

Dipartimento di Informatica, Bioingegneria,  
Robotica ed Ingegneria dei Sistemi

---

**A Framework for the Semantics-aware Modelling of Objects**

by

Andreas Scalas

Theses Series

**DIBRIS-TH-2021-XXXIII**

---

DIBRIS, Università di Genova

Via Opera Pia, 13 16145 Genova, Italy

<http://www.dibris.unige.it/>

**Università degli Studi di Genova**

**Dipartimento di Informatica, Bioingegneria,**

**Robotica ed Ingegneria dei Sistemi**

**Ph.D. Thesis in Computer Science and Systems Engineering**

**Computer Science Curriculum**

**A Framework for the Semantics-aware Modelling of  
Objects**

by

Andreas Scalas

March, 2021

**Dottorato di Ricerca in Informatica ed Ingegneria dei Sistemi**  
**Indirizzo Informatica**  
**Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi**  
**Università degli Studi di Genova**

DIBRIS, Univ. di Genova  
Via Opera Pia, 13  
I-16145 Genova, Italy  
<http://www.dibris.unige.it/>

**Ph.D. Thesis in Computer Science and Systems Engineering**  
**Computer Science Curriculum**  
(S.S.D. INF/01)

Submitted by Andreas Scalas  
DIBRIS, Univ. di Genova

• • • •

Date of submission: December 2020

Title: A Framework for the Semantics-aware Modelling of Objects

Advisors: Michela Mortara, Michela Spagnuolo  
Istituto di Matematica Applicata e Tecnologie Informatiche “Enrico Magenes”  
Consiglio Nazionale delle Ricerche

• • •

Ext. Reviewers: Mario Botsch, Hamid Laga, Sorin Hermon

## Abstract

*The evolution of 3D visual content calls for innovative methods for modelling shapes based on their intended usage, function and role in a complex scenario. Even if different attempts have been done in this direction, shape modelling still mainly focuses on geometry. However, 3D models have a structure, given by the arrangement of salient parts, and shape and structure are deeply related to semantics and functionality.*

*Changing geometry without semantic clues may invalidate such functionalities or the meaning of objects or their parts. We approach the problem by considering semantics as the formalised knowledge related to a category of objects; the geometry can vary provided that the semantics is preserved. We represent the semantics and the variable geometry of a class of shapes through the parametric template: an annotated 3D model whose geometry can be deformed provided that some semantic constraints remain satisfied.*

*In this thesis, we design and develop a framework for the semantics-aware modelling of shapes, offering the user a single application environment where the whole workflow of defining the parametric template and applying semantics-aware deformations can take place. In particular, the system provides tools for the selection and annotation of geometry based on a formalised contextual knowledge; shape analysis methods to derive new knowledge implicitly encoded in the geometry, and possibly enrich the given semantics; a set of constraints that the user can apply to salient parts and a deformation operation that takes into account the semantic constraints and provides an optimal solution. The framework is modular so that new tools can be continuously added.*

*While producing some innovative results in specific areas, the goal of this work is the development of a comprehensive framework combining state of the art techniques and new algorithms, thus enabling the user to conceptualise her/his knowledge and model geometric shapes.*

*The original contributions regard the formalisation of the concept of annotation,*

*with attached properties, and of the relationships between significant parts of objects; a new technique for guaranteeing the persistence of annotations after significant changes in shape's resolution; the exploitation of shape descriptors for the extraction of quantitative information and the assessment of shape variability within a class and the extension of the popular cage-based deformation techniques to include constraints on the allowed displacement of vertices.*

*In this thesis, we report the design and development of the framework as well as results in two application scenarios, namely product design and archaeological reconstruction.*

To my father. I wish you were here...

# Acknowledgements

This thesis is the product of several years in which I received a lot of help and assistance from a huge amount of people. I will just try to go in chronological order.

First of all, I really want to thank my family and friends, who always supported me and without whom I would not be here now.

I would like to thank Prof. Scateni, who was my first guide in the research environment together with the students of the Batcave (especially Alessandro).

I am deeply grateful to Livio, who always insisted for me to apply for a PhD position and supported me throughout all of it, explaining to me its mechanisms and difficulties. I will never forget it.

Then, I have to thank Chiara, first and most important of all my supporters, from every point of view. I appreciate it more than I can say.

After my first attempt for entering the PhD program in Cagliari, I accepted a Research fellowship in Genova, where I got to know a lot of nice people, who I really want to thank for their friendship and closeness, even in the darkest moments.

In particular, I would like to thank my invaluable supervisors. Dr. Michela Spagnuolo, for giving me the opportunity to embark on this path in the first place and for her continuous encouragement. Dr. Michela Mortara for her constant support during the four years of my stay and her conspicuous help in writing this thesis. Then, I want to thank Elia for helping me a lot, especially with my poor mathematical knowledge, and my other partners in crime for giving me the funniest years possible. I can not thank you singularly, but none the less I am very grateful to each of you.

I wish to thank all the people whose assistance was a milestone in the completion of this project, in particular Valentina, who was always available for help and discussion and to write several papers (even in the dead of the night); Prof. Botsch, who hosted me at the Bielefeld University, assisted me during our collaboration and accepted to be one of the reviewers of this thesis together with Prof. Hermon and Prof. Laga, whom I thank equally. I also want to thank the whole Computer Graphics group, formerly in Bielefeld and now in Dortmund, for the best possible hospitality and the friendly environment, as well as for their help on different occasions.

Thank you, thank you very much, cited or not, I know that without each of you this result would not have been possible.





# Abbreviations

**3D-PSM** Patient-Specific 3D Model. 26

**AI** Artificial Intelligence. 25, 32

**AR** Augmented Reality. 96, 97

**BBW** Bounded Bi-harmonic Weights. 77

**BC** Barycentric Coordinates. 75–77, 84, 133

**CAD** Computer-Aided Design. 11, 17, 24, 26, 89, 97

**CG** Computer Graphics. 6, 10, 66

**CH** Cultural Heritage. 6, 9, 12–14, 17, 20–22, 24, 25, 49, 112, 114

**CP** Control Point. 97–101, 103–105

**CVA** Corpus Vasorum Antiquorum. 13

**DoF** Degrees of Freedom. 74, 81, 82, 120

**DQS** Dual Quaternion Skinning. 71, 72

**FFD** Free-Form Deformation. 66–68

**GBC** Generalized Barycentric Coordinates. 72, 73, 76, 78, 82, 98, 99, 104, 120, 125, 129

**GC** Green Coordinates. 77, 84, 125

**GPU** Graphics Processing Unit. 77

**GUI** Graphical User Interface. 78, 88, 89, 107, 112, 125, 139

**HC** Harmonic Coordinates. 77

**HMD** Head-Mounted Display. 97, 98

**ICP** Iterative Closest Point. 12

**JSON** JavaScript Object Notation. 139

**LBC** Local Barycentric Coordinates. 78, 99

**LBS** Linear Blend Skinning. 69–73

**LMC** Leap Motion Controller. 97, 98

**LoD** Level of Detail. 27, 33, 44, 120

**MBR** Minimal Bounding Rectangle. 55, 56

**MRI** Magnetic Resonance Imaging. 13

**MVC** Mean Value Coordinates. 77, 78, 82, 99, 104, 125

**NLP** Natural Language Processing. 19, 23

**OBB** Oriented Bounding Box. 74, 88

**PMVC** Positive Mean Value Coordinates. 77

**PNRA** Programma Nazionale di Ricerche in Antartide. 123

**RoI** Regions of Interest. 24, 26, 44, 99, 100, 104

**VR** Virtual Reality. 20, 87, 96–98

# Table of Contents

<b>Chapter 1 Introduction</b>	<b>6</b>
1.1 Template: meaning and examples . . . . .	10
1.2 Overview of the framework . . . . .	18
1.3 Contributions . . . . .	20
<b>Chapter 2 Representation and preservation of Semantics</b>	<b>22</b>
2.1 Annotation systems in application domains . . . . .	24
2.1.1 Cultural Heritage . . . . .	24
2.1.2 Medicine . . . . .	26
2.1.3 Engineering . . . . .	26
2.2 Formalisation of annotations . . . . .	26
2.2.1 Information . . . . .	27
2.2.2 Selection . . . . .	27
2.2.3 Attributes . . . . .	28
2.3 Relationships among annotations . . . . .	30
2.4 Annotation persistence . . . . .	32
2.4.1 Annotation Transfer . . . . .	34
2.4.2 Transfer results . . . . .	40
2.4.3 Transfer limitations . . . . .	44
2.5 Discussion . . . . .	46

<b>Chapter 3</b>	<b>Semantics enrichment through shape analysis</b>	<b>47</b>
3.1	The Ayia Irini case study . . . . .	49
3.2	Quantitative attributes identifying artisan expertise and production process . . . . .	52
3.2.1	Experiments and results . . . . .	54
3.2.2	Inferring fixed proportions . . . . .	56
3.3	Quantitative approach to the sub-grouping of artefacts based on moulds . . . . .	57
3.3.1	Similarity assessment . . . . .	59
3.3.2	Head clustering . . . . .	59
3.3.3	Experiments . . . . .	60
3.4	Limitations and Discussion . . . . .	63
<b>Chapter 4</b>	<b>From semantics to geometry through template deformation</b>	<b>65</b>
4.1	Review of surface deformation techniques . . . . .	66
4.1.1	Variational techniques . . . . .	67
4.1.2	Free-form deformation . . . . .	67
4.1.3	Skinning . . . . .	68
4.2	Constrained deformation . . . . .	78
4.2.1	The ShapeOp library for geometric constraints . . . . .	79
4.2.2	ShapeOp extension for cage-based deformation . . . . .	81
4.2.3	Low-level and high-level constraints . . . . .	84
4.3	Discussion . . . . .	85
<b>Chapter 5</b>	<b>Results and applications</b>	<b>87</b>
5.1	Operational workflow of the system . . . . .	88
5.2	Product Design scenario . . . . .	89
5.2.1	Same “level” constraint . . . . .	90
5.2.2	Structural “continuity” constraint . . . . .	92
5.2.3	Deformation in a VR environment . . . . .	96

5.3	Archaeological reconstruction scenario . . . . .	107
5.3.1	Interactive virtual archaeological reconstruction . . . . .	107
5.3.2	A constraint based on “proportions” . . . . .	112
5.4	Discussion . . . . .	116
<b>Chapter 6 Discussion and future works</b>		<b>119</b>
6.1	Discussion . . . . .	119
6.1.1	Lack of a semantics-aware generation of cages . . . . .	120
6.1.2	Need for a “semantic” optimisation . . . . .	120
6.1.3	Work in progress . . . . .	122
6.2	Further Applications . . . . .	122
6.2.1	Investigation of biological species . . . . .	123
6.2.2	Generation of random shapes belonging to a same homogeneous class . .	123
6.2.3	Classification of shapes based on non-rigid fitting . . . . .	123
<b>Appendix A Graphical User Interface</b>		<b>125</b>
A.1	Main window . . . . .	125
A.2	Annotation window . . . . .	130
A.3	Relationships window . . . . .	137
<b>Appendix B File formats</b>		<b>139</b>
B.1	Annotation file format . . . . .	139
B.2	Graph file format . . . . .	140
<b>Bibliography</b>		<b>142</b>

# Chapter 1

## Introduction

In the last decades, there was an endeavour in research for the definition of 3D acquisition technologies (e.g., [LPC<sup>+</sup>00] ), bringing to life new and astounding techniques, such as computer tomography, magnetic resonance imaging and 3D laser scanning. These techniques have enabled highly accurate digitisation of complex 3D objects [BKP<sup>+</sup>10].

The number of areas that are profoundly changing due to the availability and usage of digital content is increasing day by day, passing from entertainment (including video-games, cinema and cartoons) to education (serious games), from industry (fabrication, predictive maintenance) to scientific research (e.g., neuroscience, mechanical engineering, astrophysics), and also in fields like Cultural Heritage (CH), where research can be conducted on the digital replica of the physical and often fragile assets.

Historically, Computer Graphics (CG) focused primarily on setting the foundations for the representation of the geometry of 3D shapes rather than their semantics, but such an approach is reaching its limitations and is not expected to satisfy the requirements of the hectic and customised usage required, for example, by social media sharing of 3D shapes [Spa16]. The reasons are several:

1. 3D content really falls in the context of big data: generally speaking, the footprint of 3D models is way heavier compared to text or images and even video clips, making the real-time streaming of such data really difficult.
2. During the last years, the attention of industry was mainly focused on the visualisation of 3D media, while the creation and manipulation of 3D models were not identified as critical, mainly due to the complexity of such actions for non-expert users.
3. Almost all the efforts spent until now focused on defining a good geometric representation for objects, without considering the meaning of the objects, their function or intended

usage and their role in a complex scenario, or at least they have only been considered as an add-on.

While the first issue is likely to be overcome by the introduction of new transmission and computational technologies (e.g., 5G wireless communications technologies and quantum computing), the solution of the remaining two really requires a paradigm shift. For this reason, in the latest years more and more research has focused on the introduction of an intelligence layer in the definition of shapes (e.g., [AIM04], [FOC08]): in this view, 3D representations embed semantic data at the geometric level, communicating information about the geometry via a formalised description of an object’s meaning.

Typical approaches in this direction study the presence of symmetries, repeating patterns or extract a *skeleton* of the object for trying to sort out low-level geometric data into a more structured and semantics-oriented representation of 3D models [Spa16].

In this thesis, we argue that, while nowadays software for the manipulation of images are reaching a good level of simplicity and effectiveness even when utilised by users with little expertise, while allowing to obtain extremely high-quality results (e.g., see Figure 1.1), the same does not hold for 3D shapes, yet there are situations in which the search, editing and reuse of 3D shapes would really be useful. The reasons are several:

1. It is relatively easy to define a part of an image (e.g., with the lasso selection) while the same selection on a 3D object is much more difficult. In 3D, indeed, there is not a unique single view of the whole object and the selection of the part of interest is more complex from the interaction point of view, while we are trying to interact with the shape acting on a 2D representation of it (typically mouse-based interaction with visualisation on monitor);
2. After managing to select one part, there are many ways for defining the deformation to be applied to it and this deformation may or may not change the overall shape of the object in several different ways. With reference to Figure 1.2, if we wanted to enlarge the flowers, scaling the whole object would not change the relationship between flowers and teapot; moreover, the teapot has several flowers: enlarging one does not necessarily mean enlarging all the others, even if intuitively they are part of the same decoration and should remain similar after one has been modified.

In this thesis, we aim at defining an approach and the methods to support the editing of 3D shapes in a “smart” way, that is, controlling how the modifications should impact the nature, functionality or purpose of an object, or on the peculiar features that discriminate that object from others of the same type, that is, features that characterise its style. We will name all these aspects as pertaining to the semantics of an object, and the editing system will therefore be named a semantics-aware modelling of objects.

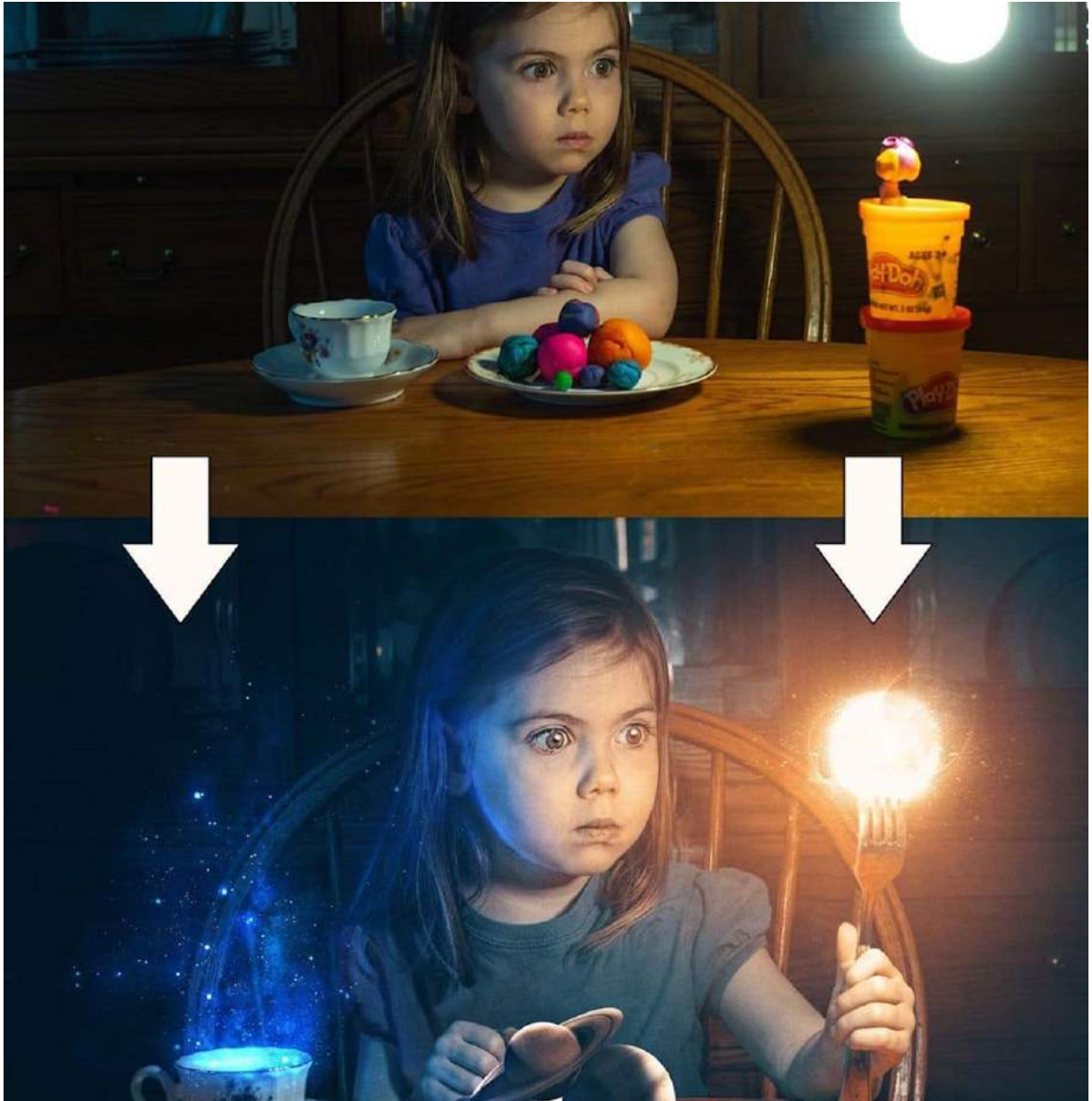


Figure 1.1: An example of the modifications obtainable using image manipulation software, such as Photoshop® (image by Kevin Carden on <https://www.christianphotoshops.com/>).





Figure 1.2: *An example of a teapot shape with some floral pattern on its surface.*

Core part of the proposed framework is the definition of an abstract representation that couples geometry and semantics of a class of shapes and allows to express geometric, structural and stylistic constraints on the object, its parts and the relations among parts. Thanks to this abstract representation, semantics-aware (and compliant) deformations can be expressed, at the degree to which the user will drive them, allowing or precluding modifications that satisfy or breaks certain constraints. We will call this abstract representation the “parametric template” of an object: “template” because it captures the main traits of the object within its class and “parametric” to convey the idea of the range within which the semantic traits are considered constraints during object editing sessions.

The research challenge addressed in this thesis was studied first in a specific context, and in particular to aid the reconstruction of broken CH artefacts, where domain experts possess strong knowledge on the stylistic properties of objects, the intra-relationships within their parts and the inter-relationships between different objects. Thanks to the component defined and developed during the research work of this thesis, CH domain experts are provided with an interactive system for merging such knowledge and their shape representation through the mechanism of 3D annotation. We believe that the constant synergy of geometry and semantics is crucial to define simpler yet smarter modelling frameworks, where even non-expert users can define and model an instance of a shape by exploiting the knowledge of domain experts in a transparent way.

## 1.1 Template: meaning and examples

As already introduced, the goal of this thesis is to set up a *framework* for allowing a semantically guided modelling of shapes belonging to the same class of *homogeneous objects* (i.e., sharing specific properties). We have in mind several applications: the manipulation of an object forbidding modifications that would change its nature or the characteristics that identify it as a part of its class; the adaptation of an object from one style to another by modifying properly the characteristics that correspond to a particular canon; the recognition/classification of an object by assessing if it exhibits the features of a certain class; the shape completion/reconstruction of a complete model through the matching of one or more compatible fragments to an archetype. All these applications assume more or less implicitly that the features identifying a class of shapes and discriminating it from others are known and encoded in a machine-readable form, either at the semantic level (the knowledge of what “makes this object a teapot”) or at the geometric one (the geometry of a reference teapot), or both.

The term *template* has meanings explicitly related to the concepts of “similarity” and “structure” and to the act of “production”. According to the Cambridge dictionary, for instance, a template is “something that is used as a pattern for producing other similar things”, “a particular model for arranging information”, “a design or pattern that is used for making copies of a shape”. The template is, therefore, a reference for other similar objects, but how much similar, and similar under which point of view? Implicitly, the template is an abstract representative for a class of objects, which share some homogeneous features, but not necessarily they are perfectly identical. A template document specifies what kind of information needs to be inserted and in which order, but the information itself will be specific to each document written following the template. We could say, the template represents a class and the documents are instances of that class.

Bringing forward this reasoning, we could claim that conversely, if an object does not follow the template, it does not belong to the class. Therefore, the template discriminates what objects belong or not to a certain category; in other words, the template represents the knowledge priors about a class of objects.

Indeed, templates have been and are still widely used in CG applications as shape priors to optimise complex problems that typically require human intervention (e.g., segmentation of medical images) or to solve under-constrained problems such as shape reconstruction from incomplete data. We briefly sketch the use of templates in several domains in the following subsections.

Last, but not least, geometric deformability arises as one of the main features of a shape template in CG applications: indeed, the template represents an ideal, reference object of a given class, and classification, recognition or completion problems must cope with the shape variability of objects within a class. Therefore, the template is typically deformed to fit some input data, and a mechanism must be in place to assess the range of this deformation, e.g., if it is acceptable (the input object is similar enough to the template and belongs to the class) or not. This is even more

true for production purposes: generating new shapes based on a template, e.g., in product customisation, content adaptation or re-purposing, requires to change a reference geometry adapting to a new setting or new constraints.

## **Templates in Computer-Aided Design**

Being able to modify interactively the geometric models associated with a product is a necessary condition to ensure the efficiency of a design process. Indeed, the geometric model of a future product is subject to several modification steps, at different stages of the design process, where different professionals are also involved. For example, the shape defined by a stylist may not be compatible with the mechanical characteristics the engineers aim for.

Therefore, this domain requires modelling systems allowing the creation of new shapes and their modification through the use of high-level operations such as copy/paste, move, size changes, remove and so on. These modifications have to be intuitive since different stakeholders are involved in the design process.

To meet these requirements, the modelling process should be shape-oriented, in the sense that the designer should think and build his/her geometric model through the use of high-level entities and not directly through the manipulation of simple primitives (faces, edges). This approach is known as feature-based modelling. The feature concept has been successfully adopted for the design of shapes defined by analytic surfaces such as planes, cylinders, spheres and so on [SM95].

Following the feature-based approach [PFGL08], the geometric model is not anymore perceived as a collection of vertices, edges and faces but as a well-organised set of features corresponding to slots, ribs, stiffeners and so on.

All the properties related to a feature type are specified within a feature class that defines a template for all its instances. This always includes the generic shape of the feature, and a few parameters, e.g. length, width, and constraints, e.g. parallel, perpendicular, that characterise this shape.

By specifying values for the parameters, an instance of the feature class can be created and then be added to a feature model [BBN06]. The feature model usually contains the feature instances, the information related to their mutual dependencies and the chronology of construction.

Still in the context of product design, but unrelated to Computer-Aided Design (CAD) systems and feature-based modelling, a recent method [ZAC<sup>+</sup>17] has been published to tackle the re-targeting of mechanical parts to adapt to new outer shapes. The system is based on the observation that, for many functional objects, the mechanical architecture remains the same while the shape varies. The approach aims at allowing novice users to re-target an existing mechanical template to a user-specified input shape. In this work, a mechanical template encodes a

parameterised mechanism, mechanical constraints that ensure a physically valid configuration, spatial relationships of mechanical parts to the user-provided shape, and functional constraints that specify an intended functionality. The algorithm interactively optimises the mechanical template while the user manipulates the placement of mechanical components and the shape.

While the previous approach fits a mechanical template to an input shape to preserve the desired functionality, another relevant work published earlier [LMS13] focused on the automatic recognition of functional parts of man-made 3D shapes in the presence of significant geometric and topological variations. The authors observed that under such challenging circumstances, the context of a part within a 3D shape provides important cues for learning the semantics of shapes. They model the context as structural relationships between shape parts and use them, in addition to the part geometry, as cues for functionality recognition. They represent a 3D shape as a graph interconnecting parts that share some spatial relationships, actually defining a template for a class of man-made objects.

## Templates in Shape Reconstruction

The construction of a 3D model from acquired data plays an important role in Computer Graphics. The challenges in reconstruction are related to noisy data and outliers, to missing data due to occlusions or simply to the sparsity of data, which significantly aggravates the problem of reconstruction. The setting of the problem can vary from the digitisation of a single static object to the acquisition of indoor and outdoor scenes or geographic areas, to the real-time reconstruction of deformable, moving subjects. In such cases, the input to the reconstruction is a point cloud acquired by laser scanning, photogrammetry or depth sensors, but 3D reconstruction can be achieved from videos (e.g., [YRCA15]) or from a few or even a single image (e.g., [BGC<sup>+</sup>15]). The solutions to such hard problems typically exploit contextual information to incorporate prior knowledge about the shape to be reconstructed.

The template shape is computed through a statistical analysis of the class of shapes and is then deformed to fit the acquired data. In the CH domain, the input data could represent partial, damaged and fragmented artefacts to be virtually completed using a reference [GSP<sup>+</sup>14]. The template is particularly effective when dealing with the creation of avatars based on scans of the real person: indeed, the acquisition process is subject to errors that might arise for differences in light, movements of the subject, poor calibration, pose complexity and occlusions. This noisy and incomplete data can be cleaned and completed by *fitting* a template shape to the input points [SKR<sup>+</sup>06], through one of the many generalisations of the Iterative Closest Point (ICP)[Zha94] algorithm (see [ARV07, BR07]) to non-rigid registration (e.g., [YWLM11, AWLB17, WAB<sup>+</sup>20]).

## **Templates in the medical field**

Template shapes are very useful when dealing with several shapes which are similar to one another. This is the case of the modelling of anatomical parts of the human body, such as bones and organs, which exhibit a known shape with a certain variability among patients. This shape variability is limited but considerable; on the other hand, identifying significant shape deviation is of particular importance in this domain, as it might indicate pathological situations.

This fact has increased the interest in the creation (or synthesis) of mean shape models to cope with the variability of anatomical structures in medical data analysis.

