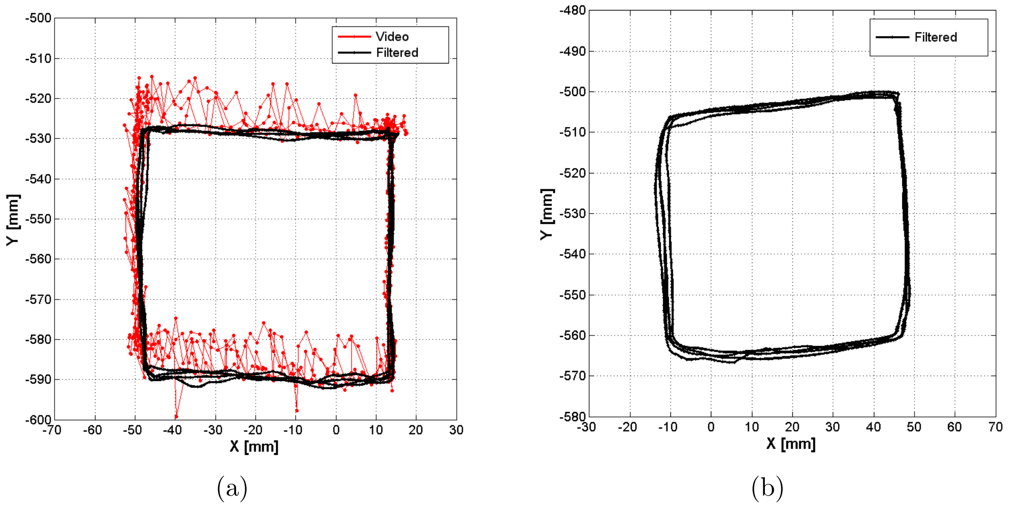


Figure 2.12: The dynamic noise estimation at 60 *cm* for the extended *SmartPen*'s tip (a) and the curved *SmartPen*'s tip (b). We tracked four times a 6 *cm* edge square and plotted the result of the stereo vision system (red line) and the filtered one (black). In (b) the filtered track is the only visible, it is not possible estimate the bent tip with only the stereo vision system.

In particular, the user draws a horizontal line and aligns the bended-tip with that line, thus the system can compute the offset angle $\phi$ between the magnetic north and that position. The angle $\phi$ allows to align the physical *SmartPen*'s tip on the horizontal plane with the virtual displacement distance starting from the position estimated by the video frames.

Both for the extended *SmartPen*'s tip and for the curved one we performed a dynamic test to verify the noise during the acquisition of a simple square (Figure 2.12). Obviously, if the *SmartPen* is equipped with a curved tip we could verify only the filtered track, since the tip position is not uniquely estimable with only the stereo vision system.

In that case the shape of the *SmartPen* is irrelevant for a sketch-based modelling use. There are not objects that can obstruct the vision rays and if the user's hand causes a occlusion, he can modify her position herself. However, the experience gained to complete the system and the acquisition sessions have been very useful.

## 2.4 Experimental results

In this section, we explore the powerful of the proposed solutions, showing several acquisition results, presented also in [69]. We acquired some common simple objects to test the accuracy of the system, and then we reserved some test to verify the *SmartPen*'s capabilities in the sketch-based modelling and how it may, in particular, benefit from the tracking system.

### 2.4.1 Reverse engineering acquisitions

The first two acquired objects are a ashtray and a telephone (Figures 2.13(e) and 2.14(e)). The system has retained the previous features and allowed to reconstruct them by tracing the 3D curve which follows the characteristics profiles of the physical objects, with a variety of different surface features.
The user uses the *SmartPen* to follow the characteristics profiles on the real object and incrementally adds curves to create the curves network, Figures 2.13(a) and 2.14(a). Then the system approximates the user-drawn curves with spline, Figures 2.13(b) and 2.14(b), and creates the underlying polyline meshes, Figures 2.13(c) and 2.14(c). The polyline mesh is used to reconstruct the final virtual model, applying diffusion flow and subdivision surface schemes and respecting the sharp edges details defined by the users (red line), Figures 2.13(d) and 2.14(d).
The use of the filter has significantly reduced the natural flicker of the human hand and has helped the next sampling phase that constructs a curves network using spline curves, visible in Figures 2.13(b) and 2.14(b).

Now we present the acquisition of a spanner (Figure 2.15), it is useful to demonstrate the accuracy of the system, despite it is not, and does not want

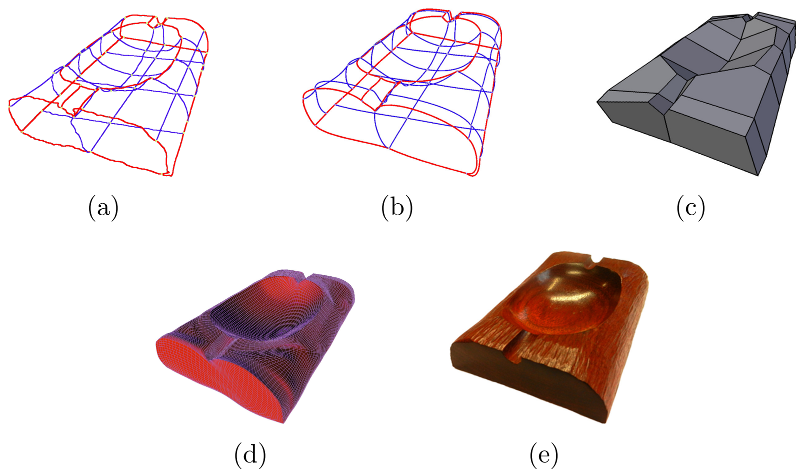


Figure 2.13: The acquisition steps of an ashtray. We can see, in sequence, the user-drawn curves (a), the spline approximation (b), the polyline mesh (c), the final result (d) and the physical object (e).

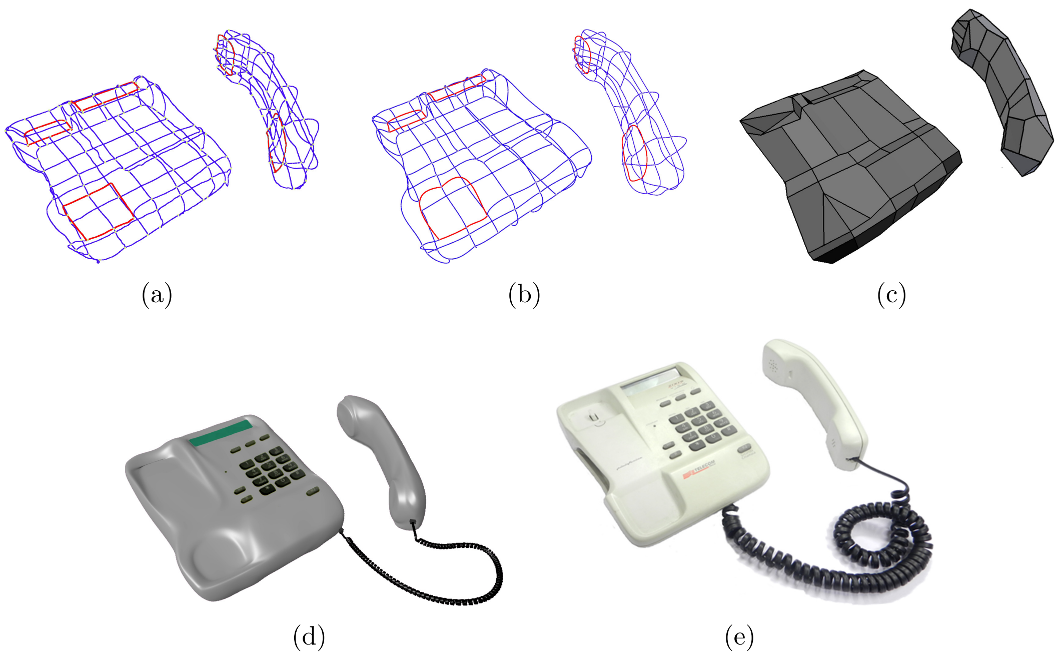
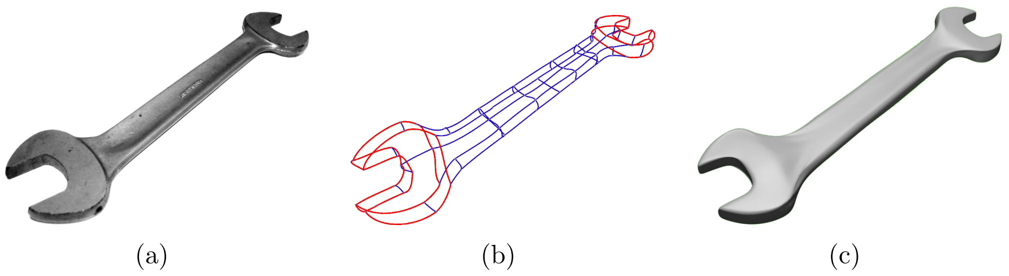


Figure 2.14: The acquisition steps of a telephone. We can see, in sequence, the user-drawn curves (a), the spline approximation (b), the polyline mesh (c), the final result (d) and the physical object (e).

to be, a professional CAD system, and we show how the outcome could be naturally integrated with a 3D printing system. The acquisition steps are obviously the same of the previous tests, but in that case the finale result, Figure 2.15(c), is prototyped using a 3D printer (MakerBot Replicator[4]) and then we compared the resulting prototype with the real object. Despite the error of the used 3D printer (about 2 *mm*), we obtained an extremely faithful prototype of the real object, with an accuracy of less than 8 *mm* (Figures 2.15(d) and 2.15(e)).



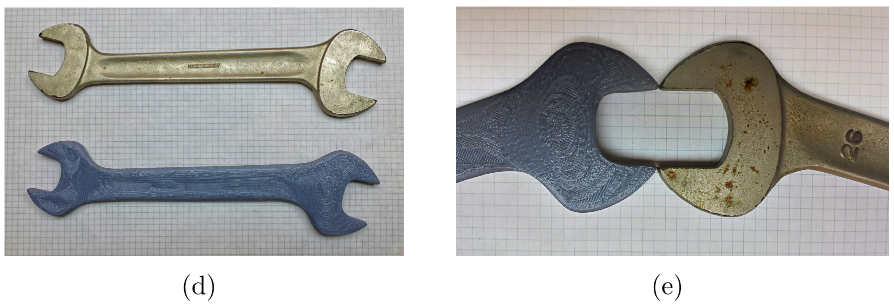


Figure 2.15: The reconstruction and prototyping of a spanner. In the first row you can see the real object (a), the spline approximation curves network (b) and the reconstructed surface (c). In the second row, the comparison between the real object and the printed one. The real spanner dimensions are $240 \times 60 \times 8$ *mm*, while the prototyped spanner dimension are $234 \times 56 \times 8$ *mm*.

The example shown in Figure 2.16 presents a result from a collaboration with The Rizzoli Orthopaedic Institute of Bologna, Italy. This kind of prosthesis, in that case a lower limb prosthesis' socket, represents the interface with the patient limb and needs to be personalized and periodically readapted. For our purposes, the acquisition of the prosthesis is a good benchmark to test the extended *SmartPen*'s tip. Indeed, we used the configuration shown in Figure 2.10(a) to acquire it.

[4]http://www.makerbot.com

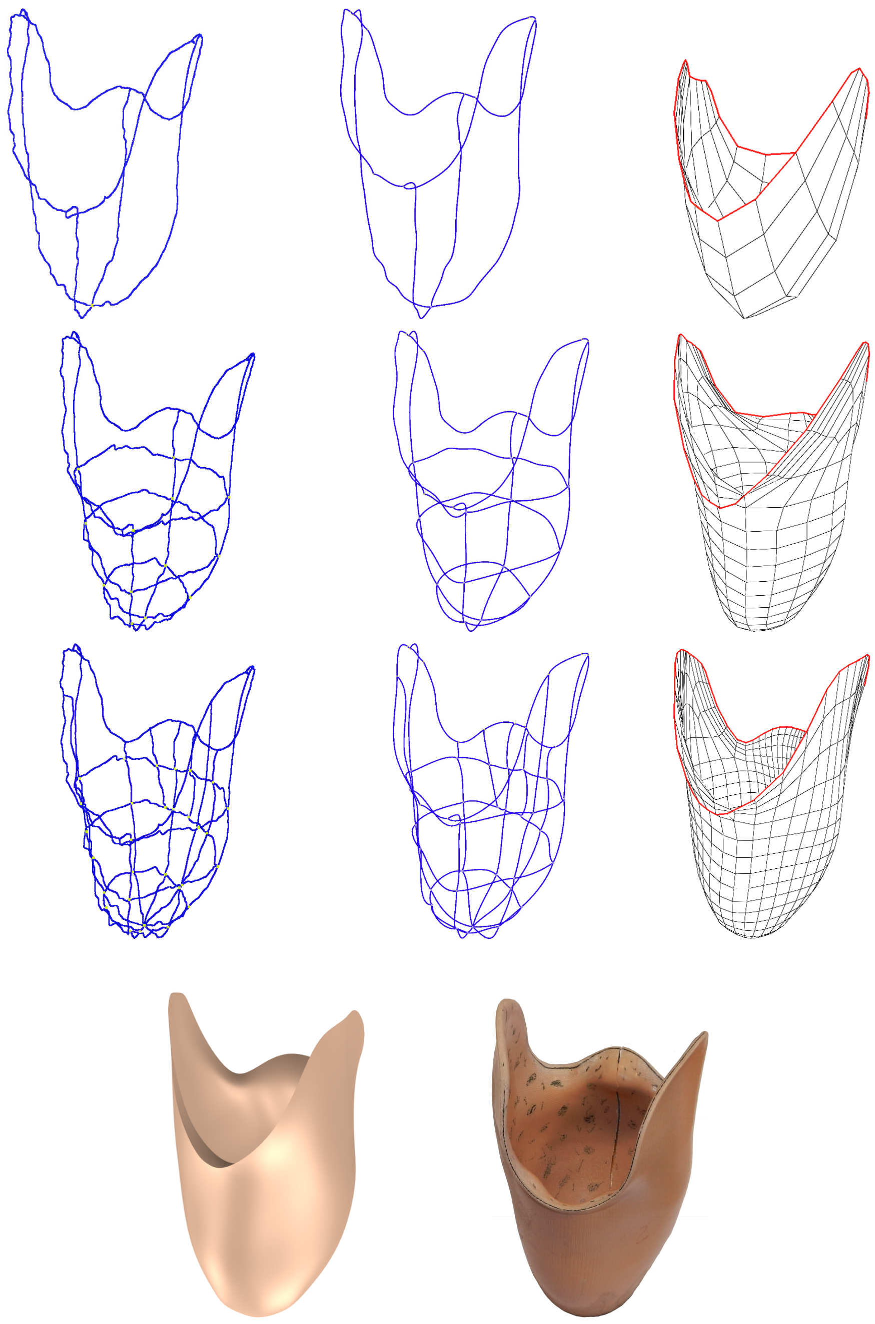


Figure 2.16: The reconstruction of the inner surface of a lower limb prosthesis' socket. Each row shows a different step of the session, while the columns show the user-drawn curves, the spline approximation and the partial result. A rendering of the final result and the original object are shown on the bottom.

That invasive implant, with free-form and concave shape, causes severe problems to the classic 3D scanning systems, principally due to several intrinsic characteristics, like the significant concavity (about 200 *mm*) and the shape of the inner surface with non-uniform thickness. As a result, it is not possible acquire only the outer surface to reconstruct the inner shape.
The images sequence clarifies the ISS process, in particular how the system refines the final result each time the user draws a curves. In each row, we placed side by side the noisy curves and the smoothed one, to evaluate the filter contribution.

The last example was studied to point out the potential of the bendable tip, such as those shown in Figures 2.10(b) and 2.10(c). The object is a small box with a complex concavity and a narrow access point, the diameter of the top hole is 20 *mm*, (Figure 2.17).
These characteristics do not allow the use neither of optical 3D scanners nor contact 3D scanners technologies to acquire the inner shape of the object, they can not reach the hidden profiles of the inner surface. Exploiting the flexibility of the bendable *SmartPen*'s tip, we were able to introduce the tip inside the box and acquire the inner profiles without disassemble the box.
One more time it is appreciable how the system requires a very low number of profiles to faithfully reconstruct the object.

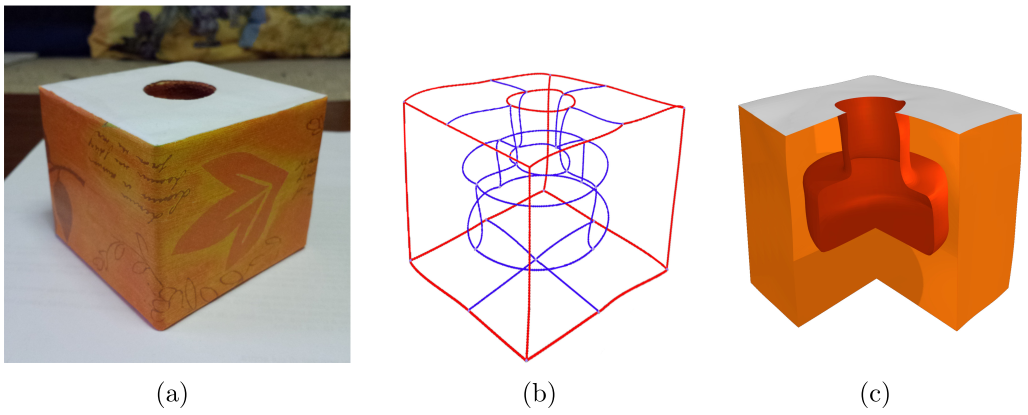

(a) (b) (c)

Figure 2.17: The acquisition of a small box with a complex concavity and a narrow access point (the hole's diameter is 20 *mm*). An image of the real object (a), the curve networks (b), where the inner profile is visible, and a section of the reconstructed object (c).

### 2.4.2 Sketch-based modelling acquisitions

For this section, we conducted some tests to verify the freehand sketching capabilities of the system and to evaluate the usability of the *SmartPen* as sketch-based modelling device.

The first example is a broken vase (Figure 2.18(d)), we reconstructed the virtual model, using the *SmartPen* to repair the missing part (Figure 2.18(c)). Then we tried to add different handles which were not present on the physical object (Figures 2.19(c) and 2.19(f)).

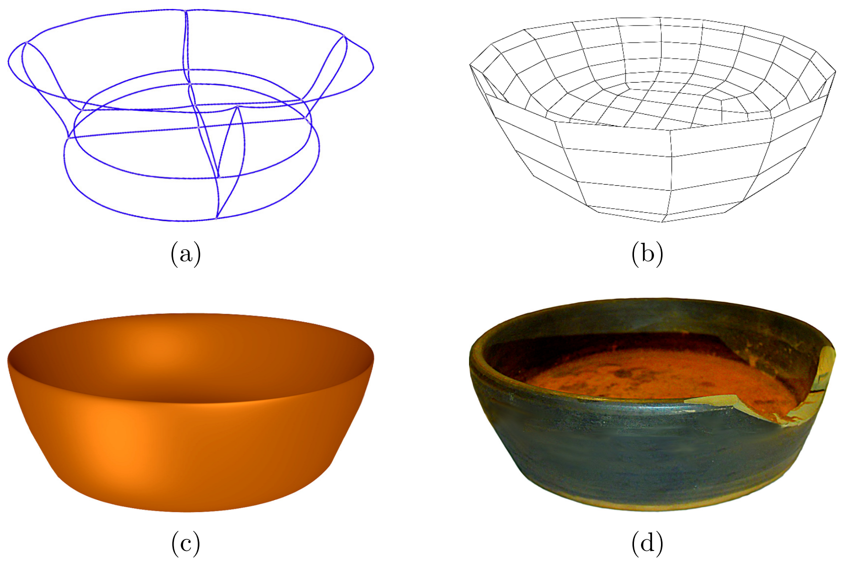


Figure 2.18: The restoration of a broken vase. The acquired curves network that complete the missing border (a), the wireframe visualization of the final result to show the regularity of the obtained surface (b) and a rendering of the reconstructed surface (c) compared to the physical broken vase (d).

The capability to add in real-time the missing part of the object is another important feature of the system, that typically is not provided by other reverse engineering systems. In the same environment, with a simple and intuitive interface you can reconstruct and enrich the virtual model just acquired.
In practice, starting from the restored virtual model (Figure 2.18(a)) we created a pair of holes on the border and on the bottom of the vase (red lines in Figures 2.19(a) and 2.19(d)), exploiting the modelling tool of the system, then we drew the freehand profile to connect them. The system has reconstructed the handles as if they were a real part of the object and you can see

a rendering in Figures 2.19(c) and 2.19(f).

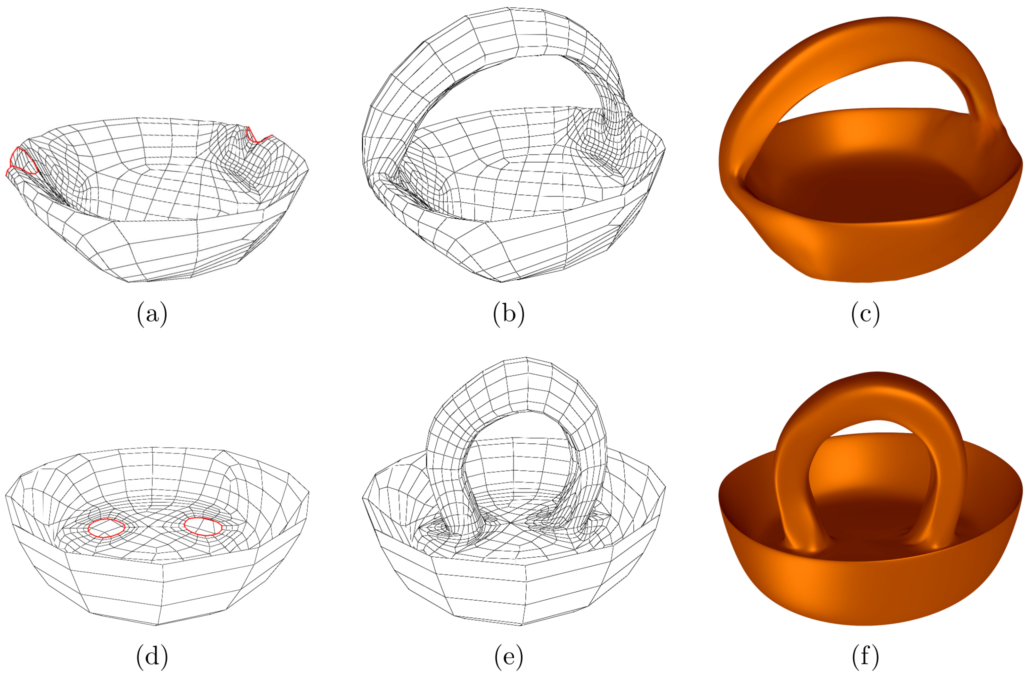


Figure 2.19: The completion of the restored vase with the addition of an handle. Each row represents a different test: the partial results after the creation of the holes (a) and (d), the final results (b) and (e) and the two relative renderings (c) and (f).

In Figures 2.20 and 2.21 we present two examples of freehand sketching, that we interpreted as a cartoon submarine and a glass vase. There are not real objects behind these virtual models, but to create the first shape we used a parallelepiped and a cylinder, respectively, and then we enriched the models with different details. The main problem is the lack of visible feedback in the real space, thus it is quite difficult reach a good perception of the proportions. Despite our reduced drawing ability, the framework supported us during the whole session, allowing to correct any mistakes.
The submarine was obtained drawing curves in the air without any modelling tool provided by the system. The final curves network appears quite regular and the resulting rendering is pleasant enough. While, for the glass vase, we sometime used the "symmetry tool" and the system reconstructed a pleasant smooth-wavy pattern.
We think that an expert designer with better drawing ability can obtain higher

quality results, simply because he is more accustomed to freehand draw. It is even true that we were able to create quite complex objects with a limited timing experience.

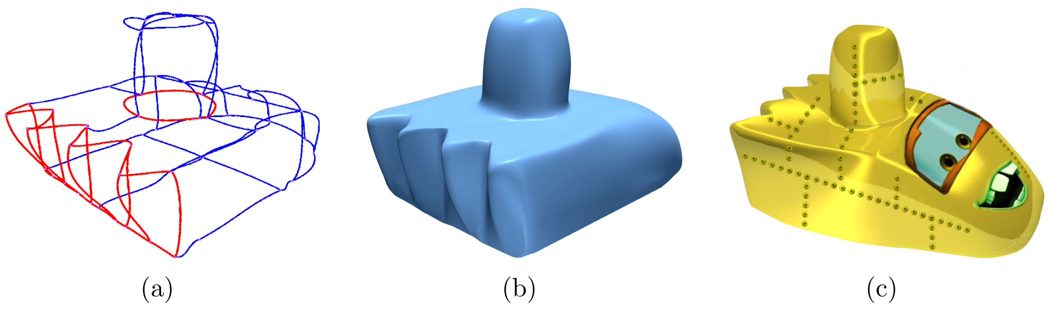


Figure 2.20: The virtual model of a cartoon submarine. It was obtained by freehand sketching: the drawn 3D curves (a), the interpolating smoothed surfaces (b) and the rendering of the reconstructed models (c).

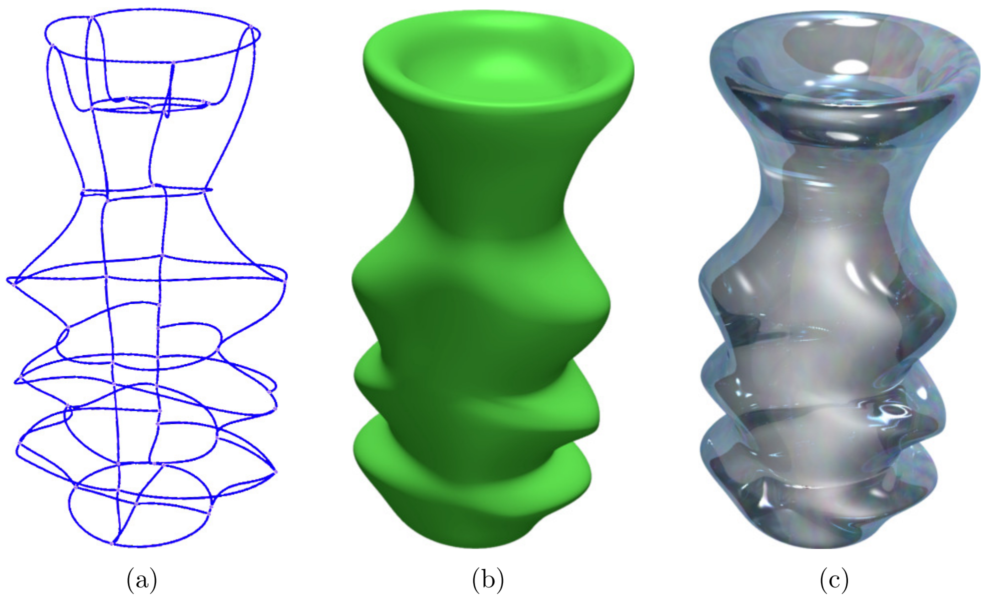


Figure 2.21: The virtual model of a glass vase. It was obtained by freehand sketching using the "symmetry tool": the drawn 3D curves (a), the interpolating smoothed surfaces (b) and the rendering of the reconstructed models (c).

## 2.5 Conclusions

We presented FIRES, a project born to be a fast and interactive reverse engineering system, evaluating if it could be even used as sketch-based modelling system. We chose it thank to several interesting features that distinguish it from traditional reverse engineering systems:

- it is a good trade-off between accuracy of the reconstructed results and hardware used, which is cheap and readily available;
- it has a simple and intuitive interface which makes available several interesting modelling solutions;
- it fuses in the same environment reverse engineering capabilities and modelling features;
- the outcome of the system is easily integrated into other modelling tools or directly used as input in a 3D printing system;
- it is able to give a immediate and interactive feedback on the reconstructed result, these characteristics allow the common user to quickly learn and to take full advantages of the system;
- the user is able to obtain the final result in a very short time.

The preliminary study of the project helped to identify some aspects of the system that could be improved. In particular, a better integration between the data collected by the stereo vision system and the inertial and magnetic sensors data was reached correctly tuning the Complementary and Kalman filters. While, the acquisition of complex concave objects is a novel features that has been introduced testing a new extensible and bendable *SmartPen*'s tip. This innovation required an update of the tip estimation procedure and a new calibration phase at the beginning of the session.

The new version of the system has passed through a preliminary users evaluation and it has been submitted for assessment by an experienced users. We organized an informal test with three different target users: experienced users in the field of computer-assisted 3D modelling and design, from the Architecture department at the University of Bologna[5]; researchers studying and designing innovative interfaces, from the *i3* research group at FBK[6] (Trento, Italy) and a designer unfamiliar with CAD or interaction technologies. After a brief demonstration they were free to explore the system. We avoided to

[5]http://www.da.unibo.it
[6]https://i3.fbk.eu

give them particular tasks, but the hands-on session aimed to highlight their feeling with the new technology.
Each of them recognized the potential of the system, in particular they found very interesting an integration within existing CAD tools and the possibility to add new details. Moreover, they were positively impressed by the accuracy of the system and the quality of the outcomes, especially when it is used as sketch-based modelling system. Some of the tested users suggested that the system could be also used as measurement system, to collect information about real objects, like the curvature or the linear distance on curved surfaces. After this first users evaluation we plan to organize a more formal user study, during which we will ask to perform several tasks both reverse engineering and freehand sketching. During this event we will involve non-expert users asking them to highlight the main shortcoming of the system.

Despite, the good feedback received, we think that the visualization module can be further improved in future. The pen-like device exploits the user's drawing capabilities and makes him able to become rapidly familiar with the system. Instead, when the system is used for sketch-based modelling, the absence of the real object, and thus of physical reference points, makes it difficult to trace lines correctly. It often happens with large or very detailed objects. We are evaluating the opportunity to employ new 3D visualization technologies, for instance a head-mounted display like Oculus[7], for a fully immersive virtual reality experience. In the next chapter, we will briefly explore low cost 3D display technologies that we tried to employ to solve a similar problem.

[7] http://www.oculus.com

# 3

# 3D modelling: interaction & visualization

*After all, a work of art can not be achieved with ideas, but with his hands.*

Pablo Picasso

The human being has always been accustomed to use his hands to get to know and interact with the world. It is a capability acquired at birth, that allows to use and manipulate objects. We are not necessarily talking about sculptors, painters, artists or craftsmen, but of all those people who daily use hands to complete tasks.
Technology solutions have always tried to bridge the gap between the digital and the real world, also as regards the interaction and visualization activities. The goal becomes much more interesting when trying to create a virtual environment quite similar to the real one, characterized by a three dimensional space and a depth perception.
This chapter will introduces the work done to evaluate how interaction and visualization technologies can be used into a 3D modelling environment. In particular, which kind of help they could give to a common user, beyond the usually gesture-based interaction.

## 3.1 Overview

Interaction and visualization have always been two complementary fields and the innovation steps have always come hand-in-hand. The simple consideration is that a new visualization mode often allows to interact in a different way, similarly new interaction devices allow to trigger different visual behaviours. We can do two examples among many possible. The first one regards the advent of 2D graphical user interfaces, which contributed to the spread of 2D input devices (the mouse is the most famous). While the second one is more recent, it concerns the devices that integrate touchable displays and cameras, which gave rise to a plenty of virtual and augmented reality applications.

Similar considerations can be made with the introduction of the third dimension, which has mainly affected the interaction field. It was a new challenge for researchers. They came to realize that 3D interaction applications had some fundamental differences with traditional one. In that case we witnessed to an evident diversification of the proposed solutions, mainly due to the heterogeneity of available hardware devices [19].

Starting from these two definitions: *"a 3D interaction is any interaction in which the user's tasks are performed directly in a 3D spatial context"* and *"a 3D user interface involves 3D interaction"*, Doug A. Bowman tried to classify application in which you can find 3D interaction, considering various technological contexts [18].

- **Desktop Computing modelling tool**: users can directly specify the object's orientation and position in the space, using 2D device in conjunction with 3D manipulation techniques;
- **Virtual Environments/Reality**: users can explore (fly through) a 3D virtual world, using 3D pointing in the desired motion direction;
- **Augmented Reality**: a physical object can be associated to a virtual object, allowing the latter to be selected, moved, and placed in the physical world;
- **Large-Screen Displays**: users can zoom into a particular area of a map simply by looking at that area;
- **Ubiquitous/Pervasive Computing**: users can copy data through different displays, by making a gesture indicating the copy direction and the copy action.

An application can anyway belong to more than one category, for instance a desktop computing application can include virtual or augmented reality characteristics.
Typically a 3D interaction can be made by gestures, if a predefined movement triggers a related behaviour of the system, or by direct control of the object,

in that case the user quite literally controls objects with his hands. A 3D interaction system of the second type represents a more interesting challenge, it can provide a means to interact with applications in a way that leverages on users' interaction skills exercised everyday in real world.

The 3D visualization is often employed to improve the understanding, analysis and exploration of data or to involve the user in a immersive entertainment experience. Many other fields are experiencing 3D visualization solutions as the stereoscopy or the volumetric displays, especially thanks to several emerging new technologies. The main reason is that it allows to improve perception, dimension and relative position of the information displayed. Even in that case, there are different methodologies to implement a 3D visualization system and typically they can be classified in two different classes.

- **Virtual Reality**: the system creates a 3D virtual environment where the user can navigate or interact with the objects. It could be a full immersion system, if it denies to the user any eye contact with the real world, for instance like head-mounted display or computer assisted virtual environment (CAVE);

- **Augmented Reality**: the system creates a 3D environment immersing virtual objects into the real world. It allows to compare virtual objects and real ones and enrich the real world with more different layers of information.

The main difference, with respect of the past, is that there is not a general purpose solution and according to your goals you can test different combinations of 3D interaction and visualization methodologies.
We recall that the main goal of our work was to study solutions to create a 3D modelling environment, suitable to a common user who does not have great experience. Hence we tried to verify if would be possible integrate 3D interaction and visualization solutions into a simple 3D modelling environment. In particular, the low-cost system should have provided a functional subset of the typical modelling operations. A "desktop computing" interaction system endowed with a simple 3D visualization assists the user that directly controls the objects with the hands.
We decided to give greater emphasis to the interaction side of the system, adopting the LEAP Motion Controller as input device. While, for the visualization side, we tested different solutions in order to verifying which can meet our needs.

## 3.2 State of the art

Although Human-Computer Interaction (HCI) is a relatively new discipline, the first researches were conducted in the 1980s, it is a very wide research field that involves computer scientists, engineers, psychologists and sociologists [57]. HCI regards the study and design of the interaction between users and computers. Despite it has evolved considerably over the years, only recently the HCI is showing more and more its potential. Thanks to a variety of new technologies, which aim is to improve the interaction and user experience, each HCI system tries to increase the efficiency of the users and reduce the training.

A 3D interaction system can be classified as a HCI system where researchers are interested in exploring new devices, experimenting new interaction paradigms and designing new software systems. During the years various interaction methodologies have been developed for 3D interaction systems. A more simple and clear classification is done in [43], where three classes were identified: *navigation*, *selection and manipulation* and *system control* and each of them can be achieved using different input solutions, for example mouse, keyboard, multi-touch input device as well as the hands.

We paid special attention to solutions that concern 3D modelling. Even though there are very interesting works about modelling with multi-touch interfaces [25, 83], with respect to our goal, we are interested in hand-based interaction, also known as Natural Interface [31]. The main reason is that this kind of interaction is the most similar to what the user is accustomed to do in real life. Moreover there are situations in which two degrees of freedom (DOFs) of the classic devices (mouse, keyboard or multi-touch input device) are not quite compatible with three dimensions space. In [84] they classified hand gesture techniques using three fundamental phases: detection, tracking and recognition. The video-based detection phase allows to recognize the hands from the background, starting from their observation through an optical equipment (infrared, thermal or classic camera). In practice, the segmentation can be carried out using several types of visual features such as skin color, shape, motion and anatomical models of hands [121], but also using some slightly more "invasive techniques", like gloves with a particular color distribution [54, 115] or with particular markers or sensors [22, 53].

Among the first works there is Surface Drawing [92], a system for creating organic 3D shapes, that facilitates the early stages of creative design. It is composed of a glove to draw surfaces, a tongs to move and scale the object, an eraser tool to remove geometry, and a magnet tool to produce small deformations. All these components, used with hands in the space, create a semi-immersive virtual environment. Despite it was a very pioneering work, equipped with many tools, it highlighted the potential of these solutions. Even in [62] they identified a weakness in the 3D modelling systems in supporting the designers' conceptual process, and they proposed an innovative user interface integrating data glove and 3D motion sensor. Combining real time actions and gestures, the proposed system creates an immersive visual design environment and makes available a limited set of operations: scaling and creation by template (sphere, cylinder, cone and cube). In [52] they obtained remarkable modelling results coupling gloves and stylus. The system employs a brush-like paradigm in order to manipulate triangle meshes. While the paradigm presented in [114] helps the user to carry out 3D assembly tasks, typically used to construct a set from a collection of props or to assemble mechanical pieces into a machine. They combine hands tracking, without gloves or markers, with the use of keyboard and mouse. The resulting system relies on a small set of gestures, easy-to-use for the final unprofessional user. Even in [32] they tried to enhance the computer-aided design (CAD) technical drawings, providing simple functionalities to select and control the camera. Hence, an untrained user can explore a 3D model using his hands, acquiring information about the whole model or each its parts. In [104] a novel handle bar metaphor was proposed as a visual control metaphor between gestures and operations. Through the Microsoft Kinect device they implement a simple set of gestures to control the object (rotation, translation and scale) and the camera (pan, rotation and zoom) in the scene. Despite the growing popularity of virtual environments a semi-immersive environment appears to be a good choice in [28]. They developed an interaction techniques based on asymmetric bimanual interaction, providing a familiar ways to conceive, create and manipulate three-dimensional shapes. The obtained results are remarkable, but the whole system is quite cumbersome to be adopted by common users. The Shape-It-Up system presented in [72] exploits a novel paradigm wherein the interpretation of gestures directly deduces the designer's intent and the subsequent modelling operations. In particular, the modelling scheme used is known as intelligent generalized cylinders, and allows to quickly create a variety of constrained and free-form shapes, while retaining the aesthetic characteristics of the shapes. A very interesting work is presented in [103], they propose a precise hand, finger and gaze tracking control for CAD modelling. Their aim is to preserve the usability level with low user fatigue, and maintain a high level of intuitiveness. The provided operations allow to perform some simple modelling tasks, like translation, rotation and zoom. The common thread of these works is summarized in [27], where a hierarchical gestural interface using Kinect is implemented to create a voxel based representation of virtual clay. The system brings out the intuitiveness of traditional clay

modelling into a virtual CAD environment, and using gestures the users can carry out some simple modelling tasks.
This kind of methodologies have also been employed in other fields such as education and learning, not only in pure CAD applications [50, 51]. The reason is that Natural Interface helps the user to interact with objects in a natural way and as she/he is used to do with.

Often 3D interaction applications implement a 3D visualization solution to create a immersive or semi-immersive 3D environment [19]. Stereoscopy is the more frequent choice, and it can be classified into two categories: active stereoscopy, if the user wears goggles that interact with the display, and passive stereoscopy, if the goggles filter the visual flow to the appropriate eye [96].
There are also other generally unused methods belonging to the autostereoscopy class, like Computer-Generated Holography (CGH) [80] and volumetric displays [96]. Even if it has already shown its potential, CGH is still in an evolution stage. It requires a high number of pixel and a fast generation techniques. While volumetric displays are quite expensive and require very large amount of bandwidth to obtain good quality images.

Therefore, it is not a coincidence if many of the works mentioned above are now proposed again, taking into account the visualization solution adopted. In [28, 52, 92] active stereo shutter glasses combined with ahead-tracking system are used to provide the correct perspective image at any location. A polarized 3D system with passive filter glasses are used in [73] to create the illusion of three-dimensional images. Head-mounted display (HMD) is another example of passive stereoscopy, and in [32, 58] is combined with markers to create a augmented reality 3D interaction environment. In [76] they compared a half-transparent mirror display, coupled with shutter glasses, with a holographic optical element display, that uses several projectors, to verify co-located issues between virtual objects and hands movements. A really innovative approach is presented in [67], where a laminar curtain of fog is used as 3D screen, enabling touch-through and see-through activities.

It is clear that 3D interaction and 3D visualization are orthogonal methodologies, where the first one represents the input of the system and the latter the output. However it is not so trivial find the right trade-off between them, in order to provide a comfortable working environment [20, 55].

In our work we proposed a 3D modelling environment based on the 3D interaction paradigm. The main goal is to avoid cumbersome equipment or extra stuff (e.g. gloves, tools) that can restrict the usability, but at the same time the user should be able to punctually interact with the objects in the scene. The open issue regards the possibility to use the 3D interaction de-

vices to perform precise modelling operations, and not only wide triggering gestures.
To help the user during the interaction phase, we studied some 3D visualization methodologies. Once again our purposes are to avoid intrusive and expensive solutions (e.g. head-mounted display, CAVE) and propose some visual cues which help the user during the 3D modelling session.

## 3.3 The 3D modelling environment

As we seen there are a considerable variety of experiences that use tools or hand gestures to control 3D object. Hence, what is new to propose another 3D modelling environment?
Nowadays, optical tracking is best technological solution that you can adopt to follow the user movements. In particular, the hardware solutions are regularly growing, and they provide equipments able to track, with an higher accuracy, hands, fingers and tools. Hence, the main reasons that drove us to propose a new 3D modelling environment are:

- evaluate if the more accuracy can be exploited to define not only gesture-based actions but also punctual actions;
- evaluate if it is possible define actions that use fingers as punctual tool to operate on the object;
- determine which old and new modelling operations can benefit from more accuracy and the hand-finger tracking;
- define a paradigm that does not require an intensive training and can be used by any users.

To address these goals, we selected the LEAP Motion device (Figure 3.1). Its accuracy and good trade-off between features provided and cost allowed to created a prototype of a 3D modelling environment. The device fits our need and can run on any commercial personal computer without other sophisticated hardware. We exploited the new hardware in order to define both navigation and interaction features. In particular, we fixed a virtual working volume above the LEAP Motion device (its tracking operative space Figure 3.2(b)) and we defined gestures and punctual actions to control each objects in the 3D environment. The prototype was created using C++ language and OpenGL library.
In the next sections we will briefly introduce the characteristics of the chosen hardware, then we will explain which interaction actions we decided to test.

The LEAP Motion device and its driver are constantly updating, thus it is possible that some of the solutions here described are, or will be, provided in the next releases.

Figure 3.1: The LEAP Motion device, whose dimensions are $76.2 \times 25.4 \times 12.7\ mm$.

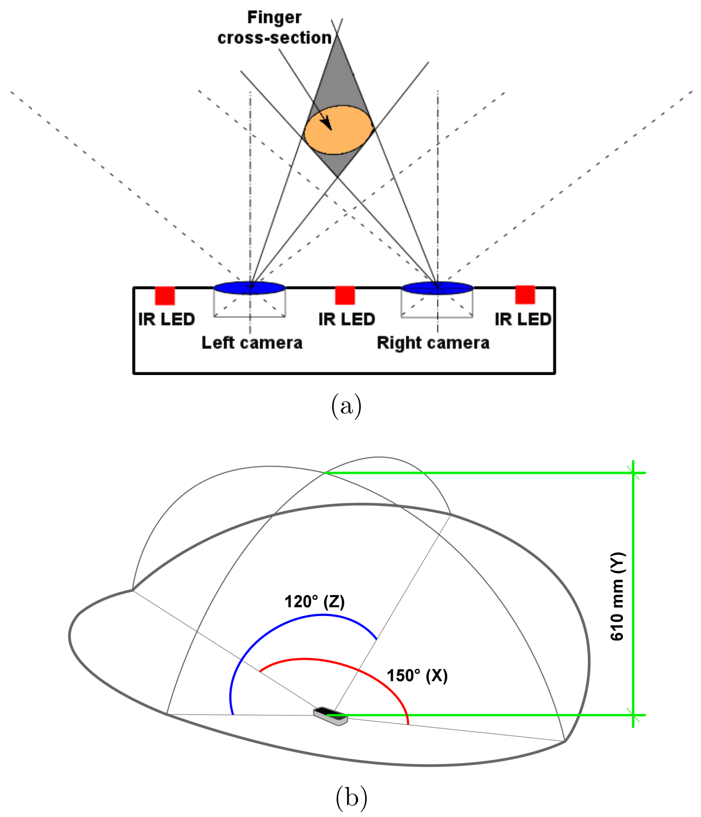


Figure 3.2: A schema of the LEAP Motion hardware components (a). The observed volume by LEAP Motion (b): the X axis is longitudinal from left to right, the Z axis is transversal from the personal computer to the user and the Y axis comes out upwards from the device. The device defines the origin of the coordinate system.

### 3.3.1 Hardware setup

The LEAP Motion peripheral device, connected through USB port, is small enough to be placed in front of the personal computer, typically facing upward (Figure 3.1). LEAP Motion device uses two monochromatic infrared cameras and three infrared LEDs (Figure 3.2(a)) to monitor a roughly hemispherical area whose sizes are 610 *mm* high, 150° wide and 120° deep (Figure 3.2(b)).

The two cameras generate almost 300 frames per second containing the 3D pattern highlighted by the LEDs, while the driver analyses the stream data and fills a data structure with all the information about the objects detected above the device. The data structure, named *Frame*, describes each single empty or full frame detected, and contains information about hands, fingers, finger-like tools and gestures. Among the recently introduced features inside the *Frame* structure we can even find: a single greyscale image from each cameras, the estimation of the forearm and the estimation for each fingers of the four bones that make up the its anatomy. However, we do not explore these new features yet.

The *Frame* data structure is very rich, but for our purposes we used a subset of the information that regards fingers, hands and gestures. We report them in Tables 3.1, 3.2 and 3.3, in order to demonstrate how a good interaction level can still be achieved with few features. The drivers are constantly updating, hence these lists could be out-of-date by the time you are reading this dissertation. Each position vector determines a point in 3D space overlying the device, and the offset from the origin, which is defined by the device, is given in millimetres.

| Fingers methods | Description |
|---:|---|
| *id()* | A unique ID assigned to the finger.<br>It is valid until the finger is visible. |
| *hand()* | The hand associated to the finger. |
| *isExtended()* | Return true if the finger is extended straight from the hand,<br>otherwise it is bent down and curled towards the palm. |
| *direction()* | The direction in which the finger is pointing. |
| *tipPosition()* | The position of the tip's finger. |

Table 3.1: A subset of the information provided by the LEAP Motion device regarding a finger.

| Hand methods | Description |
|---:|---|
| *id()* | A unique ID assigned to the hand.<br>It is valid until the hand is visible. |
| *frame()* | The frame associated with the hand. |
| *fingers()* | The fingers list attached to the hand.<br>It is in order from thumb to pinky. |
| *translation()* | The translation vector between the<br>current frame and the specified one. |
| *palmPosition()* | The palm center position in millimeters. |
| *palmNormal()* | The normal vector to the palm. |
| *stabilizedPalmPosition()* | A stabler position vector. |
| *translationProbability()* | The probability that the translation vector,<br>defined between the current frame and<br>a specified one, identifies a translational motion. |

Table 3.2: A subset of the information provided by the LEAP Motion device regarding a hand.

| Gesture methods | Description |
|---:|---|
| *id()* | A unique ID assigned to the gesture.<br>It is valid until the gesture is running. |
| *type()* | The gesture type among: circle, key tap, screen tap and swipe. |
| *hands()* | The list of hands associated with the gesture. |
| *pointables()* | The list of fingers associated with the gesture. |
| *position()* | The position of finger that performs the gesture. |
| *direction()* | The direction of finger that performs the gesture. |
| *radius()* | The radius of the circle gesture. |

Table 3.3: A subset of the information provided by the LEAP Motion device regarding the recognized gestures.

The LEAP Motion device recognizes four user's gestures by default: the *circle*, if a finger traces a circle (Figure 3.3(a)); the *key tap*, if a finger performs a tapping movement, as tapping a keyboard key (Figure 3.3(b)); the *swipe*, if a finger traces a long and linear movement (Figure 3.3(c)) and the *screen tap*, if a finger performs a vertical tapping movement, as tapping the computer's screen (Figure 3.3(d)). Gestures are not only reported through events, but rather through several configurable parameters inside the *Frame* data structure, during the whole gesture's lifetime. This allows to tune each single gesture and to implement more complex callback actions above them. In the next section we will show you how, using *circle* and *tap key* gestures and some few parameters like position and direction, we implemented some simple and intuitive interaction actions.

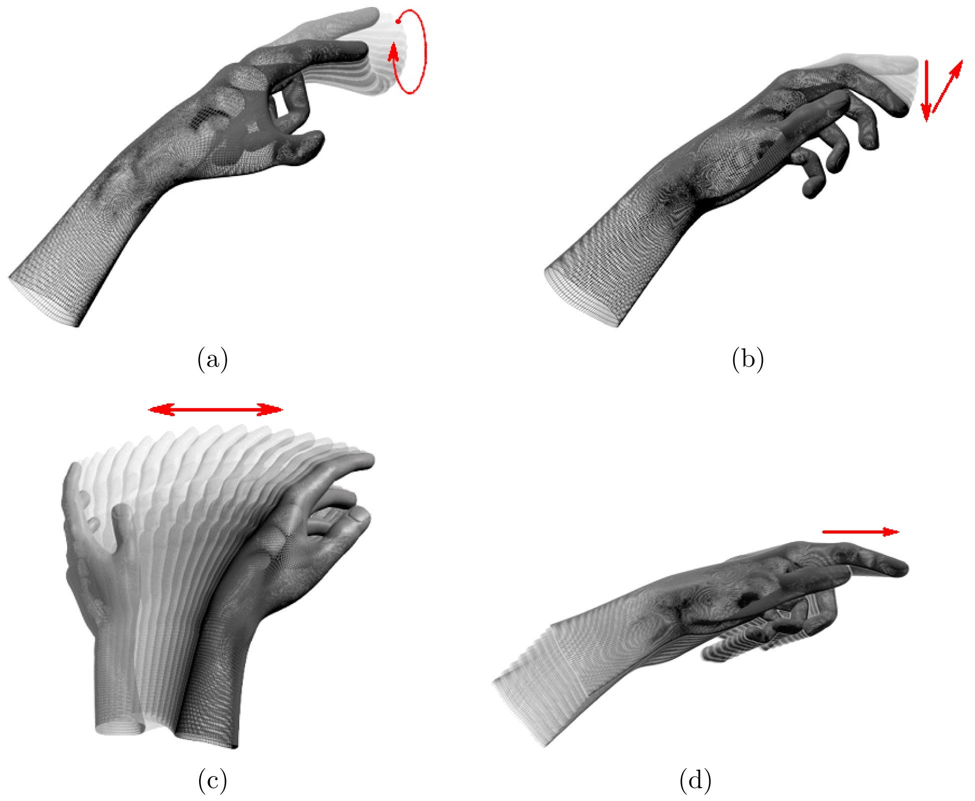


Figure 3.3: The gestures recognized by the LEAP Motion device: clockwise and counter-clockwise circle (a), downward tap (b), linear swipe (c) and forward tap (d). These images are taken from the LEAP Motion developer guide.

### 3.3.2 Interaction operations

**Environment.** First of all, we needed to define the 3D virtual environment where place the virtual objects. We decided to keep fixed the volume defined by the LEAP Motion device and align the virtual OpenGL environment to it. We performed some tests with common users and we detected that it seems

to be the most intuitive solution. The user quickly learns that the objects, visualized on the 2D screen of the personal computer, are apparently floating in mid-air above the device (Figure 3.4).
Despite the field of view of the two cameras is a wide hemispherical area, we can not consider it as an uniform parallelepiped, near the horizontal xz-plane the device significantly reduces its detection capability. Therefore we need to clip the volume observed by the device to ensure a higher stability of the system. In practice, the clipping was done shifting downward (about 200 $mm$) the xz-plane of the device with regard to the OpenGL xz-plane. The resulting working volume, whose dimension are approximately $400 \times 450 \times 400$ $mm$, allows a good freedom of interaction and natural movements, and the user finds quite familiar and not limited this environment.
We will show later how navigation actions can benefit of the clipping operation of the working volume, and how they can modify the relative position of the two spaces (device's volume and application's volume). The change will only affect a rotation of the vertical axis in order to keep fixed the horizontal xz-plane.

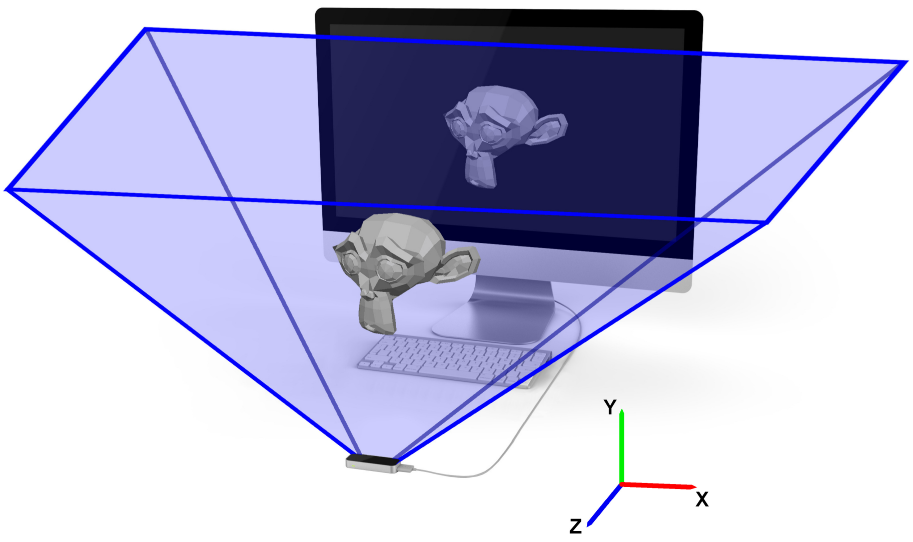


Figure 3.4: The 3D virtual space, visualized on the 2D screen, is aligned with the 3D volume of LEAP Motion device. The user has to imagine to see a mid-air floating object.