Banerjee et al. [BLP<sup>+</sup>15] explored the suitability of statistical shape analysis to produce 3D canonical models of bones [RST<sup>+</sup>07], built from homogeneous classes of 3D bones reconstructed from Magnetic Resonance Imaging (MRI) data. Grounded on previous results in landmark-guided deformation of elastic shapes [KSKL13], they generate a 3D model that captures the variability exhibited by the members of the class, while preserving important anatomical landmarks, which characterise both the function and status of the anatomical part. They argue that 3D canonical models could be used to support the diagnosis of musculoskeletal diseases, acting as reference 3D atlases (healthy average shape) on which important morphometric parameters can be evaluated and quantified.

Templates (e.g., [CJV06]) could be exploited, for example, for the reconstruction of broken bones, like skulls [YWLM11].

Finally, image segmentation of anatomical structures under challenging conditions such as tissue inhomogeneity, noise, and contrast loss can be significantly facilitated by incorporating prior knowledge, constraining the solution to remain close to a predefined template shape [PCM<sup>+</sup>14].

## **Templates in Cultural Heritage**

Particularly in the CH sector, templates are suitable representations of a-priori knowledge of shape variability within classes of objects. Indeed, qualitative and quantitative analyses of artefacts have been documented over the centuries by archaeologists and curators with great detail, cataloguing pieces according to shared types and styles and reporting shape characteristics and variations within categories. Specific objects have been particularly studied, thanks to the number of findings, and some reflect the presence of established proportions.

For instance, the Corpus Vasorum Antiquorum (CVA) is an international research project for the documentation of ceramics from the classical era. The CVA mostly publishes Greek (including Italian) pottery between the seventh millennium B.C. and late Antiquity (third-fifth century A.D.). The publications are divided into fascicles catalogued by country and museum. Today the project covers a compendium of more than 100,000 vases located in collections of 26 participating countries. The documentation of a vessel includes the description of its overall condition,

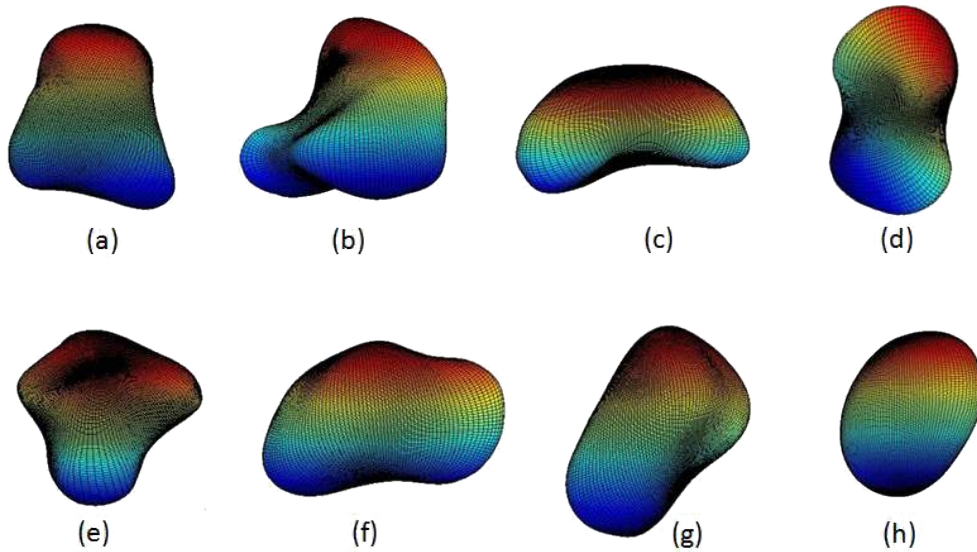


Figure 1.3: *Mean shapes for the bones of the right wrist: (a) capitate, (b) hamate, (c) lunate, (d) scaphoid, (e) trapezium, (f) trapezoid, (g) triquetrum, (h) pisiform (images from [BLP<sup>+</sup>15]).*

followed by an iconographic interpretation. Hypotheses about an artist or a workshop are also determined. Integral parts of the documentation are photographs and hand-drawings depending on the condition of the vessel and a chronological classification. Figure 1.4 shows a few examples from the Cracow fascicule 1 [PW12], set of the “Athenian black figure” pottery. On the left, the image for the Olpe with inventory number 189 (described as an “Oinochoe of SHAPE 5”). On the right, three exemplars of “Lekythos”. The structural and geometric similarity, as well as the presence of proportions within objects in the same class, is evident and the amount of information available makes it possible to derive reference measures, variances and proportions from the metric analysis of the inventory items, facilitating the “data-driven” creation of a template.

Another example refers to the representation of the human figure across cultures and styles, following proportions or *canons* [Wik20a] defining ideal body shapes and poses. This concept has been thoroughly exploited in several civilisations, e.g., in the Egyptian [Rob94], Minoan [Wei00], Greek [Tob75] and Chinese art [WH19] (see Figure 1.5).

As well as in the other fields, templates can be deployed in CH as a powerful tool for shape completion or re-assembly of damaged and fragmented findings. Indeed, archaeologists and curators often deal with incomplete and damaged objects for which a template can represent a reference, complete shape exemplifying how the original object would look [GSP<sup>+</sup>14, DZY<sup>+</sup>16, PSA<sup>+</sup>17, LBB19]. Typically, an artist produces drawings about reconstruction hypotheses using similar known artefacts of the same period and style as references, and 3D digital reconstruction or completion methods often follow a similar approach [FCD20].

Finally, the template can be used for the classification of artefacts, for instance to identify spe-



Figure 1.4: Extracts from the “Athenian black figure” pottery in CVA. From left to right: Olpe n.189, Lekythos n.345, Lekythos n.194 and Lekythos n.204. Note the inter-class differences between the olpe (a specific type of Oinochoe) and the lekythoi, as well as the intra-class stability of proportions among the lekythoi. Qualitative (e.g., form of the handle section) and quantitative descriptions are also documented.

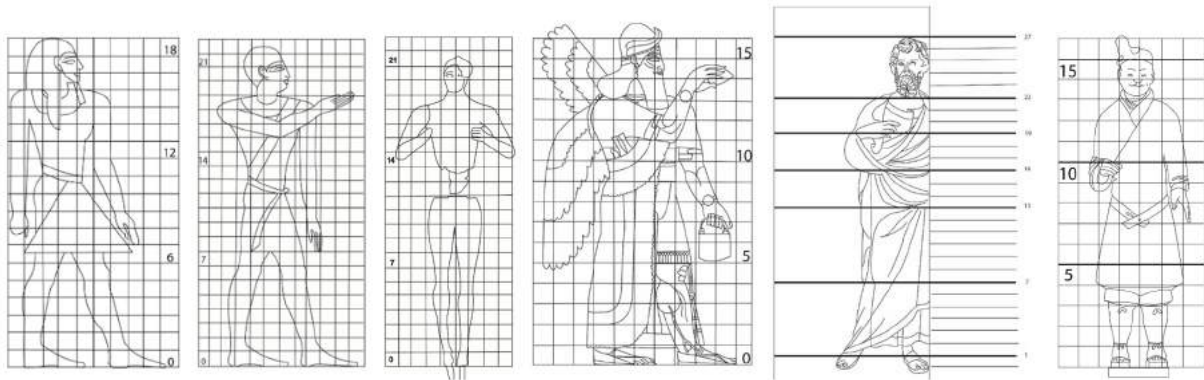


Figure 1.5: From left to right: the first Egyptian canon of proportions: the 18-square division [Rob94]; the second Egyptian canon of proportion: the 21-grid division [Ive68]; the 21-grid division for the Palaikastro Kouros statue, as proposed in [Wei00]; the 15-grid division in Assyrian human figures [Rob90]; the Cennini’s canon division in large-scale Italian Renaissance sculptures [Mor78]; terracotta warrior T10G7 with grid proposed in [WH19]. All images are taken from [WH19].



Figure 1.6: *On the left, some trees built using L-systems, on the right some buildings created using a procedural grammar (figures respectively from [Wik21] and [MWH<sup>+</sup>06]).*

cific groups of pieces such as according to a homogeneous production technique or provenance [GG20].

### **Templates in Procedural Modelling**

Procedural modelling has its roots in computer graphics techniques, such as Shape Grammars by Stiny and Gips [SG71], who were interested in developing a computational basis for design, and Lindenmayer systems or L-systems [Lin68], mathematical models for creating organic self-similar forms.

Both L-systems and procedural grammars are structural/constructive descriptions of shape classes, which offer an efficient way to generate multiple differentiated objects with a minimal number of rules (e.g., see Figure 1.6). Most current work on procedural modelling occurs within the field of computer graphics [SKU15], with applications in the urban planning, gaming, and entertainment industries (e.g., [PM01]). In recent years, archaeology and cultural heritage projects, such as the significant test cases built around ancient Rome and Pompeii, have also begun to explore the use of procedural modelling for the reconstruction of ancient sites [HMVG09].

### **Common properties of templates in the different contexts**

From the above examples in different applications and domains we can synthesise the following observations concerning templates:

1. the template represents the contextual shape priors needed to solve complex problems (e.g., in shape reconstruction);



2. the template is either an “average” shape in a class of objects of the same kind (e.g. statistical shapes of bones in medicine) or an archetype of the class;
3. the template shape is deformable to fit the input data (e.g., in shape reconstruction);
4. there must be a way to measure the distance between an object of the class and the template (e.g., for medical diagnosis);
5. the template can express how the object is structured and the construction process to achieve it (e.g., in procedural modelling);
6. the template expresses dimensions, proportions and other quantitative properties of a class of shapes (e.g., in CH);
7. the template can be used to build new objects of the same type (e.g., in CAD).

Some of the above properties address more the semantics of an object (e.g., the way it is constructed) while others refer to its spatial extent (e.g., the requirement to be deformable). This fact highlights how templates have a double nature, being both a semantic and a geometric reference for a class.

Having to define a template for a set of 3D models, an important observation regards the cardinality, the variability and the available documentation for the shapes at hand. For instance, in the medical domain, the amount of patient-specific data coming from diagnostic devices allows to build statistical shapes as templates, which already encode in themselves the variance of class elements.

In the CH sector, the amount of documentation for certain objects is so huge that deriving a template is easily affordable without many examples (e.g., amphorae are very well studied, classified into agreed taxonomies and their shape properties and variability are documented). This aspect highlights how geometric information and documentation can help each other in defining an expressive template. In the case of many objects available, the template construction can follow a data-driven approach e.g., learning the geometric features characterising a group and the range of the corresponding parameters. With few models available, still some semantics can be inferred through shape analysis techniques, but certainly external domain knowledge will be needed to define a proper template.

Variability in the class deals with the range of parameters or statistical variation in the template, and we observe that for a new query shape the evaluation of the same parameters can be used for classification purposes. In other words, the template can also express a set of constraints for class membership.

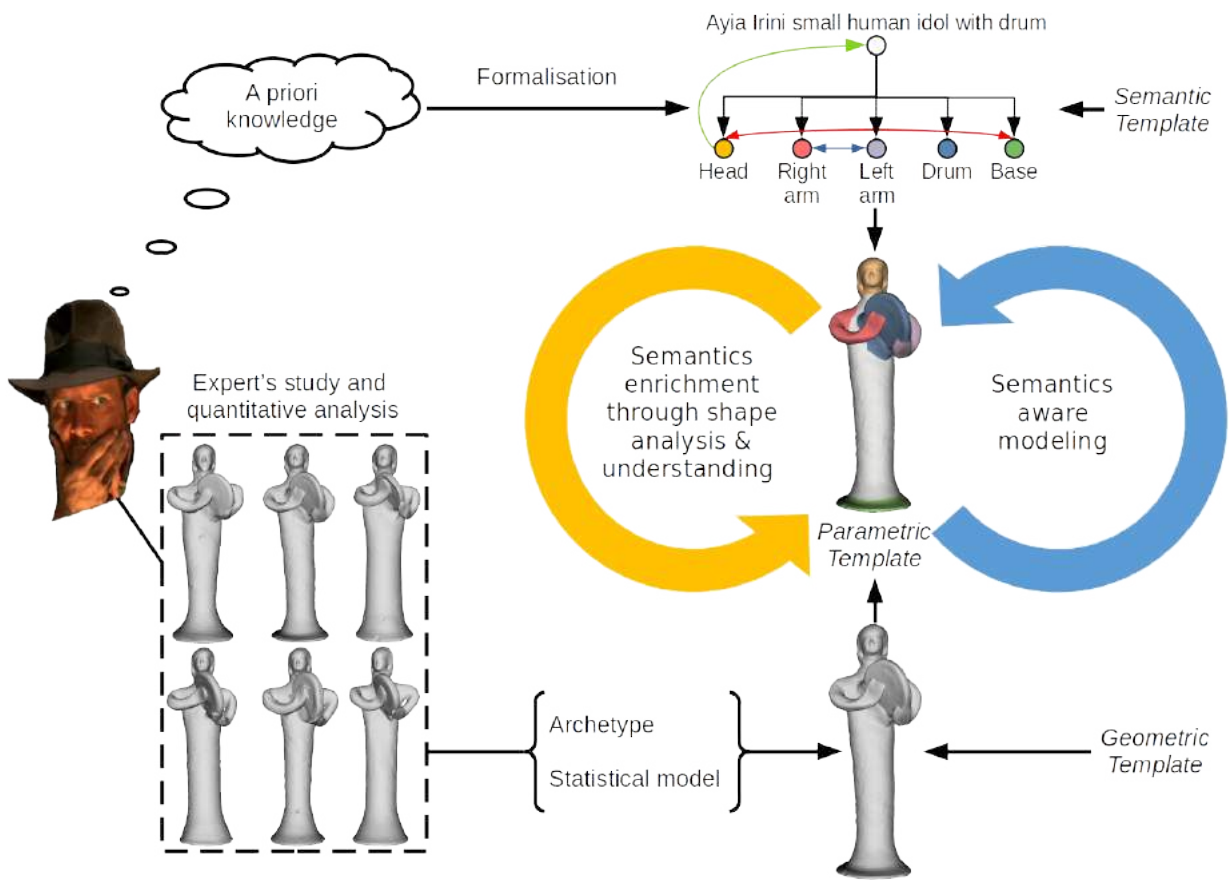


Figure 1.7: *The overview of the proposed framework.*

## 1.2 Overview of the framework

In this thesis, we focus on the formalisation of a *parametric template*, including both the geometric and semantic nature, where one helps and enriches the other, and provide a modelling framework where the parametric template supports several shape and semantics' related processes. With reference to Figure 1.7, the proposed framework has two parallel levels or views that involve the definition of a *representative shape* for the *homogeneous class* of objects and the encoding of the *semantics* of the class, that is the set of parts, properties and relationships that unite objects belonging to the same class.

The combination of these two sides generates what we call a *parametric template*, that is, at least ideally, a conceptual representation of an object, whose geometry can be instantiated at will according to application needs, e.g., in terms of re-meshing or deformation, always (and transparently) in compliance with the attributes and constraints that are proper of the object itself.

From the homogeneous class, it is possible to select one of the shapes inside the class as an archetype (a 3D shape chosen as representative), or to use existing techniques for exploiting the composition of the different shapes and creating what is called a *statistical model* to produce a *geometric template*.

Statistical models are mainly composed of two parts: the mean model, which is the “average” shape or appearance of the objects within the collections, and statistical variance with respect to the mean model [HM09]. In theory, a good statistical model should represent most of the variability that exists within the sample population using less number of variance. This kind of homogeneous class descriptions are largely used, e.g., in the biomedical field [CJV06, MVL<sup>+</sup>11, SWZ14, BLP<sup>+</sup>15]. Typical approaches for the generation of a statistical model are based on landmark-guided surface registration and standard linear statistics to compute the mean shape and the modes of variation (e.g., [KSKL13]).

In parallel, we assume that *domain experts*, with very strong knowledge related to the homogeneous class, have formalised their knowledge into the *semantic template*: a set of *properties, constraints and rules* governing the definition of the class itself. With reference to Figure 1.7, requirements for a statuette to be classified as “Ayia Irini small human idol with drum” could be that the arms have the same length, that the head is coaxial to the base and that its head’s height is  $\frac{1}{3}$  of the total height (this is just an example, not necessarily true).

Sometimes, however, this knowledge is available but not ready to use in a computational fashion. For instance, in the archaeological field it is very common to textually describe relics belonging to a particular *style* by considering proportions, repetitive patterns, materials, etc. However this approach has some issues:

1. *Natural language* is prone to intrinsic ambiguities linked to the employed terms: indeed, often a single word have different meanings and several different words can be used for referring to the same concept;
2. Textual descriptions are not machine ingestible: even if branches of research are focusing on Natural Language Processing (NLP), the management of free-form text is still an open issue;
3. Quantitative measurements are not repeatable, if the measuring process has not been properly documented (e.g., direction and reference points of a measure).

In this thesis, we start from the assumption that a representative shape exists and is given (e.g., selected from a database of shapes) as well as a formalisation of the domain expert’s knowledge (e.g., an ontology) and try to connect these two views into a single concept: the parametric template.

Of course, the parametric template is a dynamic entity: indeed we can *enrich* the semantics by means of shape analysis (see Chapter 3) or simply by allowing further documentation by

final users, in terms of free textual annotation, addition of new measurements, definition of new relationships or constraints. At the same time, the semantics (in the form of constraints set over relationships and attributes) can be exploited for generating a new geometry through *constrained deformation* (see Chapter 4). This also calls for the definition of techniques for guaranteeing the persistence of the bridge between semantics and geometry (namely, annotation persistence - see Section 2.4) when severe transformations are applied to the latter. As can be seen, the only geometry transformations which present difficulties in our settings are change of resolution and change of representation (e.g., passing from a triangular to a tetrahedral mesh). In this thesis, we will focus on triangular meshes, thus tackling only the change of resolution issue, by means of *annotation transfer*.

### 1.3 Contributions

In this work, we designed and developed a framework for the semantics-aware modelling of shapes, offering the user a single application environment where the whole workflow of defining the parametric template and applying semantics-aware deformations can take place. To this aim, we combined state of the art techniques and new algorithms to enable the user conceptualising her/his knowledge and modelling geometric shapes. We produced an implementation posted on GitHub [Sca20], providing the functionalities of annotation, measuring, documentation, and definition of constraints together with cage-based deformation, interactive reassembly and shape analysis tools (see Appendix A).

The original contributions of this thesis are the following:

1. A formalisation of the parametric template, based on the mechanism of annotation as the link between geometry and semantics of objects;
2. A method for the automatic transfer of annotations between meshes with different resolutions representing the same shape;
3. New shape analysis methods specific for the semantic enrichment of a set of clay statuettes of the “Aya Irini” collection (see Chapter 3);
4. A system for the constrained deformation based on the parametric template extending the state of the art library ShapeOp [DDB<sup>+</sup>14] for working with cages;
5. A template-based method for virtual reconstruction of archaeological findings (see Section 5.3).

The potential of this framework is shown in two applications: product customisation, also deployed in a Virtual Reality (VR) environment for collaborative design, and fragment fitting in the CH domain.

The remainder of the thesis is organised as follows:

- in Chapter 2, we present techniques for representing and preserving the semantics of the template and formalise the entities involved in the bridge between geometry and semantics;
- in Chapter 3 we discuss how geometric analysis can enrich semantics, through two exemplary applications in the CH domain: how to extract metric properties and hints about production techniques and provenance for a collection of clay statuettes, and to more finely sub-group objects in a certain class;
- Chapter 4 focuses on the definition and implementation of a constrained deformation system, where geometry is updated according to semantic rules;
- in Chapter 5, we briefly describe the functionalities of the implemented system and then present the results obtained applying it in concrete application scenarios;
- Finally, Chapter 6 presents a discussion on the achieved results and describes various directions for future research that this thesis opened up.

## Chapter 2

# Representation and preservation of Semantics

---

**Summary.** In this Chapter, we describe the concept of annotation over 3D models and give an overview of different annotation systems, focusing on those targeting the CH, medicine and engineering domains. We define a novel formalisation of annotations and organise annotated parts into a hierarchical structure induced by the annotation containment relationship, possibly enriched inserting further relationships (e.g., adjacency) among annotated parts, thus creating the *relationships graph* (Section 2.2). Finally, we introduce the problem of annotation persistence and present an original method for guaranteeing the persistence of annotations after change of resolution (Section 2.4).

---

As introduced in the previous Chapter, while techniques and frameworks dealing with the shape or geometry of objects have been extensively studied and a huge amount of pure geometric applications exist today, the semantic description of objects has often been considered as an add-on and rarely studied or included in the loop.

A semantic description of 3D objects may be understood as a description of the content employing terms which are meaningful in some domain of knowledge [CMSF11]. Consider the following example:

One of your friends is going to a furniture shop and you ask him to buy a precise object. It is a cookie holder built in the shape of the Death Star (see Figure 2.1), but your friend has not seen any Star Wars movie. So you try to describe it: it is a grey, sphere-like “starship” with several engravings and a big notch in the upper part containing other circular concentric engravings. Since the Death Star is spherical and you want to lay the cookie holder on a shelf in your living room, you require it to have a planar base. Is such a request likely to be understood in a virtual scenario, in which our avatar steps into a virtual furniture shop held by a virtual salesman? Probably not, because with the current technologies the specification of these descriptions and



Figure 2.1: *The Death Star-like cookie holder (image from <https://comicbook.com/news/death-star-cookie-jar-revealed/>).*

their use in human-computer interaction with 3D digital content is almost impossible.

The first steps for trying to overcome this problem came from the shape retrieval community: since the introduction of the first online repositories [3DC01, STA03, AIM04, SMKF04], the increase of the number of shapes called for more intelligent searching methods. For achieving this goal, different paths have been followed, starting from a search by geometric similarity (e.g., with queries defined by sketching or by example) [SMKF04] and then basing on a formal organisation of 3D models enriched with metadata to search for content also in terms of *knowledge about* the 3D shapes [AIM04].

The latter approach really went in the direction of trying to encode the knowledge directly along the shape, so that in the future it would be possible to understand what the people are saying when describing an object, e.g., using NLP techniques [FRC13], and then searching for objects that meet the requirements, i.e., which possess those characteristics (previously described and encoded) that the user is searching for.

While this paradigm is really effective in the search for objects meeting some requirements, a big improvement can be introduced by using structural decomposition of objects, thus giving the possibility to reason not only on the object as a whole but also on the arrangement of its subparts (e.g., “find a shape containing two arms and two legs”, “find a virtual character which is strong/big/tall”) [RAS07]. Structural decomposition allows, on the one side, to define new and smarter shape retrieval tools; on the other hand it sets the foundations for *part-based annotations*,

that is broadly used in several fields, from CH to CAD.

Following the concept of part-based annotations, expressing the semantics of a shape requires [RASFO7]:

- the identification of significant parts (features);
- the specification of the semantics using some kind of formalism;
- the storage of the geometry plus its semantic description in a way that could be accessed easily, by humans as well as by software agents.

Generally speaking, the purpose of annotations is the localisation of object properties on specific parts, called Regions of Interest (RoI), of it. In principle, there is no constraint on the kind of information that can be associated to a specific RoI and, indeed, it can go from text (tagging) to any kind of multimedia (e.g., images, audio and video clips) and even to other 3D shapes.

## **2.1 Annotation systems in application domains**

In the last years, several different frameworks for the definition of annotations have been introduced, in particular in the CH, medical and engineering contexts.

### **2.1.1 Cultural Heritage**

One of the critical issues in CH is the management and processing of information. Indeed, working on archaeological sites or CH artefacts usually generates huge amounts of heterogeneous data given in the form of text documents, images, data tables, video clips, etc. The exploitation of such data requires a smart solution for their collection, organisation and access by experts and specialists, but even totally novice users.

For reliability and ease of sharing, the consistency of objects' documentation is crucial and, so, the use of recognised standards, common record structure and homogeneous terminology is needed.

The coupling of experts' qualitative and quantitative analysis with digital tools really boosts the possibility for documentation to be shared, accessed and exploited while giving strong means for the repeatability of experiments and data acquisition. Just to give a brief example, suppose to have to measure a statue arm length. Where should we start measuring it? Would we use a ruler or a tape measure? The entire set of possible choices in measuring make a certain level of ambiguity to be implicit in the presentation of the measure itself.



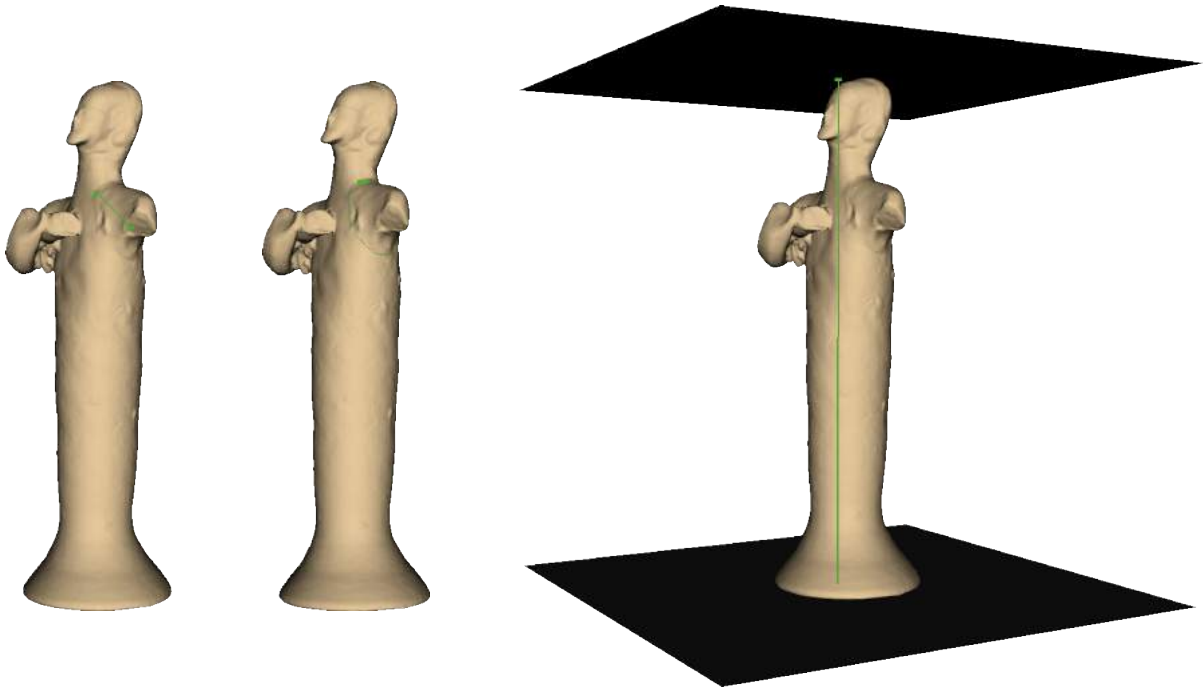


Figure 2.2: *Examples of measures taken with the ruler (Euclidean - on the left), tape (Geodesic approximation - in the middle) and bounding (still Euclidean - on the right) tools (see Section A)*

At the same time, taking measures from a physical relic introduces some risk of damaging it: indeed CH objects are often very sensitive to any kind of solicitations, from touch to illumination change.