**Navigation.** The next phase regards the definition of the navigation features. In particular, with navigation we mean the camera controls that allow to visualize the objects from different angles. Typically, in the 3D modelling

applications these operations are performed using mouse or a key combination of the keyboard. The main issue regards the limited movement capabilities of the classic interaction devices, for instance the 2D displacement vector of the mouse needs to be transformed in a 3D movements to orient the camera, but the resulting movements are quite unnatural and require several iterations to find the right position of the camera.
Exploiting the accuracy of the device we decided to use a single extended finger to control the camera (Figure 3.5): the finger's tip position defines the camera position, while the finger's direction defines its orientation. While, with an open hand moved up and down you can define the zoom in and the zoom out operations. With OpenGL functions we can easily implement these ideas, in practice we used the position of the finger's tip as "look at" vector, the finger's direction as "look to" vector and a translation of the open hand among two consecutive frames as delta to update the zoom.
These choices allow to easily navigate around the objects and even users, that do not have experience of modelling environments, have found this solution very useful: they can explore the 3D model with few intuitive gestures. The user has always to imagine that the object floats in the mid-air above the device. In that case he can use the pointing finger metaphor to change the angle of view or the open hand as zoom tuner (preserving the angle of view).

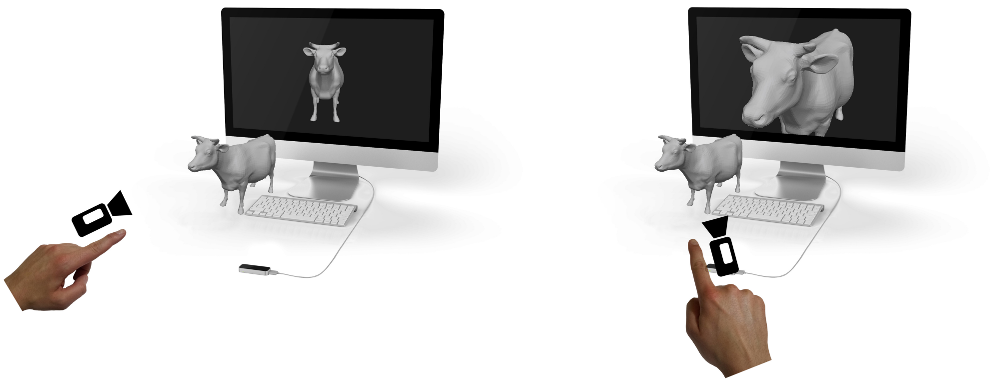
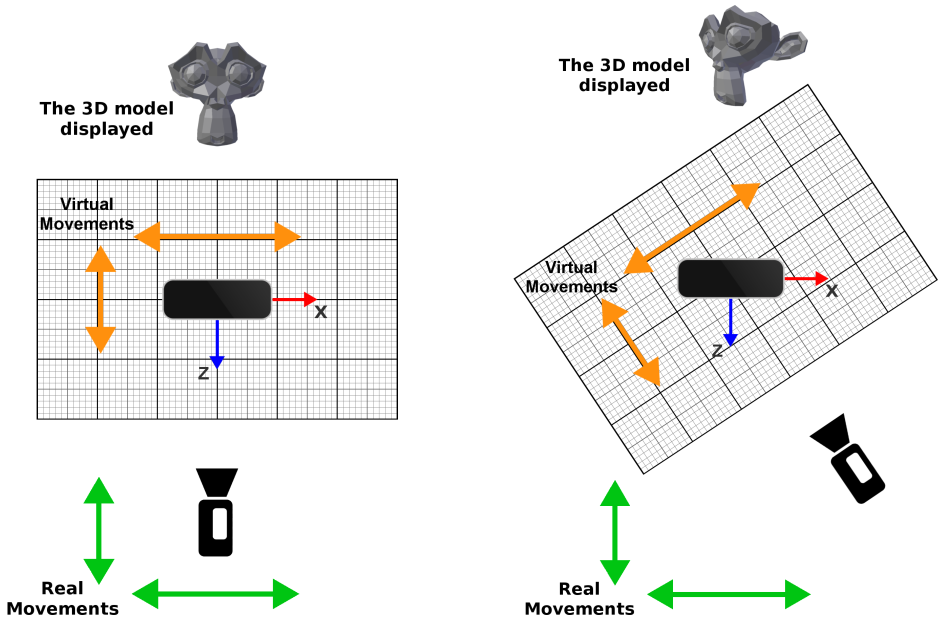

(a) (b)

Figure 3.5: An example of how you can control the camera using the metaphor of finger pointing (where framing). The cow that floats above the device is what the user has to imagine to see and he can use the extended finger to control the camera's position and orientation. On the 2D screen he can see what he frames with the finger.

The rotation of the camera around its y axis (the look up vector) implies a rotation of the horizontal xz-plane, this assumption helps to keep coherent

the movement above the device without move it.
As you can see in Figure 3.6, if you are in the default condition, on the 2D screen you can see the object framed by the camera that is positioned in front of it, and thus your real movement are exactly replicated in the 3D environment (the green and orange arrows in Figure 3.6(a)). Otherwise, if you move the camera, in order to frame the scene from another angle, on the 2D screen you can see the object framed from that angle, but each horizontal movement, detected by the device, is rotated before being shown in the virtual environment (orange arrows in Figure 3.6(b)).
In the latter case, the user, after placing the camera, may continue to perform the movements in the same direction that he would have performed in the default condition. The virtual movements are coherently with the 3D virtual environment displayed, without rotate the device.



(a) (b)

Figure 3.6: An example of xz-plane rotation. On the *left* the default condition, the camera frames the scene from the front and the device replicates the real movements (*green*) in the virtual environment (*orange*). On the *right* the condition after the rotation, the camera frames the scene from three-quarters and the device applies a rotation to the real movements (*green*) in order to align them with the virtual environment (in *orange* the rotated movements).

**Global interactions.** Up to now we have only defined a 3D virtual environment, where the camera is controlled in a more natural way. Hence we decided to explore some pure interaction activities. The goal is to verify if the user can use his hands to capture, translate, rotate or scale objects in the scene, as he would do in real world.
First of all, to help the user to recognize where his virtual hand is located with respect to the virtual 3D model (the position is defined by the 3D position of the real hand detected by the device), we drew a simple sphere for each of the fingers' tip. We recall that the user has to imagine the floating object and he can only see the 2D screen of the computer.

Typically in the real world, to pick up an object we approach the open hand to it, and then we grasp the object closing the fingers on it. Once the object is grabbed, we rotate or translate the hand to inspect or move it in the desired position. Simplifying this process, we can assume that only three fingers are necessary to capture and control an object (with two finger you can not correctly define the rotation), and we decided to evaluate if this method, based on three fingers, can be successfully adopted in our system. We have identified four features to control the object: *Capturing*, *Translation*, *Rotation* and *Scale*, and we implemented them using the three fingers paradigm.

- *Capturing* - Using the position of the three fingers' tip we defined: *(i)* a distance $\mathcal{D}$ between them as $\sum_{i=1}^{3} d_i$, where $d_i$ is the distance between the finger $i$ and its subsequent; *(ii)* a threshold $\mathcal{T}$ for this distance and *(iii)* the relative barycenter $\mathcal{B}$ of the three tips. Typically, the three fingers naturally used are thumb, forefinger and middle finger (Figure 3.7(a)). The capturing operation has been implemented as follow: if the barycenter $\mathcal{B}$ falls inside the bounding box of the object and the computed distance $\mathcal{D}$ is smaller than the threshold $T$ then the object is captured. To release the object the user can open the fingers increasing the distance $\mathcal{D}$ among them.

- *Translation* - This operation is simply derived from the condition generated by the previous operation. A captured object is moved in the 3D space in accordance with the movements carried out by hand (Figure 3.7(a)). In particular we use the vector defined by two consecutive positions of the barycenter $\mathcal{B}$ to translate the object. The movements respect the xz-plane rotation as shown in Figure 3.6.

- *Rotation* - The three fingers used to capture the object define a plane P and we are interested in two vectors: the first one lies on the plane P and is defined as difference between the position of two generic fingers, while the second is defined as the normal vector of the plane P. Using these two vectors we know the rotation of the user's hand and it is possible define a quaternion that represents each possible rotation of the object.

- *Scale* - Once again this operation is only applicable to captured objects and requires two hand: one to keep the object and one to define the scaling gesture. Indeed, the second hand is used to perform a circle gesture (Figure 3.3(a)) which radius defines a multiplier factor to control the scaling. While the spin direction (clockwise or counter-clockwise), with respect of the finger's direction that perform the gesture, defines the scale enlargement or reduction.

This first proposal implemented very natural interaction movements that reflect the movements normally made in everyday life. However, there were several hidden problems that made the system slightly unstable and sometime non-functional.
The threshold $\mathcal{T}$, required by the *Capturing* operation to identify when an object is grasped, is strongly influenced by the hand's anatomy of the user who is making the gesture. We evaluated several solutions: the simplest one is to reduce to zero the threshold. It means that the three fingers' tips must touch themselves to capture the object, but it disables the rotation such as proposed. An alternative could be to use a dynamic evaluation of the threshold, in order to calibrate it for each user. However, the main problem concerns the static position to maintain with the fingers, it can be tiring and cause too much flickering in the barycenter's position and then in the *Translation* operation.
The *Rotation* operation suffers one more time of occlusions. The three fingers used to capture the object (thumb, forefinger and middle finger) can occlude one to each other if two of them and at least one camera of the device are collinear (Figure 3.7(b)). It is a source of instability for the system, and causes stress for the user to maintain the position or to place the object.
The problem that affects the *Scale* operation is more conceptual than the previous ones. The device is able to detect each single hand that enters in its working volume, as a result we would potentially be interested to control each single object with only one hand and then also two objects simultaneously. Hence this operation, that requires necessarily two hand to be performed, has to be designed again.

The problems encountered with the first proposal have led us to conclude that intuitiveness not always means functionality. Hence, we decided to study a second solution more prone to the interaction's capabilities. Recalling that, despite the system does not want to stand out as a professional CAD environment, we need to control the objects with a good level of accuracy.
In that case we decided to be inspired by the paradigm of the graphical user interfaces for selection and interaction. In particular, the user can select which

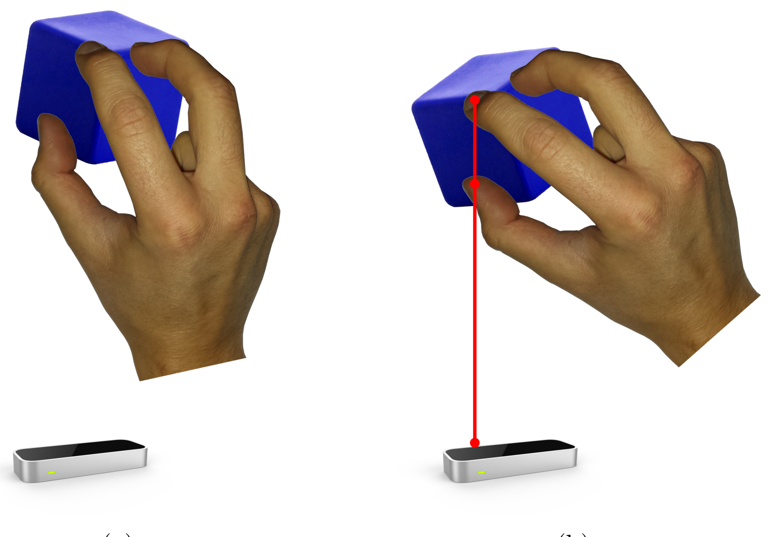


Figure 3.7: On the *left* the cube is captured and it can be translated, rotated or scaled. To set free the cube the user has to open the three fingers. On the *right* an occlusion case, two fingers and the device's camera are collinear (red line), it is a high source of instability for the system.

object wants manage using one finger and this operation binds the object to the hand to which the finger belongs. It recalls the selection procedure usually done with the mouse. As a result, each operations has undergone an improvement with respect to this new metaphor.

- *Capturing* - This operation can be done using the key tap gesture (Figure 3.3(b)). If the finger performs a key tap inside the bounding box of the object it selects the object and a visual line defines the link between the hand and the object. After the selection, the hand is free to move, and the object is associated to it until the hand is visible in the device's working volume. To deselect the object, the user can perform a second key tap gesture.
  Since the key tap gesture is difficult to be performed with small objects, we also introduced a keyboard key to replace the tap. Again, it works only if the finger's tip falls inside the object's bounding box.

- *Translation* - A bound object can be translated adopting a specific hand's configuration. The fist (all the fingers are closed) allows to control the position in the space of the object. It is comfortable to maintain and it is not subject to instability due to size variations. Even the fist recalls the position adopted to capture physical objects.

- *Rotation* - The rotation of a bound object can be control with a hand with four fingers opened. We chose this hand configuration for two reasons: it is quite simple hide the thumb under the palm, and the hand with five fingers opened can be confused with the position assumed after the fist configuration that controls the translation.
  The hand with four fingers opened has a normal and direction vector. The first one defines the pitch and roll angle, while the latter defines the yaw angle. The largest angle between pitch, roll and yaw is used as multiplier factor to rotate the object in the same direction (Figure 3.8). A greater inclination angle of the hand imposes a greater rotation to the object.

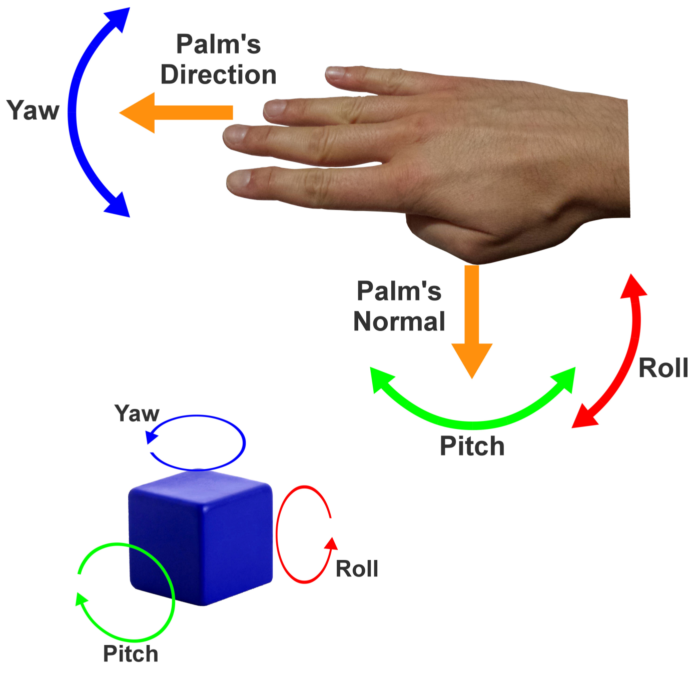


Figure 3.8: The direction and normal vectors of the palm define the three rotation direction (pitch, roll an yaw). Tilting the hand, with four opened fingers, the user can control the rotation direction and the multiplier factor: a greater inclination imposes a greater rotational.

- *Scale* - This operation can only be performed on a captured object. The main advantage is that it can be carried out by the same hand to which the object is bound, and thus the user can simultaneously scale two

object using two distinct hands. Even in that case, performing a circle gesture with one finger (Figure 3.3(a)) we can define the scaling factor (a greater circle's radius imposes a greater scale) and the scale direction (enlargement or reduction). With one finger the user controls the scale in all directions.
We also decided to allow the scale operation for each axial direction, and it can be activated performing the circle gesture with two fingers (forefinger and middle finger). The direction of the fingers defines the coordinate to be scaled (Figure 3.9).

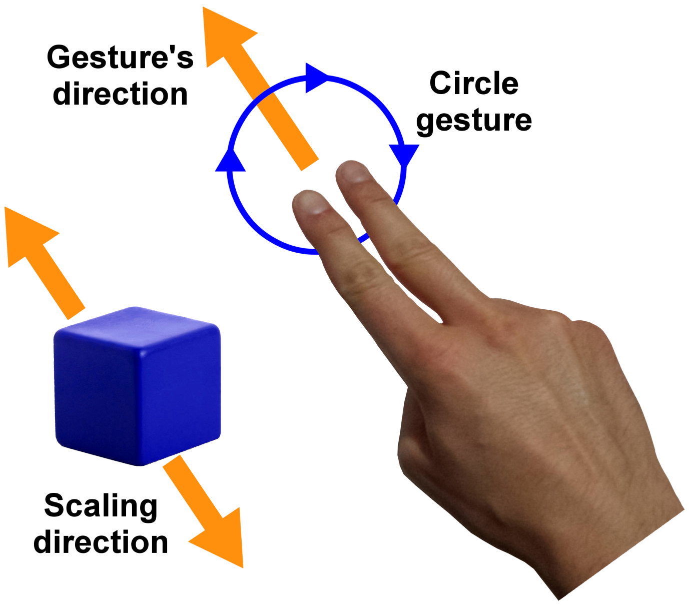
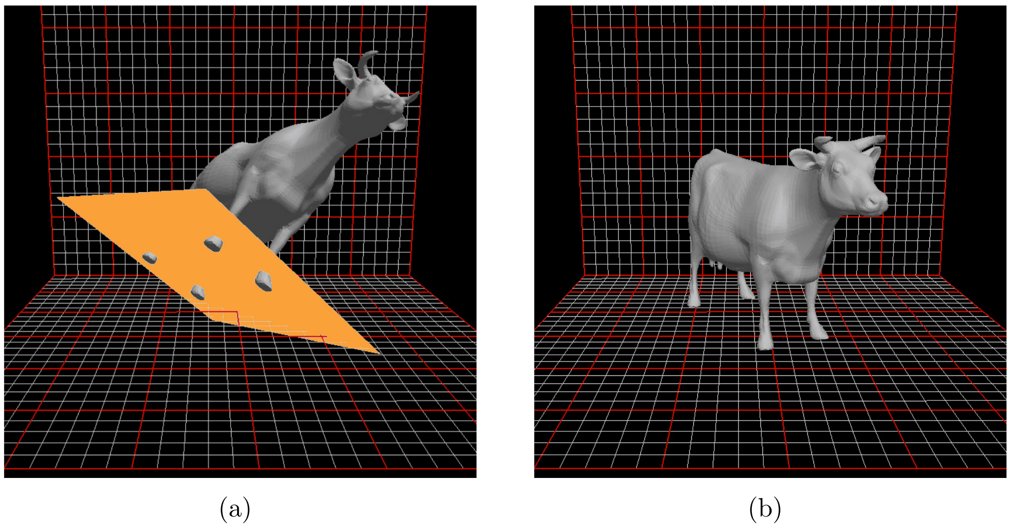


Figure 3.9: An example of scale along one direction. The circle gesture performed with two fingers defines the scale direction and the multiplier factor: a greater radius imposes a greater scale. Instead, using only one finger the object is scaled in all direction.

- *Object alignment* - In this new implementation we introduced a new feature, an alignment procedure. It may happen that by loading a model in a 3D environment it is oriented in a wrong way. This is due to two main causes: a difference between the reference system of the modelling environment in which the model was generated and the reference system of the current 3D environment or all the global transformations previously applied.
Typically, the user easily realizes of this misalignment, but he can only intervene using the classic global rotation tool. The main shortcoming is

that it requires several interaction between the object's position adjustment and the camera's movements, in order to reach the final desired position.
We decided to exploit the plane defined by the hand as the reference plane for alignment procedure. The user places the hand, with five opened fingers, in the position that will indicate the final horizontal plane for the model (Figure 3.10(a)) and then activate the alignment procedure with a keyboard key. The system computes the transformation matrix M from the current horizontal plane and the plane defined by the hand, and then applies M to the object (Figure 3.10(b)). As a result the object is rotated, in order to reach the final correct orientation with respect to current horizontal plane.



(a) (b)

Figure 3.10: An example of alignment procedure. The cow on the *left* is misaligned with respect of the xz-plane and the user defines its local horizontal plane using his hand (the light orange plane). The paws jut out from under the plane, but do not affect the procedure. The system computes the matrix M and realigns the cow, as shown on the *right*. The red&white grid identifies, respectively, the horizontal and the front planes.

This second solution has been proposed to some users with limited experience in 3D interaction systems, in order to evaluate its intuitiveness and functionality. Although the first solution strongly takes into consideration the intuitive aspects of the interaction, while the second one is more oriented toward the functionality of the proposed operations, we collected positive feedback by the users. They become familiar with the system after a short training, during which we briefly explained how it works and which gestures perform to control the objects. Furthermore, they have detected that the rotation and scale operations allow to still have a good control and precision of the result, without providing exact numerical factors.

**Local operations.** The objects that we are using are made of vertices, edges and faces, and generally it is called mesh. It is a particular discrete representation that describes the surface of the object using 3D points, the vertices, and a grid of edges to define polygonal faces. The operations presented in the previous section allow to operate on the whole object. They act on the entire set of 3D points that define the structure of the object, and they are very useful to explore, to modify the shape, to compose the scene with more objects or to carry out 3D assembly task.

However, these kind of operations are not suitable to make local changes on the object. With local operations we mean a specific category of features that allow to directly modify each single vertex of the object. They allow to operate on a restricted patch of the surface in order to enhance local details, without dealing with the whole surface.

After establishing the capabilities of the device we were using, we took as inspiration the work proposed in [81]. They detected that the classic 2D interfaces make 3D manipulation difficult and the usability issues discourage and do not attract users, especially the less experienced ones. They proposed a TUI for 3D clipping operations easy to use and to learn. In particular, they implemented a tangible pen and a tangible frame for clipping. The pen defines a virtual clipping plane, centered on the pen's tip and orthogonally oriented to it. While the frame directly defines a virtual clipping plane. With these two improvements they received good feedbacks by the users.

The good results obtained with the first interaction tests, drove us to think to an alternative use that could be done with hand and fingers in a 3D interaction environment. In particular, we introduced several new metaphors that can help the user to locally modify the object. The hand, previously used to control the translation and rotation operations, can now be considered as a virtual plane, and similarly the fingers can be used as virtual pointer.
Starting from these consideration we designed and tested some new local interaction features which are able to modify a patch of the object's surface.

- *Selection by plane* - One of the first needs that arises when you want to locally operate on a mesh, is the possibility to select the elements of the mesh, in particular the vertices. It is simpler select the edges and the faces using the vertices that constitute them.

This operation is often difficult and unintuitive if performed with 2D interactive device like mouse: your selection tools has limited degree of freedom or worst you can select only the visible vertices. It means that complex object requires several interaction between selection and repositioning to correctly select elements in a restricted area.
We decided to use the hand as selecting tool. A virtual plane, linked to the hand position, is defined, and the user can move the hand in the 3D space to place the plane in the right position with respect to the points to be selected. Using a simply collision algorithm we are able to select all the 3D point above or below the plane (Figure 3.11(a)). Also in that case, the plane position respects the xz-plane rotation as shown in Figure 3.6.

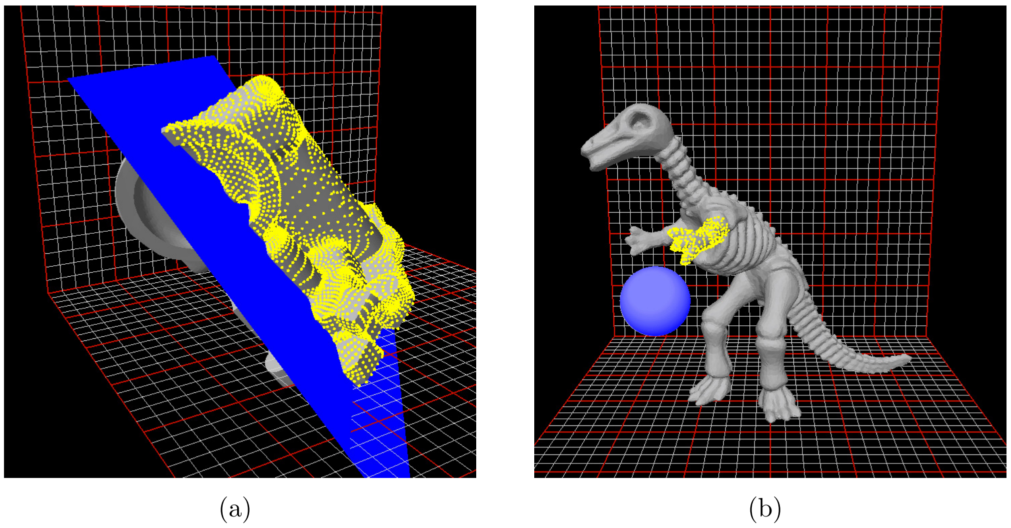


Figure 3.11: An example of selection operation, with a plane on the *left* and with a sphere on the *right*. The user moves the dominant hand in order to individualize the region of interest and then use the free hand to activate the selection. The yellow dots represent the selected vertices.

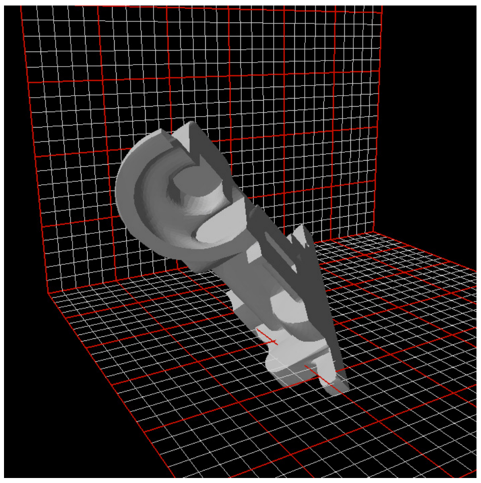
- *Selection by sphere* - The *selection by plane* is very useful, but has two shortcoming: the plane used is an infinite surface and with a plane it is difficult to select points in a concave area. For these reasons we designed another selection tool in order to operate on small and restricted surface's area.
  A finger is usually used for more precise activities in everyday life, thus we thought to reuse this idea hooking a sphere on the finger's tip. The

sphere obviously follows the position of the finger and in combination with a keyboard key can select each single vertex of the mesh (Figure 3.11(b)). If the sphere is too small or too large to reach the desired area of the object, the user can reduce its dimension with a simple circle gesture (Figure 3.3(a)).

- *Patch transformations* - The first use that can be done with the selected regions is the transformation. It includes translation, rotation and scale and in that case the user can modify only a restricted area, in order to add details or to change the local shape. This is the main difference with respect of the global transformations described in the previous section. The design of these operations (translation, rotation and scale) was carried out taking as reference the global interactions previously described. The main reasons are the usability and the intuitiveness: using the same gestures the user can control in the same way the local transformations. A fist allows to control the position of the vertices group in the 3D space, a hand with four opened fingers defines the rotation and with one or two fingers is possible scaling the vertices. The center of the rotation and scale operations is located in the barycenter of the vertices group.

- *Cut out* - The selected area can be also used to split the object (Figure 3.12(a)) or to remove undesired portions of the object (Figure 3.12(b)). Geometrically this operation consist in a restructuration of the mesh data struct in order to remove the selected vertices, but also in that case the principal issue is identify the region. The cut out operation benefits one more time of a 3D interaction environment.

- *Sculpture and extrusion* - Sculpture and extrusion identify two complementary operations (Figure 3.13). If the first one is usually used to emulate the chisel with which a sculptor removes unwanted material, the extrusion is mainly related to 3D modelling environments and allows to extract out details from the surface.
  The main problem that affects these two operation is rightly imagine the direction with respect to which you are modifying the surface. In a classic modelling environment you must place the object in the right position several time and then use the specific tool.
  We decided to use one more time the metaphor of the finger-based tool. Using a sphere on the finger's tip the user can remove or extrude the surface of the object. The advantage consists on the possibility to orient the finger with respect to the surface, without necessarily moving the object.


(a)

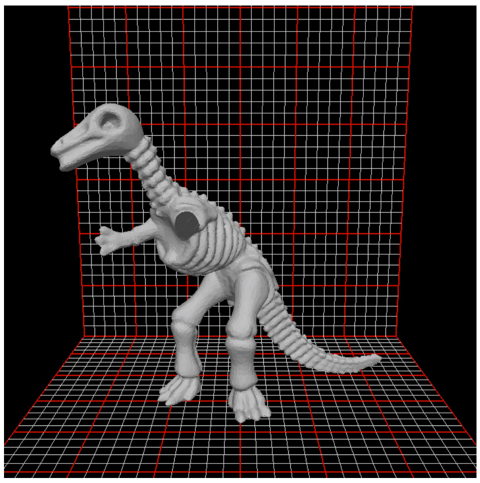
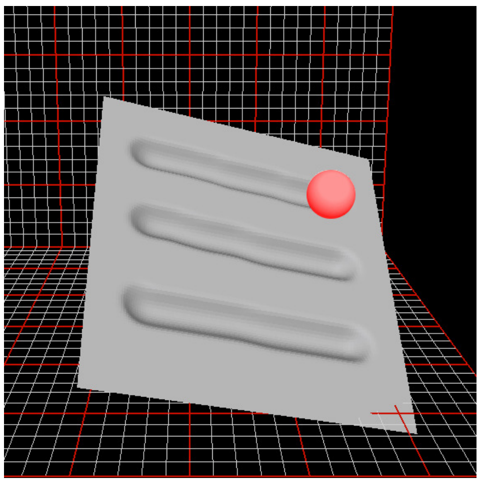

(b)

Figure 3.12: An example of cut out operation starting from the previous selected area (Figure 3.11). Using the plane linked to the hand is quit simple split the object along a desired direction (*left*). While with the sphere (*right*) the user can accurately select some portions of the model.


(a)

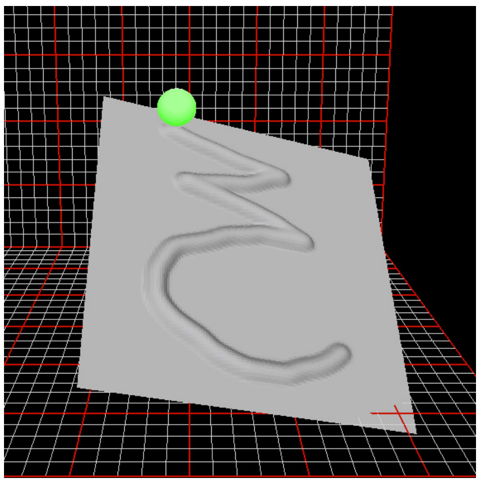

(b)

Figure 3.13: An example of the sculpture (*left*) and extrusion (*right*) operations. The sphere hooked on the finger's tip can repel or attract the object surface. The resulting detail has a hemispherical profile.

The sphere works like a magnet, it can repel (sculpture) or attract (extrusion) the surface, as shown, respectively, in Figures 3.13(a) and 3.13(b). When the user's finger is approached to the surface the nearest vertex and a set of its neighbours is computed using a radius equal to that of the sphere. The distance between each involved vertex and the sphere's center is kept constant with respect to the normal vector of the vertex. Each vertex is moved away from the sphere's center with the sculpture operation, or toward the sphere's center with the extrusion operation. The modified area will always assume a hemispherical profile, in future we would like to experiment different tool tips (e.g. cone, cube), to obtain different effect on the surface.

These local operations is a very small subset of the features that you can find in 3D modelling application, but we consider them a good test with which better understand the device's capabilities if used in a 3D interactive modelling environment. For instance, a possible extension concerns the shapes creation. Despite some previous works have not yielded significant results [92], it will be interesting evaluate if this technology can be also used for free-form creation.

### 3.3.3 Stereoscopic visualization

During the development of the interactive features of the system we detected some awkwardness by the user. Using his hands in the mid-air, the user had difficulty in understanding the relative position between the 3D models and hands. The choice to maintain the fixed horizontal xz-plane, centered on the device itself, and to allow only the rotation from the vertical axis, as shown in the Figure 3.6, has proved to be very helpful to solve the spatial orientation difficulties of the user. The possibility to take the device as a reference point partially helps the user to imagine floating objects in the mid-air above the device (Figure 3.4).

We decided to explore classic and low-cost stereoscopic technologies in order to evaluate if they could be successfully included in the system. The aim is to return a more intuitive visual feedback to the user. The choice of stereoscopic techniques has been driven by one of the main motivations of the work, it should be remembered that the idea was to develop a new paradigm using technologies and solutions within the reach of any user. For this reason, more expensive and complex solutions, like 3D shutter glasses system or head-mounted display, are excluded, even though we recognize their high innovation value.

Human beings are equipped with two eyes placed next to each other at an average distance of about 63.5 *mm*, each of which has a view of the scene from a slightly different angle. The brain takes these two poses and fuses them into one picture. In the resulting three-dimensional image the align differences are interpreted as depth information. This depth cue is very important and helps us to improve the perception of the surrounding environment or to understand the movement toward or away from us. These characteristics make the stereoscopic vision vital for seemingly simple actions such as capture and arrangement of objects.

Since several years the stereoscopic vision artificially reproduced has been considered a valid solution for graphic interfaces and assumed a prominent place also in other areas, like the video games or feature films production. Over the years different techniques, that reproduce the scheme previously described, have been developed and we can grouped them into four class.

- **Colour filter glasses -** It is one of the oldest methods of viewing 3D images and works using different colour filter to feed each eye with different images starting from a single anaglyph source. The colour theory has formed the background knowledge to test different filter combinations and the two main pairs of colours used are red-cyan and red-blue. The lack of colour fidelity is the main flaw of this technique, especially in the presence of saturated colours. Indeed, the best results are obtained with black and white images. Moreover anaglyph technique often incurs in the mutual interference problem that causes ghost image effect. For these reasons it has been replaced in many application.

- **Polarized filter glasses -** Currently, this method is the most commonly used. In that case the common starting image is filtered using glasses which have two polarizing lenses whose polarization directions differ of 90°. In that way the visible image for an eye results completely black for the other.
  This method, unlike the previous one, requires a screen able to emit polarized light. In addition, employing linear polarization the viewer has to maintain a fixed head position and parallel to the screen. Tilting the head may cause an incorrect filtering of the images for the left and right channel. This does not happen with circular polarization.

- **Shutter glasses -** This method is based on a time filter: the left and right images are alternated rapidly on the screen. The viewer wears a pair of synchronized glasses whose shutters occlude the relative frame at each time step.
  This technique requires a complex hardware setup: a pair of synchronized shutter glasses, a graphic card able to visualize two alternate video

streams and a screen with an high frame rate. Consequently, costs and results are both high.

- **Stereo pair display -** In that case, the two images are not overlapped but physically separate and presented side by side. Hence no filtering mechanism is required. Typically, this method is employed with head-mounted display.
  The images management must include the correction phase of the aberration produced by the lenses that are typically interposed between the human eyes and the display. The devices that implement this methodology have greatly evolved and the latest solutions allow to obtain images with higher brightness and sharpness. The hardware used is quite expensive and typically occludes the vision of the real world: it consists of a fully immersive solution.

All these techniques are based on the creation of a pair of images representing the same scene from a different perspective (one for each eye) exploiting the binocular disparity defined by the interocular distance. The projection, which generates the pair of images, can be done in three different ways (Figure 3.14), and depends on the mutual position between the object and the projection plane (the screen).

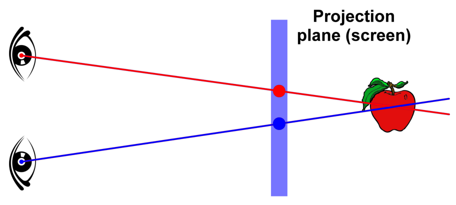
- *Positive parallax* - The object is behind the projection plane (Figure 3.14(a)), the two projections are on the same side as the respective eyes. The maximum positive parallax corresponds to the interocular distance. This solution produces an 3D object that materializes behind the screen plane.

- *Zero parallax* - The object position coincides with the projection plane (Figure 3.14(b)), hence the projections for both eyes is coincident. The 3D object materializes on the screen plane.

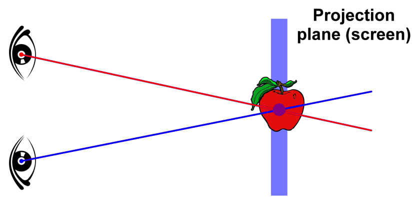
- *Negative parallax* - In this last case the object lies in front of the projection plane (Figure 3.14(c)), as a result the projection for the left eye is on the right and vice versa. When the object is located in the middle between the projection plane and the center of the eyes, the horizontal parallax is equal to the interocular distance. In that case, it will seem that the object pop out from the screen.

The couple of projections can be obtained defining two distinct virtual camera (Figure 3.15), for this purpose there are two methods.

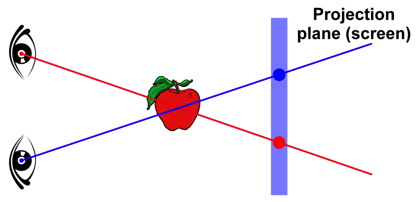
- *Off-axis* - This correct method requires a non symmetric camera frustum and each camera has its own focal point (Figure 3.15(a)).



(a)

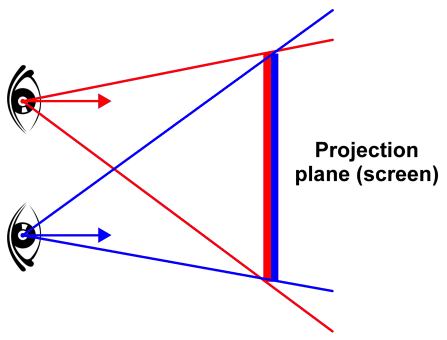


(b)

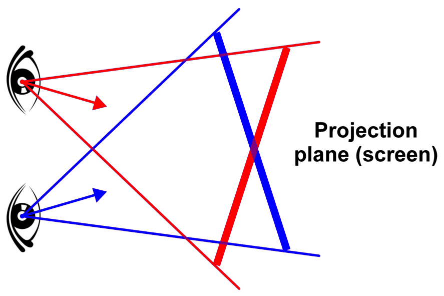


(c)

Figure 3.14: The three different projections, with positive parallax (a), with zero parallax (b) and with negative parallax (c).



(a)



(b)

Figure 3.15: On the *left* the correct *Off-axis* method that defines two non symmetric camera. While on the *right* the incorrect *Toe-in* method, that defines symmetric camera and introduces a vertical parallax.

- *Toe-in* - The cameras have fixed and symmetric aperture, and point the same focal point (Figure 3.15(b)). It introduces a vertical parallax that causes an increase of the discomfort. The vertical parallax increases with the focal aperture of the camera. Although the *Toe-in* method is incorrect, it was often used because it is widely supported by different systems.

The parameters that govern the stereoscopic effect can not be overly stressed, otherwise the human visual system would not be able to perform the fusion operation between the two projections and it would cause of sickness. In particular, with the *Off-axis* method the non symmetric frustums can be defined taking in account the field of view (FOV) of the camera, the width of the virtual screen (the projection plane) and the eyes separation, as shown in Figure 3.16. Hence the four clipping planes that define the frustum can be computed using these equations:

$$top = D_{near} * tan\frac{\theta_{FOV}}{2} \tag{3.1}$$

$$bottom = -top \tag{3.2}$$

for the top and bottom planes of each frustums, where the $D_{near}$ is near clipping distance of the frustum and $\theta_{FOV}$ is the FOV along the vertical direction. While for the left and right clipping planes we need to compute the half-width $W_{half}$ of the virtual screen, and the two asymmetric horizontal distances of the frustum, the small $D_{small}$ and the large $D_{large}$:

$$W_{half} = \rho * D_{conv} * tan\frac{\theta_{FOV}}{2} \tag{3.3}$$

$$D_{small} = W_{half} - \frac{D_{eye}}{2} \tag{3.4}$$

$$D_{large} = W_{half} + \frac{D_{eye}}{2} \tag{3.5}$$

where $\rho$ represents the aspect ratio, $D_{conv}$ is the convergence distance (the distance between the eye and the projection plane) and $D_{eye}$ the eyes separation. Starting from (3.4) and (3.5) we can define the left and right clipping plane for the left camera frustum:

$$left_L = -D_{small} * \frac{D_{near}}{D_{conv}} \tag{3.6}$$

$$right_L = D_{large} * \frac{D_{near}}{D_{conv}} \tag{3.7}$$

and for the right camera frustum:

$$left_R = -D_{large} * \frac{D_{near}}{D_{conv}} \tag{3.8}$$

$$right_R = D_{small} * \frac{D_{near}}{D_{conv}} \tag{3.9}$$

Typically the FOV is between 45° and 60°, while a good eye separation is 1/30 of the convergence distance, a larger value can be hard to resolve and is known as hyperstereo.

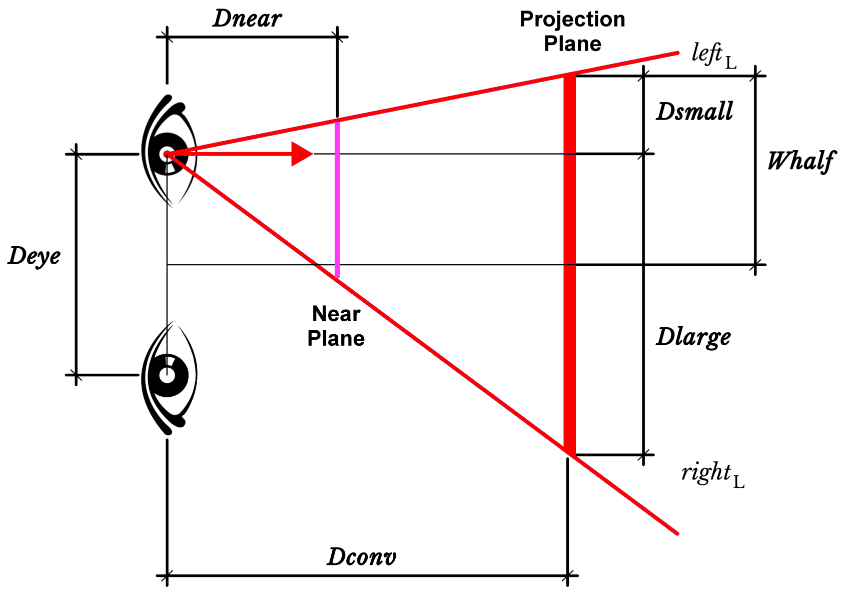


Figure 3.16: The distances used for the calculation of frustum parameters for the left eye frustum.

## 3.3.4 The visualization solution

After the study of the different methodologies to obtain stereoscopic images we decided to employ a colour filter glasses, and we opted for the anaglyph images. This cheap solution allowed us to evaluate if this family of visualization technologies can be successfully implemented in the system and if the user can benefit from it.
In particular, exploiting the graphic libraries, we set two cameras that implement the *Off-axis* method (Figure 3.15(a)), and starting from an interocular

distance which is characteristic of the human being, we defined the two cameras' frustums in order to obtain a negative parallax (Figure 3.14(c)). An example is shown in Figure 3.17, we did not have special requirements about the colors, thus we have used a pair of red and blue filters.
The aim was to obtain a pop-up effect of the 3D models from the 2D screen, and we tried to stress as much as possible this effect in order to align the image with the device coordinate system.

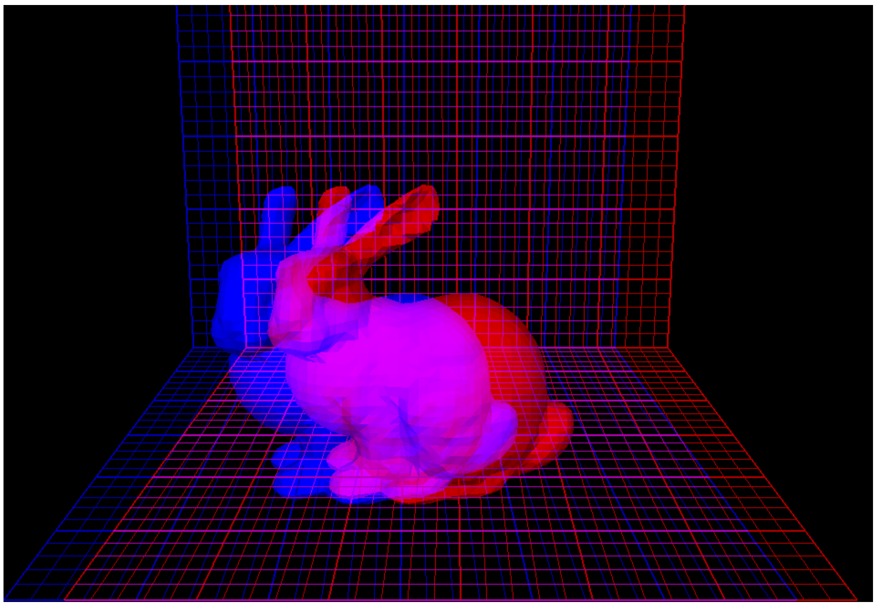


Figure 3.17: Implementing a negative parallax anaglyph we obtained an image where the blue projection one, on the left, belongs to the right eye and the red one, on the right, belongs to the left eye. To see the object comes out from the screen you must watch the image using a red&blue colour filter glasses.

We performed several test to verify how different users perceive this solution, and unfortunately we received only negative feedbacks. We know that the perception of depth depends on several subjective factors [55], but in this specific situation the accommodation-convergence mismatch, that typically affects stereoscopic equipments, plays an important role.
The convergence of the eyes is towards the actual distance of the object points in the 3D scene, but the focus is always fixed on the single images, and this may lead to eye fatigue. Hence we can not excessively stress this mismatch in order to align the virtual coordinates system with the device's coordinate system.
We even detected that it is quite difficult achieve a perfect registration be-

tween coordinates systems. It depends on several factors, like user perception of the depth or the right position of the device, that can not be directly control.
Moreover, if the user introduces his hand in the space where the anaglyph produces its images, breaks the stereoscopic effect, and this increases the discomfort. It happened even though we tried to respect the result proposed in [20], where they found that the minimum distance that allows to benefit of the 3D mid-air selection is 100 *mm*.

These kind of problems are independent of the quality of the technology used, and the same situations could occur using the other high quality techniques like polarized filter glasses or shutter glasses. A solution could be to adopt a stereo pair display, typically employed in a head-mounted display, but we want to preserve the contact with the real world. In a second time, it will be interesting explore the interaction between real object and virtual models, for example to verify if a reconstructed part will fit the real object.

Our aim is to propose a solution that is intuitive and comfortable to use, and we could not afford that the user experiences discomfort situations. For these reasons we explored a combinations of other solutions that could still help the user to improve the depth perception, without cause any discomfort.

We have identified two main lines of action, the first one regards how the interface presents the 3D scene and allowing the user to reach a better perception of the whole environment, while the second introduces some visual cues in order to help the user to improve the perception of the relative positions of the objects in the 3D scene.

In particular, to improve the perception of the 3D scene we adopted a classic solution that is typically used in 3D modelling environment. We subdivided the screen space in order to add three static orthogonal views that frame the scene from front, side and top (Figure 3.18), preserving on the bottom of the screen the 3D view. It is a very simple solution, but allows the user to have an overview of the whole scene in a single glance.

Better results were obtained introducing some visual cues for the objects and the hands. In practice, exploiting the background grid we drawn the orthogonal projections of the fingers' tips and of the spherical volume of the object (Figure 3.19). In addition when a finger falls inside the capturing volume of the object the projection changes color in order to help the user to perform the capturing task.

The users that tested both visualization solutions: stereoscopy and visual cues, found the latter more comfortable and very helpful to imagine the disposition of the object in the 3D scene. Moreover, a simple avatar for the hand allows them to quickly recognize it in the scene, and to create a match

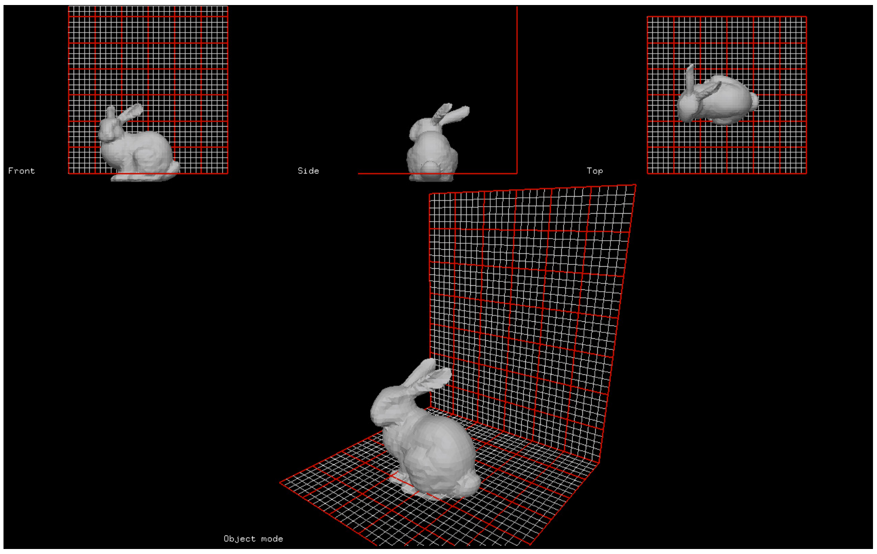


Figure 3.18: The interface introduces three static orthogonal views on the top, that frame the scene from front, side and top. On the bottom there is the 3D view, where the user can control the camera.

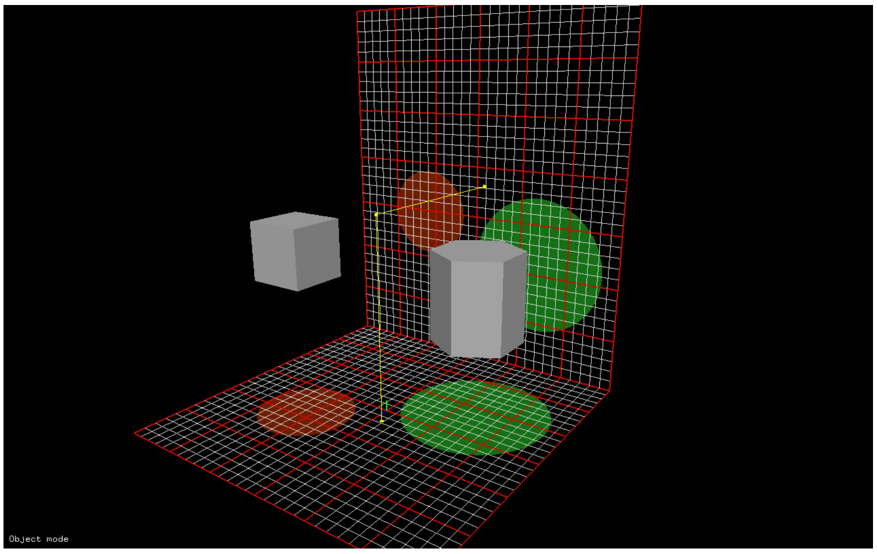


Figure 3.19: The visual cues introduced for the objects and the fingers, improve the perception of the depth. The red and green transparent disks define the projection of the capturing volume, respectively, of the cube and the prism. The straight yellow lines define the 3D position of the finger.

between their hand and its virtual representation. In particular, they reach a good interaction capabilities combining both the solutions just presented: the different views and the visual cues (Figure 3.20).
These choices are very simple and have a reduced visual effect, but they proved to be extremely intuitive and effective.

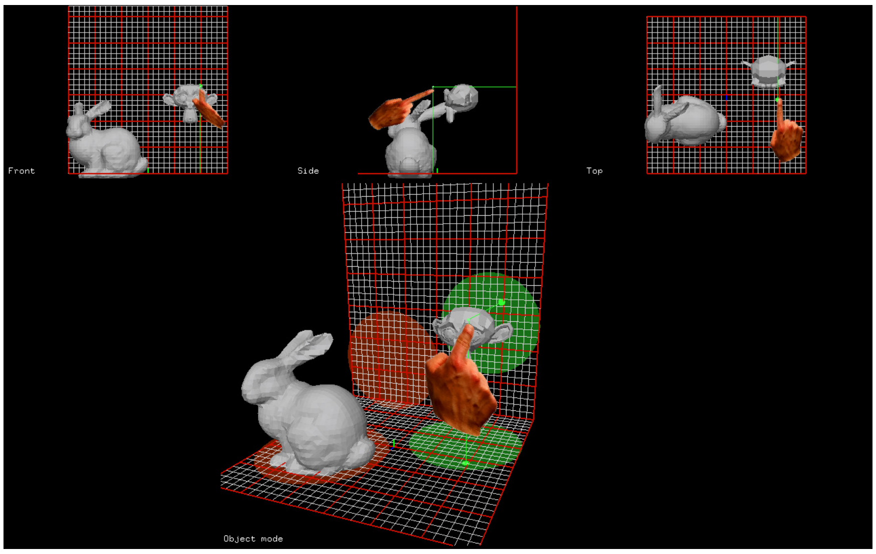


Figure 3.20: An example where both visual aids are active. It can be noticed how the finger that falls inside the capturing volume of the object, changes the colour and the size of its projected disk.

## 3.4 Conclusions

The classic 3D modelling environments or the interfaces used to navigate 3D geometries, require a large variety of command to be controlled: combination of keyboard and mouse entries. Untrained users need time and experience to learn, and they often perceive them as complex and difficult to use tools.
Informal evaluations have shown that most non-expert users perceive the tangible interface as being much easier to use and learn than the traditional 2D interface. Maybe, it is manly due to the difficulty to transform the two-dimensional movements into 3D activities.

Our goal was to explore if new interaction technologies and some classic visualization solutions could be employed to create a 3D interactive modelling environment, more prone to be used by common users who do not have great experience with these kind of tools.

In particular, we tried to verify if the new modelling environment would meet the following characteristics:

- *easy to use and intuitive* - the user needs to learn it quickly, and he needs to use it exactly as he would in the real world. It means that the interaction with hands and fingers assumes a great importance, both to perform global activities and more local tasks;

- *adequate depth perception* - the interface should provide visual cues that help non-expert user to perform tasks in the 3D environment, without precluding the vision of the real world;

- *low-cost setup* - the system must employ inexpensive and readily available technologies.

In contrast to the classic CAD tools, which segregate 3D tasks into 2D procedural inputs that require extensive training, we decided to simplify the paradigm introducing a subset of modelling features controlled by hands and fingers. In pursuit of this aim, we found that simple and intuitive solutions not always mean feasible, and we can not stress the air gestures enough because it may cause a sense of fatigue in the user.