For these reasons, in the last years CH researchers and experts are moving from a “physical” annotation of reflections and measurements to the digital equivalent and a huge amount of efforts has been spent in the definition of interactive systems supporting the annotation work.

Some examples are 3DSA [YGH13], a system for crowd-sourcing textual annotations (tagging) of objects for inferring archaeological classification of findings, CHiSEL [STLL13], an information system based on the use of the 3D representation of an object as a sort of “blackboard” where different information is represented, Aïoli [DLPDM<sup>+</sup>18], a collaborative annotation framework based on photogrammetry and high-performance cloud computing, and CHER-Ob [SKA<sup>+</sup>16], a framework providing different tools for evaluation and publication of the results of cultural heritage research and support for visualisation of different data formats. A further step forward is given by the always increasing use of Artificial Intelligence (AI)-based segmentation methods (e.g., [PGDF<sup>+</sup>20]).

### 2.1.2 Medicine

Ontology-driven annotation is not a completely new concept in medicine. Some medical software (e.g. ePAD [RAAA19], RadSem [MRS09]) already adopted this approach where the users are allowed to mark the RoI inside the 2D images (manually or automatically), and annotate it with the anatomical and/or pathological terms deriving from the ontology. Although such tools were initially limited to 2D images, they introduced the idea of the geometry-knowledge coupling in the medical field, giving means for their extension to Patient-Specific 3D Model (3D-PSM).

In the digital era, the concept of 3D-PSM is becoming increasingly relevant in computer-assisted diagnosis, surgery training on digital models, or implant design, even thanks to the availability of a wide range of advanced techniques for creating accurate and detailed 3D anatomical reconstruction (Magnetic Resonance Imaging, Computed Tomography, etc.). For this reason, in the last years there has been a surge in the interest for 3D-PSMs and, in particular, to really integrate the heterogeneous and huge quantity of data about each patient in an accessible way. This has been done by exploiting the concept of annotation by associating tags coming from an ontology [Mit08, BAC<sup>+</sup>16] or, in any case, from a controlled vocabulary [BCK<sup>+</sup>11, KC15] to portions of geometry, but also with pointers to web-resources (e.g., wiki, books) [QSO<sup>+</sup>12].

### 2.1.3 Engineering

In the engineering domain, the concept of feature is well known to identify parts associated to context-specific semantic information and modelling behaviour [SM95, DFG<sup>+</sup>94]. On the one hand, features are commonly adopted by commercial Mechanical CAD systems, allowing an easy modification of parts through meaningful parameters, which drive the shape changes according to engineering objectives. On the other hand, feature recognition systems allow the annotation of CAD models in terms of features useful for production purposes, allowing the integration with manufacturing or assembling tools and operations [JPR00].

A further step forward was taken with the introduction of the functional analysis and annotation of parts and sub-parts [WTD14], enabling both the search for shapes and their segments with certain functional properties and an effective reuse of previously defined designs.

## 2.2 Formalisation of annotations

In this thesis, a 3D shape is represented as a closed triangular mesh  $\mathcal{M} = \{\mathcal{V}_{\mathcal{M}}, \mathcal{E}_{\mathcal{M}}, \mathcal{T}_{\mathcal{M}}\}$ , where  $\mathcal{V}_{\mathcal{M}} = \{\mathbf{v} = (x_{\mathbf{v}}, y_{\mathbf{v}}, z_{\mathbf{v}}) \mid x_{\mathbf{v}}, y_{\mathbf{v}}, z_{\mathbf{v}} \in \mathbb{R}\}$  is the set of vertices of the mesh,  $\mathcal{E}_{\mathcal{M}} = \{e = (\mathbf{v}_s, \mathbf{v}_e) \mid \mathbf{v}_s, \mathbf{v}_e \in \mathcal{V}_{\mathcal{M}}\}$  is the set of its edges and  $\mathcal{T}_{\mathcal{M}} = \{t = (e_1, e_2, e_3) \mid e_1 \in \mathcal{E}_{\mathcal{M}} = (\mathbf{v}_1, \mathbf{v}_2), e_2 \in \mathcal{E}_{\mathcal{M}} = (\mathbf{v}_2, \mathbf{v}_3), e_3 \in \mathcal{E}_{\mathcal{M}} = (\mathbf{v}_3, \mathbf{v}_1), \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathcal{V}_{\mathcal{M}}\}$  the set of its triangles.

An annotation is defined as a triplet  $\mathcal{A} = (I, S, P)$ , where  $I$  is a piece of information,  $S$  is the *geometric selection* of parts of the mesh the information refers to, and  $P$  is a set of properties of the annotation. Each of the components of the triplet can be defined in several ways and needs to be subject to careful analysis and correct formalisation. In the following, we present an in depth view of  $I$ ,  $S$  and  $P$  respectively in subsection 2.2.1, 2.2.2, 2.2.3.

## 2.2.1 Information

While in principle information related to a piece of geometry could be of any type (a descriptive text, a link to a multimedia content, a quantitative measure), in this thesis we will consider information as a concept in the form of a textual tag, which is assigned to a geometric portion (tagging). We assume concepts to be selected from a shared and controlled vocabulary in order to avoid typical issues introduced by natural language (ambiguity, different levels of expertise, different language, granularity, etc.). Note that, while a simple controlled vocabulary is extremely useful for the reduction or even avoidance of these issues, employing more structured organisations of terms such as paronomies (e.g., [CVHS20], thesauri (e.g., [get]) or ontologies (see [RM08]) allow to infer information exploiting the already built network of connections and relations between objects. In this thesis, we will not follow this direction, but rather encode relations among entities in the form of *graphs* of relationships between annotations (see Section 2.3). Additional details can be anyhow encoded as annotation properties or attributes (see Section 2.2.3).

## 2.2.2 Selection

There are many ways for defining the geometric selection  $S$ . For example, in [PCDS20] clipping volumes are defined for selecting specific parts of the shape in real time, so that the selection remains well defined at varying Level of Detail (LoD). In this thesis, we define  $S$  as a homogeneous subset of mesh vertices, edges or triangles (i.e., no selection can be made both of vertices and triangles). We take into account three types of annotations according to their geometric selection (or nature, or dimension): points, poly-lines and regions.

- A *Point* annotation  $\mathcal{A}_p$  corresponds to a set of single mesh vertices and identifies landmarks on an object, that are typically used for reference or measurements, such as the nose tip, the sellion and the eye corner for the human face (e.g., [Cae20]). So,  $S_{\mathcal{A}_p} = V_{\mathcal{A}_p} \subseteq \mathcal{V}_{\mathcal{M}}$
- A *Poly-line* annotation  $\mathcal{A}_l$  is formed by one or more connected sets of edges (i.e., adjacent vertices) that can either be open or closed. So,  $S_{\mathcal{A}_l} = \{l\}$ , where

$$l = \{e_i^l = (v_s, v_e) \in E_{\mathcal{A}_l} \mid i = 1 \vee i = |l| \vee (\exists e' \in l : e' = (-, v_s) \vee \exists e' \in l : e' = (v_e, -))\}$$

where  $E_{A_l} \subseteq \mathcal{E}_{\mathcal{M}}$  is the set of edges associated to  $A_l$ . Given the edge-vertices relationship in the definition of the triangular mesh, one can obtain the vertices associated to the poly-line annotation as  $V_{A_l} = \{\mathbf{v} \in \mathcal{V}_{\mathcal{M}} \mid \exists e \in E_{A_l} : e = (-, \mathbf{v}) \vee e = (\mathbf{v}, -)\}$ . Annotations of this type identify feature lines such as crease lines, character lines, highlight lines in CAD.

- A *Region* annotation  $\mathcal{A}_r$  is a bi-dimensional patch formed by one or more connected sets of faces. A region can be formed by multiple components and may have holes. So,  $S_{\mathcal{A}_r} = \{r\}$ , where  $r = \{t \in T_{\mathcal{A}_r} \mid \exists t' \in T_{\mathcal{A}_r} : t \cap t' = e \in \mathcal{E}_{\mathcal{M}} \neq \emptyset \vee |r| = 1\}$ , where  $T_{\mathcal{A}_r} \subseteq \mathcal{T}_{\mathcal{M}}$  is the set of triangles associated to  $\mathcal{A}_r$ . Given the triangle-vertices relationship in the definition of the triangular mesh, one can obtain the vertices associated to the region annotation as  $V_{\mathcal{A}_r} = \{\mathbf{v} \in \mathcal{V}_{\mathcal{M}} \mid \exists t \in T_{\mathcal{A}_r} : \mathbf{v} = \mathbf{v}_1^t \vee \mathbf{v} = \mathbf{v}_2^t \vee \mathbf{v} = \mathbf{v}_3^t\}$ . Notice that, since we are using triangular meshes as shape representation, this kind of annotation regards both regions having a *superficial* (e.g., a drawing on a vase) and a *volumetric* nature (e.g., the head of a statue).

Here the whole object is considered as an annotated region, labelled according to the template class (e.g., a “teapot”). This will be helpful both for enriching the contextual search (e.g., search for all the “teapot” shapes) but even for the organisation of annotations.

### 2.2.3 Attributes

Each annotation can be described by a set of properties  $P$ , quantitative or qualitative characteristics defining some aspect of that specific part. We name those properties *Attributes*. The whole object (i.e., the root of the annotation containment tree, see subsection 2.3) maintains the global attributes of the shape (e.g., total height of the object, site of excavation of the archaeological finding, etc.).

*Qualitative* attributes describe the annotated part qualitatively, e.g., a “beard” region may have a “beard style” attribute specifying whether it is curly, wavy, or straight. We expect qualitative attributes to be textual tags from a vocabulary of terms that are meaningful for the attributes of a certain concept.

On the other hand, *quantitative* attributes are numbers, measurements or more complex descriptors of the geometrical component of an annotation. For instance, an “eye” may have a “width” attribute specifying the horizontal span of that piece of geometry. A region annotated as “beard” might be investigated more through a surface curvature analysis; the output curvature map might be the value of a quantitative “bumpiness” attribute. This last example also shows how qualitative and quantitative attributes may refer to the same aspect from different perspectives: a “straight” beard with high “bumpiness”, with the curvature map evidencing linear structures, might mean that the beard presents deep engraved lines. This example highlights how some qualitative attributes may be defined by geometric rules and be, to some extent, inferred by quantitative ones.



Figure 2.3: *Typical cases of measures taken in the physical world (fragments coming from the Salamis collection studied in the GRAVITATE project [PWM<sup>+</sup> 16]) (images from the British Museum repository at <https://www.britishmuseum.org/collection>).*

While the meaning of a measure could be known by convention (e.g., “height” for a statue is likely to be the Euclidean distance between the base and the topmost point, with the statue having an explicit upright orientation) or defined in the knowledge formalisation in terms of landmarks (e.g., the width of an eye is the distance between the internal and external apex), we provide “pins” to specify and store the measurements reference. This addresses a common archaeological documentation issue: often objects are described along with a “length” value, without specifying the direction and extreme points of the measurement, nor the tool to measure it (or equivalently the distance involved, Euclidean or geodesic). Things get even more vague when pieces lacking a principal direction or orientation are concerned, e.g., terracotta fragments (see Figure 2.3).

Attributes play a twofold role: on the one side, it is likely that attributes for different objects in the same class will stay the same or change in a controlled pattern (e.g., within a range). There-

fore, attributes and relationships between attributes will be subject to *constraints* to guarantee a valid intra-class deformation (see Chapter 4). On the other side, annotations and their attributes provide a documentation mechanism for the 3D object, supported by interactive selection and measuring tools, that allows quantitative and qualitative descriptions of a specific object rather than of the class of objects in general.

## 2.3 Relationships among annotations

To better exploit the information provided by annotations, they are organised into a hierarchy, where the parental relationship is given by the annotation *containment* (here denoted by  $\sqsubset$  and  $\sqsupset$ ). There are different types of annotation containment. In particular:

- A point annotation  $\mathcal{A}_p$  can be contained
  1. into another point annotation  $\mathcal{A}'_p$  if and only if all of its associated vertices can be found into the set of vertices associated to the other.
  2. into a poly-line annotation  $\mathcal{A}'_l$  if and only if all of its associated vertices can be found into its poly-lines:  $\mathcal{A}_p \sqsubset \mathcal{A}'_l \iff V_{\mathcal{A}_p} \subseteq V_{\mathcal{A}'_l}$ ;
  3. into a region annotation  $\mathcal{A}'_r$  if and only if all of its associated vertices can be found in any of the selected triangles:  $\mathcal{A}_p \sqsubset \mathcal{A}'_r \iff V_{\mathcal{A}_p} \subseteq V_{\mathcal{A}'_r}$ ;
- A poly-line annotation  $\mathcal{A}_l$  can be contained
  1. into another poly-line annotation  $\mathcal{A}'_l$  if and only if all of its associated edges can be found into any of the others' poly-lines:  $\mathcal{A}_l \sqsubset \mathcal{A}'_l \iff E_{\mathcal{A}_l} \subseteq E_{\mathcal{A}'_l}$ ;
  2. into a region annotation  $\mathcal{A}'_r$  if and only if all of its associated vertices can be found in any of the selected triangles:  $\mathcal{A}_l \sqsubset \mathcal{A}'_r \iff V_{\mathcal{A}_l} \subseteq V_{\mathcal{A}'_r}$ .
- A region annotation  $\mathcal{A}_r$  can be contained only in another region annotation  $\mathcal{A}'_r$  if and only if all of its associated triangles can be found in the set of triangles associated to the other:  $\mathcal{A}_r \sqsubset \mathcal{A}'_r \iff T_{\mathcal{A}_r} \subseteq T_{\mathcal{A}'_r}$ .

Following this procedure, a tree structure can be populated, with the whole shape as root and all the other annotations as child nodes, following the containment relationships. An example of this hierarchical organisation of the annotations can be seen in Figure 2.4. This first structure will then be enriched with other types of relationships, defining a relationships (hyper)graph.

Some of these relationships are naturally defined and can be automatically extracted directly from the geometry; beside *containment* we consider in our implementation the *adjacency* between two parts; other notable relationships like symmetries, co-axiality, co-linearity (in case of points

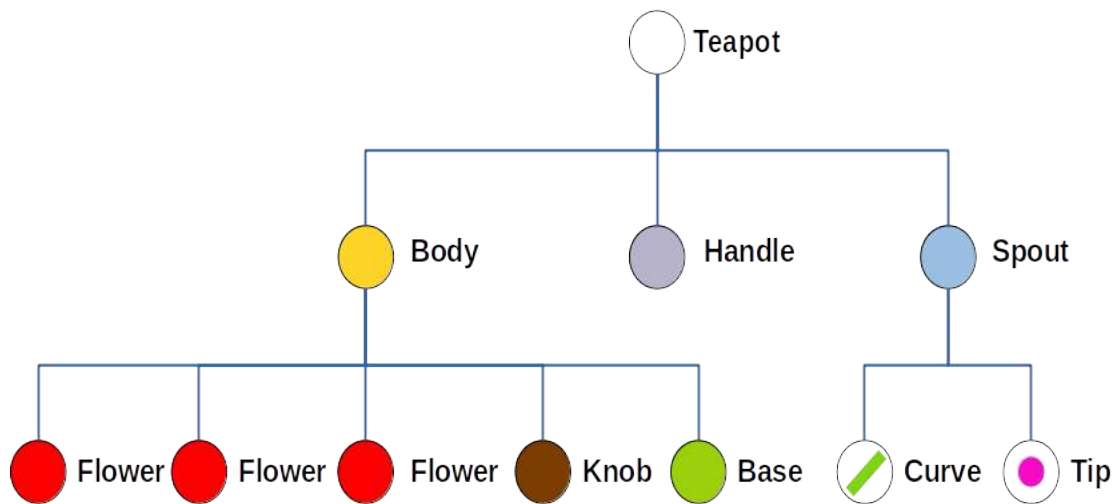


Figure 2.4: A possible annotation of a teapot shape. Here, the different types of selection are shown: point (“Tip”), poly-line (“Curve”) and region (e.g., “Flower”). In the lower part we show the hierarchical structure of annotations induced by containment.

selections), part similarity, etc. [LMS13, BDS<sup>+</sup>12], that are deeply studied in previous works, could be easily added to the framework in future versions.

On the other side, there are relationships that are not directly inferred from the geometry, because they require a more complex reasoning, like semantic attributes. Examples can be the *expertise* level of an artisan crafting certain parts of a statuette, which we analyse in Section 3.2 for building tools supporting archaeological studies, the *erosion* severity of certain parts of a building but even the style of components of an object. These kind of relationships are necessarily stated by a domain expert, who possess a strong knowledge on the homogeneous class and has a priory knowledge about them, even if future improvements in AI approaches are likely to substitute human expertise with automatic tools.

Of course a vast number of relationships could be defined among annotations, some of them maybe more salient than others in specific domains or for particular object classes. In this thesis we provide a small set of example relationships, but the system allows for defining new ones.

Laga et al. [LMS13] focused on man-made objects and used containment, adjacency, symmetry, side contact, co-axiality and horizontal support as part relations. Similarly to their approach, we insert additional relationships in the hierarchy defined by annotations containment as additional arcs with type, either directed or not. The additional relationship arcs are likely to create loops, so that the annotation hierarchy will not have a tree structure any longer. Relationships between annotations may simply translate into a relationship among their attributes, e.g., a “same length” relationship between the two arms of a human shape is verified if the two “arm” annotations have the same value of the *length* attribute).

The above example introduces the idea that the relationship graph, encoding the semantics of a homogeneous class of objects, can be exploited for defining *constraints* on the deformation (see Chapter 4). An example of relationship graph for chairs is depicted in Figure 2.5.

## 2.4 Annotation persistence

In the proposed framework, annotations represent the link between semantics and geometry and the glue between the two sides of the parametric template. In this thesis, we pursue the continuous synergy between these two natures, exploiting geometric analysis to enrich the semantics and using the semantics to drive geometric deformations.

However, asynchronous changes on either the semantic or geometric side could compromise the consistency of annotations. For example, a computational intensive algorithm could require a decrease in geometric resolution of the 3D model at hand, or the annotations defined by two users may use a different language for defining the same concept.

While language-related issues can be mitigated by the use of a shared and controlled vocabu-



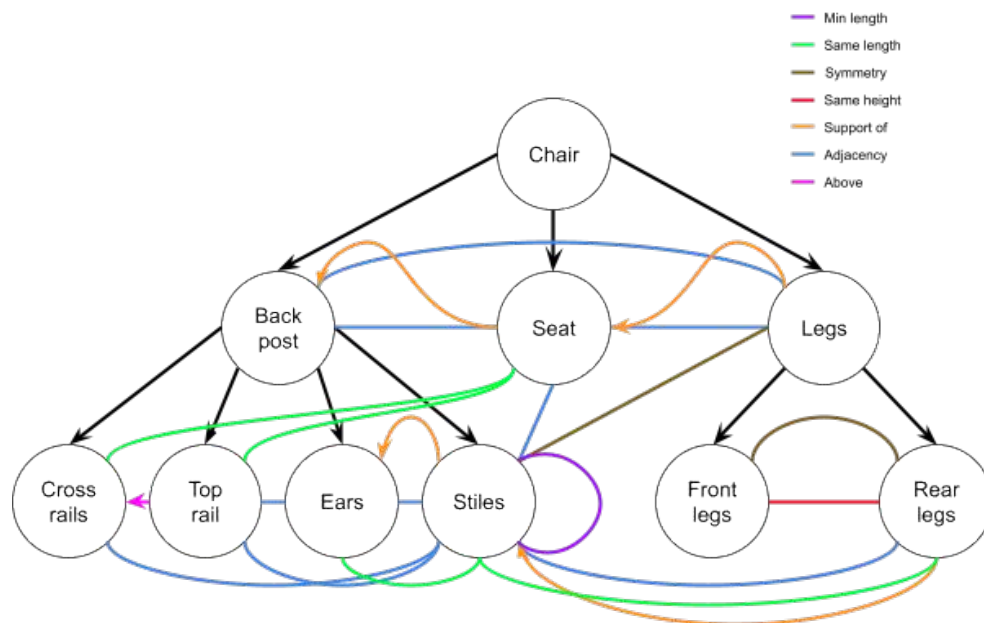


Figure 2.5: An example showing some relationships between the different parts composing a simple chair shape.

lary, geometric changes can indeed damage the annotations [HF07]. The main issues can be synthesised in these categories:

- Change of resolution: the object representation' resolution is changed, e.g., in a multi-LoD application or for specific requirements of the system, and so coarsened or refined in the vertices and triangles density;
- Change of arrangement: the spatial displacement of vertices' positions is changed, e.g., due to a geometry processing algorithm (e.g., Laplacian fairing);
- Change of representation: e.g., passing from a triangle mesh to a quadrilateral mesh or a tetrahedron mesh.

Therefore, mechanisms are required to guarantee consistency of annotations regardless of the transformation applied to the geometry.

Issues introduced by geometric changes are easily solved by defining geometric selections in terms of *indices* of vertices, edges and triangles respectively for the point, poly-line and region annotations (see Section 2.2.2). So, even if the simplexes involved in the representation are moved in space, the annotations would not be corrupted.

Conversely, the change of resolution is a very difficult issue. This could be solved by employing a spatial indexing of annotations, such as the one in [PCDS20]. This technique defines a portion

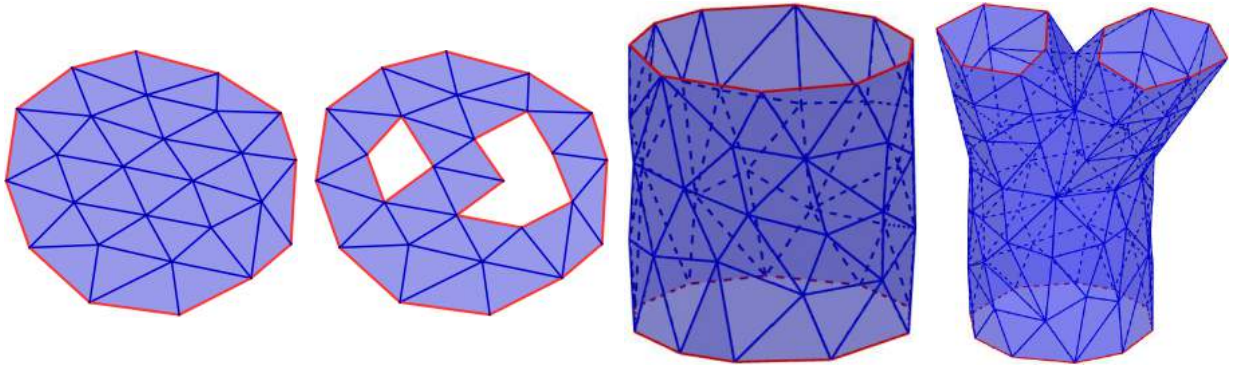


Figure 2.6: *Different types of selection (from left to right disk, holes, cylinder and fork). Notice that holes and fork are topologically equivalent so that the same solution solves both the configurations.*

of space into which the geometric embedding of the annotation could be found. Of course, changing the resolution would not damage the selection, because the corresponding portion of shape would be still enclosed in the defined volume. However, this type of techniques can not easily deal with deformations of the shape, re-introducing the change of displacement issue (e.g., elongating one finger of the hand can make it partially fall off the defined volume).

For this reason, we implemented an *annotation transfer* method to guarantee the survival of annotations after changes in resolutions, obtained by projecting the annotations from one source representation to another [SMS17, SMS20].

### 2.4.1 Annotation Transfer

Let a region annotation be defined as  $A_r = (I, S, P)$ , where  $I$  is some information of any type,  $S$  is the geometric selection of interesting parts on the geometry of the mesh and  $P$  is a set of attributes or properties of the annotation.

Let a selection patch  $S_i \subseteq \mathcal{T}_{\mathcal{M}}$  be a connected, 2-manifold subset of triangles in  $\mathcal{M}$ , such that its boundaries, named patch *outlines*  $O_{ij} \subset \mathcal{E}_{\mathcal{M}}$ , are *ordered* sets of vertices, connected in pairs by edges of  $\mathcal{M}$ .

Each selection patch is therefore connected, but can have an arbitrary number of boundaries. In the simplest case, the selection is topologically equivalent to a disk and the outline is a single boundary curve. We describe the solution to this configuration in [SMS17]. However, the selection can have multiple boundaries, e.g., in case of holes (see Fig. 2.6).

Starting from an annotated patch  $S_s$ , totally enclosed by a single outline  $O_s$  and belonging to a source mesh  $\mathcal{M}$  we want to identify a set of triangles onto a target mesh  $\mathcal{M}_t$  that well approxi-

mates  $S_s$ , which is the corresponding target patch  $S_t$  on  $\mathcal{M}_t$ . Note that the transferred patch  $S_t$  will consist of elements of  $\mathcal{M}_t$ , that is, we do not insert vertices on  $\mathcal{M}_t$  to define  $S_t$ .

In our study, an important hypothesis is that the source and target meshes are perfectly aligned, which makes the procedure easier. With this, we do not mean that the association between the vertices of the source and target shape is known in advance, but rather that the two shapes are nothing more than a representation, at different resolution, of the same object. So, the pose, orientation and position of them should be the same. In cases that do not respect this requirement from the beginning, it is necessary to adopt a prior shape alignment phase, using one of the many methods offered by the state of the art [TCL<sup>+</sup>13].

For each vertex in  $O_s$ , the transfer procedure identifies the best match among vertices in  $\mathcal{M}_s$  and  $\mathcal{M}_t$ . This builds an ordered set of matched vertices  $O_m$ , which belong to  $O_t$  but are not adjacent in  $\mathcal{M}_t$  on a general basis. So, more vertices of  $\mathcal{M}_t$  might belong to  $O_t$ , especially when the transfer targets a higher resolution mesh. Then, the complete outline of  $S_t$  can be reconstructed by tracing the shortest paths between successive pairs of vertices in  $O_m$ .

The main steps of the procedure are listed in Algorithm 1.

---

**Algorithm 1** Transfer a patch outline  $O_s$  onto a new mesh  $\mathcal{M}_t$ .

---

```

1: procedure PATCHTRANSFER( $O_s, \mathcal{M}_t$ )
2:   for each  $(\mathbf{v}_s^i, \mathbf{v}_s^{i+1})$  in  $O_s$  do
3:      $\mathbf{v}_t^i \leftarrow \text{FINDCORRESPONDENCE}(\mathbf{v}_s^i, \mathcal{M}_t)$ 
4:      $\mathbf{v}_t^{i+1} \leftarrow \text{FINDCORRESPONDENCE}(\mathbf{v}_s^{i+1}, \mathcal{M}_t)$ 
5:      $O_t \leftarrow O_t \cup \text{SHORTESTPATH}(\mathbf{v}_t^i, \mathbf{v}_t^{i+1})$ 
6:   end for
7: end procedure

```

---

Since  $\mathcal{M}_s$  and  $\mathcal{M}_t$  differ, maybe considerably, the transfer will necessarily introduce a distortion in the shape of the selection area. The amount of such distortion depends on four factors:

- the geometric difference between  $\mathcal{M}_s$  and  $\mathcal{M}_t$ ;
- the shape of the original selection  $S_s$ ;
- the method chosen for the vertex correspondence procedure (see Section 2.4.3);
- the metric employed for the distance in the shortest path computation (see Algorithm 3).