During this activity, we found some interesting things. The first regards the 3D interaction devices, typically they are employed as advanced 2D interaction devices. Nevertheless, they have reached a high accuracy which allows to exploit mid-air gestures to perform punctual tasks.
The visualization of the 3D contents has several shortcomings, and in this specific case a classic technique, like stereoscopy, has proved inadequate. It will be interesting explore the holographic technique when it will reach a good maturity.

Then we can conclude that a system entirely controlled by the gestures would become extremely complex: too many gestures combination to remember in order to enable all features. Hence the best choice is a fusion between air gestures and keyboard keys or mouse entries. For these reason some solutions presented could be successfully integrated in other simple 3D modelling environments.

For the future, we identified several interesting topics. The first regards the freehand free-form creation. Up to now we modified existing objects, but could be interesting create new ones studying new interaction operations. Another interesting topic concerns how to use the row images (one of the last features of the Leap Motion device) and consequently, how to introduce physical objects to be put in relation with the virtual models. Then we might integrate other advanced operations which can benefit from the 3D interaction environment.

For the moment, we have performed some generic tests in order to verify the fairness of insights used to create this first prototype. As a result, we plan to perform a more formal user study, verifying how the same tasks are performed in a classic 3D modelling environment and in our system.

We think that these kind of applications can also be used in other contexts, for instance, they may represent a valid support to perform cognitive rehabilitation tasks, or some educational activities.

# 4

# Advanced modelling operations

*The difference, of course, is that while in physics you are supposed to figure out how the world is made up, in computer science you create the world.*

Linus Benedict Torvalds

Computers are one of the technologies that have mostly contributed to simplify and support the fulfilment of countless activities in the human being life. They can boast a two manifold contribution: they are very useful to *emulate* tasks that would be usually performed with coarser instruments, and they can *create* situations that otherwise would not be possible represent in real life.
In the first case, the challenge is to correctly reproduce the conditions that lead to completion of a given task, respecting properties and attributes of the real entities involved in the virtual reconstruction working environment. While creation capabilities leave much more space to fancy solutions. You can obtain remarkable results, going beyond the physical principles that govern the real objects.
Both of these principles can be applied in a 3D modelling environment, and that's what usually happens. Once again the aim is to bridge the gap between complex activities and non-expert users.

## 4.1 Overview

The 3D modelling of virtual objects was born in the middle of the last century, and it has always been a crucial step in different fields to create more complex products. It is possible to identify three different ways in which a three-dimensional model of an object can be generated.

- **Manual modelling**: the more classical technique to realize virtual models. The user has available several tools and methods for constructing objects, for example, starting from some primitive shapes (e.g. cube, cone, sphere) or using mathematical solutions (e.g. Bézier's curves, NURBS curves and surfaces).
- **Three-dimensional scanning**: this technique allows you to reconstruct a virtual model by scanning a real object. The acquisition of the points, thanks to which it is possible to reconstruct the object, can be done either by direct contact of a specific instrument or exploiting different light source.
- **Procedural modelling**: is a technique assisted by software tools able to automatically or semi-automatically generate the desired geometry based on a set of rules. It is widely used to generate models of fluids, tissues, hair and vegetation.

The different techniques to generate a virtual model allow the use of the 3D modelling in different application fields, each of which may require a different interpretation of the model itself: *solid modelling*, the resulting model is considered as formed by a full volume; *volumetric modelling*, the model uses an implicit surfaces (isosurfaces) to visualize the values of a continuous function whose domain is just the 3D space; *surface modelling*, the model is represented by its external surface. The user is provided different more or less complex modelling functions (e.g. skinning, extrusion, revolution, parametric patches, sphere modelling) depending on the task that he wants to accomplish and the type of representation used.
In particular, with surface modelling three different representation can be adopted.

- *Polygonal approximation* - the surface is formed by polygons, a grid (mesh) of vertices, edges and faces which describe it. A greater number of polygons produces a smoother result. Although the approximation of natural surfaces is very rough, this methodology is very simple, provides a good approximation of mechanical parts and it is used during rendering phase.
- *Parametric surfaces* - the surface is defined by a set of control points, which the final result can approximate or interpolate. The final object has a high quality surface, but they are more complex and require a mathematical background to better exploit the control of local form.

- *Algebraic surfaces* - the surface is analytically described by polynomials. Typically, they are used in molecular modelling because are not suitable for realistic scenes. They are easy to calculate and to use, if you know the theory that governs them, but they have several limitations in describing the objects' edges, for this reason they are almost never used.

We are interested in the polygonal approximation, the discrete representation of the model's surface, mainly for two reasons: it is by far the easiest to understand in the eyes of a non-expert user, that does not require a specific training, and if continuous representations have various procedures to manipulate them, which implementation did not require special effort, the discrete representation requires *ad hoc* solutions. For these reasons, the development of modelling features for discrete virtual model is by far more interesting challenge.

In particular, a methodology capable of deforming the virtual objects in a physically plausible way has been studied. We will see how this technique based on total curvature energy is able to preserve surface's details.
In addition, we exploited one of the main advantages of the virtual models: the ability to perform actions not possible in the real world, as the bodies' interpenetration. We continued the work started in our previous thesis [7], regarding on the Boolean operations on virtual objects whose surface is represented in a discrete manner.

## 4.2 State of the art

The possibility to deform curves, surfaces and solid models has gained popularity in computer graphics after they were proposed in the late of 1980s by Terzopoulos [111]. The aim is to create realistic animations involving various applied forces in a simulated physical world. They exploited the elasticity theory to construct differential equations, whose numerical solution faithfully replicates elastic behaviour and more general inelastic behaviour. Terzopoulos was the first to introduce the theory of multidimensional deformable models in a Lagrangian dynamics setting, based on deformation energies in the form of generalized splines in order to control the continuity [110].

The Terzopoulos' approach has established a main direction and we can classify all shape deformation solutions proposed since then, in two main categories according to the way in which they act on the object to be deformed [13]: *space deformation*, if the objects is modified by deforming their embedded space, or *surface deformation*, if the deformation is directly defined on the surface of the object.

All *space deformation* methods use a control structure, for instance a lattice, to immerse the object, and then every structure deformation is propagated to the object itself.
In Free-Form Space Deformation (FFD) the deformation is indirectly applied to the object: the user moves the points of the control structure. The object's points are expressed as a linear combination of the control points of the structure and blended with some different bases functions, for instance Bézier [95], B-spline [35, 56] or T-splines (a generalization of NURBS) [101].
The main defect of the FFD is the unnaturalness of the deformation through the control structure, that has led to the Direct Manipulation FFD (DMFFD). In [39] the user directly moves some object points, the system computes the points' displacements of the control structure and then accordingly updates the rest of the object. Another way to improve both FFD and DMFFD is to employ the radial basis functions [12], thanks to the nature of the handle point and the deformation function physically used the deformation is extremely more natural.
Nonlinear methods are presented in [106], where an energy functional optimizes the local deformation gradients, and in [15], where the object is voxelized and the deformation is driven by a nonlinear elastic energy.
The robustness, the quality and the efficiency of these methods are strongly affected by the complexity of the control structure, while they are less affected by quality of the original surface and do not depend on the underlying surface representation [101].

There are a plenty of different *surface deformation* methods, well summarized in [16]. The Transformation Propagation method linearly propagates the deformation within a region, and the main challenge is how to define the propagation function. In [5] the geodesic distances is employed, while in [78] they use the euclidean distances.
The Shell-Based Deformation techniques minimize two deformation energies: stretching and bending [112]. Its peculiarity is that these two energies are inspired by the world of physics.
An interesting method is proposed in [17], the Multi-scale Deformation decomposes the object surface into two frequency bands: high frequencies for details and low frequencies for global shape. Then they are able to deform the low frequencies preserving the higher ones.
Other two detail-preserving techniques have been proposed as surface deformation methods based on differential coordinates: the Gradient-Based Deformation and the Laplacian-Based Deformation. In [120] the original surface gradients are used as target for the least-squares method applied on the deformed surface. While in [61, 105] they have replaced the gradient with the Laplacian operator on vertices.

Typically the linear methodologies can be solved very efficiently but they can lead to counter-intuitive results for large-scale deformations or in some case they can even fail. While non-linear deformation techniques require more complex numerical schemes, but can obtain better results [13]. Both with linear and non-linear methods some constraints can be added to improve results quality: pyramid coordinates [98], handle-aware isoline technique [2], volumetric graph Laplacian [40], skeleton-based inverse kinematics [99] and shell-based minimization coupled by a non-linear elastic energy [14].

From the perspective of the usability, the deformation metaphor used, that concerns the manner with which the user defines the deformation, it is very important. The user is provided different approach to define the deformation: by handle point, if he moves some "handle" points of the object; by curve-based, if the deformation is imposed by sketching curves or by control point, if the user manages the object in an indirect way [33].

Boolean operations are a prerogative of virtual environments and allow to obtain new complex models as combination of others, which can be usually simpler. Union (or merge), intersection and difference are the basis of constructive geometry and their computation has proven to be a challenging and complicated task that often depends on the chosen representation for the model. Even though, the importances of Boolean operations even in CAD/CAE/CAM environments is undisputed as pointed out in this work [107], where they are employed to construct heterogeneous material objects.

Since the early works presented at the end of the 1980s, it was clear that the challenge was to develop robust methods for Boolean operations, suitable to operate with the boundary representation (B-rep). In [85] they discussed algorithms for merging B-reps solids that are combined by regularized Boolean operations. They detected two categories of constraint that a polyhedron needs to adhere: topological and geometric constraints, for instance the solids must be algebraically closed to obtain closed resulting solids and the regularization procedure may discard extraneous regions of the boundary ("dangling" face or edge). Even in [10] a method for computing approximate results of Boolean operations on B-Rep solids is described, and it is optimized using multi-resolution subdivision surfaces in a neighbourhood of the intersection. A different approach is presented in [34], they compute Boolean operations on B-rep solids but the main interest is to reduce the complexity by converting a 3D problem into a 2D one. In particular, the search for intersections is reduced in 2D space by means of projections on the planes of each single face.

The intersection problem between planar polygons, and consequently the Boolean operations, has been repeatedly faced. The motivation is that it can be useful for the classification of the 3D intersection between boundaries. In [86] they used the concept of simplicial chains and algebraic operations to compute Boolean operations between general polygon, and this allows them to reduce the study of special cases. An evolution of the previous algorithm is presented in [79], preserving the simplicity of the method and introducing a subdivision step they have been able to reduce less than one-third the running time required. Another optimized solution is proposed in [66], in that case the algorithm does not need to be adapted to work with polygons with holes or with regions composed of sets of polygon.

One of the first interesting solutions on polyhedral shapes is presented in [102], where they adopted approximate arithmetic to compute Boolean operations between triangular mesh or polygonal mesh. The triangle-mesh variant of the algorithm needs to convert the polygonal region into a set of triangles. Despite, the method as a whole is not fully robust geometrically, they are able to preserve the connectivity of the boundary.
In the literature, there are numerous solutions that offer Boolean operations between triangle-based mesh, thanks to the lower computational complexity. In [82] they classified all possible intersections between coplanar triangles, and they can deal all arbitrary closed solids (convex or concave). Another algorithm based on triangular mesh is presented in [23], in that case they take advantage of graphics hardware to treat the triangulation phase, but it is strictly required that the mesh is "watertight" and topologically complete. While in [91] the focus is on model with large numbers of triangles, paying attention to problems due to degenerate triangles or intersections of nearly coplanar triangles. In [119] they proposed a fast and robust Boolean operations algorithm able to deal with B-reps polyhedra, assuming that the B-rep is a triangulated manifolds. Even in [47] they used B-rep objects whose geometry is given by discrete data, but in addition they demonstrated the extension of the algorithm to triangular meshes.

Another interesting topic is how to improve the efficiency of these methods. In [6, 37] they used a data structure to represent Nef polyhedron optimized for Boolean operations. While in several works the common solution has been to employ octree data structure, to reduce the space to be inspected. In [113] they are able to compute approximate Boolean operations on large polyhedral solids, and in [30] they perform exact evaluation on triangular meshes. The algorithm presented in [123] uses binary tree to store vertices information of the triangular meshes and to search edge-face intersection, in that way the proposed solution for Boolean operations is more efficient.

Since deforming in a physically plausible way a discrete model is a challenging topic in digital surface processing, in this work we investigate a new

variational model for detail preserving surface-based deformation of a mesh, based on total curvature energy. The purpose is to create a simple and intuitive deformation model for discrete model, able to preserve the details of the surface.
In addition, the triangular meshes are more simple and easy to use then unstructured one, but constitute a limit for next uses of the model. We propose a algorithm for Boolean operations able to manage generic meshes, without convert the model into a triangular mesh.

## 4.3 Deformation by discrete elastica

In the last two decades we have witnessed an exponential growth in interest in computer animation and game design, that led to the development of a multitude of editing methodologies for discrete models, including deformation methods [33].

Surface based-deformation methods of discrete models have been recently widely investigated as a valid alternative to rigging and caging. Its natural and intuitive way to apply and control the deformation has attracted our interest, with the aim of a future integration in an environment like the one shown in the Chapter 3.
In particular, modelling a physically plausible deformation of a flexible object from its naturally planar state is a fundamental and challenging topic. Starting form the work presented in [87] on deformation, we propose a new model for detail preserving surface-based deformation of a body, based on total curvature energy, that can include different constraints for different results [8].

The main requirement for physically based surface deformation is an elastic energy that measures how much an object has been deformed from its initial configuration. The idea is to improve the traditional Laplacian method for surface editing by using the total curvature as a better aesthetic measure for deformation of elastic bodies.
The equilibrium configuration reached by the deformation due to external localized forces and interactively applied by the user is modulated by imposing geometrical constraints, that allow to better formalize the physical metaphor. The constraints offer two main advantages: they allow the user to hint the desired deformation shape, without specifying it exactly, and they retain the local surface features, preserving the relative orientation and size.

### 4.3.1 The energy-based deformation model

The deformation of a non-rigid body is a change in shape due to different applied forces as pulling, pushing, bending or twisting. It can be defined elastic if the original shape restores itself upon removal of the external deformation forces, inelastic otherwise. We focused on the first elastic deformations step, neglecting the phase responsible for recovering the original shape. The main idea is to use the curvature-based energy as a better aesthetic measure for deformation of elastic bodies.

The curvature of a curve in a given point is by definition the reciprocal of the radius of the osculating circle in that point. For each point $s$ on a surface $S \in \mathbb{R}^3$ you can compute the normal vector $\overrightarrow{n}$. A plane $P$ that contains $\overrightarrow{n}$ and intersects the surface in $s$ yields a curve $\gamma$ with a own curvature. Each unit vector $\overrightarrow{v}$ orthogonal to $\overrightarrow{n}$ identifies a different plane $P$, and the set of all vectors $\overrightarrow{v}$ define a circumference $C$. As a result the surface curvature is a function $f : C \rightarrow \mathbb{R}$, and since $C$ is compact and $f$ is continuous, the maximum and minimum values of $f$ are the two principal curvatures $k_1$ and $k_2$ of the surface in $s$.
Typically, a curvature energy may be expressed in terms of principal curvatures of a surface: the mean curvature ($\mathcal{H} = k_1 + k_2$), the Gaussian curvature ($\mathcal{K} = k_1 k_2$), and the total curvature ($\mathcal{T} = k_1^2 + k_2^2$).

Let $\mathscr{M}$ be a fixed reference surface and $X : \Omega \subset \mathbb{R}^2 \rightarrow \mathscr{M} \subset \mathbb{R}^3$ be a function of parametrization. A deformation is a function $d$ that maps $\mathscr{M}$ to a certain deformed model $\mathscr{M}'$, by adding to each point $X(u,v) \in \mathscr{M}$ a displacement vector $d(u,v)$, such that $\mathscr{M}' = X'(\Omega)$ and $X' = X + d$. A reasonable approximation for elastic thin-shell energy which measures stretching and bending is the following [13]:

$$\int_\Omega k_s \|I' - I\|^2 + k_b \|II' - II\|^2 dudv \tag{4.1}$$

where $I$ ($I'$) and $II$ ($II'$) represent the first and second fundamental forms for $\mathscr{M}$ ($\mathscr{M}'$) and $k_s$ and $k_b$ modulate, respectively, the resistance to stretching and bending. The solution of this integral over $\Omega$ is too complex hence two simplifications are recommended.
*First simplification step.* The elastic thin-shell energy as defined in (4.1) leads to a difficult non-linear optimization problem. Therefore, it is common to linearize the objective function replacing the change of the first and second fundamental forms ($I' - I$ and $II' - II$) by the first-order and second-order partial derivatives of the displacement function $d$ [16]. As a results the stretching (membrane) energy is defined as:

$$E_S(d) = \frac{1}{2} \int_\Omega k_s (\|d_u\|^2 + \|d_v\|^2) dudv. \tag{4.2}$$

while the bending (thin plate) energy is defined as:

$$E_B(d) = \frac{1}{2}\int_\Omega k_b(\|d_{uu}\|^2 + 2\|d_{uv}\|^2 + \|d_{vv}\|^2)dudv. \tag{4.3}$$

*Second simplification step.* Choosing the parametrization domain $\Omega$ equal to the surface $\mathscr{M}$, such that $d : \mathscr{M} \to \mathbb{R}^3$ is defined on the manifold $\mathscr{M}$ itself, keeps the parametrization of the surface as close to isometric as possible. This second simplification is well known in literature and turns the Laplace operator $\triangle$, with respect to the parametrization $X$, into the Laplace-Beltrami operator $\triangle_{\mathscr{M}}$, with respect to the manifold $\mathscr{M}$. Therefore when the parametrization is isometric (4.2) becomes:

$$E_S(d) \simeq \frac{1}{2}\int_{\mathscr{M}} k_s\|\nabla_{\mathscr{M}} d\|^2 d\mathscr{M} \tag{4.4}$$

and (4.3) becomes:

$$E_B(d) \simeq \frac{1}{2}\int_{\mathscr{M}} k_b\|\triangle_{\mathscr{M}} d\|^2 d\mathscr{M} \tag{4.5}$$

and their minimization, performed efficiently by applying variational calculus, yields to their Euler-Lagrange equations, respectively, for the **stretching energy** (4.4):

$$-\triangle_{\mathscr{M}} d = 0 \tag{4.6}$$

and for the **bending energy** (4.5):

$$\triangle^2_{\mathscr{M}} d = 0 \tag{4.7}$$

Deformation of solids can only consider the stretching energy, while the deformations of non-rigid surfaces (so called thin-shells) requires both. Hence the **combination energy** $E_C$ of the previous ones, as proposed in [16], is given by:

$$-k_s\triangle_{\mathscr{M}} d + k_b\triangle^2_{\mathscr{M}} d = 0 \tag{4.8}$$

where stretching and bending are combined together and modulated by the weights $k_s$ and $k_b$. The linearization in (4.4) and (4.5) causes artifacts for large deformations and (4.6) and (4.7) require suitable boundary constraints.

Instead, the total curvature energy fuses together the mean and Gaussian curvature. Thin flexible structures are governed by a surface bending energy, the so-called Canham-Helfrich model [38]:

$$E(\mathscr{M}) := \int_{\mathscr{M}} \alpha + \beta(H - H_0)^2 - \gamma K d\mathscr{M} \tag{4.9}$$

where $H_0$ denotes the spontaneous curvature which plays an important role in thin-shell. The Canham-Helfrich model can be reduced to the total curvature energy with $\alpha = H_0 = 0$, $\beta = 1$ and $\gamma = 2$:

$$E_T(\mathscr{M}) := \int_{\mathscr{M}} (H^2 - 2K) d\mathscr{M} = \int_{\mathscr{M}} (k_1^2 + k_2^2) d\mathscr{M} \tag{4.10}$$

The minimization of this functional leads one more time to the Euler-Lagrange equation for the **total curvature energy**:

$$\triangle_{\mathscr{M}} H(d) - 2H(d)(H^2(d) - K(d)) = 0 \tag{4.11}$$

The equation (4.11) is a fourth-order partial differential equation (PDE), since the term $\triangle_{\mathscr{M}} H(X)$ involves fourth-order surface derivatives, and to be well posed it requires independent boundary conditions.

Each application that involves a deformation process can be interested either to a dynamic time dependent simulation, or directly to solve the final rest state. We are interested in the latter which requires the resolution the Euler-Lagrange formulation subject to user-defined boundary constraints (Section 4.3.2).
In practice, it demands to define two specific regions of the surface: the fixed one $\mathscr{F} \in \mathscr{M}$ and the handle one $\mathscr{H} \in \mathscr{M}$. The second $\mathscr{H}$ region allows to define the displacement and represents the target of the deformation (Figure 4.1). The free region $\mathscr{M} \backslash (\mathscr{F} \cup \mathscr{H})$ has to be recomputed by solving the PDE each time the user moves the handle region $\mathscr{H}$.

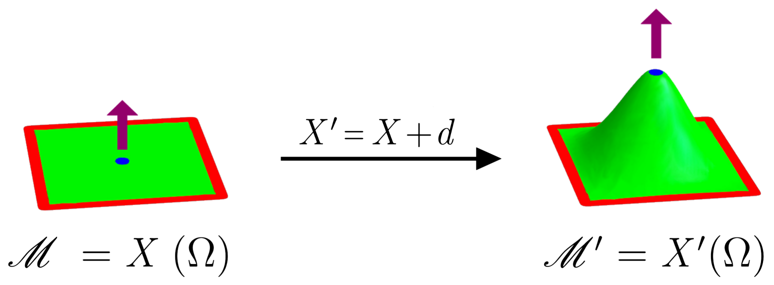


Figure 4.1: An example of how to apply the deformation. The red border represents the fixed region $\mathscr{F}$, the blue disk is the handle region $\mathscr{H}$, while the purple arrow represents the external force. The system automatically updates the free green region.

Whichever energy model you choose to use the deformation problem can be formalized by:

$$min_{X'} \; E(X' - X) \tag{4.12}$$

and we will show how all these requirements and the deformation model can be easily integrated in a system such as the one presented in Chapter 3.

### 4.3.2 Linear and non-linear constraints

The deformation model (4.12), based on one of the energy forms seen in the previous section, can be enriched with constraints. In particular, two types of constraints have been taken into account: positional constraints and detail-preserving constraints. As a result the previous definition of the deformation model can be updated as follow:

$$min_{X'} E(X' - X) \quad \text{subject to} \quad \Phi(X') \tag{4.13}$$

where $\Phi(X')$ represents linear or non-linear constraints.

**Positional constraints.** We recall that in order to define the deformation the user moves the $\mathscr{H}$ region of the surface. If the user establishes that the new position of the $\mathscr{H}$ region must be absolutely respected, just as happens in classical editing tools, we are in the presence of hard positional constraints. Otherwise, soft positional constraints allow the user to indicate the imprecise locations of the $\mathscr{H}$ region, without specifying it exactly, in order to hint the desire shape.
This kind of constraint can be modelled allowing that the final target position $C$ can approximate the real reached position $X'$:

$$\frac{1}{2}\int_{\mathscr{M}} (X' - C)^2 d\mathscr{M} \tag{4.14}$$

The positional constraints are an example of linear constraints and we will show in Section 4.3.3 that there are two way to impose this constraint.

**Detail-preserving constraints.** This second kind of constraint is a non-linear constraint and tries to preserve local differential properties of the surface under the deformation.

Let $\delta = \triangle_{\mathscr{M}}(X)$, the surface deformation is obtained by:

$$\frac{1}{2}\int_{\mathscr{M}} (\triangle_{\mathscr{M}} X' - \delta)^2 d\mathscr{M} \tag{4.15}$$

it forces the new position to resemble its undeformed Laplacian as closely as possible, in view of the fact that $\triangle_{\mathscr{M}} X = -\mathcal{H}\overrightarrow{n}$, where $\mathcal{H}$ is the mean curvature and $\overrightarrow{n}$ are the outward surface normals. Equation (4.15) leads to a linear least-squares problem.
Laplacian deformation methods, based on the minimization of (4.15) [16,105], or other methods, based on differential coordinates [120], fail to yield intuitive results for translational deformations. They try to preserve the orientation of

the normals with respect to the global coordinate system, while translation deformation does not cause a change in surface gradients, or normal vectors.

We desire a method able to preserve the features of the surface, it means that it should preserve the normals' relative orientation and possibly the feature's size.
We introduced the local transformations $T$ restricted to rotation and isotropic scaling. The differential representation $\delta$ is transformed into the deformed pose $\delta = T\delta$. As a result, the deformed positions $X'$ are then obtained by replacing (4.15) with the following non-linear term:

$$\frac{1}{2}\int_{\mathscr{M}} (\triangle_{\mathscr{M}} X' - \hat{\delta}(X'))^2 d\mathscr{M} \tag{4.16}$$

The term $\hat{\delta}(X')$ includes the effects of local rotations and leads to deal with a non-linear least-squares problem.

### 4.3.3 Model discretization

Up to now we treated the deformation model for a two dimensional manifold of arbitrary topology $\mathscr{M}$ embedded in $\mathbb{R}^3$, while we are interested in meshes. A mesh $\mathcal{M}$ is a discrete surface: a piecewise-linear approximation of $\mathscr{M}$. $\mathcal{M}$ is defined by a set of triangles $T_i, i = 1, \ldots, N_t$, which cover the surface, and a list of vertices $X_i, i = 1, \ldots, N_v$, where $X_i \in \mathbb{R}^3$ is the i$^{th}$ vertex with associated the normal vector $\overrightarrow{n}_i$. To discretize the deformation model we need to discretize each single component.

The first discretization regards the Laplacian operator. A vertex $X_i \in X$, usually defined by Cartesian coordinates $X_i = (x_i, y_i, z_i)$, can be represented in differential coordinates as $\delta_i = (\delta_i^x, \delta_i^y, \delta_i^z)$. In matrix-vector form:

$$\begin{aligned} \mathrm{L}x &= \delta^x \\ \mathrm{L}y &= \delta^y \\ \mathrm{L}z &= \delta^z \end{aligned} \tag{4.17}$$

where the large and sparse matrix $\mathrm{L} \in \mathbb{R}^{N_v \times N_v}$ represents the connectivity matrix of the mesh and it is the discretization of the local Laplace-Beltrami operator $\Delta_{\mathscr{M}}$ on the mesh $\mathcal{M}$.

Given a smooth surface $\Delta_{\mathscr{M}} = 2\mathcal{H}\overrightarrow{n}$ [29], the discrete approximation of mean curvature vector $\mathcal{H}_i\overrightarrow{n}_i$ associated to the vertex $X_i$ can be derived using the following discrete form:

$$\mathcal{H}_i\overrightarrow{n}_i = \mathrm{L}(X_i) = \frac{1}{2A_i}\sum_{j\in N(i)} w_{ij}(X_j - X_i) \tag{4.18}$$

where $N(i)$ is the set of first-ring neighbour vertices of vertex $X_i$, $A_i$ is the Voronoi area surrounding $X_i$, and the weights $w_{ij}$ are positive numbers which satisfy the normalization condition $\sum_{j \in N(i)} w_{ij} = 1$. Different geometric discretizations of the Laplacian can be obtained for different choices of the weights in (4.18), we chose the most commonly used [68] so-called cotangent weights:

$$w_{ij} = (\cot \alpha_{ij} + \cot \beta_{ij}) \tag{4.19}$$

where $\alpha_{ij}$ and $\beta_{ij}$ are the two angles opposite to the edge $(X_j, X_i)$ (Figure 4.2(a)).

The Gaussian curvature can be extended to discrete surface $\mathcal{M}$ keeping the Gauss-Bonnet theorem true by:

$$\int_{\mathscr{M}} \mathcal{K} d\mathscr{M} := \sum_i \mathcal{K}_i \quad \text{and} \quad \mathcal{K}_i = \frac{1}{A_i}(2\pi - \sum_{j \in N(i)} \theta_j) \tag{4.20}$$

where $\theta_j$ are the incident internal angles at $X_j$ (Figure 4.2(b)). Also in that case the equation (4.20) is the most commonly used for triangular meshes [108].

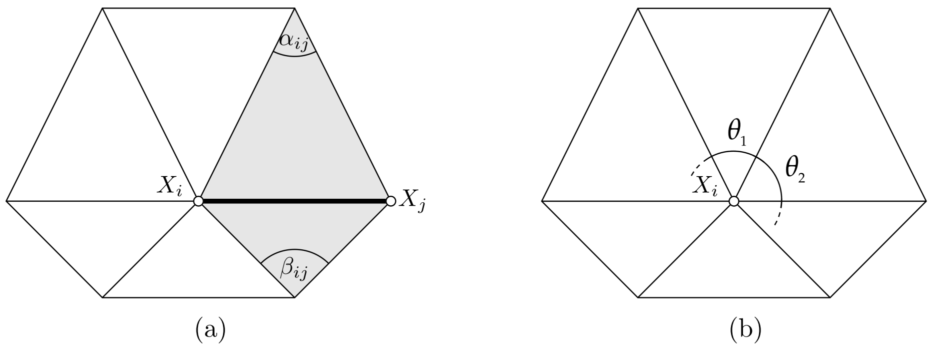


Figure 4.2: On the *left* the two opposite angles $\alpha_{ij}$ and $\beta_{ij}$ used to define $w_{ij}$, they belong to the two triangles that share the edge $(X_j, X_i)$. On the *right* the incident internal angles $\theta_i$ for the vertex at $X_i$.

All the discretizations presented allow to define the deformation energy models for the discrete surface $\mathcal{M}$ (Table 4.1), considering the displacement vector $X' = X + d$ and $G = \mathcal{H}^2 - \mathcal{K}$.
Each model shown in Table 4.1 leads to a generic linear system $AX = b$ with a sparse $N_v \times N_v$ coefficient matrix, where $\mathscr{M} \backslash (\mathscr{F} \cup \mathscr{H})$ vertices are treated as unknowns, while $\mathscr{F} \cup \mathscr{H}$ vertices are incorporated into the right-hand side of the system.

| Energy | Discrete form | Linear System |
|---|---|---|
| $E_S$ (4.6) | $\mathrm{L}d = 0$ | $\mathrm{L}X' = \mathrm{L}X$ |
| $E_B$ (4.7) | $\mathrm{L}^2 d = 0$ | $\mathrm{L}^T\mathrm{L}X' = \mathrm{L}^T\mathrm{L}X$ |
| $E_C$ (4.8) | $k_s\mathrm{L}d + k_b\mathrm{L}^2 d = 0$ | $(k_s\mathrm{L} + k_b\mathrm{L}^2)X' = (k_s\mathrm{L} + k_b\mathrm{L}^2)X$ |
| $E_T$ (4.11) | $\mathrm{L}^2 d - 2\mathrm{L}Gd = 0$ | $(\mathrm{L}^2 - 2\mathrm{L}G)X' = (\mathrm{L}^2 - 2\mathrm{L}G)X$ |

Table 4.1: The discrete forms of the four deformation energy models.

We have seen in (4.13) that the constraints can be incorporated as penalty factor of the discrete energy functional. The discrete form of the positional constraints can be added as follow:

$$min_{X'}\ E(X' - X) + \frac{\lambda_1}{2}\|X' - C\|^2 \tag{4.21}$$

where $\lambda_1 > 0 \in \mathbb{R}^n$ is the penalty coefficient, and $C$ is the vector of the prescribed positions of the vertices. The coefficient $\lambda_1$ allows to control the weight of the positional constraint and in order to approach the interpolation of the constraints $C$, it has to be chosen sufficiently large. However, the condition number of the matrix grows with $\lambda_1$, then a higher weight can cause numerical problems.
The positional constraints affect the final position of the $\mathscr{H}$ vertices and with hard positional constraint (the indicated position must be respected) there are two ways to apply them: *(i)* the application solves (4.21) but the final computed positions of the $\mathscr{H}$ vertices are discarded; *(ii)* the system $AX = b$ is constructed considering for $X$ only the vertices $\mathscr{M}\backslash(\mathscr{F} \cup \mathscr{H})$, with the guarantee that the reduced matrix L is not singular since it is derived from the original connectivity matrix of the mesh $\mathcal{M}$ in which the rows and columns of the vertices $\mathscr{H}$ were moved to the right-hand side.

The detail-preserving constraints can be added in the same way, in that case also the term $\hat{\delta}(X')$ is unknowns in the deformation process. The proposed energy functional to minimize, to solve the mesh deformation problem, is the following:

$$min_{X',\hat{\delta}}\mathscr{L}(X', \hat{\delta}),$$

$$\mathscr{L}(X', \hat{\delta}) := E(X' - X) + \tfrac{\lambda_1}{2}\|X' - C\|^2 + \tfrac{\lambda_2}{2}\|LX' - \hat{\delta}\|^2 \tag{4.22}$$

The minimum of (4.22) can be determined by the alternating minimization procedure, solving:

$$\begin{aligned}\hat{\delta}^{(k+1)} &= \operatorname{argmin}_{\hat{\delta}} \mathscr{L}(X'^{(k)}, \hat{\delta}) \\ X'^{(k+1)} &= \operatorname{argmin}_{X'} \mathscr{L}(X', \hat{\delta}^{(k+1)})\end{aligned} \tag{4.23}$$

Since $\mathscr{L}(X', \hat{\delta}^{(k+1)})$ is continuous differentiable in $X'$, the solution $X'^{(k+1)}$ of the second minimization in (4.23) for the total curvature energy is obtained by imposing:

$$\begin{aligned}0 = \;& \nabla_{X'}\mathscr{L}(X', \hat{\delta}^{(k+1)}) = \\ & (\mathrm{L}^2 - 2\mathrm{L}G)(X' - X) + \lambda_1(X' - C) + \lambda_2(\mathrm{L}^T(\mathrm{L}X' - \hat{\delta}^{(k+1)}))\end{aligned} \tag{4.24}$$

It means that for each new step $k$, new $X'$ and $\hat{\delta}$ values are computed in order to reach the final stable state of the deformation process.
In matrix-vector form, the solution of (4.24) for the new mesh vertices $X'$, is given by solving the following system:

$$\begin{bmatrix} & (\mathrm{L}^2 - 2\mathrm{L}G) \\ \mathbf{0} & \sqrt{\lambda_1} I_n \\ \mathbf{0} & \sqrt{\lambda_2}\mathrm{L}^T\mathrm{L} \end{bmatrix} X' = \begin{bmatrix} (\mathrm{L}^2 - 2\mathrm{L}G)X \\ \sqrt{\lambda_1} C \\ \sqrt{\lambda_2}\mathrm{L}^T\hat{\delta}^{(k+1)} \end{bmatrix} \tag{4.25}$$

where the block $A = (\mathrm{L}^2 - 2\mathrm{L}G)$ represents the total curvature energy, $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix which requires a resorting of the rows of L, and $C \in \mathbb{R}^n$ is a vector of elements $c_i$ for each of the $n$ positional constraint. The overdetermined system has dimension $(N_v + 2n) \times N_v$ and it is full rank, thus it has a unique solution in the least-squares sense. The system (4.25) is solved by the conjugate gradient method, where we terminate the iterations as soon as the norm of the residual is less than or equal to $10^{-4}$. The use of an iterative solver allows us to avoid storing the large dimension matrices, the only requirement is matrix-vector products. The complete description of the *discrete elastica non-linear deformation* (DEND) algorithm used to solve (4.25) can be found in [87].

### 4.3.4 Deformation results

In this section, we present some examples to show how the proposed deformation model performs to deform structured and unstructured polygonal meshes. All the examples have been produced on a standard consumer-level LINUX PC, indeed the final algorithm does not require high computational capabilities.

The implementation of the deformation model is easy and intuitive and allow the user to select regular/irregular regions $\mathscr{F}$ that he/she wants to keep

fixed, so as the areas $\mathscr{H}$ that he/she will move to a target position. The positions of the remaining vertices $\mathscr{M}\backslash(\mathscr{F}\cup\mathscr{H})$ will be automatically determined by the system.
This feature makes the system particularly suitable to be used by a non-expert user, the metaphor of the fixed and handle vertices is similar to the way a real object is deformed.

For the time, the application has no collision detection which would allow for handling collision occurring between deformed parts of a deformable body. Collision detection is a complex constraint which increases considerably the complexity of the system (4.25). Nevertheless, we will show that the deformation model enables to apply small to large deformations on middle-large detailed meshes, while keeping the shape of the details in their natural orientation.

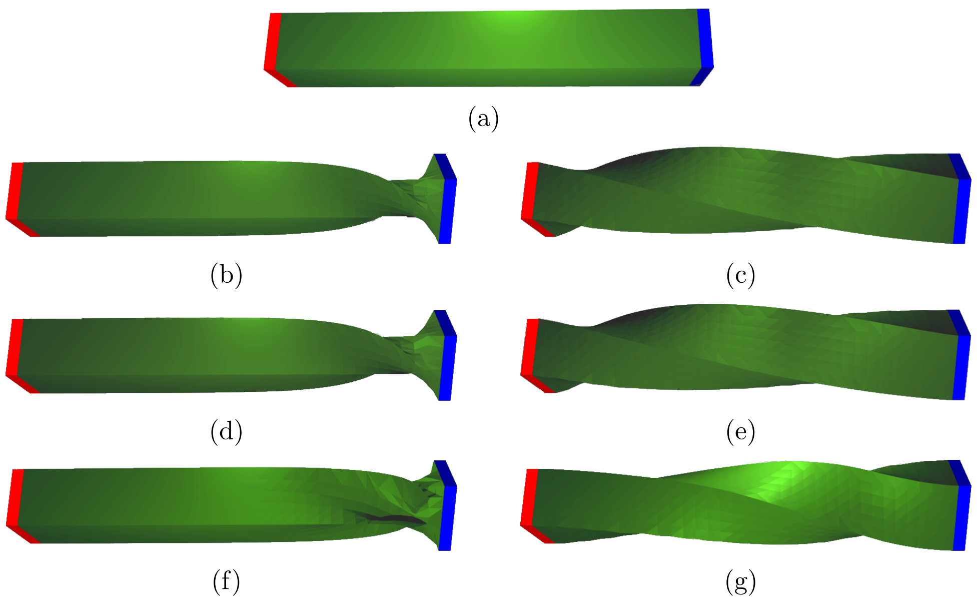


Figure 4.3: Deformation of the bar mesh with 856 vertices. The deformation is achieved by anchoring the red region and twisting blue one of 135° in one step.

In the first examples, shown in Figures 4.3 and 4.4, we compared the different energy models described in Table 4.1, applying linear and non-linear constraint.
In both figures, the first row shows the original model, a bar mesh with 856 vertices (Figure 4.3(a)) and a cylinder mesh with 1088 vertices (Figure 4.4(a)), while each subsequent row shows the deformation result obtained using, re-

spectively, bending (Figures 4.3(b), 4.3(c), 4.4(b) and 4.4(c)), combination (Figures 4.3(d), 4.3(e), 4.4(d) and 4.4(e)) or total curvature energy (Figures 4.3(f), 4.3(g), 4.4(f) and 4.4(g)). Again, in the left column we present the results obtained with linear constraint, while on the right we used non-linear constraint. The bar was twisted of 135° and the cylinder was bended of 120°, for both the operation was performed in a single step. You can note as the red region represent the fixed vertices $\mathscr{F}$ and the blue one the handled vertices $\mathscr{H}$.

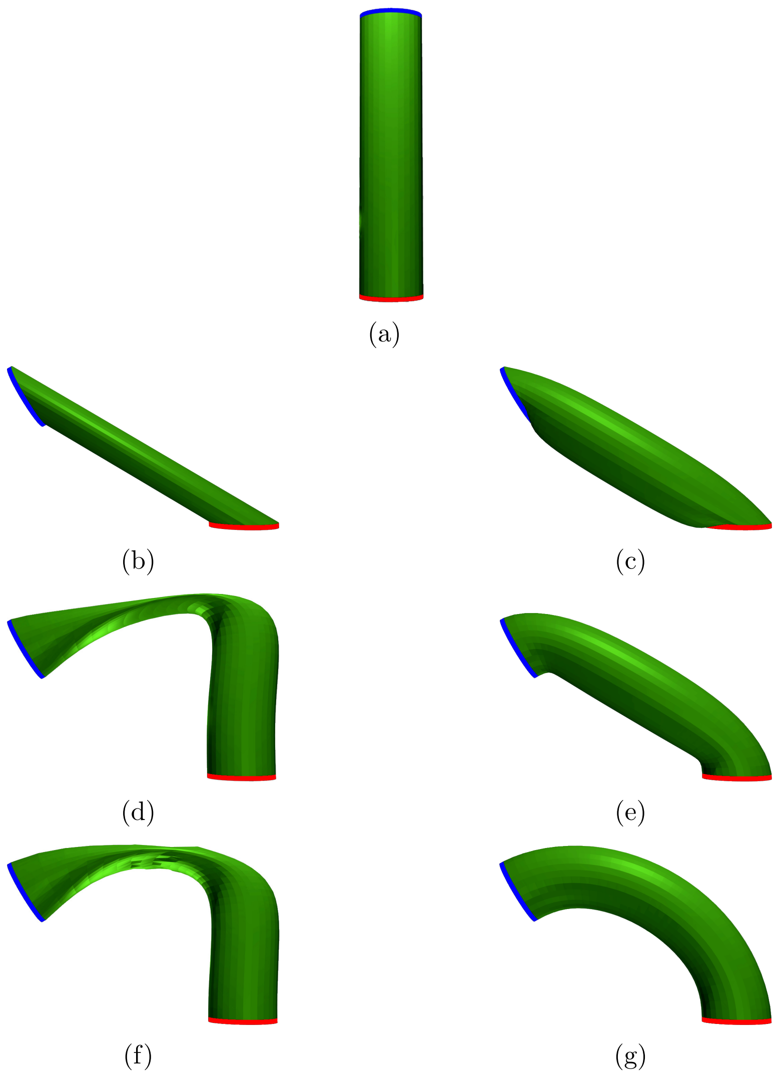
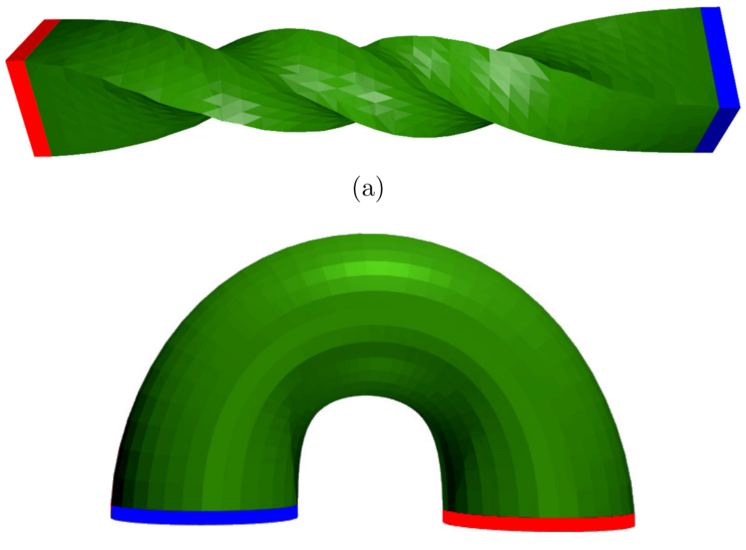


Figure 4.4: Deformation of the cylinder mesh with 1088 vertices. The deformation is achieved by anchoring the red region and bending the blue one of 120° in one step.

The results clearly show the weaknesses of the linear deformation approach. If the deformation results of the bar are quite uniform for all different energy models coupled with non-linear constraints, we only get a uniform bending along the whole cylinder shape using the total curvature energy.

In linear theory the behaviour of the deformable model is physically correct only for small displacements (about 10% of the mesh size), it is less realistic for larger deformations. Hence we tested the behaviour of the total curvature energy model with non-linear constraints when it is applied to obtain large deformations.
Once again we used the bar and the cylinder and Figure 4.5 illustrates two large deformations obtained by interactively applying 5 steps for a total 300° twisting on the bar model (Figure 4.5(a)), and 3 steps for a 180° bending on the cylinder mesh (Figure 4.5(b)).
Large deformations is the main disadvantage of linear elasticity, and interactive methods usually prevent large deformations, since each step remain reasonably small. Nevertheless, the total curvature energy with non-linear constraints provides well-shaped and aesthetically pleasing results also with large deformation step. The other considered energies are not been able to produce the same outcomes.


(a)

(b)

Figure 4.5: Large deformation test: 5 steps for a total 300° twisting on the bar model (a) and 3 steps for a total 180° bending on the cylinder mesh (b).

We also tested the detail-preserving capability of the deformation model,

and the result is shown in Figure 4.6. The first row show how to apply a simple deformation to transform a plane with 2115 vertices (Figures 4.6(a)) into a bumpy plane 4.6(b). The red vertices have been anchored while the blue ones have been translated.
The bumpy plane is then deformed by bending the two sides of the mesh using the total curvature energy model with (Figure 4.6(c)) and without (Figure 4.6(d)) detail-preserving (non-linear) constraints. Using non-linear constraints the deformation model is able to preserve both the normals' relative orientation and the the feature's size.

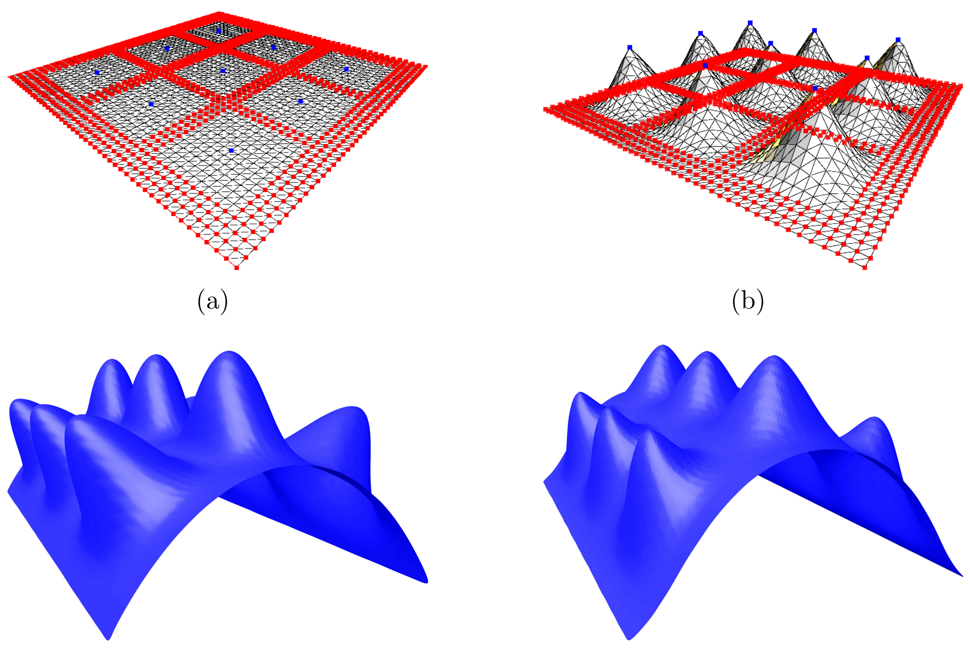


Figure 4.6: Details preserving test. The first row shows the step to transform the original plane (a) into a the bumpy plane (b), achieved anchoring the red vertices and translating the blue ones. The second row shows the deformation results obtained applying total curvature energy with (c) and without (d) detail-preserving (non-linear) constraints, both obtained by bending the two sides of the mesh.

The last examples show the flexibility and the potential of the deformation model based on the total curvature energy with non-linear constraints. We tested open and closed surfaces representing elastically deformable models. The deformation of a torus in Figure 4.7 shows once again, the robustness of the deformation model and its capability to deal with large deformations.

The Figure 4.8 shows the deformations of a thin plate model, represented by the flag mesh. Translating the right side of the flag mesh and anchoring the left side we obtained a waving simulation.
The human hand, shown in Figure 4.9, demonstrates how it is possible obtain aesthetically pleasing results deforming a complex model.
Finally, the Figure 4.10 shows how a medium-large mesh with 10098 vertices can be correctly deformed preserving the smooth regions and the local features of the surface.

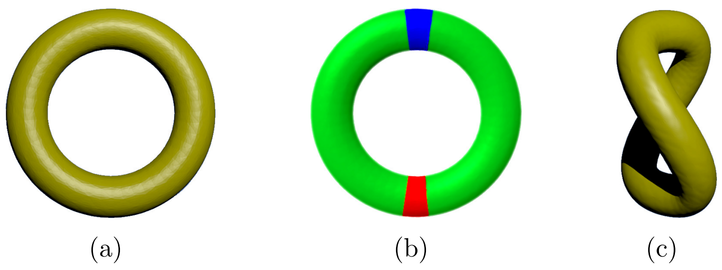


Figure 4.7: A twist of torus mesh. The original model (a), the selected region to apply the twisting (b), and the obtained result (c).

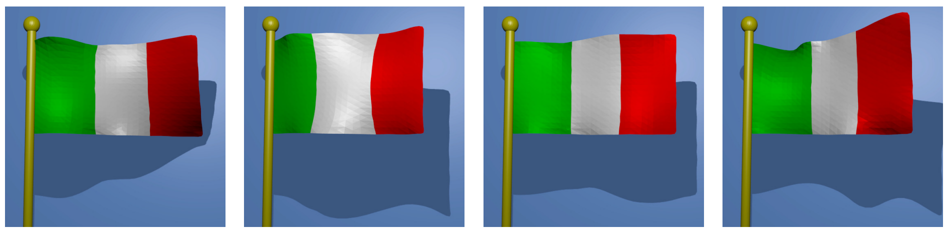


Figure 4.8: An example of deformation to produce wind effect by translating the right side and anchoring the left side of the flag mesh.

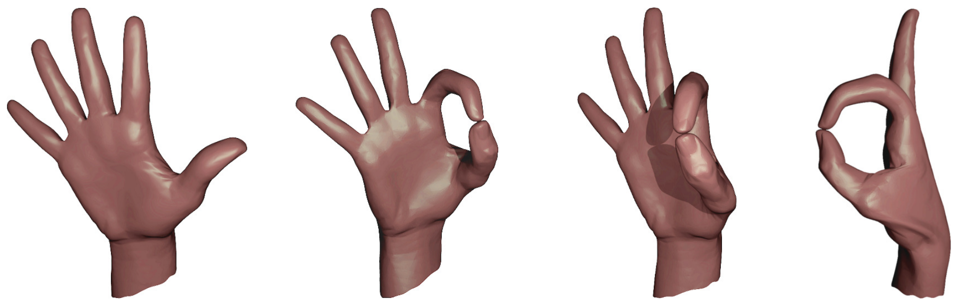


Figure 4.9: The deformation of a human hand mesh to show, from different points of view, the aesthetically pleasing result obtained.

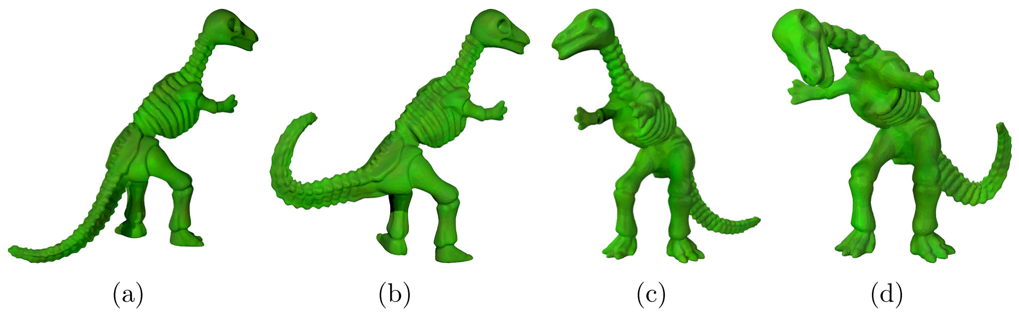


Figure 4.10: A sequence of dino deformations, the mesh is medium-large (10098 vertices) and rich in details. The original model (a) and (c) has been deformed in the richest regions: the tail (b) and the neck (d).

## 4.4 Boolean operations

In the introduction of the chapter we already highlighted how the Boolean operations are a prerogative of virtual environments, they allow to fuse together objects in different ways that are rather not possible in real world. Among the many possible uses, Boolean operations represent a key component of the constructive solid geometry (CSG), a solid modelling method with which complex objects can be constructed by assembling elementary (primitive) geometric solids. The basic operations available are three: union, difference and intersection, resulting from set theory.
Boolean operations further demonstrate their potential when coupled with subdivision surface methods. The outcome of the assembling process can be refined in order to obtain very smooth surfaces, with pleasing junction points. This is possible if Boolean operations yield mesh with a good quality tessellation.

The combination of these characteristics in addition to the easy understanding that characterize the Boolean operations led us to investigate a more general tool, considering a Boolean operations naturally integrable into a modelling system based on the 3D interaction.
However, most of the available algorithms that implement Boolean operations requires triangular mesh [23, 82, 91, 119], while experiences with quadrangular or unstructured[1] mesh are almost non-existent.

[1]In a structured mesh the elements of $\mathcal{V}$ (vertices) and $\mathcal{F}$ (faces) are compliant between them: they have the same valence, for example triangular or quadrangular meshes. An unstructured mesh is composed of heterogeneous elements of $\mathcal{V}$ and $\mathcal{F}$, and extraordinary

The need to have an efficient and robust Boolean operations algorithm able to manage arbitrary polygonal meshes is very much felt. For instance, Blender, a well-known modelling tool, in a very recent version (2.62) includes CARVE[2] a library to deal Boolean operations on generic polyhedra.

Starting from these consideration and from the assumption that even the unstructured meshes have coplanar faces, we further developed our previously implementation of a Boolean operations algorithm that works with unstructured meshes [7]. It allows to automatically perform Boolean operations among structured or unstructured and open or closed meshes. The simplicity and intuition with which you can apply Boolean operations make it suitable for non-expert users and it sounds an excellent tool to be integrated into an 3D interactive modelling environment.

Keep unchanged the initial structure of the meshes' surface (triangular, quadrangular or unstructured) has several advantages. First of all, the initial structure is left unchanged, which would otherwise be difficult to reconstruct. This means that the user is not required additional effort to imagine and interpret known mesh, as can for example happen if the system forces the triangulation of a simple cube. Moreover, you have the freedom to choice the surface subdivision method to use, without mandating the use of schemes suitable only for triangular meshes or quadrangular meshes.
For these reasons, we decided to carry on our previous work improving and extending its features.