Indeed, given  $S_s$ ,  $\mathcal{M}_s$  and  $\mathcal{M}_t$ , we can define a vertex correspondence procedure such that the transferred patch  $S_t$  either totally encloses  $S_s$  or only partially overlaps it. Intuitively, the partial overlap approach is preferable to minimise distortion (see sub-section 2.4.2). This is synthesised in Algorithm 2.

---

**Algorithm 2** Find the correspondence as the mapping of a vertex  $\mathbf{v}$  onto the one ( $\mathbf{v}_C$ ) of the target mesh  $\mathcal{M}_t$  which is closest to the projection and whose normal matches the one of  $\mathbf{v}$ .

---

```

1: procedure FINDCORRESPONDENCE( $\mathbf{v}$ ,  $\mathcal{M}_t$ )
2:    $d \leftarrow \infty$ 
3:    $\mathbf{r} \leftarrow \text{getNormal}(\mathbf{v})$ 
4:   for each  $t'$  in  $\mathcal{T}_{\mathcal{M}_t}$  intersected by  $\mathbf{r}$  do
5:      $d' \leftarrow \text{distance between } \mathbf{v} \text{ and } t'$ 
6:     if ( $d' < d$ )  $\wedge$  ( $\text{getNormal}(t') \cdot \mathbf{r} \geq 0$ ) then
7:        $t \leftarrow t'$ 
8:        $d \leftarrow d'$ 
9:     end if
10:  end for
11:   $\mathbf{v}_C \leftarrow \text{vertex of } t \text{ closest to } \mathbf{v}$ 
12: end procedure

```

---

It may happen that different vertices of  $O_s$  are mapped onto the same target vertex, and this may produce a badly-shaped (i.e., non manifold) target selection. To avoid this case, we add a check for rejecting duplicated vertices in  $O_t$ .

Note that the projection is made with a raycast-like approach, checking, for each vertex in  $O_s$ , if its ray intersects any triangle in  $\mathcal{M}_t$ . Indeed, the simple search for the closest vertex may fail in case of high resolution reduction and/or very close features with no relation with the current vertex, while issues related to noisy normals can be solved or at least reduced by de-noising the surface and/or re-computing the normal by averaging with respect to the neighbours.

This raycast-like approach results in a very high computational complexity ( $O(n)$  only for the projection, where  $n$  is the number of triangles of the target model). So, we improved the efficiency of the approach by inserting all the target vertices in a KD-tree (managed with the nanoflann library [BR14]) and checking only triangles inside a sphere with a pre-defined radius.

---

*KD-tree radius selection.* Note that the definition of the radius for the sphere impacts very much the procedure. In fact, if the radius is too small the intersection check would not find any triangle, whereas if it is too large the whole process would be slowed down drastically, equal to the complexity without optimisation in the worst case.

So, the best radius value is a trade-off between performance and retrieval power. Furthermore, note that it would be better to have two different radius values: one from low-to-high and one for high-to-low transfers. In fact, going from low to high resolution it is likely to have a lot of target vertices very close to the source one, so that we can reduce the radius (we usually use  $\frac{1}{1000}$  of the bounding box diagonal for the low-to-high transfers), while the opposite is true from high to low resolution (we used  $\frac{1}{100}$  of the bounding box diagonal).

---

Once corresponding vertices have been found, we take all successive couples in  $O_m$  and we search onto  $\mathcal{M}_t$  the shortest path between them. This is done with an approximation of the Dijkstra algorithm [Dij59], where the search is interrupted as soon as the target vertex has been reached. The pseudo-code for the shortest path algorithm can be found in Algorithm 3

---

**Algorithm 3** Finds the shortest path between the given pair of vertices.

---

```

1: procedure SHORTESTPATH( $\mathbf{v}_1, \mathbf{v}_2$ )
2:    $D.add((\mathbf{v}_1, 0))$ 
3:    $F.push(\mathbf{v}_1)$ 
4:   do
5:      $\mathbf{v} \leftarrow F.pop()$ 
6:     for each  $\mathbf{v}'$  in the first ring neighbourhood of  $\mathbf{v}$  do
7:        $d \leftarrow D(\mathbf{v}) + dist(\mathbf{v}, \mathbf{v}')$ 
8:       if  $\mathbf{v}'$  has not been visited then
9:          $D.add((\mathbf{v}', d))$ 
10:         $P.add((\mathbf{v}', \mathbf{v}))$ 
11:         $F.push(\mathbf{v}')$ 
12:       else if  $D(\mathbf{v}') > d$  then
13:          $D(\mathbf{v}') \leftarrow d$ 
14:          $P(\mathbf{v}') \leftarrow \mathbf{v}$ 
15:       end if
16:     end for
17:   while  $\mathbf{v} \neq \mathbf{v}_2$ 
18:    $path \leftarrow P$ 's elements sequence from  $\mathbf{v}_2$  to  $\mathbf{v}_1$  (reversed)
19: end procedure

```

---

With reference to Algorithm 3, there are many different possibilities for the definition of the *dist* function. In the present work, we have tested three different types of distances:

- The common *Euclidean* distance.
- The Euclidean distance to the segment between the current pair of target vertices (from now on we refer to this as the *Segment* distance).
- The combination of the above (from now on we refer to this as the *Combined* distance).

The results obtained using these three distances are compared in Section 2.4.2.

Note that it is not guaranteed that the path between successive target vertices will not intersect other parts of  $O_t$ . In practice, this issue is solved by assigning an infinite weight to arcs incident in vertices already in  $O_t$  in the check for the shortest path computation. Notice that this solution

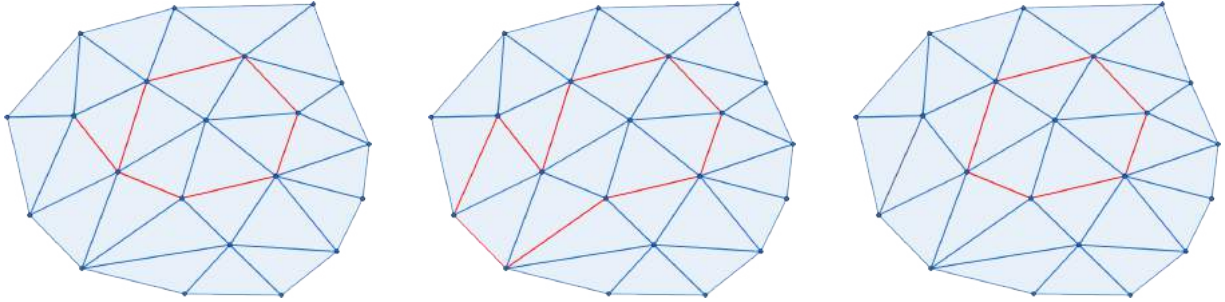


Figure 2.7: *On the left, a transferred outline with a “spike”, on the middle the result obtained with the infinite weight on the used arcs for the Dijkstra algorithm, on the right the pruned result.*

even allows to manage the “spikes” (see Figure 2.7) that are sometimes created by the procedure. While the obtained result is good in general, we allow the user to decide whether to follow this path or post process the outline by simply pruning the generated spikes (the results can be quite different, see Figure 2.12).

Finally, we combine the paths computed in the previous step between couples of successive target vertices to obtain a closed outline.

Once the ordering of the outline of the target selection is completed, obtaining the enclosed patch is straightforward (region growing with the triangle on the left of any edge in  $O_{ij}$ ).

The next step regards the transfer generalisation to regions of arbitrary topology. Indeed, selections may exhibit significant variations: some examples of selections are shown in Fig. 2.6. As one can see, the difference between those cases consists in the number of boundaries. Furthermore, we can envisage multiple patches to be part of the same selection.

To generalise the approach to selections of arbitrary topology, we project all the outlines of a selection onto the target mesh as described above; the only issue is how to identify the interior of the target selection properly. In the previous solution, the ordering of the single target outline was induced by the ordering of the source outline and, as such, defined a priori and uniquely; the same paradigm holds for additional boundaries, provided that they are ordered consistently, i.e., all the boundaries are ordered so that the interior of the selection lies on the left of the oriented boundary edges.

We then define a selection  $S$ , of a Region annotation (see Subsection 2.2.2), as a set of patches  $S_i$ , each one uniquely identified by a set of boundaries  $O_{ij}$ , each one ordered so that the interior of  $S_i$  always lays on the left side of the edge  $(\mathbf{v}_k, \mathbf{v}_{k+1}) \in O_{ij}$  for every  $k$ . With this assumption, the selection area is uniquely defined by its outlines.

## Outlines extraction

In general, we have no clue about what are the boundaries of a region selection, nor about their number. So we have to define how to obtain a set of outlines, ordered as already said, from a set of triangles ( $T_S \subseteq \mathcal{T}_M$ ), not necessarily connected.

For each  $t \in T_S$  we check if any of its edges separate it from an external triangle and, in that case, the separating edges are marked. In this way we obtain an unordered and not necessarily connected set of edges  $E_S \subset \mathcal{E}_M$ . Then, starting from a vertex  $\mathbf{v}$  of any edge in  $E_S$ , we check for each  $\mathbf{v}'$  in its 1-neighbourhood if  $(\mathbf{v}, \mathbf{v}')$  is an edge in  $E_S$  and, in that case, we remove it from  $E_S$  and put it in  $O_i$ . This process is repeated iteratively until  $O_i$  is completed, taking each time  $\mathbf{v}'$  as  $\mathbf{v}$  (when  $(\mathbf{v}, \mathbf{v}')$  is in  $E_S$ ) and ignoring the already used vertices. Then, if  $E_S$  is not empty, we initialise a new outline component and we repeat the whole process until there are edges in  $E_S$ . This allow us to obtain a set of closed outlines. At last, for each  $O_i$  enclosing  $S$  we check if the triangle on the left of its first edge is in  $T_S$ . If this is not true, we invert the outline. The whole process is reported in Algorithm 4. Using the presented approach, we managed to obtain the result presented in Fig. 2.10.

---

**Algorithm 4** Extracts the *ordered* outlines that uniquely identify the selection (encloses  $T$ )

---

```
1: procedure OUTLINESEXTRACTION( $T_S$ )
2:    $E_S \leftarrow$  edges in  $T_S$  with only one incident triangle in  $T_S$ 
3:   while there are elements in  $E_S$  do
4:      $\mathbf{v}_0 \leftarrow$  first vertex of any edge in  $E_S$ 
5:      $\mathbf{v} \leftarrow \mathbf{v}_0$ 
6:     do
7:        $o.push(\mathbf{v})$ 
8:        $e \leftarrow$  first edge incident to  $\mathbf{v}$  that is in  $E_S$ 
9:        $\mathbf{v} \leftarrow e.oppositeVertex(\mathbf{v})$ 
10:       $E_S.pop(e)$ 
11:     while  $\mathbf{v} \neq \mathbf{v}_0$ 
12:      $O.push(o)$ 
13:   end while
14:   for each  $o \in O$  do
15:     if left triangle of any edge in  $o$  is not in  $T$  then
16:       revert  $o$ 
17:     end if
18:   end for
19: end procedure
```

---

## 2.4.2 Transfer results

The annotation transfer presents some interesting properties. First, it guarantees to always achieve a result; in the worst case, the target area may be degenerate (the annotated area collapses to a point due to a very large disparity of mesh sampling density). In most of the cases, the transfer works well, identifying target areas very similar to the source (see Table 2.1).

The quality of the transfer procedure has been studied in the context of the GRAVITATE project [PWM<sup>+</sup>16], tackling problems related to the re-unification, re-assembly and re-association of archaeological terracotta fragments of the Salamis collection, now divided among several museums throughout Europe. In particular, the transfer has been extensively used for the propagation of the so called “facets”<sup>1</sup>. In the project, facets were computed (the faceting extraction pipeline is explained in [ED17]) for 117 fragments in the Salamis collection (that were selected for their archaeological interest) on models at 50K resolution, due to the time complexity of the faceting algorithm. The resulting facets, annotated, were then transferred to the other resolutions used in the project (100K, 1M and full resolution) using our method.

More precisely, depending on the available resolutions, the presented method has been applied, for each distance type, for 105 transfers from 50K to 100K, 42 transfers from 50K to 1M and for 117 transfers from 50K to the full resolution version of the model.

The results of these transfers has been summarised in Table 2.1. For a visual evaluation of the degree of precision of the method see Figure 2.8, 2.9 and 2.11.

Notice that those in Table 2.1 are average values. Time is expressed in seconds and it is reliable only for the  $50K \rightarrow 100K$  and the  $50K \rightarrow 1M$  transfers, because the full resolution spans from  $\sim 46K$  to  $\sim 27M$  vertices, so the average is not representative (further details can be found in the supplementary material of [SMS20]).

As can be seen, the Euclidean distance and the Segment distance produce better results in terms of perimeter distortion and area distortion, respectively. This is probably because of the nature of the Segment distance, which privileges paths closer to the starting segment, thus leading to a connecting path having a sort of zigzag pattern; this behaviour keeps the area distortion very low, while incrementing the perimeter distortion in turn. Still, if we analyse the ratio between perimeter and area (and vice versa) we see that the Euclidean distance stays the best and the same goes for the circularity ( $P^2/A$ ). So, we believe that the advantage of reducing the area distortion of the Segment distance is compromised by the corresponding worsening of the perimeter distortion, at the extent that it reduces the overall precision of the transfer. As for the Combined distance, its only advantage is it never gives the worst result, but it always stands in a sort of middle place.

---

<sup>1</sup>The facets are different regions of a model which belong to a specific view of a fragment. There are three different kinds of facets, that are the *external*, *internal* and *fracture* facets.



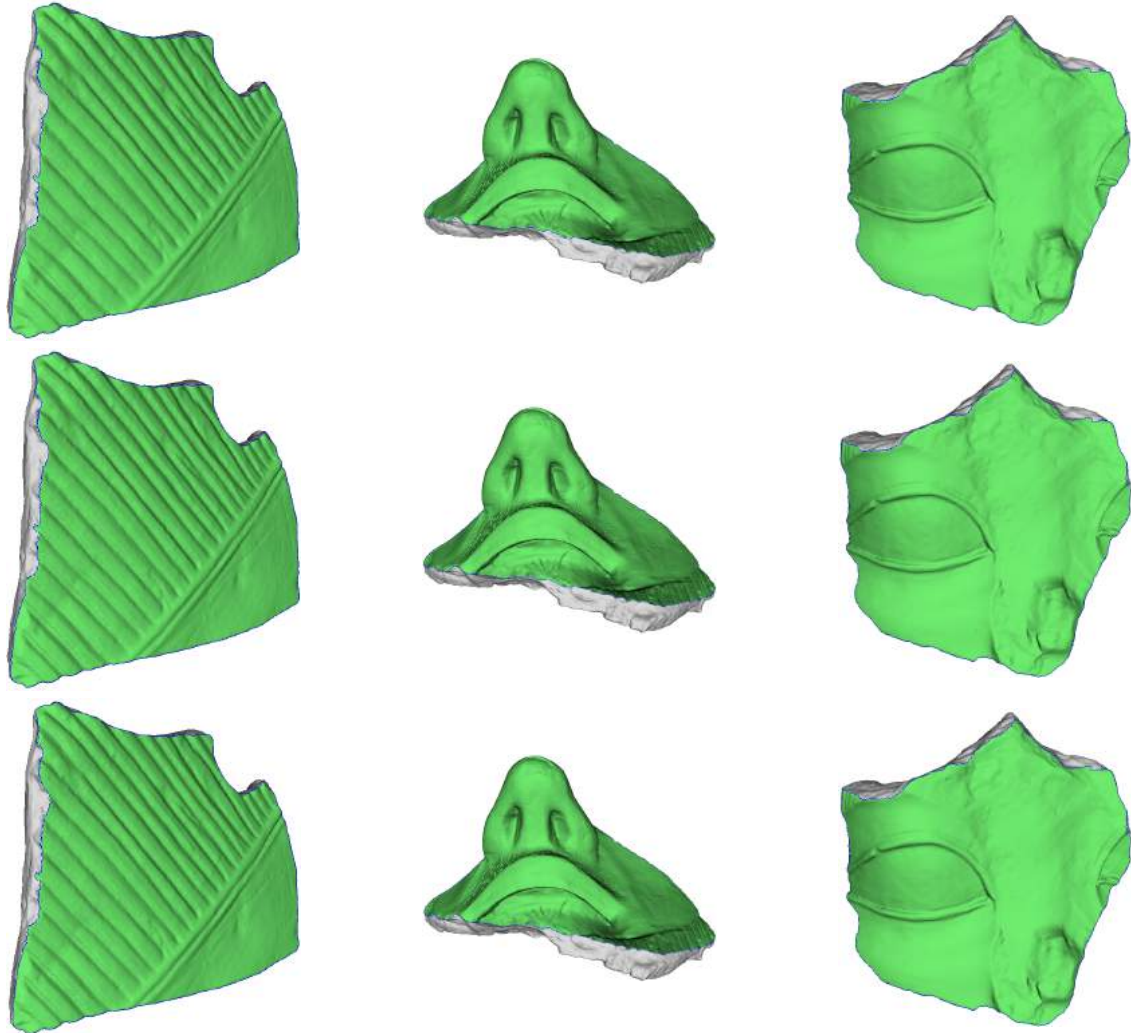


Figure 2.8: *Different facet transfers (going downward) from 50K to 1M vertices resolution and backwards*

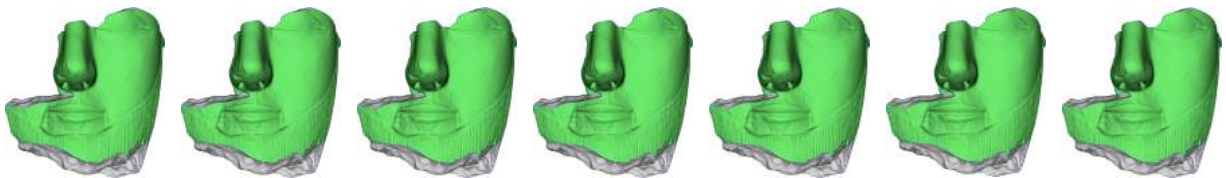


Figure 2.9: *Transfer of a facet between 50K, 100K, 1M, full resolutions (respectively) and backwards*

Table 2.1: The results of the application of the procedure, in the GRAVITATE context, for the facet transfer. The columns refer to the distortion of the target region (in percentage) with respect to:  $A \rightarrow$  Area,  $P \rightarrow$  Perimeter,  $C \rightarrow$  Circularity ( $P^2/A$ ); the  $P/A$  and  $A/P$  columns are obvious. The best results between Euclidean, Segment and Combined distance are highlighted in bold. The transfers have been performed on a computer with 64 GB DDR4 RAM and Intel® Core™ i7-7700 CPU. These values are all obtained applying the specified descriptors to the facets before and after the transfer and then computing their ratio.

S $\rightarrow$ T	Metric	Time (s)	A	P	P/A	A/P	C	Average
50K $\rightarrow$ 100K	Segment	0,82	<b>0,0170%</b>	1,2818%	1,2687%	1,2888%	2,5305%	1,2774%
	Euclidean	<b>0,81</b>	0,0182%	<b>1,2305%</b>	<b>1,2158%</b>	<b>1,2341%</b>	<b>2,4280%</b>	<b>1,2253%</b>
	Combined	0,82	0,0174%	1,2435%	1,2297%	1,2484%	2,4544%	1,2387%
50K $\rightarrow$ 1M	Segment	<b>6,44</b>	<b>0,1675%</b>	6,7841%	6,6310%	7,1235%	12,9461%	6,7304%
	Euclidean	6,47	0,1808%	<b>5,4278%</b>	<b>5,2584%</b>	<b>5,5609%</b>	<b>10,3909%</b>	<b>5,3638%</b>
	Combined	6,45	0,1694%	6,0404%	5,8836%	6,2672%	11,5545%	5,9830%
50K $\rightarrow$ clean	Segment	8,67	<b>0,0805%</b>	4,3484%	4,2782%	4,5804%	8,3411%	4,3257%
	Euclidean	<b>8,63</b>	0,0927%	<b>3,5538%</b>	<b>3,4688%</b>	<b>3,6547%</b>	<b>6,8423%</b>	<b>3,5225%</b>
	Combined	8,65	0,0821%	3,9051%	3,8321%	4,0672%	7,5121%	3,8797%

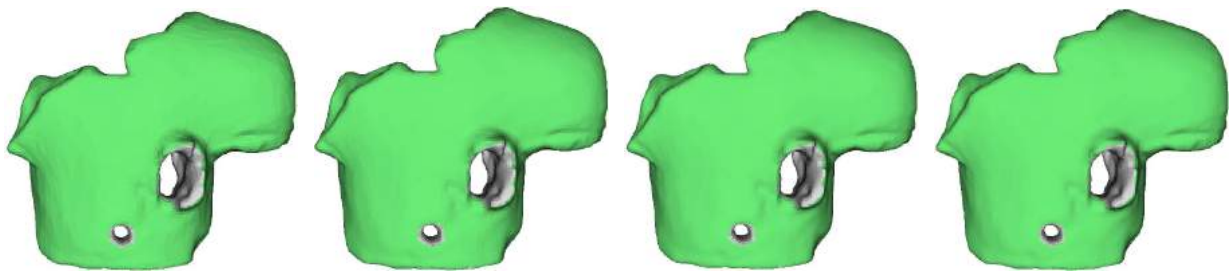


Figure 2.10: A selection with different boundaries at different resolutions (from left to right: 5K, 10K, 30K and  $\sim$  50K)

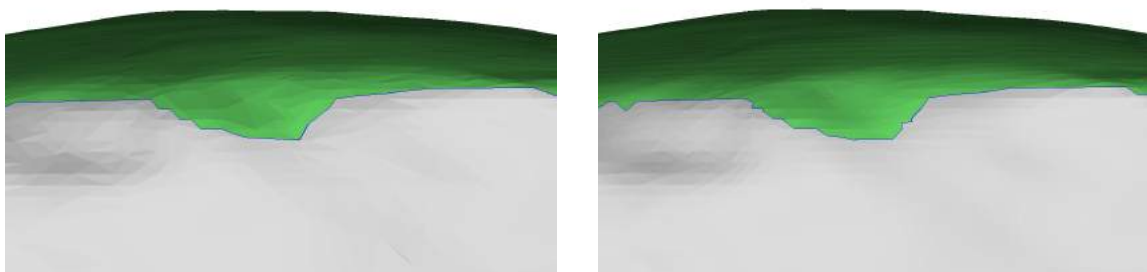


Figure 2.11: Zoomed-in view of a fragment at, respectively, 50K and 1M vertices resolution

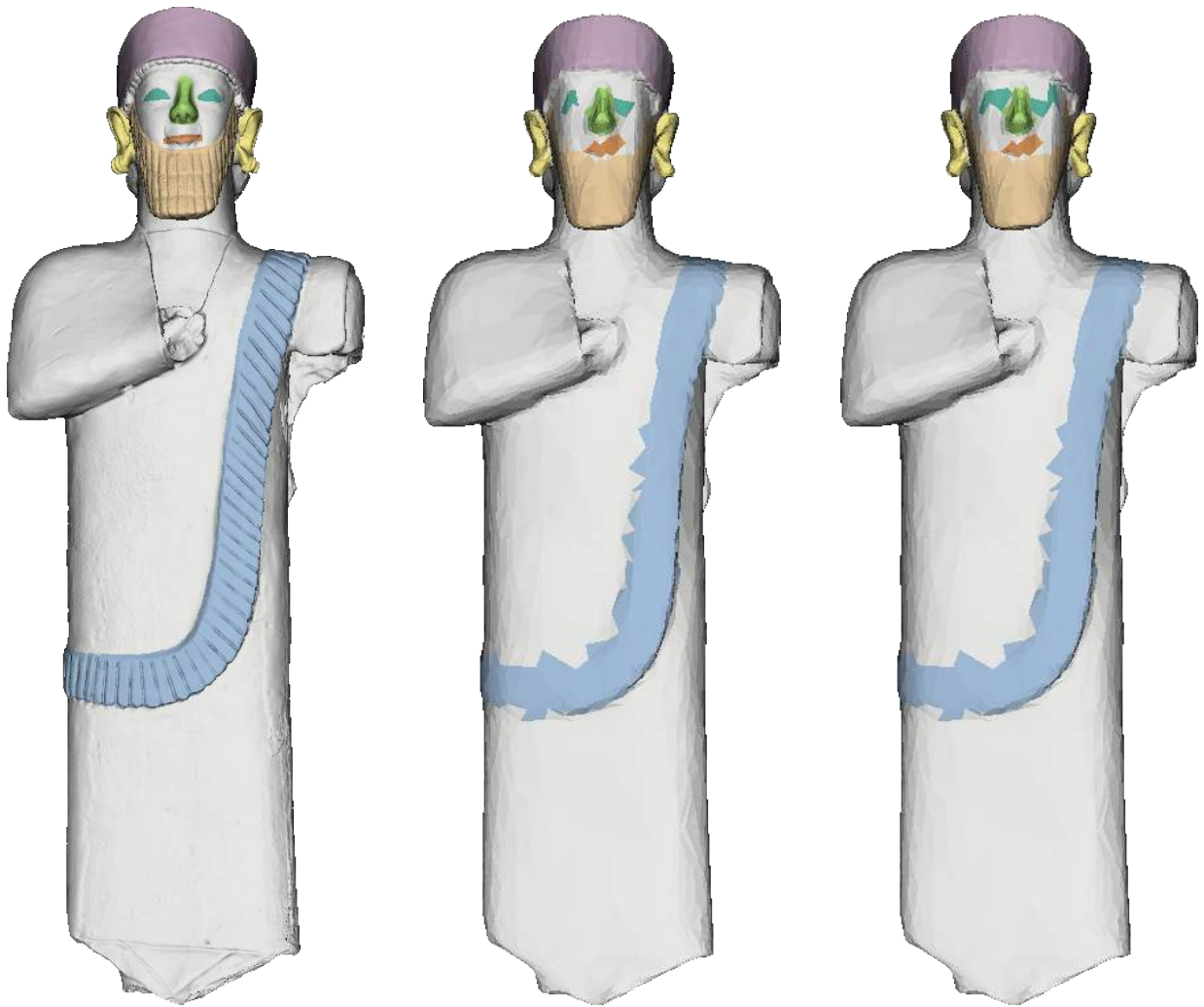


Figure 2.12: *Transfer results on aggressively reduced resolution. From left to right: annotation on the 1M vertices model and transferred annotation on a simplified version (~5K vertices) using a pruning technique on the spikes produced by the algorithm and the infinite weight on used arcs for the shortest path computation.*

### 2.4.3 Transfer limitations

The proposed transfer approach presents some issues:

- *Degeneracy*: the area of a patch, expressed as an outline composed of more than two vertices, might map to a degenerate outline which encloses an empty area (see Figure 2.13a);
- *Wrong projection*: a vertex in the higher resolution mesh can be nearer to a triangle which isn't actually connected to the annotation even if its normal points in the right direction (see Figure 2.13c; this is the case when the shape exhibits several nearly overlapping layers);
- *Distortion*: when some parts of an annotated area are greatly simplified, there could be an excessive distortion, in terms of area and shape (see Figure 2.13b).
- *Intersection*: if two or more boundaries of the same patch are very close, the transfer distortion could lead to an intersection of the boundaries, causing problems in the region growing for obtaining the inner triangles (Figure 2.13d-f).

Currently, the transfer procedure has been defined only to tackle region annotations (see Section 2.2.2). In future developments, we plan to address point and poly-lines annotations. Note that allowing and formalising dishomogeneous transfer (between selections of different dimensions, e.g., region to line) would solve the degeneracy issue in transferring regions: in fact, the mapping of an entire area onto a point or line can be viewed as a way for keeping track of the presence of an interesting feature opening for innovative representation of the object annotations where semantic LoDs could be implemented.

Another interesting improvement could be to consider the insertion of new vertices on the target mesh to better approximate the original RoI shape in case of huge distortions or to prevent degeneracies. This approach is potentially dangerous as it could introduce inconsistencies across different users in shared applications.

A final observation regards different kinds of correspondence between vertices. Indeed, defining the correspondence between source and target vertices strongly impacts the resulting target area, going from a totally enclosing to a totally enclosed transferred annotation. However, the latter approaches would introduce more distortion, depending on the arrangement of the vertices on the model. An example can be viewed in Figure 2.14

We believe that the presented method, even if currently specialised to tackle the change of resolution issue only, could be exploited and generalised to tackle the change of representation as well. For instance, the change from a triangular to a quadrilateral mesh could be faced in principle by triangulating the quad mesh and falling again in the projection from a triangular mesh to another. An interesting problem would be projecting from a triangular to a tetrahedral mesh: the exterior

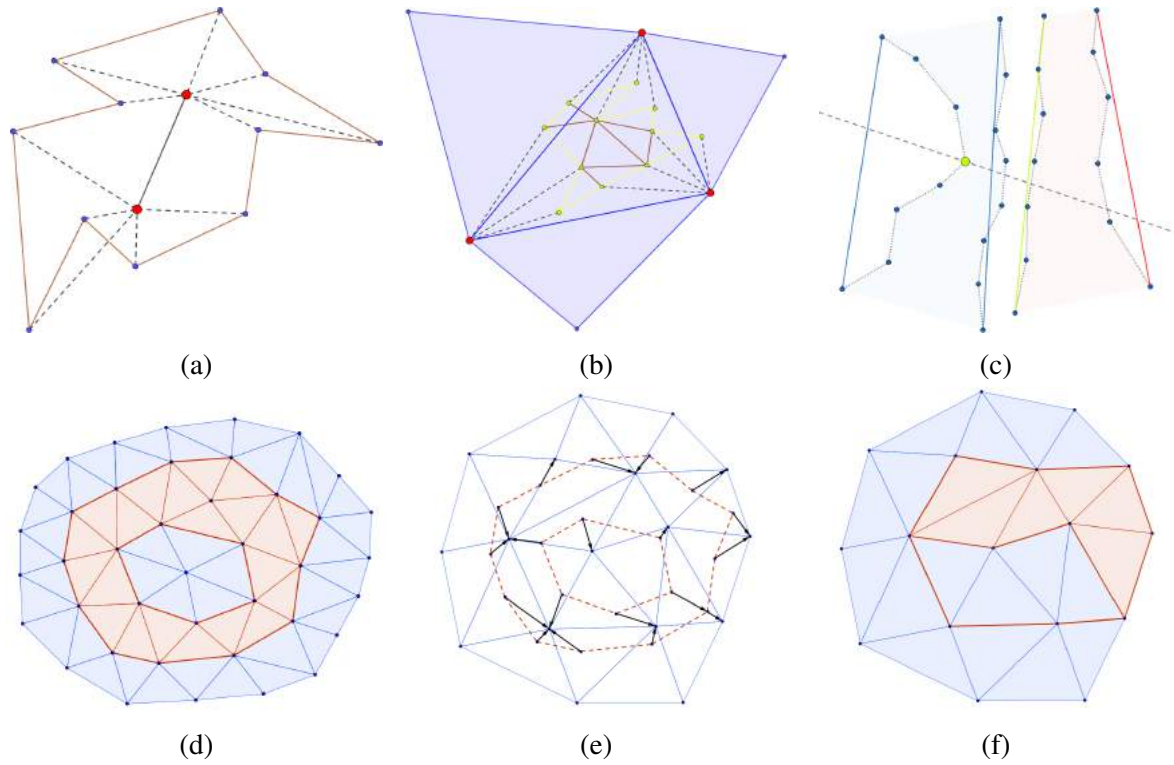


Figure 2.13: *In the first row, (a) the degeneracy, (b) distortion and (c) wrong projection issues. In the second row, the intersection issue, where (d) represents the original annotation, (e) an overlapping of it onto the target mesh and (f) is the result.*

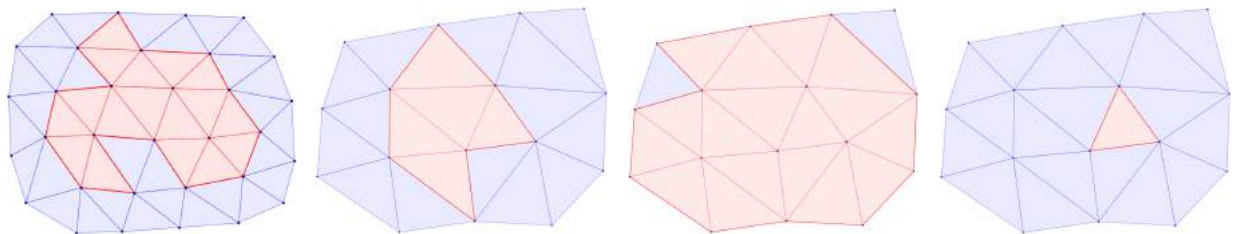


Figure 2.14: *An example of different vertices correspondence. From left to right: the original annotation, the partial overlapped, total enclosing and total enclosed results.*

shell is again a triangle mesh to triangle mesh case, while how to project the annotation from the surface to inner tetrahedra will require a more in-depth analysis and could be topic for future development.

## **2.5 Discussion**

In this Chapter, we presented our formalisation for the semantic part of the parametric template, providing definitions of annotations, attributes and relationships; in this work the mechanism of annotation represents the tool for bridging the gap between the semantics and its geometric counterpart. Once this bond is created, we need methods to guarantee that it remains unchanged after geometric changes. Therefore, we presented the annotation transfer approach whose aim is the persistence of annotations after resolution change.

Results show that the annotation transfer works pretty well even after severe changes in mesh resolution, until the size of the annotated feature is compatible with the level of detail of the target mesh. Especially in the downsampling case, the target mesh could be so coarse that the corresponding part actually fades, thus providing unreliable results. Future research will investigate how to determine the minimum resolution required to preserve an annotation as well as local refinement in order to cope with high distortions.

## Chapter 3

# Semantics enrichment through shape analysis

---

**Summary** In the previous Chapter, we have defined the annotation mechanism as the bridge to bind semantics to portions of geometry; semantics is given by an available formalisation of knowledge about the object class and is embedded in the geometry through manual annotation by a domain expert. Once the annotation takes place, we need tools to guarantee the association between geometric selections and knowledge remains consistent with geometric deformations, and we proposed a method for annotation persistence to cope with severe changes in mesh resolution. However, geometry also carries shape information that can be automatically extracted by shape analysis methods and enriches the semantics itself. Indeed, shape descriptors, from metric measurements to more complex characteristics like curvature, thickness or straightness, could provide precise numeric attributes, analyse the variability range of attributes within a class, facilitate expert’s study on the digital piece and support to find new domain knowledge, which could be put back in the loop.

In this Chapter, we present two approaches based on 3D shape analysis to support expert’s study and new contextual knowledge discovery in the archaeological research domain. Both applications focus on a collection of ancient clay statuettes, namely the “Ayia Irini small human idols” collection. In the first place, we define and deploy several shape descriptors to determine the level of expertise of the artisan (e.g., roundness, straightness) and study the production process (e.g., identify rules of proportions among parts). Secondly, we apply geometric based clustering to highlight the use of different moulds related to the workshop of production. Results highlight ranges in the variability of measures and stable proportions, which can be used to set attributes and relationships on annotations. The head clustering also suggests a further sub-classification of statuettes and in turn the use of a specific template for each sub-class.

---

As introduced in Section 1.1, the encoded semantics, initially derived from a shared knowledge formalisation or defined by a domain expert, can continuously be enriched by applying shape analysis tools for extracting measures or other geometric properties as well as relationships be-

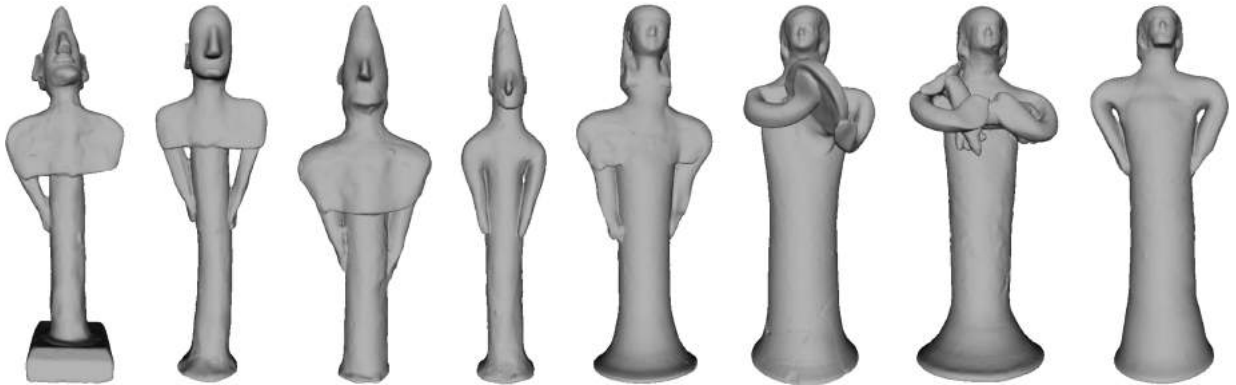


Figure 3.1: A subset of the statuettes from the Ayia-Irini collection

tween different parts (and/or their attributes). In fact, even if the knowledge of domain experts is very deep, there can be still room for investigation; with reference to the archaeology context, the work of researchers is indeed the analysis of the material to find out new undiscovered pieces of knowledge.

Shape analysis would be particularly useful especially in those fields where common *measures* and other geometric properties are important for classification purposes and/or for elaborating research hypothesis (e.g., in human physiognomy there are some well known proportions between measures taken from the eyes, the nose, etc. [ZJB<sup>+</sup>18]).

In particular, we focus on defining techniques to derive indicators for stylistic features of archaeological findings. These indicators can be used to:

1. derive quantitative attributes identifying a collection, to better specify the semantics and the constraints on the class [SMVS18, SVM<sup>+</sup>18] (see Section 3.2)
2. suggest new sub-grouping of archaeological collections, providing new hypothesis on production, provenance and reunification of pieces, and driving the selection of more specific sub-class templates [SVM<sup>+</sup>19] (see Section 3.3).

In the following, we first describe the archaeological collection representing our experimental case study and then the developed solutions for the two above problems, namely the quantitative descriptors to characterise the collection and the identification of a possible sub-classification of the statuettes based on manufacturing.



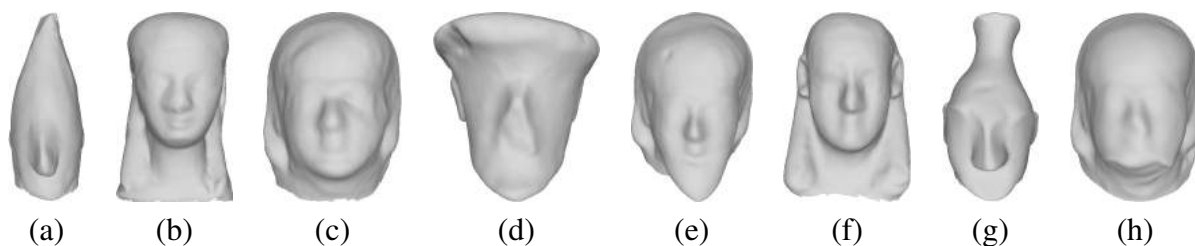


Figure 3.2: *The head of some artefacts in the data-set*

### 3.1 The Ayia Irini case study

Documentation in the CH sector is the process of recording information about collections; for reliability and ease of sharing, it is crucial that objects are documented consistently, using recognised standards, common record structure and homogeneous terminology. According to the UNESCO guidelines, writing a textual description of the shape and appearance of the object is highly recommended [Sti08], possibly including measures. The typical approach for archaeological analysis is mainly qualitative and, as such, object descriptions are rich but subjective and prone to ambiguities. Furthermore, even when some measures are reported, they are limited to global dimensions (e.g., height), are approximate (e.g., a curved surface measured by ruler) and are often not uniquely defined: for example, measuring the length of the arm of the statuette in Figure 3.4 involves choosing a distance metric (e.g., Euclidean versus geodesic) and the extreme points: where exactly does the arm begin? So, measurements by different archaeologists may be quite dissimilar, the exact procedure followed very hard to specify in textual notes.

Conversely, the quantitative approach is based on objective metrics to produce replicable results and, coupled with digital tools, can assist the qualitative one in different ways. Firstly, the use of a 3D model as a digital twin of the artefact allows pinpointing precise landmarks (measurement extrema, feature lines, areas of interest) and define them uniquely. Secondly, a variety of shape analysis tools can be applied to get quantitative descriptors directly in the digital domain, with no risk of damage to the real artefact. Finally, the study of shape descriptors allows for feature identification, automatic comparison of models (or parts of them) and classification.

Indeed, archaeologists are used to classify and categorise the material remains coming from past cultures [PB10]. It is a way to understand and interpret such material.

As Irving Rouse declares in his work on artefacts' classification in archaeology, 'analytic classification consists of forming successive series of classes, referring to different features of artefacts. Each class is characterised by one or more attributes which indicate a custom to which the artisan conformed, for example, a manufacturing technique, or a concept which he expressed in the artefacts, such as a design' [Rou60].

In this Chapter, digital shape analysis tools are suggested as a quantitative support for archae-

ological research and, in the long term, for the task of suggesting new possible classifications, with particular focus on the collections analysed in the framework of the GRAVITATE project [PWM<sup>+</sup>16]. Among the case studies under investigation in the project, the Ayia-Irini collection of small terracotta figurines seemed appropriate to demonstrate the beneficial support of quantitative analysis to answer different archaeological research questions [Vas16, Vas17].

The Ayia-Irini collection consists of almost 2000 votive clay statues and statuettes of different size, shape, and style found in an ancient Cypriot sanctuary. The site was brought to light by a small group of Swedish archaeologists during the Swedish Cyprus Expedition led by the archaeologist Ejnar Gjerstad [GLSW35] in the 20-th century. The collection includes medium and large human statues (man-size and over), animals, Minotaurs, horsemen, etc. After the excavation, the artefacts were divided between Sweden and Cyprus and currently they are conserved in five different museums.

Here, the focus is on the so called ‘small human idols’, mostly attributed to the Cypro-Archaic period (700-500 BC). Gjerstad’s classification of the ‘small human idols’ was based on visual investigation of criteria such as iconography (e.g., arm position, dress, headgear), technique (handmade, wheel-made, and moulded), clay colour, slip type (also in this case according to the colour: e.g., light/white slip), and decoration (e.g., black and red).

As Gjerstad himself admits, his first idea of classifying according to all noted typological differences (shape, technique, representation, etc.) would have created more than fifty types. For this reason, he decided to organise the material in less and bigger groups, deriving sixteen (16) groups that he named Types [GLSW35, pp. 785-786]. This classification presents ambiguities since many criteria overlap between Types. Some of these variations are explained by Gjerstad as advancement in the production technique, without taking other possibilities into consideration, such as artisans’ diverging ambition and skill.

For example, various experimental studies on ceramic prove that the technique used and the manufacture level, the form and the shape of the artefact, the weight and the size, and the thickness of the material, are some important parameters that could help us to interpret the artisanal signature of a workshop, or of an individual, and her/his level of expertise [DD86, pp. 19-36] [BS09] [Bot16, pp. 32-33].

Seventy years later, another interpretation of the Ayia-Irini material based on stylistic similarities suggested that stylistic features could indicate the provenance; such hypothesis brought to the creation of sub-groups pertinent to those different areas of production or imitation [Fou07, pp. 89-92, 127-132].

In both studies, even if greatly carried out and representative of interesting results, the analyses of the Ayia-Irini material followed a qualitative approach.

The study reported in this Chapter takes into account 103 statuettes of the collection, which were chosen from the different hosting institutions and digitised, both through laser scanning and

photogrammetric technique, in order to produce 3D replicas to be quantitatively analysed. All sampled statuettes represent male standing figures (up to 25 cm height maximum), sometimes holding weapons, animals, or music instruments. They belong to three of the Gjerstad ‘small human idols’ types: Type 5, 6 and 7. The Type 5 and 6 comprehend 60 handmade statuettes sharing similar characteristics, which make difficult a definite attribution to a class respect to another. Moreover, some items present common characteristics that make us hypothesise the presence of production patterns that could be meaningful for the identification of workshops or even different artisans’ hands. The 43 statuettes attributed to Type 7 are made by three integrated techniques (handmade, wheel-made and moulded) and share more defined characteristics, such as the use of several moulds for the production of the heads. Moreover, the archaeologists state that for the Type 7 statuettes 5 moulds were used to produce their heads but no attribution of how many and which statuettes correspond to each mould was done [Fou07, pp. 89-92, 127-132]. Also, some similar features of the hand-made heads suggest a possible same artisan’s production.

The traditional classification approach followed by the past studies on the Ayia-Irini material and the issues identified, led to some interesting archaeological questions:

- Can quantitative analysis support the qualitative one and suggest new, objective classifications of archaeological material?
- Is the presence of fixed measurements, ratios between the parts and geometric similarities expression of a “serial” production to be attributed to specific workshops or even artisans’ hands?
- Is it possible to identify different levels of expertise according to the uniformity of the statuettes’ clay width and other dimensions?
- How many moulds can be quantitatively identified in the sample? How many artefacts come from the same mould? Is it possible to identify a ‘chronological sequence’ through the comparison of the heads which seem to come from the same mould?

In the following, we describe our effort to identify the proper shape descriptors to support answering each of these questions. Firstly, we aim at identifying indicators for the artisan’s expertise and for the presence of fixed proportion in the production; secondly, we investigate new sub-groupings based on the use of a specific mould.

## 3.2 Quantitative attributes identifying artisan expertise and production process

As previously mentioned, the statuettes of the sample under study are diverse with respect to several factors, such as the production technique. With reference to Figure 3.1, the bodies of the statuettes are all tubular-shaped, but those on the left are produced by hand, while the remaining are obtained by wheel for the “body” part, by mould for the head and by hand for arms, neck and accessories (animals, musical instruments, etc.).

The production technique is particularly important to identify the proper dimensions to be measured and useful to possibly detect the presence of different hands/level of expertise. For example, an experienced craftsman is able to produce by hand a more rounded and straight body than a novice; in the same way, operating the wheel requires good skills to achieve homogeneous thickness of the clay.

The level of expertise of the craftsman’s hand could then be quantified using:

1. An estimation of *roundness* of the tubular part, for hand-made statuettes;
2. A measure of how much the *thickness* of the material varies over the tubular parts of the shape (thickness’ homogeneity);
3. A *straightness* measure for the tubular part (e.g., distance from the principal axis), for wheel-made statuettes.

In our publication [SVM<sup>+</sup>18], a set of slices of the shape extracted by virtually slicing the model with parallel, horizontal planes at increasing heights is analysed to estimate the above dimensions. The focus will be on the lower part of the body, by slicing the tubular part at different predefined heights.

The roundness is computed slice by slice as the average distance between the shape and its equivalent circle (circle centred on the centroid of the shape which has the same area of the slice). This can be computed even as the best-fitted circle using circular regression, however, the equivalent circle is computationally much lighter to compute and, since all the slices are sufficiently round, it is a good approximation of the fitted one (see Figure 3.3).

Wheel-made statuettes present an inner cavity (due to the production technique used) which, for scanning limitations, the digital counterpart represents only partially. However, the cavity can be detected in the lower slices, as an inner planar hole. Therefore, thickness is computed as the average distance between the inner and outer boundary of each slice, thickness variance is then extracted over the set of slices.

Finally, straightness depends on the definition of the axis of the statuette. A linear regression system for interpolating the barycenter of the slices can be set up, but this can be too sensitive to

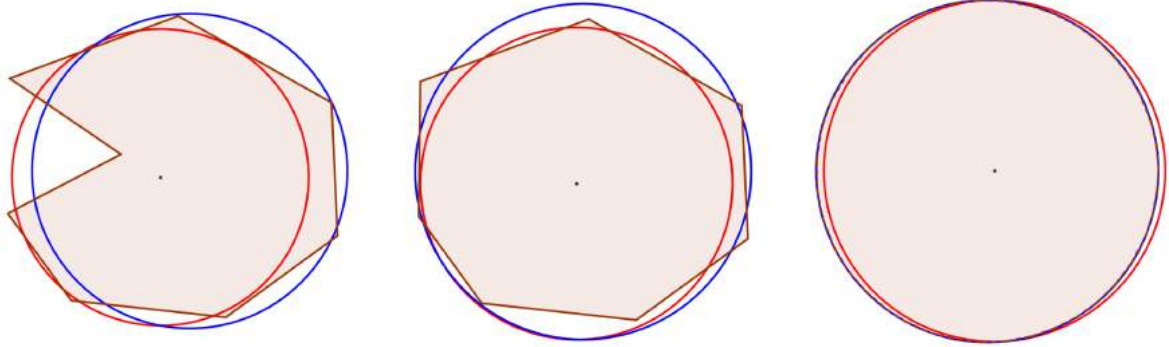


Figure 3.3: *Difference between fitted circle (blue one) and equivalent circle (red one) computed from the original shape (brown). As it can be seen, as the shape approaches that of a circle, the differences between fitted and equivalent one become thinner.*

noise (any imperfection over a slice can tilt the line). Another approach follows the extraction of the axis of rotational symmetry, but the features applied onto the tubular body (e.g., arms) affect its computation. For this reason, the focus will be on the tubular part, which exhibits a good symmetric structure and allows for a simple extraction of the axis (being a generalised cylinder). In particular, the choice fell on the vertical line passing through the barycenters' average. The average distance between the principal axis and the barycenters of the slices is computed and taken as a measure of straightness.

With the aim of answering the second research question, and so seeking fixed proportions between parts, we investigated the following entities:

1. Total size of the statuette;
2. Clay amount (volume);
3. Position and length of the arms;
4. Ratio between body parts measures.

In a first place, the overall volume of the object was considered as an indicator for both the total size and the clay amount<sup>1</sup>. This measure, however, can not be sufficiently precise due to the partial lack of the inner representation of the object.

<sup>1</sup>The volume  $V$  can be easily computed from a closed triangular mesh, following the divergence theorem of Gauss, as  $V = \sum_{i=1}^m \mathbf{c}_i \cdot \mathbf{n}_i \cdot a_i$ , where  $\mathbf{c}_i = \frac{1}{3}(\mathbf{v}_i^1 + \mathbf{v}_i^2 + \mathbf{v}_i^3)$  is the centroid of the  $i$ -th triangle,  $\mathbf{n}_i = (\mathbf{v}_i^2 - \mathbf{v}_i^1) \times (\mathbf{v}_i^3 - \mathbf{v}_i^1)$  its normal,  $a_i = \frac{1}{6} \|(\mathbf{v}_i^2 - \mathbf{v}_i^1) \times (\mathbf{v}_i^3 - \mathbf{v}_i^1)\|$  the corresponding area and  $\mathbf{v}_i^1, \mathbf{v}_i^2, \mathbf{v}_i^3$  are the vertices of the triangle (see [LK84]).



Figure 3.4: *The point-wise curvature on the statuette A.I. 245, following the colour bar on the right. Notice how arms and head joints are highlighted by curvature.*

Concerning the arm position and length, the precise identification of the junction area between arms and chest is needed, and this is not trivial. It is commonly understood that the statuettes were produced in separate pieces (the body, the arms, the accessories and the head), which were successively joined together. In many cases the junction of the arms or of the head is not clearly visible. However, sometimes a tiny relief of clay marks the joint location: indeed, in some statuettes, the mean curvature calculated on the mesh highlights that rim (see Figure 3.4).

After the identification of joints, it is possible to measure parts and look for fixed ratios among different statuettes or within the same statuette (e.g., different statuettes have arms with the same size, the head of a statuette is always  $\frac{1}{3}$  of the torso, and so forth).

### 3.2.1 Experiments and results

In order to optimise the above-mentioned analyses, automatic ways to identify regions where specific tools apply are needed: e.g., the tubular part of the body for measuring roundness, the head for mould identification. At a finer detail, there may also be the need to identify features' location (e.g., nose or ears).