### 4.4.1 Data structure

The storing of the mesh in a file is typically carried out by following particular conventions, which mainly produce an ordered list of vertices and faces. They allow an easy sharing of the information among different applications. Conversely, advanced modelling tools require complex data structures to better organize the mesh's information and to speed up access to each mesh component (vertices, edges or faces) or identify the set of neighbours of a specific component. Boolean operations are one of the editing tool that require a data structure suitable for this kind of queries.

There are two ways to organize the mesh information in a data structure:

- **Space partition:** in that case the cubic space containing the mesh is recursively partitioned, and each node, represented by a cube, is subdivided into eight octants. The aim is to identify which mesh component falls around a specific 3D position. A data structure that implement this

vertices make them more flexible.

[2] http://carve-csg.com

procedure is called *octree* and it is suitable for spatial indexing, nearest neighbor search and collision detection.

- **Surface mapping:** this representation is not interested in the spatial position of the mesh components, but rather constructs a map to navigate the mesh surface by moving from any element to another one connected to it. *Winged-edge* data structure is a possible mapping based on edges' orientation. The connectivity information makes the data structure invariant under both global and local space transformations (translation, scale and rotation).

The aim to integrate the Boolean operations in a 3D interactive modelling environment, such as presented in the Chapter 3, attracted us towards *winged-edge* data structure.

All editing operations presented in the previous chapter can modify the 3D position of the vertices, it means that an *octree* data structure must be updated each time the user interacts with a model or it must be created when a Boolean operation is selected. In particular, punctual and local transformations like translation and deleting, require updating of each vertex affected by the transformation.

Otherwise, a *winged-edge* data structure can be created when the model is loaded, and remains constant during the whole session. Even after deleting a vertex the data structure can be quickly updated, using its own connectivity information to correctly remove pointers.

## 4.4.2 The original algorithm

In this section we present our first implementation of an algorithm for Boolean operations, explaining its different phases and highlighting the deficiencies.

Let $\mathcal{M}_A = \{\mathcal{V}_A, \mathcal{E}_A, \mathcal{F}_A\}$ and $\mathcal{M}_B = \{\mathcal{V}_B, \mathcal{E}_B, \mathcal{F}_B\}$ two unstructured mesh, each of which is constituted by a set of vertices $\mathcal{V}$, edge $\mathcal{E}$ and faces $\mathcal{F}$, then the algorithm is able to compute the four combinations resulting from set theory: $\mathcal{M}_A \setminus \mathcal{M}_B$, $\mathcal{M}_B \setminus \mathcal{M}_A$, $\mathcal{M}_A \cap \mathcal{M}_B$ and $\mathcal{M}_A \cup \mathcal{M}_B$. The algorithm is even able to automatically classify the various products resulting from the intersection of the two models, without any intervention from the user. There are four distinct phases that characterize the Boolean process:

I) **Intersections detection.** In this first phase, the two sets $\mathcal{I}_{AB}$ and $\mathcal{I}_{BA}$ of intersection points between the two meshes are determined. $\mathcal{I}_{AB}$ and $\mathcal{I}_{BA}$ are constructed by verifying exhaustively the intersections produced by $\mathcal{E}_A$ in $\mathcal{F}_B$ and $\mathcal{E}_B$ in $\mathcal{F}_A$ (Figure 4.11(a)) respectively. At the end,

any duplicates are removed.
The research of the intersections is performed using a robust triangle-based method, but the triangulation of the faces is only virtual and it is able to correctly manage concave faces. This allows to preserve the original structure of the surface.
The triangulation of a face is real and persistent if and only if the whole intersection profile completely falls within a face, indeed only the *winged-edge* data structure is able to manage a face with a hole.

II) **Profile intersection reconstruction.** The intersections identified in the previous phase can be of different nature: they can belong to a vertex, to a edge or fall inside the face itself. The goal of this phase is to produce an ordered list of intersections $\mathcal{P}$ using the previous two sets $\mathcal{I}_{AB}$ and $\mathcal{I}_{BA}$. $\mathcal{P}$ represents the polygonal of the intersection profile (Figure 4.11(b)).
In order to properly sort the intersection points the research is alternated between the two sets $\mathcal{I}_{AB}$ and $\mathcal{I}_{BA}$. Indeed, order two intersection points that belong to face's edges is a simple task and can be done using the information of a single set $\mathcal{I}$ (e.g. $\mathcal{I}_{AB}$). While, it is more complex order a set of intersection points $L$ that fall within the face. In this second case, the information of the other set $\mathcal{I}$ (e.g. $\mathcal{I}_{BA}$), where the edges' index that created the intersections $L$ is stored, become fundamental.

III) **Profile insertion.** In this phase the polygonal $\mathcal{P}$ is inserted in both meshes, taking care not to introduce duplicate vertices and preserving the surface orientation (Figure 4.11(c)).
Three different situations may arise: *(i)* a face is not affected by the polygonal and it will be copied; *(ii)* the polygonal affects the edge of the face, hence some vertices will be added to it; *(iii)* the polygonal crosses the face, which will be split respecting the face orientation.

IV) **Patches classification.** Starting from an edge of $\mathcal{P}$ a breadth-first search is used to identify the faces that belong to each patch. This last step produces two patches for each mesh, useful to create the Boolean results (Figures 4.11(d), 4.11(e) and 4.11(f)). The possible combinations are shown in Table 4.2.

The *winged-edge* data structure proved to be very useful in the phases II, III and IV, where a quick navigation in the neighbourhood of the point of interest is required. The same researches would be less efficient than using a structure like the *octree*.

| | $\mathcal{M}_B^1$ | $\mathcal{M}_B^2$ |
|---|---|---|
| $\mathcal{M}_A^1$ | $\mathcal{M}_A \cap \mathcal{M}_B$ | $\mathcal{M}_A \setminus \mathcal{M}_B$ |
| $\mathcal{M}_A^2$ | $\mathcal{M}_B \setminus \mathcal{M}_A$ | $\mathcal{M}_A \cup \mathcal{M}_B$ |

Table 4.2: Patches combination to produce the results of Boolean operations.

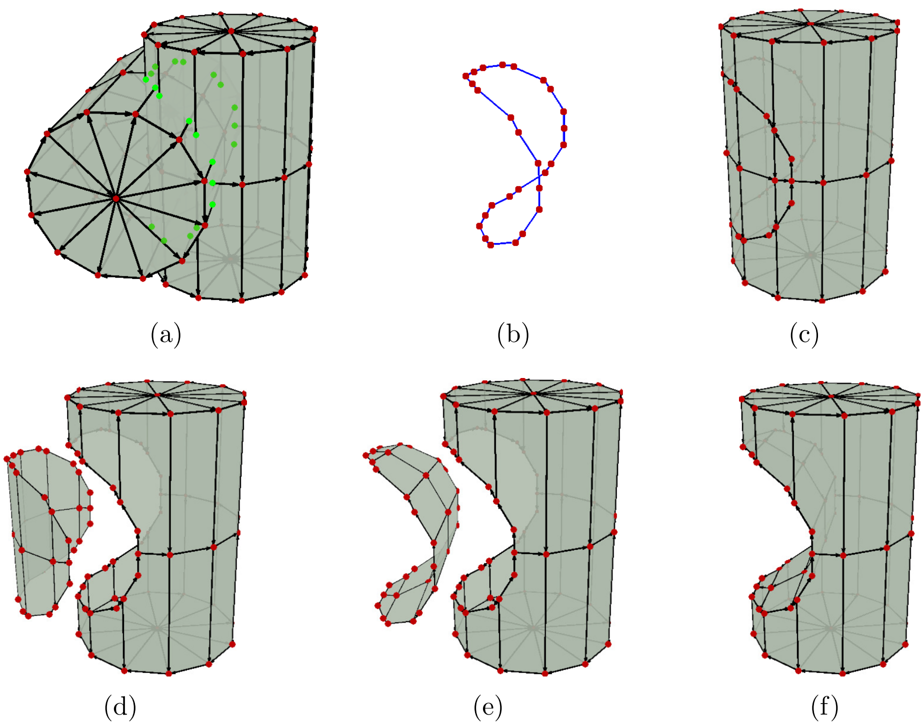


Figure 4.11: Boolean operations phases. Let consider in (a) the left cylinder the mesh $\mathcal{M}_A$, and the right one the mesh $\mathcal{M}_B$. Phase I (a): the algorithm identifies the intersections (light green dots). Phase II (b): the polygonal $\mathcal{P}$ is reconstructed. Phase III (c): the polygonal $\mathcal{P}$ is inserted in each single mesh. Phase IV (d), (e) and (f): the algorithm separates and classifies the patches and produces the resulting combinations, in that case $\mathcal{M}_B \setminus \mathcal{M}_A$.

This first version of the algorithm has evidently some flaws:

- Brute-force search to identify the intersection points during phase I is very expensive, especially with large meshes.
- The profile reconstruction phase does not manage a face twice crossed by the profile $\mathcal{P}$.

- The algorithm requires that the meshes intersect each other only once and it does not manage the tangent faces.

The CARVE library previously quoted tries to solve all these issues, even if sometime it introduces unnecessary triangles and does not correctly handle tangency. Our interest was to continue the study of our algorithm improving its features. The aim is to propose a light library, to integrate in a 3D interactive environment, able to automatically produce all the resulting Boolean combinations.

### 4.4.3 The extensions introduced

We have worked to propose some extensions to the Boolean operations algorithm, trying to preserve the autonomy feature that makes the algorithm simple and intuitive in the eyes of a common user.

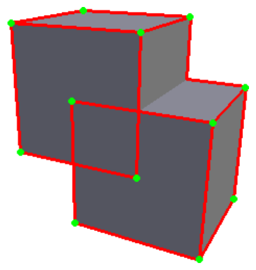
First of all, we decided to improve the initial intersections' research starting from a very simple consideration: the bounding box $\mathcal{B}$ of a mesh $\mathcal{M}$ is itself a mesh, in particular it is a cube. Hence we used the original algorithm to compute the intersection region of the two bounding box $\mathcal{B}_A$ and $\mathcal{B}_B$, and then we reduced the search of the intersection points to those elements that belong to $\mathcal{B}_A \cap \mathcal{B}_B$.
This would have been one of the few cases where *octree* data structure would make a difference, but we have already explained the reasons why the *octree* were excluded. The new approach, coupled with *winged-edge* data structure, is very efficient if the intersection between the two meshes is much smaller compared to their size and can significantly reduce the search.

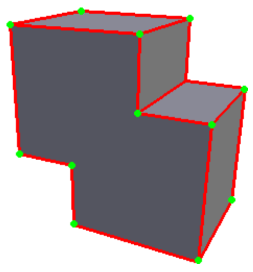
The problem of a face twice crossed by the profile $\mathcal{P}$ can occur both in the case of single intersection between the meshes and in the case of multiple intersections. We will see later how we handled the latter case.
It was necessary to refine the threshold with which the intersections edge-face are classified and assigned. We tried to bind the threshold to the average size of the mesh edges, but this solution still fails with intersections near the vertex. Hence, we used a more strict threshold and for each new intersection found the algorithm uses labels to verify that the neighbours edges had not already generated the same intersection point.

Labeling the intersection and the new threshold used helped the alternating procedure with which the profile $\mathcal{P}$ is created, making it able to manage twice crossed faces.
The single intersection between meshes was clearly too restrictive, but its solution was quite simple, especially after the changes just described. In particular, the phase II was repeated several times to generate different polygonal

$\mathcal{P}_i$, one for each intersection between meshes, each of which is then inserted during phase III.
Instead, the phase IV has requested a more robust patches classification and a classic collisions detection algorithm has been used. Let $r$ a half-line outgoing a vertex belonging to the mesh $\mathcal{M}_A$ and not belonging to any profile $\mathcal{P}_i$, then we compute the number of intersections between $r$ and $\mathcal{M}_B$. An odd number of intersections identifies a patch that falls inside the mesh $\mathcal{M}_B$, otherwise the patch is external. After the patches classification the algorithm can match them using the common information defined by each profile $\mathcal{P}_i$.

We have started to modify the algorithm so that it can manage the tangency. The idea is to use the bounding box optimization to identify which elements of the two meshes generate a tangency. In particular, the elements affected by the tangency are separately processed and excluded from subsequent phases that generate the profiles $\mathcal{P}_i$.


(a)


(b)

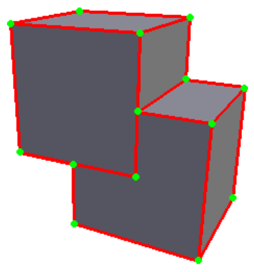

(c)

Figure 4.12: Comparison between two correct approaches to solve the same Boolean union of two tangency cubes (a). The pure Boolean union (b) obtained with our algorithm and a merge Boolean union (e) obtained with CARVE library[3].

Also in that case, our goal is to avoid to introduce any extra elements into the result and to produce a more intuitive pure union between the two meshes and not a merge operation. Consequently, the Boolean union does not preserve any

[3]The CARVE result was manually corrected, a bug introduces some overlapped edges.

elements belonging to one of the two meshes and present in the intersection, and does not require the triangulation of the tangency region.
In Figure 4.12 we show the Boolean union of two cubes. The pure union does not introduce extra elements belonging to the intersection region (Figure 4.12(b)), while a merge union preserves vertices and edges of one of the two meshes (Figure 4.12(c)).

### 4.4.4 Boolean results

Boolean operations on discrete models are intuitive and well-known in the literature, nevertheless, the solution typically proposed work on triangulated meshes or create new triangles around the junction point. In addition, we recall that our aim is to realize editing tools for non-expert users. Therefore, in this section we will present some simple examples in order to highlight the key characteristics of our approach based on unstructured mesh when used by novice users in a 3D interactive modelling environment.

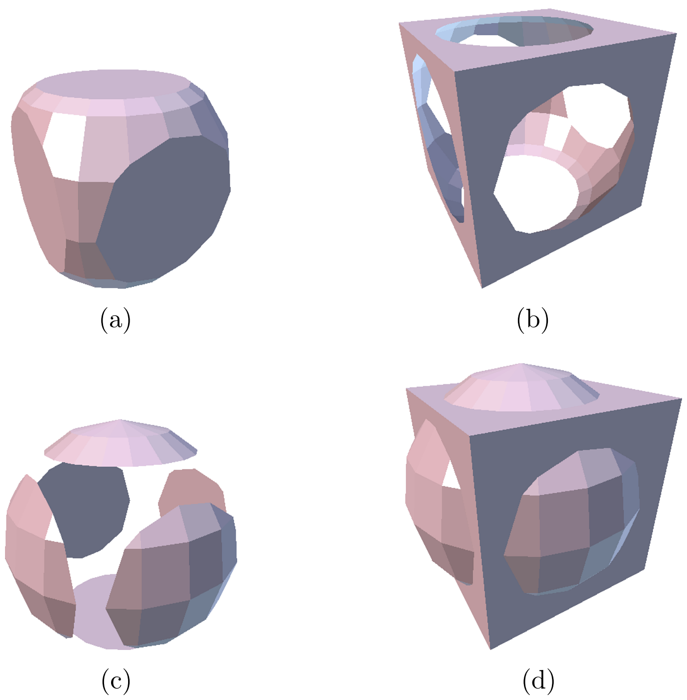
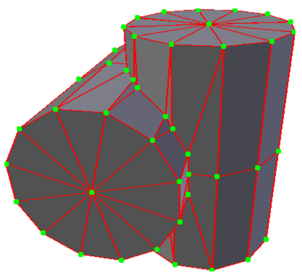
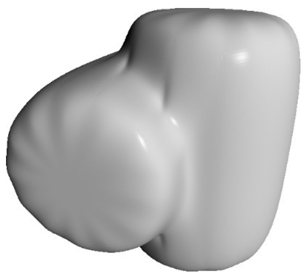
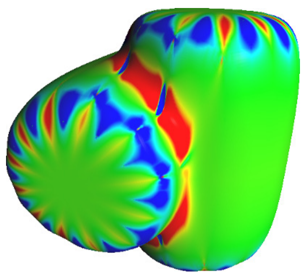


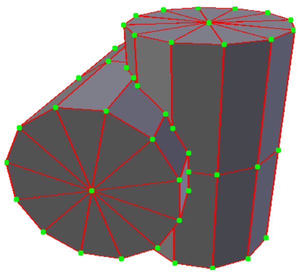
Figure 4.13: An example of multiple intersections. Given a cube mesh and a sphere mesh, the algorithm is able to automatically produce the four combinations presented in Table 4.2: cube ∩ sphere (a), cube \ sphere (b), sphere \ cube (c) and cube ∪ sphere (d).

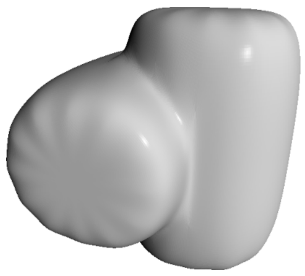
The first example, shown in Figure 4.13, presents the four combinations automatically generated by the algorithm. In particular, it is an example of multiple intersections between a cube mesh and a sphere mesh. In practice, the user places the objects in the desired position, then requires to compute the Boolean operations. The system constructs the polygonal to be inserted and uses them to split each meshes in patches. During the final phase the algorithm reassembles the patches in order to produce the intersection (Figure 4.13(a)), union (Figure 4.13(d)) and the two differences (Figures 4.13(b) and 4.13(c)), and corrects the patches' orientation to meet the manifold constraint.


(a)


(b)


(c)


(d)


(e)

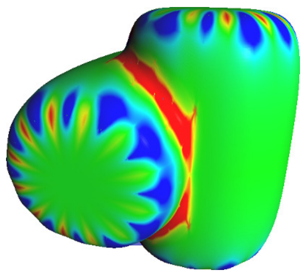

(f)

Figure 4.14: The union of two cylinders. In the first row the result obtained using an algorithm that creates triangles around the junction point, while in the second row the outcome of our algorithm. To highlight the artifacts on the junction point we present a render of the final surface after 5 steps of Catmull-Clark subdivision schemes (b) and (e), and the colour map of the Gaussian curvature (c) and (f).

In literature there are several subdivision surface refinement schemes that can be classified in interpolating and approximating, or by the type of polygon that they operate on. However, if the goal is the ease of use, it is not desirable to put the user who does not have experience with subdivision surface schemes in the position of having to choose a priori the right scheme because bound by the outcome of the Boolean operation. If using a triangulation algorithm before proceeding with the refinement, the subdivision scheme, a well

established tool to refine a mesh, would act in a completely different way with respect to the outcome of the Boolean operation.
In Figure 4.14 we wanted to compare the refined result of the outcome of a Boolean operations algorithm that introduces triangles around the intersection profile (Figure 4.14(a)) with that of our algorithm (Figure 4.14(d)). We applied an union operation between two cylinders out of phase and we refined the two results with 5 steps of Catmull-Clark subdivision schemes (Figures 4.14(b) and 4.14(e)), a scheme that accepts arbitrary initial meshes. In the first case the new triangles introduced produce artifacts near the junction point. Colouring the smooth surfaces with a map corresponding to the Gaussian curvature (Figure 4.14(c)) the artifacts become more visible, highlighting a variation of the curvature. Despite our algorithm creates irregular faces that will generate extraordinary vertices, after the subdivision steps the surface appears smooth and uniform, even on the junction point (Figure 4.14(f)).

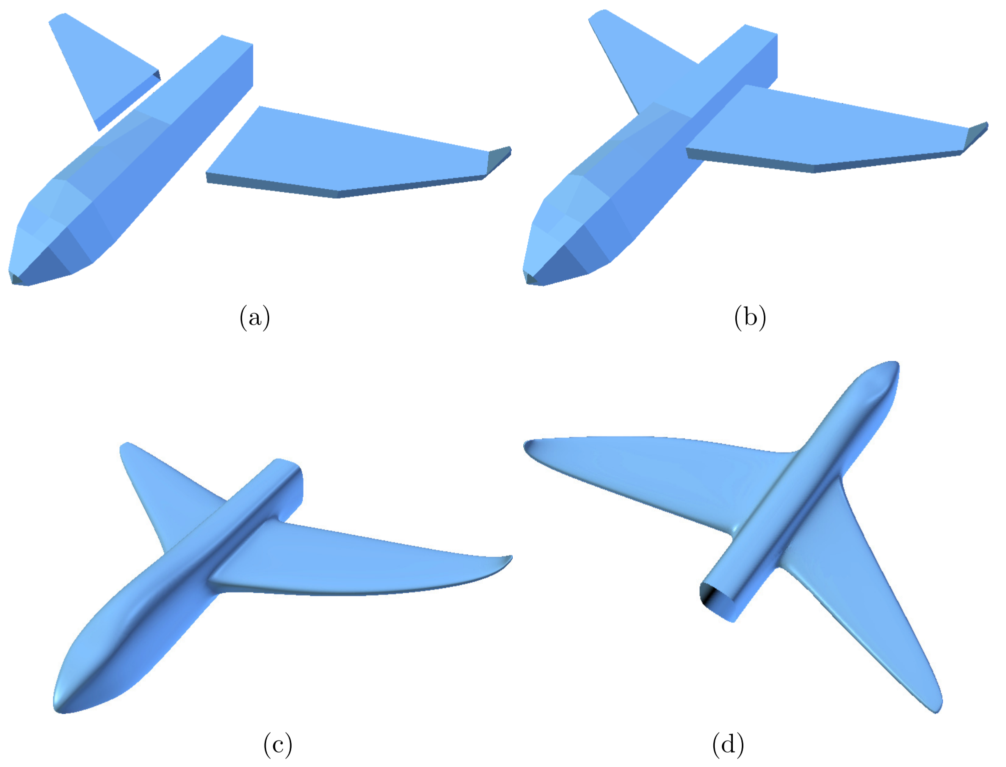
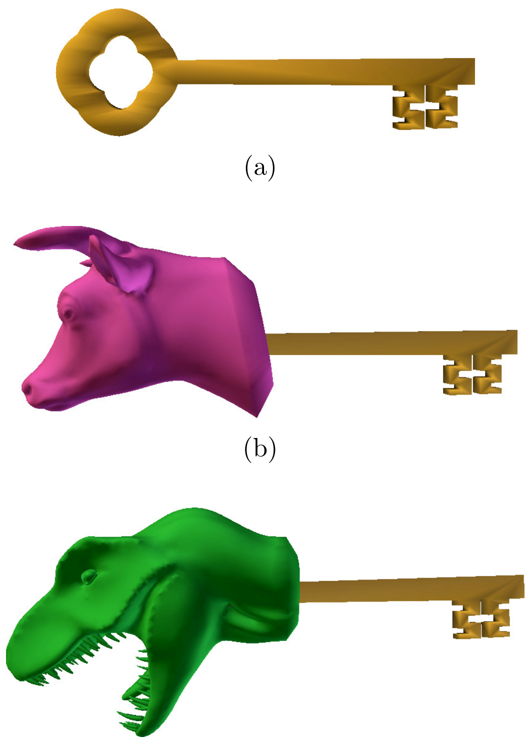


Figure 4.15: Two different wings assembled on the same fuselage. All opened surfaces in (a) are fused together, using two executions of the union procedure (b). A rendering of the final results (c) and (d) show the smooth and uniform surface.

We recall that we have studied editing tools suited to be used by non-

expert users, and in the introduction of this dissertation we hypothesized several use of a new 3D interactive modelling environment. For instance, a user who wants reconstruct broken or missing parts of objects, or more simply who wants create fancy objects. The next two examples (Figures 4.15 and 4.16) show a practical application of Boolean operations, both are obtained using our Boolean operations tool integrated in the 3D environment presented in the Chapter 3.
In Figure 4.15(a) the user, who wants to create a unique version of a toy aeroplane, verifies which wing is best suited to his idea. The Boolean operations algorithm manages also open surfaces and joins each part in a single model (Figure 4.15(b)).



(a)

(b)

(c)

Figure 4.16: Two examples of fancy objects created in a 3D interactive modelling environment, assembling a key (a), respectively, with a cow head (b) and a dinosaur head (c).

In Figure 4.16 the user creates two fancy objects. The different tools, previously presented also in the Chapter 3, were used to assembly a key with different handles. The user used the plan to cut the two heads of the cow and dinosaur and placed the items in the desired position with the interaction gestures, then applied Boolean operations to create the final model. As a result, any tilts are due to the manually alignment. This test performed by a generic user revealed the potential of the system, he noticed that is quite simple interact with the objects to create new models exploiting Boolean operations.

## 4.5 Conclusions

In this chapter we presented two advanced modelling operations, deformation and Boolean operations, both for discrete model. Once again, the aim was to identify which editing tools can be simplified and adapted to be used in a 3D interactive modelling environment by user without modelling experience.

Despite the background deformation model proposed is quite complex to be presented to a non-expert user, it remains transparent and good results can be achieved using a simple interaction metaphor. The choice of the two regions with which apply the deformation is very intuitive and easily integrable in a 3D interactive modelling environment, like the one shown in the Chapter 3.

Exploiting the total curvature, combined with non-linear constraint, we defined a better aesthetic measure for deformation of elastic bodies. It allows to obtain high qualitative results whose deformation recall the real deformation. The proposed variational model satisfies several requirements, like local influence, details preservation, realistic effects and structural preservation. Nevertheless, there are some open challenge that we will consider. The first one regards the self-intersect: the model can be surrounded with bounding boxes and using collision dynamics method the surface integrity can be preserved. The second interesting challenge regards the efficiency. Currently deformation editing can be achieved in real-time only for medium size objects, but we believe that with optimized implementation of linear system solvers the real-time (or semi real-time) deformation can be extended to all models.

We presented a solution for Boolean operations among opened and closed unstructured meshes. It is able to automatically compute union, difference and intersection, resulting from set theory.

The Boolean operations for discrete models was widely treated in the literature, but the proposed solutions often require a triangle-based mesh. We shown how this requirement can be very restrictive, it involves the change of the surface structure of the mesh and can limit the intuitiveness with which interpreting the resulting surface. We focused on a simple method that does

not need triangles and does not introduce extra elements that preserves the original surface structure of the meshes.
The library for Boolean operations has been already integrated in the 3D interactive modelling environment presented in the Chapter 3. Users have found the library very simple and intuitive, thanks to the ability to position models with their hands and to automatically combine them.

Deformation and Boolean operations are just two of the many advanced editing tools for discrete models. In future, it will be interesting to deepen new features inspired by the current modelling systems.
We recall that the aim is to adapt some classical methods in order to be used by non-expert users. The challenge is to propose solutions easy to use, which results are qualitatively appreciable.

# 5

# Conclusions

*The journey of a thousand miles begins with one step.*

Lao Tzu

The 3D modelling, introduced several decades ago, has significantly transformed the designers' and modellers' work, continuously providing more and more advanced editing tools to support the process by which they implement their ideas. The expert users have available methods to manage discrete and continuous virtual models, during all different phases that characterize the modelling process, summarizable in creation, interaction and editing.

Nowadays, despite many technological innovations like hand gesture input devices, advanced 3D visors or 3D printers have been submitted, the 3D modelling is still not a widespread activity and continues to be bound to professional environments. The main limitation is the complexity of use of the modelling tools currently available. They require a intensive training, high skills and significantly hindering the use by common users. However, just these new technologies recently introduced suggest new directions for 3D modelling.