The paradigm of slices is used to ideally segment the statuette into meaningful sub-parts, and use clustering to group similar slices (Figure 3.5). To this end, a set of descriptors are employed on slices extracted from the entire height of each statuette:

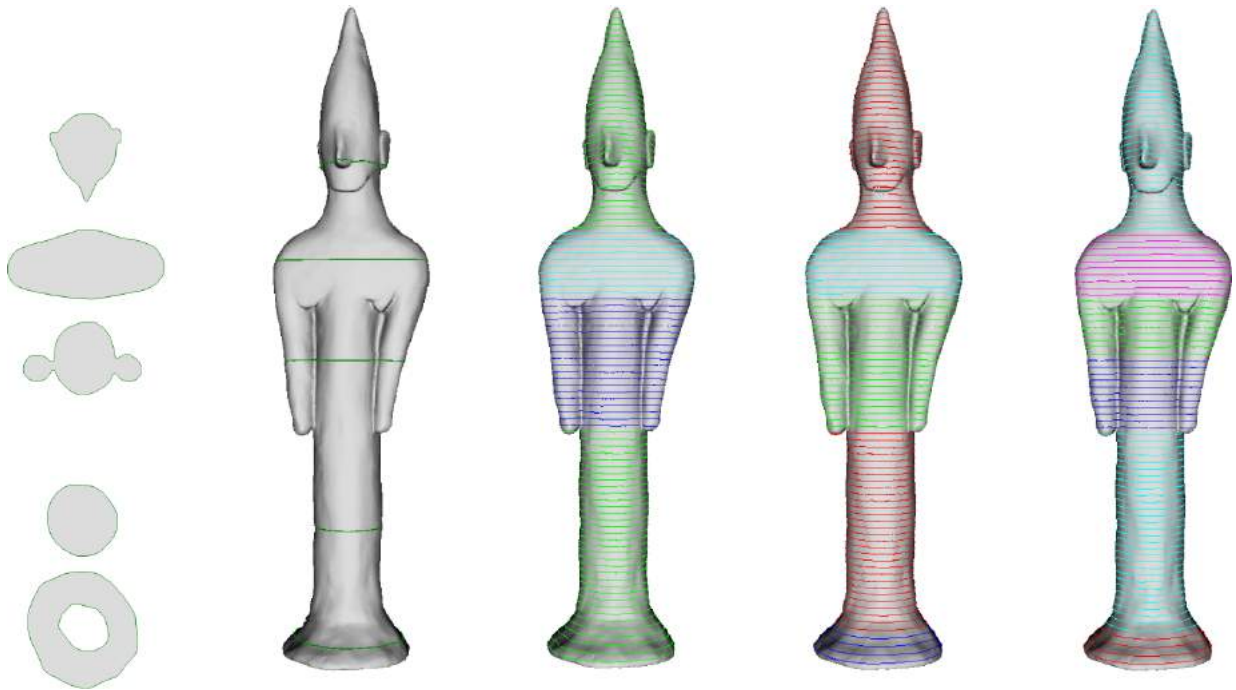


Figure 3.5: *Slice clustering. On the left, the A.I.130 statuette with a few slices highlighted; on the right, resulting clustering of the full set of slices into 3, 4 and 5 clusters, respectively.*

1. Elongation:  $1 - \frac{m}{M}$ , where  $m$  and  $M$  are the minor and major axis of the Minimal Bounding Rectangle (MBR) (the smallest oriented rectangle, in terms of area, containing the shape);
2. Solidity:  $\frac{A}{A_{CH}}$ , where  $A$  is the area of the shape and  $A_{CH}$  is the area of its *convex hull* (the smallest, in terms of area, convex shape enclosing the shape)
3. Compactness:  $\frac{4\pi A}{P^2}$ , where  $P$  is the perimeter of the shape;
4. Circular variance: error with respect to the equivalent circle;
5. Rectangular variance: error with respect to the MBR;
6. Euler number: number of connected components - number of holes;
7. Hole-area ratio:  $\frac{A}{A_H}$ , where  $A_H$  is the total area of all the holes.

Then, the clustering is obtained with a k-means approach [Mac67] (see Figure 3.5). The problem of identifying the proper number of clusters still remains.

In the second experiment, an attempt is made to obtain automatic recognition of wheel-made versus hand-made statuettes (archaeologists already have this classification, but it is useful for us to

automatically launch the proper descriptors depending on the case). The statuette clustering and a successive comparison of the performance of two descriptors applied to slices are performed:

- **Eccentricity:** the ratio between the axes of the MBR;
- **Roundness:** the average error between the shape and the circle which best fits the points of the shape.

These descriptors are properly applied to the lower body. Once the segmentation into meaningful parts will be satisfactorily solved, the slices will be automatically selected. At this stage, however, 100 slices from the lowest part of each statuette are considered (about  $\frac{2}{5}$  of the statue), then the average values and variance are taken: the variance for wheel-made ones is expected to be lower. Comparing the results with the ground-truth (archaeological classification) the classification based on eccentricity gets a F1-score of 0.6993 while roundness reaches the value of 0.8167, both with some manual tuning.

These first results are very promising: in fact, the classifier based on roundness reaches a very good score.

### 3.2.2 Inferring fixed proportions

Understanding if the statuettes were produced following rules about proportions between parts would be really important to enrich the set of constraints defining the belonging to a certain class (see Chapter 4), e.g., for classification purposes.

An example of such a study can be seen in Table 3.1, where 4 different measures are taken (using the tools presented in Chapter 5 from the statuettes of the Type 7 (see Figure 3.6):

- *total height*: the Euclidean distance between the base and the topmost point, with the statue having an explicit upright orientation;
- *base diameter*: the Euclidean distance between the two points on the base which are farthest away from each other (this is because the bases produced by the same artisan are likely to have more or less the same diameter in proportion with the rest of the statuette, both because of her/his skills and for stability purposes);
- *arm length*: this has been taken as the (approximated) geodesic measure between the hypothetical joint of the arm (near the neck) and the end of the “hand”, i.e., the point attached to the hold object or to the body, passing on the “exterior” of the arm itself (the idea is to understand if the arms could come from clay cylinders all of similar length);



Table 3.1: *Some measures taken from the Type 7 statuettes. The values in the first 4 rows are expressed in cm.*

	Average	Min	Max	Median	St. deviation
total height	20.03720	17.74204	25.23407	19.80382	1.35701
base diameter	5.79426	4.96149	6.84970	5.77085	1.03491
arm length	8.23578	6.78108	12.76251	8.04783	0.43048
head's height	4.35963	3.08510	7.34923	4.18244	0.86102
arm length/total height	41%	31%	52%	41%	4.7%
base diameter/total height	29%	23%	35%	29%	2.8%
head's height/total height	22%	15%	35%	21%	3.9%
arm length/base diameter	143%	117%	186%	140%	16.2%
arm length/head's height	194%	114%	281%	186%	37.6%
base diameter/head's height	137%	81%	194%	139%	25.6%

- *head's height*: the Euclidean distance between the joint of the head and the topmost point, with the statue having an explicit upright orientation (the idea is to understand if the chosen size of the different parts could depend on the employed mould).

To try and understand if the proportions are fixed, some ratios between the measures are computed and a statistical study applied, e.g., analysing the variability of the measures. Of course, one cannot expect the variability to be low, because even while using fixed proportions ancient artisans were limited by the tools and techniques of their epoch, but the first results in this direction are very promising. Indeed, the standard deviation reported in Table 3.1 suggests that the total height of the statue could present a fixed proportion with respect to head's height, arms' length and base diameter.

### 3.3 Quantitative approach to the sub-grouping of artefacts based on moulds

In this section, we present a preliminary study, published in [SVM<sup>+</sup>19], aiming at re-analysing the material already available for the Ayia-Irini collection utilising a quantitative approach [Vas16, Vas17, SVM<sup>+</sup>18] to complement the previous results with traditional methods on the same material [GLSW35] with the aim of identifying potential sub-groups for the Ayia Irini classification.

Note that this kind of approaches can be beneficial when dealing with classes with an high inner variability: indeed, in these cases sub-grouping the class can be a way for generating a more tight-fitting geometric template and a semantic template more rich in shared features and relationships (see Section 1.1).

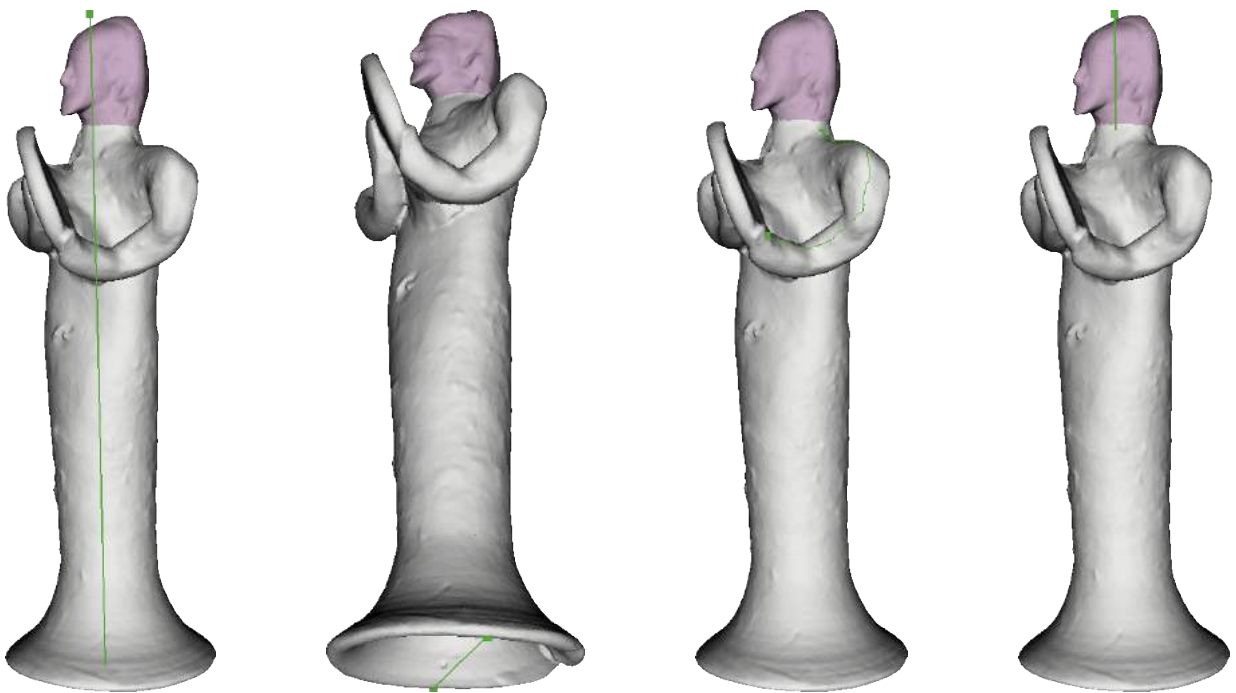


Figure 3.6: *The measures taken from the statuettes of Type 7, respectively total height, base diameter, arm length and head height.*

The 3D analysis in this study will try to answer the following archaeological questions: can characteristics in the hand-made heads been quantitatively measured to help us identify a production by the same artisan? How many moulds can be identified in the sample? How many and which artefacts come from the same mould?

In the following experiments, the focus is on the statuettes' heads and their mutual similarity to group figurines whose heads are most similar. Each output class should possibly represent figurines created by the same hand or produced with the same mould.

The procedure works as follows: firstly, the head part from the mesh of the whole figure is extracted. Then, two state-of-the-art algorithms are exploited to i) assess the similarity among faces and ii) cluster similar faces into classes. In this preliminary work, the heads were manually segmented using MeshLab [CCC<sup>+</sup>08].

Any mesh defect (e.g., isolated vertices, duplicated faces) were fixed using MeshLab and ReMesh [AF06]. Since apparently the moulds consisted only of a front half, the head back may bias the comparison. Therefore, faces are also extracted for Type 7 statuettes (i.e., those with moulded faces, wheel-made body and hand-made arms and accessories) by removing the head's back using MeshLab.

### 3.3.1 Similarity assessment

MeshSIFT has been chosen and employed to assess the similarity between pairs of head meshes. It was first introduced by Smeets et al. in 2013 for expression-invariant face recognition [SKDP13]. Albeit the invariance to face expression is not a requirement in this setting, this method has been employed because it is robust to incomplete data (heads and faces are represented as open meshes). Moreover, the method has proven to achieve a good recognition rate in international contests like the “SHREC ’11: Face Scans”[VJD<sup>+</sup>11]; an open implementation in MATLAB is available online [CDJF19]. In brief, MeshSIFT computes correspondences between salient points on the two meshes, the number of matched pairs being the estimate for how similar the two meshes are (further details can be found in the original article).

Comparing  $n$  heads in pairs produces a  $n \times n$  matrix  $\mathbf{S}$  of integers such that  $\mathbf{S}_{i,j}$  contains the similarity value  $s_{ij}$  between face  $i$  and face  $j$ .

The maximum values by row and by column lay on the diagonal (self-similarity). The range of similarity values for each head/face differs a lot: in this experiment, the self-similarity value ranged between 1466 for the model with inventory A.I.1249 and 160 for A.I.1295. Furthermore, the clustering algorithm employed for the next step needs distances among elements rather than similarities.

Therefore, the similarity scores for each model (with the maximum being the self-similarity value) are normalised in the range  $[0, 1]$ ; then, data are reversed into matrix  $\mathbf{D}$  to represent the distance between two heads as  $d_{ij} = 1 - s_{ij}$ .

### 3.3.2 Head clustering

To cluster heads based on their normalised distance, DBSCAN [EK SX96] has been applied to the first three components of the eigenvectors extracted from  $\mathbf{D}$  (this triplet being also used as coordinates to plot heads in the score space). DBSCAN is one of the most used and cited clustering methods: it is designed to discover clusters of arbitrary shape in noisy applications (e.g., see Figure 3.7). We refer to the original article for details. The main idea is the following:

A point  $q$  is *density-reachable* from a point  $p$  if their distance is less than a specified threshold  $\epsilon$  and if  $p$  is surrounded by a sufficient number of points. In this way, the *density-connectivity* can be stated as: two points  $p$  and  $q$  are density connected if there exists a point  $o$  such that both  $p$  and  $q$  are density-reachable from  $o$  (the latter definition guarantees that the points on the boundary of the clusters are grouped with those in the interior).

So, any cluster in the database will satisfy the following properties:

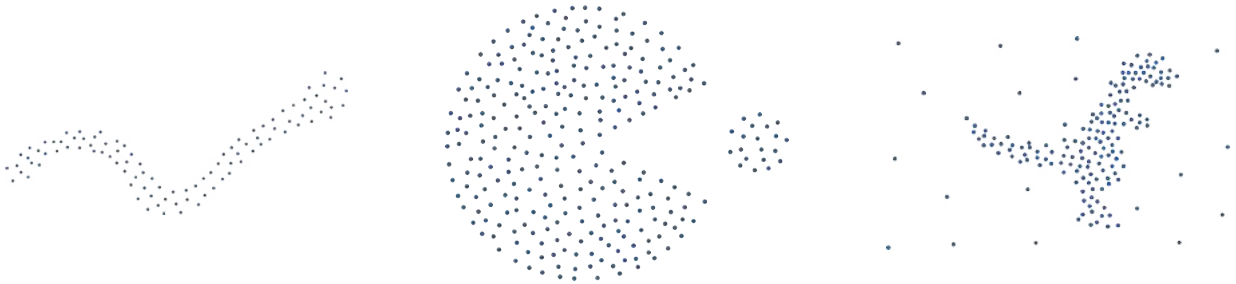


Figure 3.7: *Some examples of clusters of points that are detected by DBSCAN.*

1. Each pair of points in the cluster are mutually density-connected;
2. If a point in a cluster is density connected to another point, that point is part of the same cluster.

The DBSCAN algorithm (we employed the MATLAB R2019a implementation) requires two parameters: the minimum number of points to define a cluster (we set it to 1) and the value of the threshold  $\epsilon$ , which relates to the density of clusters and affects the clustering granularity. Higher values of  $\epsilon$  determine few, huge clusters; conversely, a smaller  $\epsilon$  will produce more, smaller sets. If  $\epsilon$  is not set, the algorithm self-estimates this parameter.

Different values for  $\epsilon$  has been tested. Since values are normalised in the range  $[0,1]$ , the maximum distance between points in the same cluster is arbitrarily fixed to be below the 10% of the maximum distance (so 0.1). A sort of *average distance* inside the clusters was also used, evaluated by extracting the  $k$  nearest neighbours (where  $k$  is the expected size of the cluster) for each point and then computing the average of the mean distance between them. The knn-search algorithm provided in [FBF77] and available on MATLAB has been used. In the following, we will refer to this value as  $knn-\epsilon$ . Finally, the self-calculated  $\epsilon$  provided by DBSCAN was also tested.

In the following, we will present the results achieved using the  $knn-\epsilon$  threshold (details can be found in [SVM<sup>+</sup>19]).

### 3.3.3 Experiments

Three experiments to analyse and interpret the manufacturing of the statuettes for classification purposes have been performed: in the first two, moulded and hand-made heads were analysed separately while in the last one the same procedure was applied to whole data-set.

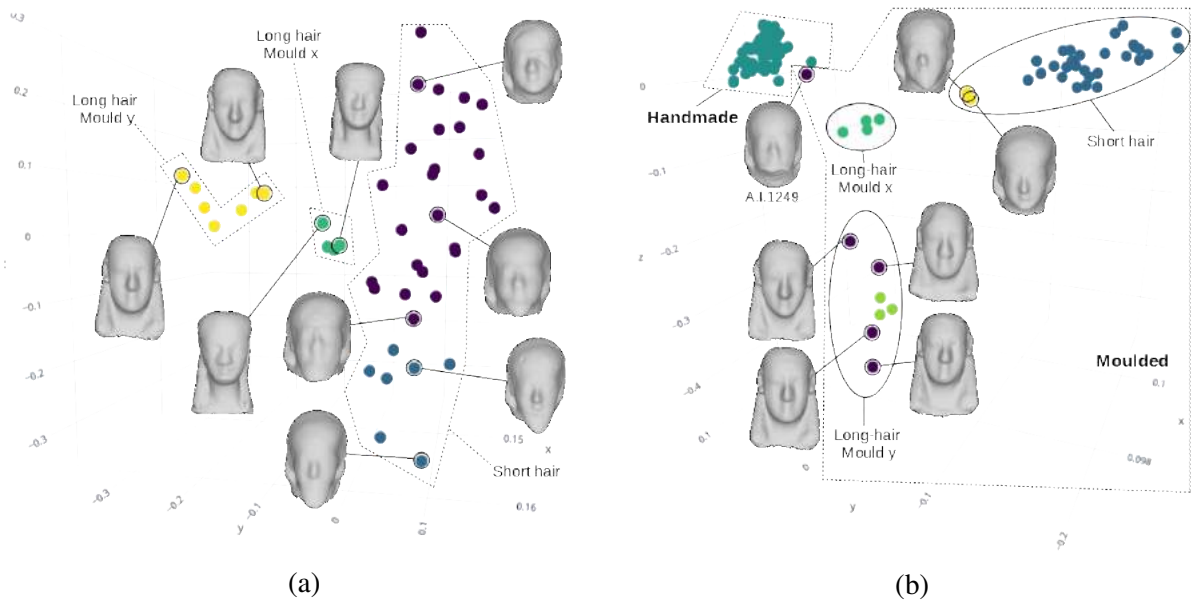


Figure 3.8: On the left, the results of the clustering (using the  $knn-\epsilon$  threshold) of all the moulded faces; on the right, the results of the clustering (using the  $knn-\epsilon$  threshold) on the whole data-set.

### Intra-class clustering: moulded faces

The first experiment was focused on the 43 moulded heads of the Type 7 statuettes, to estimate how many moulds were used for their production and which statuettes were produced with each mould. In this test only faces were taken into account (after back side of the head removal).

The clustering highlights two main classes, namely #1 “short hair” and #2 “long hair” heads, according to their most apparent characteristic. Also, though testing different values of  $\epsilon$ , the stable and constant presence of two separate and well definite clusters within the “long hair” heads group has been noticed. With reference to Figure 3.8a, using  $knn-\epsilon$  as threshold highlights two subgroups (in yellow and green) in the “long hair” class, and other two (in blue and violet) in the “short hair” class. The result suggests that the “long hair” heads might come from two different moulds, while the “short hair” heads appear very close to each other, suggesting a production from the same mould but, at the same time, with a distinction.

Indeed, 13 heads out of the 32 in the “short hair” class show a small piece of clay representing a beard, manually added after moulding.

Moreover, at finer clustering levels (i.e., at smaller values of  $\epsilon$ ) the subdivision between “heads with beard” and “heads without beard” within the “short hair” class emerges, but two sub-classes are not always precisely delineated, possibly because the added handmade beards bias the comparison.

Therefore, the experiment on the “*short hair*” subset has been repeated with three different settings:

- analysing only the 18 faces that have no added beards (“heads without beards”);
- analysing all the 32 faces after manually removing the beards from the 14 (“heads with beard”);
- analysing all the 32 faces after manually removing the beards from the 14 “heads with beard” and the chin from the 18 “heads without beard”

The first experiment was done to check if any subdivision within the “short hair” faces “without beard” appeared. With  $knn-\epsilon$  threshold, one main group was created, with only three external items. Only a further quantitative comparison among the faces could clarify if these deviations, rather than indicating different moulds, could be due to mould degradation, different pressure on the clay, or degradation of the artefacts.

The aim of the second experiment was to check if any new clustering could have been identified. The results show a definite subdivision between faces “without beard” and faces once “with beard”. The reason could be that modified faces present a lack and therefore the recognition happened only on the superior part, creating a group *per se*.

So, the test has been repeated eliminating the chin, bearded or not, from the 32 faces, to make the group homogeneous.

The third experiment produced a big group with some small other clusters, which can be possibly introduced by the erosion and/or by different pressure on the material during the production.

### **Intra-class clustering: handmade heads**

The same approach has been tested on the 60 handmade statuettes (Types 5 and 6). They were more troublesome, principally because of the peculiarities induced by the mere use of the hands for production. In this case, the whole head mesh has been analysed, since the artefacts are entirely handmade.

As mentioned, this group is characterised by a range of different features: “long hat” (Figure 3.2a), “short hat”, “truncated hat” (Figure 3.2g), “turban hat” (Figure 3.2d), and so forth, terms used by the archaeologists just to describe them qualitatively and therefore in a subjective and not homogeneous way. Changing the value of  $\epsilon$  during the experiment, it appears that certain subgroups are stable, such as the one composed of statuettes with “long hat” heads, with “short hat” heads or the one represented by statuettes with “truncated hat” heads. The latter group is particularly interesting for the interpretation of the production of the statuettes: a parallel

investigation on the same material showed the identification of traces of the very same type of decoration and pigments, suggesting a probable production by the same hand [VGH].

### **Inter-class clustering**

The aim of this last experiment was, firstly, to understand if the followed procedure is able to guess the classification of the heads into handmade and moulded and, secondly, to test whether other interesting information (i.e., a new classification) could be extracted from the whole dataset. Since the handmade heads are analysed integrally, the entire head was used for the moulded elements as well.

The unsupervised analysis indeed shows the principal subdivision of the artefacts between moulded and handmade statuettes according to their production technique (Figure 3.8b, where the handmade are the teal ones and the wheel made the remaining).

Moreover, previous tests identified the presence, within moulded statuettes, of a “short hair” (blue and yellow) and a “long hair” subgroups (lime and green), and at least two possible moulds, x and y in the “long hair” heads. It is worth to note that the group of heads created with the hypothetical mould y could be still divided in two further parts (the lime dots represent the heads produced with the mould y, while other points here demarcated as outliers - violet - should be part of the same group). In the future, these artefacts will be further analysed.

## **3.4 Limitations and Discussion**

In this Chapter, we have discussed the combined use of several shape descriptors to support the domain expert in his/her reasoning on a shape in order to reveal new contextual knowledge which, in turn, can be used to enrich the semantics itself. We focused on an example from the archaeological domain, where different descriptors were able to support quantitative attribute extraction and showed potential to help answering complex archaeological questions. However, the methodology as well as the single tools have a general value; adding more shape descriptors will provide the expert users more tools to use and combine in his/her study, to test hypotheses and document new findings.

The work described in this Chapter has some limitations. Firstly, the automatic recognition of parts have proved to be very challenging. The part segmentation using slices was not fully successful, and this hindered the composition of a fully automatic procedure for shape understanding. For instance, some measures to investigate the presence of fixed ratios among parts had to be taken manually, making measures prone to subjectivity. Similarly, the study on heads required some heads to be segmented manually. Concerning arm joint detection, we still have to investigate curvature patterns that could highlight clay artefacts due to hand pressure.

The next steps will regard the study of additional descriptors to identify new different subgroups. In the longer term, classifications either compliant with the archaeological ones, but even totally divergent are envisaged. Finally, it has been noticed that different statuettes have similar weights, so an attempt to understand if there is a connection based on total amount of clay used for the construction of different statuettes will be done. The question about if it is possible to estimate quite precisely the weight (the volume) of a statuette despite the lack of inner representation issue will be further investigated.



## Chapter 4

# From semantics to geometry through template deformation

---

**Summary** We approach now the core contribution of this thesis, that is, the geometric deformation guided by semantics. The subject of this Chapter is the definition of the constrained deformation technique, based on the bridge between geometry and semantics defined in Chapter 2; the implementation of the semantics-aware deformation based on constraints with examples and applications is instead presented in Chapter 5.

In the following, we present a review of the state of the art for surface deformation techniques, with particular focus on skinning. Then, we focus on cage-based deformation as the most suitable technique for the requirements of our application and discuss its potential and drawbacks.

We ground our approach to constrained deformation on a previous technique (ShapeUp [BDS<sup>+</sup>12]), which is designed for working directly on vertices positions, while our aim is to introduce cages to speed up the deformation. Therefore, we present our extension in order to manage constrained deformation through cage-based techniques. Finally, we discuss the pros and cons of this approach.

---

There are several levels at which the semantics can be exploited for acting on the geometric side of the template. Ideally, the structural information that can be found in the relationships graph (see subsection 2.3), along with annotation attributes, could be exploited for creating a dummy geometry from scratch, composing shape primitives (like a stick-man for the human shape). In our case, a reference geometry is available for a class of homogeneous shapes and we want to use the semantics to change this geometry, i.e., to deform it in such a way that the semantics is preserved. In other words, we can produce a new geometry acting not directly on the shape but rather on the transformations applied to it, by setting up a framework for semantically constraining the deformations. It is left to the domain expert to select those attributes and relationships that are crucial for an object to be considered part of the homogeneous class (e.g., what are the properties of a shape that defines whether it is a “teapot”?) and stating that they have to be

satisfied, at least up to a certain precision.

In this Chapter, we briefly present (a non exhaustive compendium on) the state-of-the-art of the deformation techniques; among these, we will apply in this work cage-based deformation (see [NS13]), because of the freedom it gives to the manipulation, its interactive speed and its simple implementation; then, we define the constraints, how we can mathematically formalise them exploiting some state-of-the-art techniques (e.g., [IMH05, BDS<sup>+</sup>12]) and how we can adjust the shape so that the constraints on attributes and relations are satisfied up to a certain error threshold or, otherwise, discard the deformation.

## 4.1 Review of surface deformation techniques

Computer animation has attracted a lot of attentions since the foundation of CG (a brief history of computer animation can be seen in <https://youtu.be/IhQp6eol76c>).

Historically, humans have always tried to achieve shape animation: for example, in ancient times drawings were made on pottery or other media so that the same character was pictured in different poses (e.g., animated vessel in Shahr-e Sukhteh [Wik20b] or Vitruvian Man of Leonardo da Vinci). It was not until the early 18th century that the first machines for producing animations were studied and produced (e.g., thaumatrope, phenakitoscope, zoetrope, flip-book, praxinoscope) exploiting the human brain's interpolation power to make images appear to move.

Then, in the early 20th century, the first cartoons made using the technique of stop-motion and drawn key-frames [SB85] were produced with the addition first of the sound (*Steamboat Willie* with Mickey Mouse) and then of colour with several techniques (the first successful one being Kinemacolor). After the foundation of new companies such as Walt Disney Studios and Warner Brothers Cartoons, cartoons production rose more and more until the first cases of animated TV series on prime-time (Hanna Barbera's *The Flintstones*). Afterwards, cartoon became ubiquitous.

In the latest 20th century, animation started the paradigm shift from pure analogic towards totally digital techniques, with *Toy Story* being the first fully computer-animated feature film released. While traditional animation was obtained by drawing key-frames that were shown with a certain frequency and with some (also drawn) in-betweens inserted to give the illusion of smoothness in the transition from a key-frame to the successive one, nowadays key-frames are commonly obtained by animators exploiting shape deformation techniques applied to 3D shapes, while interpolated in-betweens are generated by means of morphing techniques (e.g., [Ale02b, AFNS12]).

Here, we review the state of the art for interactive 3D shape deformation techniques, which will be a key feature of the proposed framework. All this techniques are based on the concept of *handle*, i.e., an intermediate entity that can be manipulated by the user to apply the desired deformation on the target shape. Basing on how the transmission is propagated from the handles to the target shape, they can be grouped into *variational*, *Free-Form Deformation (FFD)* and

*skinning* techniques.

### 4.1.1 Variational techniques

Commonly, the handles manipulated by the users in variational approaches are the vertices (called control points) of the shape representation. Once the user has moved one or more control points, the shape deforms consequently minimising an objective function (called *deformation energy*) [BS08], so that the deformation itself is posed as an optimisation problem, which typically requires iterative solvers [JDKL14].

Examples are techniques based on Radial Basis Functions [NFN00, BK05] and As-Rigid-As-Possible deformations [ACOL00, SA07].

This methods allow to obtain high-quality shape-preserving deformations, but they are very slow at the extent that they can not be used for interactive applications for deforming high resolution shapes; other drawbacks are the lack of means for changing the topology of the shape and for performing global shape changes. Since our purpose is to provide an interactive application to users, we will not use this kind of techniques directly (as can be seen in Section 4.2.2, we will re-introduce variational techniques as means for including *constraints* in the deformation, but in such a way that they will work on a reduced representation with respect to the mesh to be deformed - the corresponding *cage* - thus reducing the weight of computations).

### 4.1.2 Free-form deformation

Firstly introduced by the pioneering work of Sederberg and Parry [SP86], these techniques base on the definition of a control lattice, whose nodes can be manipulated by the user without any kind of constraint on the target shape, apart for it to be enclosed by the lattice.

This kind of methods are widely-used in commercial software (such as Autodesk 3Ds Max and Maya [3DS20, May20]) for providing smooth deformations and preserving the volume of the skin [MJBF02].

Using FFD, a complex shape can be deformed by positioning the control vertices of the coarse control grid. A more general extension of FFD (Extended FFD) was later presented by [Coq90]. Moreover, Hsu et al. [HHK92] provided a method that allows the user to control the FFDs by manipulating the object directly and, finally, the method in [AB97] uses an independent deformation function to provide a more flexible FFD while defining constraints on the preservation of the shape's volume.

Although FFD techniques are really simple to be implemented and extremely efficient, the difference in shape and resolution between the target shape and the control lattice greatly reduce

the amount of control allowed to the user, while the indirectness of the deformation propagation often leads to difficulties in predicting the result of the manipulation (FFD act on the *volume* into which the shape is embedded, rather than on the shape itself) [RF17].

To try and reduce the issues of these methods, cage-based deformation techniques have been introduced as a specialisation of this concept.

### 4.1.3 Skinning

Skinning is the process of controlling deformations of a given object using a set of deformation primitives. A typical example of an object often subjected to skinning is the body of a virtual character. In this case, the deformation primitives are rigid transformations associated with bones of an animation skeleton, but other kind of primitives are also possible (e.g., cages).

Given this broad definition, even the variational techniques fall within the class of skinning techniques so, according to [JDKL14], we will refer in the following only to the so called direct methods. Direct methods compute the resulting deformations using closed-form expressions, i.e., without any numerical optimisation. They are often very fast and trivially parallelisable, which makes them particularly attractive for interactive, real-time applications and GPU implementations.

#### Skeleton-based deformation techniques

This kind of techniques exploit the concept of *control skeleton* as a handle for manipulating the target shape.

The concept of skeleton is relatively simple and intuitive: in nature, vertebrate beings, like mammals, reptiles, birds, etc., have a inner structure called skeleton, which drives and controls their motion. Analogously, in computer animation, a control skeleton is an inner structure composed by *bones* and *joints* connected by bones, through which the user controls the character movements defining a set of poses.

There have been several attempts (see [TDS<sup>+</sup>16]) in defining how to automatically extract such skeletons from the target shape, mainly grouped into “medial-axis” based techniques (e.g., [WP00, WP02a, PFW<sup>+</sup>03]), where a force field is used to shrink the shape until it become a medial surface that is simplified to obtain a *curve skeleton* later sampled to generate the discrete control skeleton, and segmentation based approaches (e.g., [DMMT<sup>+</sup>07, CCXS11, dBA16]) that first segment the target shape into meaningful parts and then extract one or more line segments from each of them that are later joined.

Notice that not all the shapes allow to define a control skeleton (e.g., see Figure 4.1) at least in some parts, or anyhow the definition of such handles would be hardly usable (what use would



Figure 4.1: *Some shapes on which it is difficult to define a control skeleton, for example on the “head” of the jellyfish or on the whole muffin or, at last, in fragmented shapes like that of a statue on the right (images respectively from [open3dmodel.com](http://open3dmodel.com), [hum3d.com](http://hum3d.com) and from the STARC repository of the Cyprus Institute [public.cyi.ac.cy/starcRepo](http://public.cyi.ac.cy/starcRepo)).*

it be to have a control skeleton for a car shape?). So, control skeletons are better suited for the deformation of the so called *articulated* shapes, i.e., shapes having articulated parts.

In the following, we present a list of techniques for defining the propagation of the deformation from the control skeleton to the target shape.

**Linear blend skinning** Linear Blend Skinning (LBS), also known as skeleton-subspace deformation, (single-weight-)enveloping, or matrix-palette skinning, is the basic and most well known algorithm for direct skeletal shape deformation, defining a de-facto standard in videogames and animation communities.

It is difficult to trace the roots of LBS. Some of the early ideas appeared in the pioneering works [NM82] and [MTLT89]. Perhaps the first paper that gives an exact mathematical description of LBS is due to [LCF00], but it only mentions the algorithm is well-known and implemented in commercial software packages [JDKL14].

Linear blend skinning assumes the following input data:

- Rest pose shape, typically represented as a polygon mesh. The mesh connectivity is assumed to be constant, i.e., only vertex positions will change during deformations. We denote the rest-pose vertices as  $\mathbf{v}_1, \dots, \mathbf{v}_n \in \mathbb{R}_3$ . It is often convenient to assume that  $\mathbf{v}_i$  are in fact  $\mathbb{R}_4$  vectors with the last coordinate equal to one (homogeneous coordinates).
- Bone transformations, represented using a list of matrices  $\mathbf{T}_1, \dots, \mathbf{T}_m \in \mathbb{R}_{3 \times 4}$ . The matrices  $\mathbf{T}_i$  can be conveniently defined using an animation skeleton; in this case they correspond to spatial transformations aligning the rest pose of bone  $i$  with its current (animated)

pose.

- **Skinning weights.** Each vertex  $v_i$  of the shape has an associated set of weights  $w_{i,1}, \dots, w_{i,m} \in \mathbb{R}$ , where every weight  $w_{i,j}$  describes the amount of influence of bone  $j$  on vertex  $i$ . Common requirements are  $w_{i,j} \geq 0$  and  $w_{i,1} + \dots + w_{i,m} = 1$  (partition of unity).

LBS computes deformed vertex positions  $\mathbf{v}'_i$  according to the following formula:

$$\mathbf{v}'_i = \sum_{j=1}^m w_{i,j} \cdot \mathbf{T}_j \cdot \mathbf{v}_i = \left( \sum_{j=1}^m w_{i,j} \cdot \mathbf{T}_j \right) \mathbf{v}_i \quad (4.1)$$

Notice that, while this formula is referred to bone transformations  $\mathbf{T}_j$  applied to a control skeleton, the same approach applies to different types of handles by simply changing the definition of the deformation primitives (e.g., cage vertices' position in space).

LBS works very well when the blended transformations  $\mathbf{T}_j$  do not differ significantly in their rotation components. Despite its fast and straightforward implementation, LBS suffers from some visible artefacts when the joint is rotated of more than  $90^\circ$ . In a rotating joint, the skin is expected to rotate around the joint as well, maintaining the volume. However, the linear model interpolates skin vertices positions linearly between the bones, shrinking the nearby volume. This is a very well known issue of the method which is called “candy-wrapper” artefact, that depends on the fact that a linear combination of rotations is no longer a rotation [Ale02a]. Geometrically, this is a consequence of the fact that the Lie group of 3D transformations,  $SO(3)$ , is not a linear (or “flat”) space, but a curved manifold [JDKL14].

Later on, this issue has been first reduced by the use of more deformation weights in the so-called multi-linear methods and then totally removed moving to non-linear methods).

**Multi-linear blend skinning** One of the main issues of LBS and, at the same time, one of its main advantages, is the simplicity of the propagation scheme. It starts from the assumption that the only entities that will change during the deformation are the transformations  $\mathbf{T}_0, \mathbf{T}_1, \dots, \mathbf{T}_m \in \mathbb{R}^{3 \times 4}$ , which means that the LBS scheme is a linear function which accepts as input a set of transformations ( $12 \times m$  scalars, since each transformation is a  $3 \times 4$  matrix) and produces in output the deformed positions ( $3 \times n$  scalars).

Now, while LBS associate only one weight for each vertex-bone pair, it is possible to derive a more general skinning model by adding more weights (with a maximum of 36 weights) associating a weight to each entry of the transformation matrices ( $3 \times 4$  matrix = 12 entries) for each spatial coordinate of the vertices ( $3$  coordinates  $\times$  12 entries in a matrix = 36 possible weights). We refer to [JDKL14] for the details. Currently, no work employs all 36 weights to the best of our knowledge.

In [WP02b], a first extension of this model (Multi-weight enveloping) has been presented for trying to reduce the issues of LBS by simply changing the last matrix assigning *twelve* different weights for each vertex-bone couple  $(i, j)$ . This approach, while using twelve times the number of weights with respect to LBS, still is not the most general approach possible. However, it is able to solve the candy-wrapper artefact while, on the other hand, loses the very intuitive interpretation of the single weight associated to each vertex-bone pair: this leads to difficulties in designing good weights for the deformation, a problem that, always in [WP02b], has been solved by extracting them from a set of examples. Unfortunately, these examples are not always available.

Another possibility, that has been proposed in [MMG06], is Animation Space, an approach for reducing the number of weights per vertex-bone couple to 4, while still enforcing the invariance to world-space rotation and translation. However, Animation Space has the same limitation of the previous approach, so that even in [MMG06] the weights are learned from examples.

**Non-linear blend skinning** While concretely solving the candy-wrapper artefact (even if keeping sometimes some shrinking of the volume), multi-linear methods defines weights with a very unintuitive nature, at the extent that typical multi-linear methods learns the weights from examples.

Another non-linear approach tries to bypass the candy-wrapper artefact by taking into account the fact that the rotations that are being blended in LBS are in the Lie group of 3D transformations  $SO(3)$  which is a curved manifold [JDKL14]. In particular, common non-linear methods employ the concept of quaternion (or, better, unit quaternion, i.e., a quatenion  $\mathbf{q} : \|\mathbf{q}\| = 1$ ) as a tool for defining rotations. This is because, unlike matrices which are the most commonly used tool for expressing rotations, the set  $Q_1$  of unit quaternions offer the so called *double cover* property, i.e.,  $Q_1$  covers  $SO(3)$  twice, so that the quaternions  $\mathbf{q}$  and  $-\mathbf{q}$  represent exactly the same rotation matrix [Han06]. This allows to distinguish between  $360^\circ$  and  $0^\circ$  rotations, thus not shrinking the volume applying huge rotations. However, in the first version a  $720^\circ$  is again equal to  $0^\circ$  and this issue has been solved generalising the algorithm to more than two rotations in [BF01] by means of an iterative procedure, or avoided using linear combination of unit quaternions in [Hej04] and [Kv05].

These early non-linear methods all suffer from the bad handling of the translations: they try to handle rotations (defined with unit quaternions) and translations (defined with translation vectors) separately and, in this way, introduce artefacts [KCvO08]: indeed, splitting a rigid transformation into a rotational and a translational component, a specific pivot point (which by default is set near the object's centre of mass) is defined as centre of rotation.

In Dual Quaternion Skinning (DQS) [KCvO07, KCvO08], a dual quaternion-based definition of rigid transformations is used to simultaneously represent rotations and translations, leading to better results with reference to the previous approaches but introducing another well-known

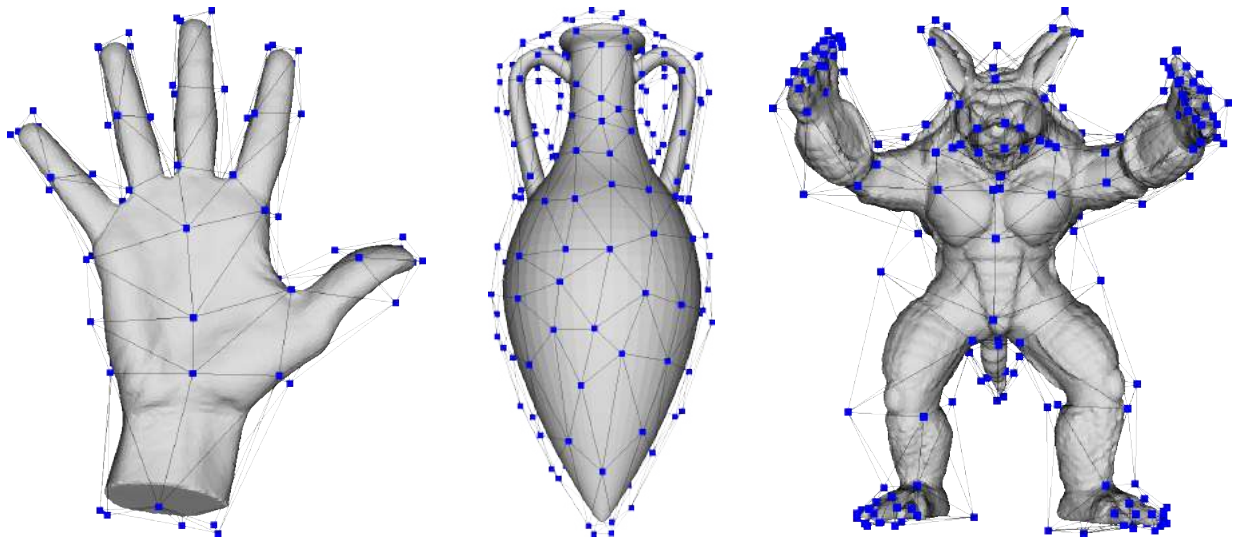


Figure 4.2: *Some shapes with the corresponding control cage.*

artefact, called *bulging artefact* [KS12, KCGF14], while at the same time not allowing non-rigid transformations. The solution of these issues inspired several research efforts from different perspectives, going from the combination of LBS and DQS (e.g., [KS12, OBP<sup>+</sup>13]) to the definition of other kinds of deformation primitives (e.g., [SF98, JBPS11]).

For further information on the presented techniques or a broader coverage of the state-of-the-art in skeleton-based skinning we refer the reader to [JDKL14] and [RF17].

### Cage-based deformation

While skeleton-based skinning techniques have become a de-facto standard in the animation community, deformations required in other application contexts may be better obtained with the employment of cages.

Let us denote a cage  $\mathcal{C}$  any closed polygonal mesh that envelops another polygonal mesh  $\mathcal{M}$  to be deformed (see Figure 4.2). The cage is usually a simplified version of the mesh  $\mathcal{M}$  and contains much less vertices. This tool finds several application contexts, such as the collision detection and in general everywhere a coarser version of a mesh could be useful e.g., for reducing the computational complexity.

Here, however, we will focus on the usage of cages for manipulating the shape of objects. This kind of manipulation is based on the usage of the vertices of  $\mathcal{C}$  as handles, i.e, the user selects and moves the vertices of the cage, and the model mesh will deform accordingly. Cage-based deformation techniques base on the concept of Generalized Barycentric Coordinates (GBC), which give means for defining the value of a certain function over a point in space (and so even



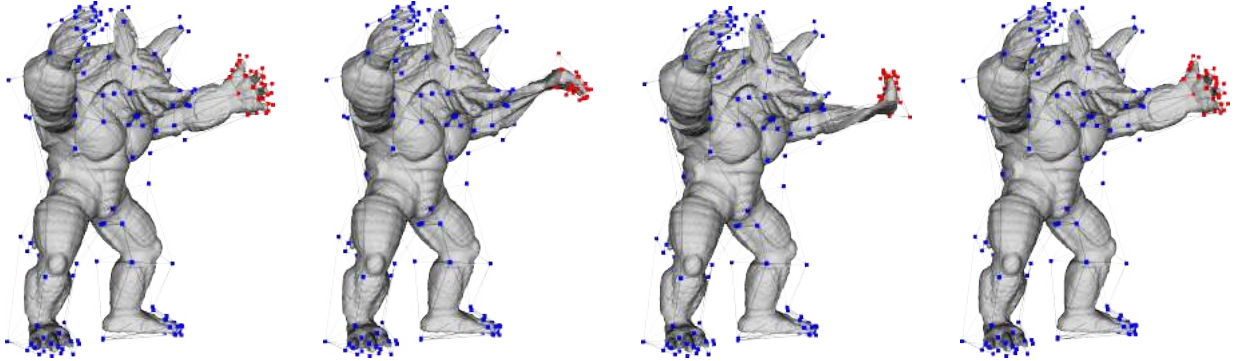


Figure 4.3: *The candy wrapper artefact in cage-based deformation.*

over the model mesh vertices) in terms of a linear combination of some control points (cage vertices) [NS13]:

$$f(\mathbf{v}_i) = \frac{\sum_{j=1}^m w_{ij} f(\mathbf{c}_j)}{\sum_{k=1}^m w_{ik}}, \quad (4.2)$$

These kind of techniques falls into the LBS group and inherit from it pros and cons (e.g., candy-wrapper artefact - see Figure 4.3). The (modified) formula for the computation of the deformed mesh vertices positions  $\mathbf{v}'_i \in \mathcal{V}_{\mathcal{M}}$  is computed as:

$$\mathbf{v}'_i = \sum_{j=1}^m b_{i,j} \cdot \mathbf{c}_j \quad (4.3)$$

where  $b_{i,j} = \frac{w_{ij}}{\sum_{k=1}^m w_{ik}}$  is the deformation weight associated to the couple composed of the mesh vertex  $\mathbf{v}_i$  and the cage vertex  $\mathbf{c}_j$ . Indeed, with reference to equation 4.2, taking as function to be interpolated the position in space of a vertex, i.e.,  $f(\mathbf{x}) = \mathbf{x}$ , equation 4.3 would be the resulting formula. A deeper comparison between GBC application and LBS is presented in [JDKL14].

Notice that equation 4.2 can be seen in matrix form as

$$\mathbf{V}_{\mathcal{M}} = \mathbf{B} \cdot \mathbf{V}_{\mathcal{C}} \quad (4.4)$$

where  $\mathbf{V}_{\mathcal{M}}$  is the  $n \times 3$  matrix obtained stacking the position in the space of the mesh vertices and  $\mathbf{V}_{\mathcal{C}}$  is the  $m \times 3$  matrix obtained in the same way for the cage vertices. This will be useful in subsection 4.2.2 and in subsection 5.2.3.

In the following, we briefly present techniques for (semi-)automatically generating cages and the properties of GBC and their different formalisation.

**Cage generation** The optimal cage size in terms of vertex cardinality is given by a trade-off between the number of Degrees of Freedom (DoF) available to the user for the deformation and the manipulation and computational complexity. A fine cage (e.g., having the same number of vertices of  $\mathcal{M}$ ) will give the user a lot of control points allowing very detailed deformations, which, however, are more complex to perform: she/he should be skilled in order to manage many control points and obtain the desired shape. Moreover, this will increase considerably the computational time so that possibly a standard hardware will not suffice to achieve an interactive deformation. Conversely, a cage with few points enables a coarser control on the deformation but allows the modification to be much more intuitive and a faster computation.

In brief, a “good” cage should have the following properties [CLM<sup>+</sup>19]:

- tight envelopment of the original model without either intersecting it, or self-intersecting;
- its control points should be close to the parts of the model one would like to deform or bend;
- be coarse enough to be easily manipulated, yet fine enough to capture the necessary details, thus it might need variable resolution;
- endow the symmetries present in the model.

There have been several attempts on the automatic generation of cages, each one with pros and cons, which can be roughly grouped into:

- *Template based* methods [YCSZ13, JZvdP<sup>+</sup>08] generate new cages using a set of cage templates with predefined topologies. This allow reuse of cage parts but requires a well defined set of topologies and, even in that case, it is almost impossible to generate tight-fitting cages for complex shapes.
- *Simplification based* methods [BCWG09, DLM11, XZG13, SVJ15] follow a resample-and-offset approach, which is very simple and quite fast, but can introduce cage self-intersections and not always guarantee that the mesh is inside the cage.
- *Bounding shape based* methods [XLG09, XLG12, XLX14] obtain the cages by iteratively refining the Oriented Bounding Box (OBB) of the shape. This can cause cages to be bad fitting and, because of the extraction of the OBB (repeated many times in some methods), these approaches are inefficient and slow.
- *Interaction based* approaches [CF14, LD17, CLM<sup>+</sup>19] where a certain level of interaction with the user is required to generate the cage. In [CF14] and [CLM<sup>+</sup>19], a starting skeleton is required as a guideline for the generation. While implicitly encoding the structure of the shape to be deformed (especially if the skeleton has been generated taking into

account the segmentation in parts of the shape), these methods are quite fast and require little interaction with the user, but they can be used only if the shape to be deformed allows the definition of a skeleton, partly losing the advantages of using a cage: typically shapes allowing the definition of a skeleton are better deformed using the latter handling abstraction, even if there are some exceptions (e.g., cases in which the scaling of parts of the shape is required in addition to the articulation - see [CTL<sup>+</sup>20]). The method defined in [LD17], instead, generate cages starting from some user-specified cut slides: while allowing the creation of tight fitting and high genus cages, this method requires huge efforts and a certain level of skills from the user.

In this thesis, we employed a very simple resample-and-offset approach (details can be found in Appendix A). This, of course, can give very bad results on complex shapes (e.g., self-intersections, loose-fitting cage, etc.) but we decided to employ it because of its simplicity. We leave to future works the definition of a semantics-based technique, where a cage will be built with more control points in proximity to *features* (annotations) of the object and less in more generic areas, while enforcing that some of the defined relationships reflect on the generated cage (e.g., sets of control points influencing two symmetric areas should be symmetric all the same). We believe that this approach will produce a cage respecting all (or at least a good percentage) of the desiderata expressed above (note that the *perfect* cage is not something that can be built a priori, but depends on the specific deformation required by the user).

**Barycentric Coordinates and their generalisation** Barycentric Coordinates (BC) were first introduced in 1827 by Möbius [Mö27]. Their basic philosophy can be explained with an example: suppose that we have a line segment  $l$  delimited by two points  $\mathbf{x}_1$  and  $\mathbf{x}_2$  (see Figure 4.4a). In the example, the purpose is to derive a colour shading onto  $l$  passing from a point to another (interpolating the colour function  $c(\cdot)$  of  $\mathbf{x}_1$  and  $\mathbf{x}_2$  onto the line segment). To do this, a BC function for each control point is defined, namely  $w_1$  for  $\mathbf{x}_1$  and  $w_2$  for  $\mathbf{x}_2$ , as:

$$w_1(t) = \frac{l_2}{l_1+l_2} \quad w_2(t) = \frac{l_1}{l_1+l_2}$$

Where  $t$  is the position over the line segment and  $l_1$  and  $l_2$  its distances, respectively, from  $\mathbf{x}_1$  and  $\mathbf{x}_2$ . Finally, the colour value at each point over the line segment can be derived as:

$$c(t) = w_1(t)c(\mathbf{x}_1) + w_2(t)c(\mathbf{x}_2)$$

This simple and yet powerful tool has been extended by Möbius himself to any kind of triangle. The key is to use the area ratio rather than the distance one (see Figure 4.4(b)). So, any function values  $\phi(\mathbf{x}_1)$ ,  $\phi(\mathbf{x}_2)$ ,  $\phi(\mathbf{x}_3)$  can be interpolated inside the triangle as:

$$\phi(\mathbf{t}) = w_1\phi(\mathbf{x}_1) + w_2\phi(\mathbf{x}_2) + w_3\phi(\mathbf{x}_3)$$

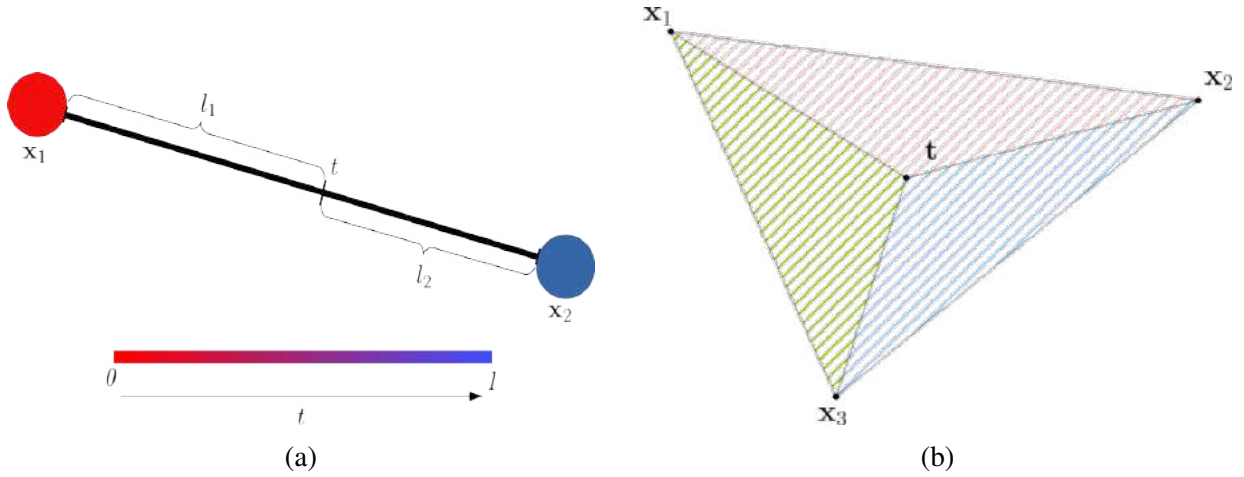


Figure 4.4: *The Barycentric Coordinates: on the left the colour gradient application, on the right the sub areas of a triangle given a point  $t$ .*

$$w_i = \frac{a_i}{a_1 + a_2 + a_3} \quad a_i = \text{area}(\mathbf{x}_{i+1}, \mathbf{x}_{i+2}, \mathbf{t}), \forall i \in \mathcal{Z}_3$$

Later, BC were developed in many works [Flo97, Flo03, FHK06, MBLD02, HF06]. Here, however, we will focus on the linking between a cage and its interior. Let  $\mathbf{c}_1, \dots, \mathbf{c}_m \in \mathbb{R}^3$  be a set of control points which are the vertices of a closed control cage, and let  $\Omega$  be the domain bounded by the cage. The goal is to find a function  $w_j : \Omega \rightarrow \mathbb{R}$  for each  $\mathbf{c}_j$ , such that  $\{w_1(\mathbf{x}), \dots, w_m(\mathbf{x})\}$  is a set of GBC of  $\mathbf{x} \in \Omega$  with respect to the control points  $\mathbf{c}_j$ . These coordinate functions are used for interpolating function values  $f(\mathbf{c}_1), \dots, f(\mathbf{c}_m)$ , given at the control points over the interior of  $\Omega$  by 4.2. In this thesis, we will use both  $w_{ij}$  and  $w(\mathbf{x})_j$  for referring to the same entity, i.e., a value associated to a couple composed of a point in the deformation domain and a cage vertex.