We imagined a scenario in which non-expert users can easily perform all the steps that characterize the creation and management of 3D virtual models, assuming just few reasons that will bring the common users to approach this new type of activity. For instance, the repair of real objects through the reconstruction by 3D printing of broken or missed parts, the creation of fancy objects, the modify of a 3D avatar. Up to envisage new markets where we can buy the 3D virtual model of the desired object, and modify it afterwards in order to adapt the shape to our needs before the printing.

In this context, the work introduced in this dissertation is focused on the study of methods and algorithms that allow to bridge the gap between modelling methodologies and non-expert users. We identified three main phases that characterize the management of 3D virtual models: the creation, the navigation and interaction and the editing. In particular, we evaluated a subset functionalities for each topics and our approach has always had the novice user as the main focus of the work, reducing complexity of the proposed solutions. Moreover, considering the major number of open challenges offered by the discrete representation of the virtual models we adopted the polygonal approximation (mesh) to represent the virtual object surface.

In the first Chapter, we proposed an innovative software solution able to reconstruct physical object. It is coupled with a very promising pen-like device endowed with several low cost sensors. Even better, it is able to create new virtual object by sketch-based modelling, allowing the user to draw three-dimensional virtual objects in the mid-air just as he would for two-dimensional figures on paper. The whole system is very simple and intuitive and fuses in the same environment reverse engineering capabilities and freehand modelling features. It has attracted the interest of common users, but even users familiar with CAD technologies have recognized its potential specifically in the ability to quickly add new details.

Then we addressed the navigation and visualization problems in 3D virtual modelling environments. The classic 3D interfaces used to navigate 3D geometries require a large variety of command to be controlled, a combination of keyboard and mouse entries that untrained users need time and experience to learn. We investigated the use of a new 3D interaction gesture-based devices as tool able to inspect the scene and to carry out global and local changes to the model. The optical devices have achieved high accuracy allowing us to directly use the 3D position of the hand in the air. Even though a full gesture-based environment could become too complex, 3D gestures can be successfully employed to simplify most 3D activities. We conducted studies to improve the depth perception in a 3D modelling environment, but the results have been unsatisfactory for now and we have proposed a partial solution that helps the user to understand the mutual position of objects and user's hands in the scene.

Typically, a novice user needs of a small number of functionalities that make it possible to modify the shape of the object as he will. For this reason, in the third Chapter we started to investigate two advanced editing methods: deformation and Boolean operations. Keeping in mind that the complexity must remain completely transparent to the user, we proposed intuitive and integrable methods able to emulate the real physical deformation and to exploit a virtual models peculiarity like objects interpenetration. The main contribution regards the possibility to obtain high quality results in a simple way.

The deformation procedure requires that the user selects the fixed region of the model and specifies the target position of the handle one, the system automatically computes the final rest position of the whole object. While the Boolean operations tool does not have particular constraints and can be activated after placing the objects in the desired position.

There are still formidable challenges to be tackled in order to create 3D interactive modelling environments suitable for use by inexperienced users. Some of them concern the visualization in a 3D modelling environment that can improve the depth perception in systems such as those presented in the first two chapters of the dissertation. Other interesting challenges regard the optimization of complex editing methods, in order to provide algorithms able to automatically modify the virtual model in real-time. Finally, it will be compelling find new editing operations to be adapted to this new 3D modelling paradigm.

# Bibliography

[1] F. Abbasinejad, P. Joshi, and N. Amenta, *Surface patches from unorganized space curves*, in Computer Graphics Forum, vol. 30, Wiley Online Library, 2011, pp. 1379–1387.

[2] O. K.-C. Au, H. Fu, C.-L. Tai, and D. Cohen-Or, *Handle-aware isolines for scalable shape editing*, in ACM Transactions on Graphics (TOG), vol. 26, ACM, 2007, p. 83.

[3] S.-H. Bae, R. Balakrishnan, and K. Singh, *Ilovesketch: as-natural-as-possible sketching system for creating 3d curve models*, in Proceedings of the 21st annual ACM symposium on User interface software and technology, ACM, 2008, pp. 151–160.

[4] C. V. Beccari, E. Farella, A. Liverani, S. Morigi, and M. Rucci, *A fast interactive reverse-engineering system*, Computer-Aided Design, 42 (2010), pp. 860–873.

[5] G. H. Bendels and R. Klein, *Mesh forging: editing of 3d-meshes using implicitly defined occluders*, in Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Eurographics Association, 2003, pp. 207–217.

[6] G. Bernstein and D. Fussell, *Fast, exact, linear booleans*, in Computer Graphics Forum, vol. 28, Wiley Online Library, 2009, pp. 1269–1278.

[7] F. Bertini, *Modellazione di mesh 3d non strutturate*, Bachelor's degree thesis, University of Bologna, 2008.

[8] F. Bertini and S. Morigi, *Deformation by discrete elastica*, in WSCG 2013 Full Paper Proceedings, 21st International Conference in

Central Europe on Computer Graphics, Visualization and Computer Vision 2013, in co-operation with EUROGRAPHICS Association, M. M. Oliveira and V. Skala, eds., Union Agency, 2013, pp. 223–232.

[9] M. Bessmeltsev, C. Wang, A. Sheffer, and K. Singh, *Design-driven quadrangulation of closed 3d curves*, ACM Transactions on Graphics (TOG), 31 (2012), p. 178.

[10] H. Biermann, D. Kristjansson, and D. Zorin, *Approximate boolean operations on free-form solids*, in Siggraph, vol. 1, 2001, pp. 185–194.

[11] M. Billinghurst, H. Kato, and I. Poupyrev, *Tangible augmented reality*, ACM SIGGRAPH ASIA, (2008), pp. 1–10.

[12] M. Botsch and L. Kobbelt, *Real-time shape editing using radial basis functions*, in Computer graphics forum, vol. 24, Wiley Online Library, 2005, pp. 611–621.

[13] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon mesh processing*, CRC press, 2010.

[14] M. Botsch, M. Pauly, M. H. Gross, and L. Kobbelt, *Primo: coupled prisms for intuitive surface modeling*, in Symposium on Geometry Processing, no. EPFL-CONF-149310, 2006, pp. 11–20.

[15] M. Botsch, M. Pauly, M. Wicke, and M. Gross, *Adaptive space deformations based on rigid cells*, in Computer Graphics Forum, vol. 26, Wiley Online Library, 2007, pp. 339–347.

[16] M. Botsch and O. Sorkine, *On linear variational surface deformation methods*, Visualization and Computer Graphics, IEEE Transactions on, 14 (2008), pp. 213–230.

[17] M. Botsch, R. Sumner, M. Pauly, and M. Gross, *Deformation transfer for detail-preserving surface editing*, in Vision, Modeling & Visualization, Citeseer, 2006, pp. 357–364.

[18] D. A. Bowman, J. Chen, C. A. Wingrave, J. F. Lucas, A. Ray, N. F. Polys, Q. Li, Y. Haciahmetoglu, J.-S. Kim, S. Kim, et al., *New directions in 3d user interfaces.*, IJVR, 5 (2006), pp. 3–14.

[19] D. A. Bowman, E. Kruijff, J. J. LaViola Jr, and I. Poupyrev, *3D user interfaces: theory and practice*, Addison-Wesley, 2004.

[20] G. Bruder, F. Steinicke, and W. Sturzlinger, *To touch or not to touch?: comparing 2d touch and 3d mid-air interaction on stereoscopic tabletop surfaces*, in Proceedings of the 1st symposium on Spatial user interaction, ACM, 2013, pp. 9–16.

[21] V. Carbone, M. Carocci, E. Savio, G. Sansoni, and L. De Chiffre, *Combination of a vision system and a coordinate measuring machine for the reverse engineering of freeform surfaces*, The International Journal of Advanced Manufacturing Technology, 17 (2001), pp. 263–271.

[22] L. Y. Chang, N. S. Pollard, T. M. Mitchell, and E. P. Xing, *Feature selection for grasp recognition from optical markers*, in Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, IEEE, 2007, pp. 2944–2950.

[23] M. Chen, X. Y. Chen, K. Tang, and M. M. Yuen, *Efficient boolean operation on manifold mesh surfaces*, Computer-Aided Design and Applications, 7 (2010), pp. 405–415.

[24] C. K. Chui and G. Chen, *Kalman filtering, with real time applications*, Springer, 2009.

[25] A. Cohé and M. Hachet, *Beyond the mouse: Understanding user gestures for manipulating 3d objects from touchscreen inputs*, Computers & Graphics, 36 (2012), pp. 1119–1131.

[26] K. Das, P. Diaz-Gutierrez, and M. Gopi, *Sketching free-form surfaces using network of curves*, in Proc. Eurographics workshop sketch-based interfaces and modeling. Eurographics, 2005.

[27] D. Dave, A. Chowriappa, and T. Kesavadas, *Gesture interface for 3d cad modeling using kinect*, Computer-Aided Design and Applications, 10 (2013), pp. 663–669.

[28] B. R. De Araújo, G. Casiez, J. A. Jorge, and M. Hachet, *Mockup builder: 3d modeling on and above the surface*, Computers & Graphics, 37 (2013), pp. 165–178.

[29] M. P. Do Carmo, *Riemannian geometry*, Springer, 1992.

[30] F. R. Feito, C. J. Ogáyar, R. J. Segura, and M. Rivero, *Fast and accurate evaluation of regularized boolean operations on triangulated solids*, Computer-Aided Design, 45 (2013), pp. 705–716.

[31] M. Fiorentino, R. Radkowski, C. Stritzke, A. E. Uva, and G. Monno, *Design review of cad assemblies using bimanual natural interface*, International Journal on Interactive Design and Manufacturing (IJIDeM), 7 (2013), pp. 249–260.

[32] M. Fiorentino, A. E. Uva, G. Monno, and R. Radkowski, *Augmented technical drawings: a novel technique for natural interactive visualization of computer-aided design models*, Journal of Computing and Information Science in Engineering, 12 (2012), p. 024503.

[33] J. Gain and D. Bechmann, *A survey of spatial deformation from a user-centered perspective*, ACM Transactions on Graphics (TOG), 27 (2008), p. 107.

[34] Y. Gardan and E. Perrin, *An algorithm reducing 3d boolean operations to a 2d problem: concepts and results*, Computer-Aided Design, 28 (1996), pp. 277–287.

[35] J. Griessmair and W. Purgathofer, *Deformation of solids with trivariate b-splines*, in Proceedings of eurographics, vol. 89, 1989, pp. 137–148.

[36] T. Grossman, R. Balakrishnan, and K. Singh, *An interface for creating and manipulating curves using a high degree-of-freedom curve input device*, in Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, 2003, pp. 185–192.

[37] P. Hachenberger, L. Kettner, and K. Mehlhorn, *Boolean operations on 3d selective nef complexes: Data structure, algorithms, optimized implementation and experiments*, Computational Geometry, 38 (2007), pp. 64–99.

[38] W. Helfrich, *Elastic properties of lipid bilayers: theory and possible experiments.*, Zeitschrift für Naturforschung. Teil C: Biochemie, Biophysik, Biologie, Virologie, 28 (1973), p. 693.

[39] W. M. Hsu, J. F. Hughes, and H. Kaufman, *Direct manipulation of free-form deformations*, in ACM SIGGRAPH Computer Graphics, vol. 26, ACM, 1992, pp. 177–184.

[40] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum, *Subspace gradient domain mesh deformation*, in ACM Transactions on Graphics (TOG), vol. 25, ACM, 2006, pp. 1126–1134.

[41] T. Igarashi, S. Matsuoka, and H. Tanaka, *Teddy: a sketching interface for 3d freeform design*, in ACM SIGGRAPH 2007 courses, ACM, 2007, p. 21.

[42] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al., *Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera*, in Proceedings of the 24th annual ACM symposium on User interface software and technology, ACM, 2011, pp. 559–568.

[43] J. Jankowski, M. Hachet, et al., *A survey of interaction techniques for interactive 3d environments*, in Eurographics 2013-STAR, 2013.

[44] G. Johnson, M. D. Gross, J. Hong, and E. Yi-Luen Do, *Computational support for sketching in design: a review*, Foundations and Trends in Human-Computer Interaction, 2 (2009), pp. 1–93.

[45] S. Jordà, G. Geiger, M. Alonso, and M. Kaltenbrunner, *The reactable: exploring the synergy between live music performance and tabletop tangible interfaces*, in Proceedings of the 1st international conference on Tangible and embedded interaction, ACM, 2007, pp. 139–146.

[46] J. Jorge and F. Samavati, *Sketch-based interfaces and modeling*, Springer, 2010.

[47] B. Kaan Karamete, S. Dey, E. L. Mestreau, R. Aubry, and F. A. Bulat-Jara, *An algorithm for discrete booleans with applications to finite element modeling of complex systems*, Finite Elements in Analysis and Design, 68 (2013), pp. 10–27.

[48] L. B. Kara and K. Shimada, *Sketch-based 3d-shape creation for industrial styling design*, Computer Graphics and Applications, IEEE, 27 (2007), pp. 60–71.

[49] O. A. Karpenko and J. F. Hughes, *Smoothsketch: 3d free-form shapes from complex sketches*, in ACM Transactions on Graphics (TOG), vol. 25, ACM, 2006, pp. 589–598.

[50] H. Kaufmann and D. Schmalstieg, *Mathematics and geometry education with collaborative augmented reality*, Computers & Graphics, 27 (2003), pp. 339–345.

[51] H. Kaufmann and D. Schmalstieg, *Designing immersive virtual reality for geometry education*, in Virtual Reality Conference, 2006, IEEE, 2006, pp. 51–58.

[52] Y. J. Kil, P. Renzulli, O. Kreylos, B. Hamann, G. Monno, and O. G. Staadt, *3d warp brush modeling*, Computers & Graphics, 30 (2006), pp. 610–618.

[53] J.-H. Kim, N. D. Thang, and T.-S. Kim, *3-d hand motion tracking and gesture recognition using a data glove*, in Industrial Electronics, 2009. ISIE 2009. IEEE International Symposium on, IEEE, 2009, pp. 1013–1018.

[54] L. Lamberti and F. Camastra, *Real-time hand gesture recognition using a color glove*, in Image Analysis and Processing–ICIAP 2011, Springer, 2011, pp. 365–373.

[55] M. Lambooij, M. Fortuin, I. Heynderickx, and W. IJsselsteijn, *Visual discomfort and visual fatigue of stereoscopic displays: a review*, Journal of Imaging Science and Technology, 53 (2009), pp. 30201–1.

[56] H. J. Lamousin and W. N. Waggenspack Jr, *Nurbs-based free-form deformations*, Computer Graphics and Applications, IEEE, 14 (1994), pp. 59–65.

[57] J. Lazar, J. H. Feng, and H. Hochheiser, *Research methods in human-computer interaction*, John Wiley & Sons, 2010.

[58] M. Lee, R. Green, and M. Billinghurst, *3d natural hand interaction for ar applications*, in Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference, IEEE, 2008, pp. 1–6.

[59] A. Levin, *Interpolating nets of curves by smooth subdivision surfaces*, in Proceedings of the 26th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 1999, pp. 57–64.

[60] H. Lin, W. Chen, and H. Bao, *Adaptive patch-based mesh fitting for reverse engineering*, Computer-Aided Design, 39 (2007), pp. 1134–1142.

[61] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rossi, and H.-P. Seidel, *Differential coordinates for interactive mesh editing*, in Shape Modeling Applications, 2004. Proceedings, IEEE, 2004, pp. 181–190.

[62] N. Looi, C. Loo, and K. Pau, *Real time actions and gestures as intuitive user interface for creative 3d modeling*, in Active Media Technology, 2005.(AMT 2005). Proceedings of the 2005 International Conference on, IEEE, 2005, pp. 487–490.

[63] R. Love, *Blessing the nations in the 21st century: A 3d approach to apostolic ministry*, International Journal of Frontier Missions, 25 (2008), pp. 31–37.

[64] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, *Estimation of imu and marg orientation using a gradient descent algorithm*, in Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on, IEEE, 2011, pp. 1–7.

[65] R. Mahony, T. Hamel, and J.-M. Pflimlin, *Nonlinear complementary filters on the special orthogonal group*, Automatic Control, IEEE Transactions on, 53 (2008), pp. 1203–1218.

[66] F. Martínez, A. J. Rueda, and F. R. Feito, *A new algorithm for computing boolean operations on polygons*, Computers & Geosciences, 35 (2009), pp. 1177–1185.

[67] D. Martinez Plasencia, E. Joyce, and S. Subramanian, *Mistable: reach-through personal screens for tabletops*, in Proceedings of the 32nd annual ACM conference on Human factors in computing systems, ACM, 2014, pp. 3493–3502.

[68] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, *Discrete differential-geometry operators for triangulated 2-manifolds*, in Visualization and mathematics III, Springer, 2003, pp. 35–57.

[69] B. Milosevic, F. Bertini, E. Farella, and S. Morigi, *A smartpen for 3d interaction and sketch-based surface modeling.* Submitted to ACM Transactions on Graphics (TOG), 2014.

[70] J. Mitani, H. Suzuki, and F. Kimura, *3d sketch: sketch-based model reconstruction and rendering*, in From geometric modeling to shape modeling, Springer, 2002, pp. 85–98.

[71] S. Morigi and M. Rucci, *Reconstructing surfaces from sketched 3d irregular curve networks*, in Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling, ACM, 2011, pp. 39–46.

[72] S. Murugappan, H. Liu, K. Ramani, et al., *Shape-it-up: Hand gesture based creative expression of 3d shapes using intelligent generalized cylinders*, Computer-Aided Design, 45 (2013), pp. 277–287.

[73] S. Nam and Y. Chai, *Spacesketch: Shape modeling with 3d meshes and control curves in stereoscopic environments*, Computers & Graphics, 36 (2012), pp. 526–533.

[74] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, *Fibermesh: designing freeform surfaces with 3d curves*, in ACM Transactions on Graphics (TOG), vol. 26, ACM, 2007, p. 41.

[75] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge, *Sketch-based modeling: A survey*, Computers & Graphics, 33 (2009), pp. 85–103.

[76] P. Olsson, F. Nysjo, S. Seipel, and I. Carlbom, *Physically co-located haptic interaction with 3d displays*, in Haptics Symposium (HAPTICS), 2012 IEEE, IEEE, 2012, pp. 267–272.

[77] G. Orbay and L. B. Kara, *Sketch-based surface design using malleable curve networks*, Computers & Graphics, 36 (2012), pp. 916–929.

[78] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross, *Shape modeling with point-sampled geometry*, in ACM Transactions on Graphics (TOG), vol. 22, ACM, 2003, pp. 641–650.

[79] Y. Peng, J.-H. Yong, W.-M. Dong, H. Zhang, and J.-G. Sun, *A new algorithm for boolean operations on general polygons*, Computers & Graphics, 29 (2005), pp. 57–70.

[80] T.-C. Poon, *Digital holography and three-dimensional display: Principles and Applications*, Springer, 2006.

[81] W. Qi and J.-B. Martens, *Tangible user interfaces for 3d clipping plane interaction with volumetric data: a case study*, in Proceedings of the 7th international conference on Multimodal interfaces, ACM, 2005, pp. 252–258.

[82] H. Qu, M. Pan, B. Wang, Y. Wang, and Z. Wang, *Boolean operations on triangulated solids and their applications in 3d geological modelling*, Advances in Spatio-Temporal Analysis, 5 (2007), p. 21.

[83] S. Radhakrishnan, Y. Lin, I. Zeid, and S. Kamarthi, *Finger-based multitouch interface for performing 3d cad operations*, International Journal of Human-Computer Studies, 71 (2013), pp. 261–275.

[84] S. S. Rautaray and A. Agrawal, *Vision based hand gesture recognition for human computer interaction: a survey*, Artificial Intelligence Review, (2012), pp. 1–54.

[85] A. A. Requicha and H. B. Voelcker, *Boolean operations in solid modeling: Boundary evaluation and merging algorithms*, Proceedings of the IEEE, 73 (1985), pp. 30–44.

[86] M. Rivero and F. R. Feito, *Boolean operations on general planar polygons*, Computers & Graphics, 24 (2000), pp. 881–896.

[87] M. Rucci, *Geometric Surface Processing and Virtual Modeling*, PhD thesis, University of Padova, 2013.

[88] E. Sachs, A. Roberts, and D. Stoops, *3-draw: A tool for designing 3d shapes*, IEEE Computer Graphics and Applications, 11 (1991), pp. 18–26.

[89] B. Sadri and K. Singh, *Flow-complex-based shape reconstruction from 3d curves*, ACM Transactions on Graphics (TOG), 33 (2014), p. 20.

[90] S. Schaefer, J. Warren, and D. Zorin, *Lofting curve networks using subdivision surfaces*, in Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM, 2004, pp. 103–114.

[91] M. Schifko, B. Jüttler, and B. Kornberger, *Industrial application of exact boolean operations for meshes*, in Proceedings of the 26th Spring Conference on Computer Graphics, ACM, 2010, pp. 165–172.

[92] S. Schkolne, M. Pruett, and P. Schröder, *Surface drawing: creating organic 3d shapes with the hand and tangible tools*, in Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, 2001, pp. 261–268.

[93] R. Schmidt, A. Khan, K. Singh, and G. Kurtenbach, *Analytic drawing of 3d scaffolds*, in ACM Transactions on Graphics (TOG), vol. 28, ACM, 2009, p. 149.

[94] R. Schneider and L. Kobbelt, *Geometric fairing of irregular meshes for free-form surface design*, Computer aided geometric design, 18 (2001), pp. 359–379.

[95] T. W. Sederberg and S. R. Parry, *Free-form deformation of solid geometric models*, in ACM SIGGRAPH Computer Graphics, vol. 20, ACM, 1986, pp. 151–160.

[96] I. Sexton and P. Surman, *Stereoscopic and autostereoscopic display systems*, Signal Processing Magazine, IEEE, 16 (1999), pp. 85–99.

[97] O. Shaer and E. Hornecker, *Tangible user interfaces: past, present, and future directions*, Foundations and Trends in Human-Computer Interaction, 3 (2010), pp. 1–137.

[98] A. Sheffer and V. Kraevoy, *Pyramid coordinates for morphing and deformation*, in 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on, IEEE, 2004, pp. 68–75.

[99] X. Shi, K. Zhou, Y. Tong, M. Desbrun, H. Bao, and B. Guo, *Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics*, in ACM Transactions on Graphics (TOG), vol. 26, ACM, 2007, p. 81.

[100] A. Shtof, A. Agathos, Y. Gingold, A. Shamir, and D. Cohen-Or, *Geosemantic snapping for sketch-based modeling*, in Computer Graphics Forum, vol. 32, Wiley Online Library, 2013, pp. 245–253.

[101] D. Sieger, S. Menzel, and M. Botsch, *A comprehensive comparison of shape deformation methods in evolutionary design optimization*, in Proceedings of the 3rd International Conference on Engineering Optimization, 2012.

[102] J. Smith and N. A. Dodgson, *A topologically robust algorithm for boolean operations on polyhedral shapes using approximate arithmetic*, Computer-Aided Design, 39 (2007), pp. 149–163.

[103] J. Song, S. Cho, S.-Y. Baek, K. Lee, and H. Bang, *Gafinc: Gaze and finger control interface for 3d model manipulation in cad application*, Computer-Aided Design, 46 (2014), pp. 239–245.

[104] P. Song, W. B. Goh, W. Hutama, C.-W. Fu, and X. Liu, *A handle bar metaphor for virtual object manipulation with mid-air interaction*, in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2012, pp. 1297–1306.

[105] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, *Laplacian surface editing*, in Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM, 2004, pp. 175–184.

[106] R. W. Sumner, J. Schmid, and M. Pauly, *Embedded deformation for shape manipulation*, in ACM Transactions on Graphics (TOG), vol. 26, ACM, 2007, p. 80.

[107] W. Sun and X. Hu, *Reasoning boolean operation based modeling for heterogeneous objects*, Computer-Aided Design, 34 (2002), pp. 481–488.

[108] T. Surazhsky, E. Magid, O. Soldea, G. Elber, and E. Rivlin, *A comparison of gaussian and mean curvatures estimation methods on triangular meshes*, in Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on, vol. 1, IEEE, 2003, pp. 1021–1026.

[109] K. Takayama, D. Panozzo, A. Sorkine-Hornung, and O. Sorkine-Hornung, *Sketch-based generation and editing of quad meshes*, ACM Transactions on Graphics (TOG), 32 (2013), p. 97.

[110] D. Terzopoulos, *On matching deformable models to images: Direct and iterative solutions*, in Topical Meeting on Machine Vision, Technical Digest Series, Vol. 12, Washington, DC, March 1987, Optical Society of America, pp. 160–167.

[111] D. Terzopoulos and K. Fleischer, *Deformable models*, The visual computer, 4 (1988), pp. 306–331.

[112] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, *Elastically deformable models*, in ACM SIGGRAPH Computer Graphics, vol. 21, ACM, 1987, pp. 205–214.

[113] C. C. Wang, *Approximate boolean operations on large polyhedral solids with partial mesh reconstruction*, Visualization and Computer Graphics, IEEE Transactions on, 17 (2011), pp. 836–849.

[114] R. Wang, S. Paris, and J. Popović, *6d hands: markerless hand-tracking for computer aided design*, in Proceedings of the 24th annual ACM symposium on User interface software and technology, ACM, 2011, pp. 549–558.

[115] R. Y. Wang and J. Popović, *Real-time hand-tracking with a color glove*, in ACM Transactions on Graphics (TOG), vol. 28, ACM, 2009, p. 63.

[116] G. Wesche and H.-P. Seidel, *Freedrawer: a free-form sketching system on the responsive workbench*, in Proceedings of the ACM symposium on Virtual reality software and technology, ACM, 2001, pp. 167–174.

[117] B. Xu, W. Chang, A. Sheffer, A. Bousseau, J. McCrae, K. Singh, et al., *True2form: 3d curve networks from 2d sketches via selective regularization*, ACM Transactions on Graphics, 33 (2014).

[118] K. Xu, K. Chen, H. Fu, W.-L. Sun, and S.-M. Hu, *Sketch2scene: Sketch-based co-retrieval and co-placement of 3d models*, ACM Transactions on Graphics (TOG), 32 (2013), p. 123.

[119] S. Xu and J. Keyser, *Fast and robust booleans on polyhedra*, Computer-Aided Design, 45 (2013), pp. 529–534.

[120] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, *Mesh editing with poisson-based gradient field manipulation*, in ACM Transactions on Graphics (TOG), vol. 23, ACM, 2004, pp. 644–651.

[121] X. Zabulis, H. Baltzakis, and A. Argyros, *Vision-based hand gesture recognition for human-computer interaction*, The Universal Access Handbook. LEA, (2009).

[122] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes, *Sketch: an interface for sketching 3d scenes*, in ACM SIGGRAPH 2007 courses, ACM, 2007, p. 19.

[123] S. Zheng, J. Hong, and K. Jia, *Boolean operations on triangulated solids*, in Assembly and Manufacturing (ISAM), 2013 IEEE International Symposium on, IEEE, 2013, pp. 348–351.

*Computer science research is different from these more traditional disciplines. Philosophically it differs from the physical sciences because it seeks not to discover, explain, or exploit the natural world, but instead to study the properties of machines of human creation. In this it is analogous to mathematics, and indeed the "science" part of computer science is, for the most part mathematical in spirit.*

- Dennis Ritchie -