With application to shape deformation, GBC should satisfy certain properties to allow high-quality results. Here is a list of properties that are commonly considered crucial [ZDL<sup>+</sup>14]:

1. Reproduction:  $\sum_{j=0}^m w_j(\mathbf{x}) \cdot \mathbf{c}_j = \mathbf{x}, \forall \mathbf{x} \in \Omega$ ;
2. Partition of unity:  $\sum_{j=0}^m w_j(\mathbf{x}) = 1$ ;
3. Non-negativity:  $w_j(\mathbf{x}) \geq 0, \forall i$ ;
4. Lagrange property:  $w_j(\mathbf{c}_k) = \begin{cases} 0 & \text{if } j \neq k, \\ 1 & \text{otherwise;} \end{cases}$
5. Linearity: functions  $w_j$  are linear on cage edges and faces;

6. Smoothness: functions  $w_j$  vary smoothly on  $\Omega$ ;
7. Shape “preservation”: the deformation mapping should be at least quasi-conformal;
8. Locality: a control point only influences its nearby regions and a point  $\mathbf{x} \in \Omega$  is influenced by a small number of control points, i.e., the vector  $[w_1(\mathbf{t}), \dots, w_m(\mathbf{t})]$  is sparse.

The first methods that generalised the concept of BC to volumes enclosed by polygonal meshes were the simultaneous works from Floater [FKR05] and Ju et al. [JSW05] and worked exploiting the Mean Value theorem, from which their generalisation take the name. While allowing to compute the BC with a closed form expression, and so enabling parallel computation, and being very simple to implement, these generalisations have some fatal defects:

- Negative values: these coordinates do not satisfy point 3 in the list of desired properties, which leads to counter-intuitive deformations that result from inconsistent deformation directions between the control points and the cage interior;
- Global behaviour: these coordinates do not satisfy point 8 in the list of desired properties, thus leading to global modifications any time a cage vertex is moved, resulting in a non-intuitive deformation.

For these reasons, there have been quite a few works that tried to overcome the issues of Mean Value Coordinates (MVC):

- Harmonic Coordinates (HC) [JMD<sup>+</sup>07] were defined by finding the solution to the Laplace equation under well chosen boundary conditions: while this approach avoids the introduction of negative values and gives a first degree of control over the locality of the deformation, it doesn’t provide a closed form expression, thus not enabling parallel computation;
- Positive Mean Value Coordinates (PMVC) [LKCOL07] solve the negativity issue of MVC while reducing the computation time required by HC, but inherit the global behaviour from MVC and requires a Graphics Processing Unit (GPU) implementation;
- Green Coordinates (GC) [LLCO08] have the goal of providing quasi-conformal deformations and achieves it by taking into account the cage triangles and their normal in the computation of the deformed vertices positions: this method, as well as MVC provides a closed form expression while ensuring non-negativity, however it does not have a very local behaviour (even if still more local than MVC);
- Bounded Bi-harmonic Weights (BBW) [JBPS11] are not, strictly speaking, BC because they do not satisfy point 5 in the list of desired properties but they allow a complete integration between different types of handling abstractions, namely points, skeletons and cages, and allow a very local deformation. This method does not provide a closed form expression;

- Local Barycentric Coordinates (LBC) [ZDL<sup>+</sup>14, TDZ19] directly enforce local behaviour during deformation while satisfying linear precision. Again, this method do not provide a closed form expression.

Other notable generalisations of GBC have been defined in [HS08, MLS11, BLTD16, APH17, DCH20].

In this thesis, we employed MVC because of their simple implementation and the compatibility with the ShapeOp library (see subsection 4.9). However, in the future we plan to move to a more sophisticated technique, namely LBC, for obtaining more local deformations.

## 4.2 Constrained deformation

In the context of this thesis, constraints are rules that a shape should comply to belong to a class. To express this idea, the template has annotations, annotations share relations and have attributes (as formalised in Section 2.2) and, most importantly, the expert can specify constraints over them. Unconstrained relations and attributes can therefore be manipulated at will, while constrained entities are limited or completely fixed. To make an example, the 103 “small human idols” statuettes in the Ayia Irini terracotta collection [Vas17] have a height of 17 to 25 centimetres; the collection includes other, bigger figures with real human size, which the archaeologists have distinguished in a separate class.

In principle, the user can set constraints on attributes regardless their nature (qualitative and quantitative, textual and numeric, etc.); however, in this work we focus on constraints that can be checked and modified by geometric analysis algorithms. This includes “low level” constraints, directly related to geometric measurements and quantitative properties of the surface (e.g., area measure, relation of co-linearity), and “high level” constraints, which, even having a qualitative nature, can be defined and assessed through one or more geometric properties (e.g., the example of straight beard in sub-section 2.2.3). We refer to low level constraints as geometric, and to high level constraints as semantic constraints, because the set of specific geometric constraints corresponding to an high-level constraint will be transparent to the user, who will instead express the semantic constraint through its intuitive meaning.

Constraints can apply either to a single annotation or to multiple annotations. A single annotation can be constrained with respect to its position, shape and attributes (e.g., orientation, scaling, length), while a set of annotations can be constrained according to their mutual arrangement, relations or proportions (e.g., three points must stay aligned, or one line should be twice longer than another). Constraints are, again, set by the user, who can express them through an ad-hoc constraint window in the system Graphical User Interface (GUI) (see Appendix A).

Practically speaking, constraints (both low and high level) are implemented as sets of one or

more constraints defined on the geometry. In this thesis, we employed and extended the ShapeOp library [DDB<sup>+</sup>15], which provides an optimisation engine to meet geometric constraints defined over point clouds.

### 4.2.1 The ShapeOp library for geometric constraints

The ShapeOp library (based on the ShapeUp approach [BDS<sup>+</sup>12]), is an open source, header only library for both static and dynamic geometry processing, using a unified framework for optimisation under constraints.

In ShapeOp, a constraint is defined on a set of points and concerns a collection of admissible positions or configurations for them. The library tries to obtain the best possible configuration for the set of constrained points by relying on an iterative two-steps minimisation approach.

Let  $C = \{C_1, \dots, C_m\}$ , be the set of defined constraints and  $o = |C|$ . Then, for each iteration, the two steps are:

1. *Local solve or projection*: the current configuration of points  $\mathbf{x}$  is projected on the closest position that satisfies each constraint singularly, obtaining a set of projected configurations:

$$P_i(\mathbf{x}) = \min_{\mathbf{y} \in C_i} \|\mathbf{y} - \mathbf{x}\|_2^2 \quad (4.5)$$

2. *Global solve*: a new configuration of points  $\mathbf{x}'$  is obtained by minimising the distances from the projected configuration collection  $P_i(\mathbf{x})$ . So this step solves:

$$\min \phi(\mathbf{x}) = \min \sum_{i=1}^o (w_i \cdot \|\mathbf{x}' - P_i(\mathbf{x})\|_2^2) \quad (4.6)$$

where  $\phi$  is called the *proximity function* and  $w_i$  is a weight used for controlling the importance of constraint  $C_i$ .

A simplified example of such a process is reported in Figure 4.5

The optimisation solver provided in the ShapeOp library is enriched with a set of built-in constraints and external forces for architectural geometry and physics simulation, that can be used without further implementation [BDS<sup>+</sup>12]. These include some constraints to require points co-linearity, co-planarity, co-circularity and co-sphericity, some others to restrict the strain applicable to edges, triangles and tetrahedrons, some constraints for area and volume preservation, Laplacian fairing and bending resistance. Further details on this constraints' formalisation can be found in [BDS<sup>+</sup>12]. Luckily, to extend the library with additional constraints it is sufficient to provide the necessary *projection* operation for the new constraints.

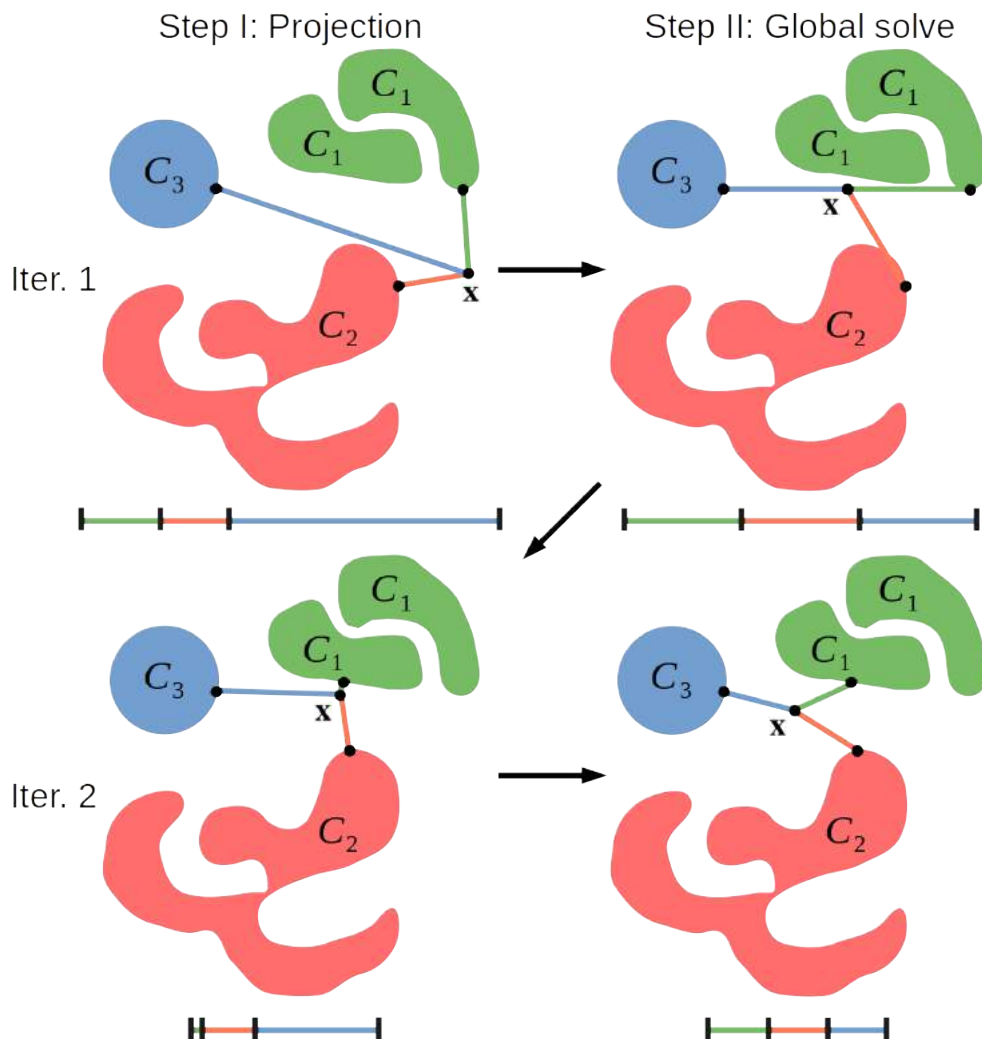


Figure 4.5: A simple example of optimisation of geometry configuration under 3 different and unrelated constraints  $C_1$ ,  $C_2$  and  $C_3$ . Step I:  $x$  is projected on the respective set of admissible positions obtaining the 3 black points; Step II: a new configuration is obtained by minimising the distance from the projected points.



Therefore, we enrich the ShapeOp library with new constraints and combine several constraints together in order to make it possible to express an high-level constraint intuitively, while implementing it through proper low-level geometric constraints (see Chapter 5).

Indeed, a domain expert could ignore what a Laplacian operator provokes to the surface, while this is a crucial ingredient to many deformations if the outcome surface is expected to look smooth. In other words, we are going to define macro-operations, whose outcome can be intuitively expressed, consisting of the application of a series of low level geometric constraints on the initial object, that will remain hidden to the user. The semantic constraints will likely let the user manipulate the shape without bothering of maintaining the essential features (either functional or stylistic) proper of its category.

## 4.2.2 ShapeOp extension for cage-based deformation

As already mentioned in the previous subsection, ShapeOp is designed for working directly with the position of points, which allow the approach to be general and usable both with point clouds, triangle and tetrahedral meshes.

Our aim, however, is to allow interactive constrained deformation, which requires to use low-resolution meshes or, rather, to speed-up the deformation using the techniques at the state-of-the-art (namely skinning - see 4.1.3).

Indeed, using skinning it is possible to deform the target shape by modifying the chosen handles, here cages, whose DoF are hugely fewer than those of the target shape. So, since ShapeOp uses vertices to define constraints and their positions for computing the transformations required by the constraints to be satisfied, we modified the library in order to make it work with the skinning handles rather than with the shape vertices directly.

Let  $\mathbf{V}_{\mathcal{M}}$  be the  $n \times 3$  matrix obtained stacking the position in space of all the vertices in  $\mathcal{M}$  and  $\mathbf{V}_{\mathcal{C}}$  the  $m \times 3$  matrix obtained in the same way with the cage vertices.

As can be seen in [BDS<sup>+</sup>12], if we define  $\mathbf{V}_i \subseteq \mathbf{V}_{\mathcal{M}}$  the vector of  $n_i$  vertices involved in the shape constraint  $C_i$ , then the proximity function  $\phi(\mathbf{x})$  can be re-formulated as

$$\phi(\mathbf{V}_{\mathcal{M}}) = \sum_{i=1}^o (w_i \cdot \|\mathbf{N}_i \cdot \mathbf{V}_i - P_i(\mathbf{N}_i \cdot \mathbf{V}_i)\|_2^2) \quad (4.7)$$

where  $\mathbf{N}_i$  is the *centring matrix*<sup>1</sup> that is used to centre the vertices in  $\mathbf{V}_i$  on their mean.

This formulation is possible because shape projections are invariant under translation. Equation

---

<sup>1</sup>The centring matrix is defined as  $\mathbf{N}_i = \mathbf{I}_{n_i} - \frac{1}{n_i} \cdot \mathbb{O}_{n_i}$ , where  $\mathbb{O}_{n_i}$  is the matrix of all ones of size  $n_i \times n_i$ .

4.7 can then be re-formulated by rewriting  $\phi(V_{\mathcal{M}})$  as:

$$\phi(\mathbf{V}_{\mathcal{M}}) = \|\mathbf{Q} \cdot \mathbf{V}_{\mathcal{M}} - \mathbf{p}\|_2^2 \quad (4.8)$$

where the matrix  $\mathbf{Q}$  combines all weighted mean-centred constraint vertices, and  $\mathbf{p}$  integrates all projections. The alternating optimisation scheme for each iteration then becomes:

1. For fixed  $\mathbf{V}_{\mathcal{M}}$ , compute the projection vector  $\mathbf{p}$ .
2. For fixed  $\mathbf{p}$ , solve the normal equations  $\mathbf{Q}^T \cdot \mathbf{Q} \cdot \mathbf{V}_{\mathcal{M}} = \mathbf{Q}^T \cdot \mathbf{p}$  for updating  $\mathbf{V}_{\mathcal{M}}$ .

Since  $\mathbf{Q}$  only depends on the shape constraints, the matrix  $\mathbf{Q}^T \cdot \mathbf{Q}$  can be pre-factorised using sparse Cholesky factorisation<sup>2</sup>, which hugely reduces computation time.

Now, substituting  $\mathbf{V}_{\mathcal{M}} = \mathbf{B} \cdot \mathbf{V}_C$ , where  $\mathbf{B}$  is the matrix of employed GBC, in equation 4.4, we obtain:

$$\begin{aligned} \mathbf{Q}^T \cdot \mathbf{Q} \cdot \mathbf{B} \cdot \mathbf{V}_C &= \mathbf{Q}^T \cdot \mathbf{p} \\ \mathbf{B}^T \cdot \mathbf{Q}^T \cdot \mathbf{Q} \cdot \mathbf{B} \cdot \mathbf{V}_C &= \mathbf{B}^T \cdot \mathbf{Q}^T \cdot \mathbf{p} \end{aligned} \quad (4.9)$$

Thus, solving for the position of the cage vertices in space, we obtain the configuration of the mesh vertices that better fits all the constraints. Notice that, again,  $\mathbf{Q}$  and  $\mathbf{B}$  do not depend on the unknowns, so the matrix  $\mathbf{B}^T \cdot \mathbf{Q}^T \cdot \mathbf{Q} \cdot \mathbf{B}$  can be pre-factorised using sparse Cholesky factorisation as well.

Beside the computation efficiency, another noticeable characteristic of this approach is that it follows the philosophy of skinning, i.e., it moves the biggest part of the computation time in pre-processing for allowing interactive speed during deformation. A comparison of the computation time required by ShapeOp and our extension can be viewed in Table 4.1.

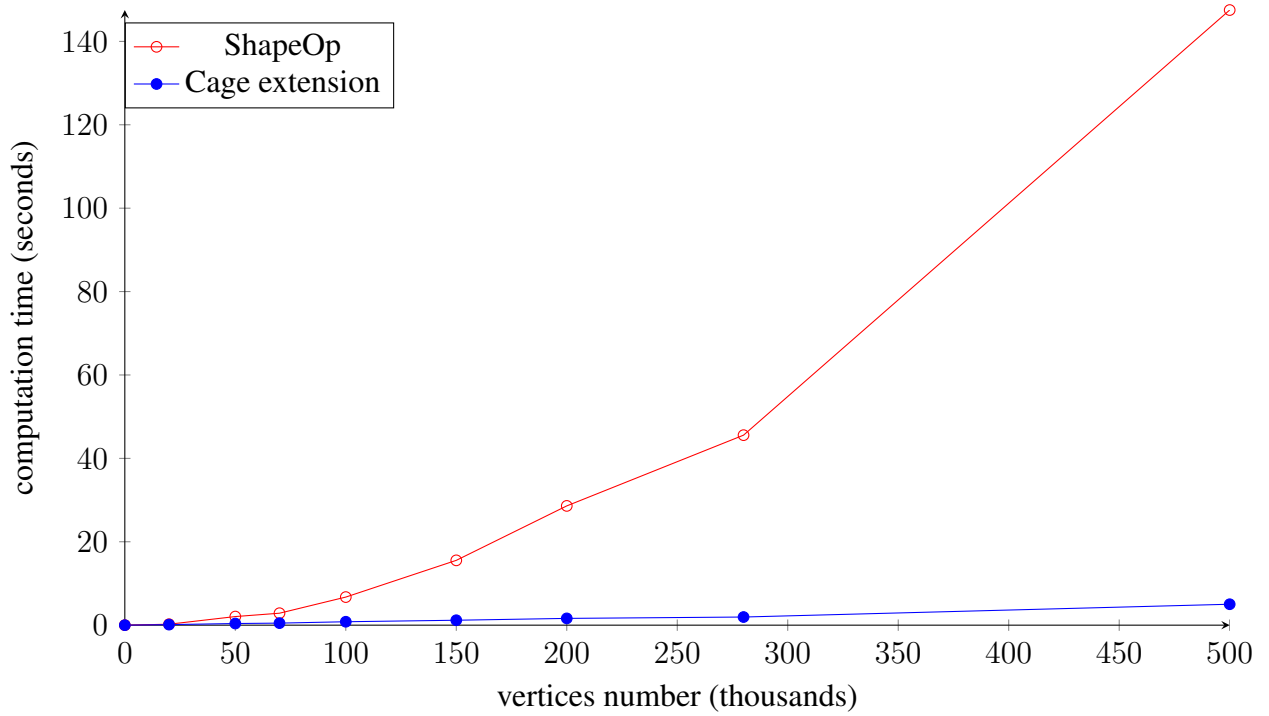
These advantages of course are not for free: first of all, the quality of results really depends on the characteristics of the employed GBC. This can be seen in Figure 4.6(d), where the optimisation made on the cage tries to make the red part planar but the ears and the tail tip are affected as well. This happens because of the employed GBC (MVC), which have a global behaviour. Furthermore, since the DoF are way less than those of the model mesh, it may happen that there is no possible configuration satisfying a constraint: again, in Figure 4.6(d) the red part is not totally planar, even if the ‘‘Plane’’ constraint is the only one imposed on the shape. Of course, this problem is even more severe when there are several constraints, possibly in contrast. Thus, the choice of the resolution and topology of the cage is extremely important and should be a trade-off between the computational complexity and the goodness of the deformation.

---

<sup>2</sup>The Cholesky factorisation of a symmetric and positive-definite matrix  $\mathbf{A}$  is a decomposition into  $\mathbf{A} = \mathbf{L} \cdot \mathbf{L}^T$ , where  $\mathbf{L}$  is a lower triangular matrix with real and positive diagonal entries, which is used for solving efficiently numerical problems. A symmetric matrix  $\mathbf{A}$  is positive definite if, for any vector  $\mathbf{x}$ , the product  $\mathbf{x}^T \cdot \mathbf{A} \cdot \mathbf{x}$  is positive.

Table 4.1: The timings required for the solution (with maximum number of iterations allowed set to 50) of the optimisation problem described in Figure 4.6 on a computer with 16GB DDR4 RAM and Intel® Core™ i7-7700HQ CPU. The same problem has been solved on meshes at higher and higher resolution, namely at 20, 50, 70, 100, 150, 200, 280 and 500 thousands of vertices resolutions. Values are expressed in seconds.

	20K	50K	70K	100K	150K	200K	280K	500K
ShapeOp	0.228	2.064	2.866	6.735	15.543	28.601	45.566	147.518
Cage extension	0.133	0.393	0.49	0.819	1.18	1.622	1.95	5.019



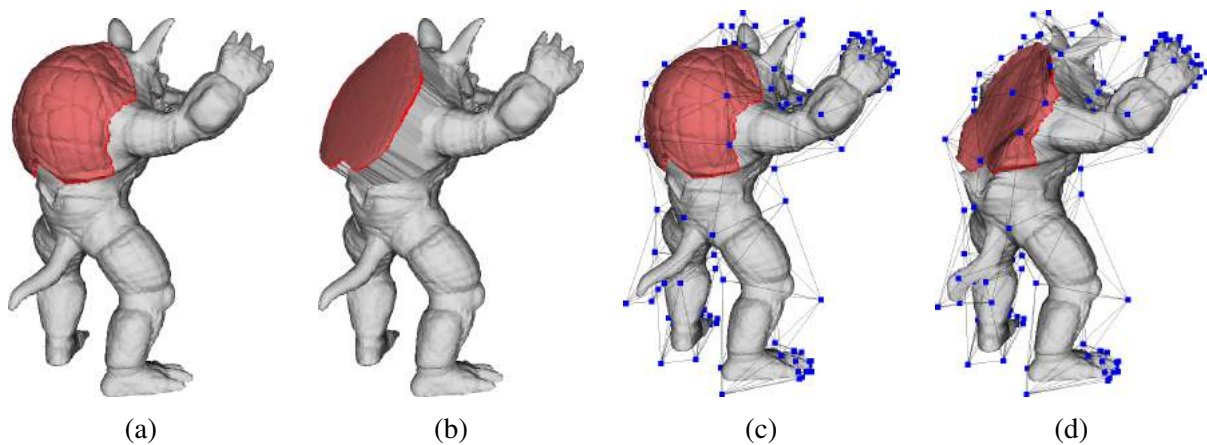


Figure 4.6: *Extension of ShapeOp for working with cages. Here a constraint imposing that all the vertices have to stay on the same plane (called “plane” constraint in the library) has been applied to the red part: (a) rest shape, (b) resulting shape after 2 iterations of the optimisation made directly on the model mesh, (c) rest shape with corresponding cage, (d) resulting shape after 2 iterations of the optimisation made on the cage.*

Finally, ShapeOp works with points for allowing an application to almost every kind of shape representations and this is one of its major pros but it introduces difficulties to work with every possible generalisation of BC, for example GC works by taking into account triangles during the deformation, so that working with vertices only does not work correctly.

### 4.2.3 Low-level and high-level constraints

As already stated in the previous subsections, ShapeOp is a library meant for geometry processing. Here however, we are trying to enforce geometric approaches with the semantics given by the formalisation of some domain expert’s knowledge. Thus, we introduced a bridge between geometry and semantics.

As introduced in Chapter 2, such a bridge is given by 3D part-based annotations, that define *parts* of the template shape and their attributes, and by the relationships among them. The idea is to let the domain expert define constraints over attributes and relationships, including “low level” constraints, that are directly related to geometric measurements and quantitative properties of the surface (e.g., area measure, relation of co-linearity), and “high level” constraints, which, even having a qualitative nature, can be defined and assessed through one or more geometric properties.

Our approach, then, is to allow the domain expert to specify what a certain relationship means in terms of geometric entities in a transparent way: she/he will not need to have a deep geometry

knowledge. For example, the user can require that the selected part “keeps its original shape”, and the system will call for the “Similarity” constraint in ShapeOp. Thus, we will provide a vocabulary of high-level constraints, representing intuitive concepts, already implemented as sets of low-level constraints.

Note that, since the manipulation of a shape here builds an optimisation problem, we can exploit the constraint-specific “error” (i.e., the residual of the corresponding cost function) as an index of constraint satisfaction extent. Such information can then be reported to the user, who could accept or discard the produced deformation, or we could exploit it to build a shape classification tool (see Sections 5.3 and 6.1). This, of course, requires a careful definition of how to extract the error measure so that it could be comparable with that of the other constraints. In the developed framework, we compute some statistics (namely average, median, min and max values) for the normalised errors (i.e., since the residual is given by the sum of residuals over a set of vertices, we divide the result by the number of vertices) that refer to low-level constraints regarding an high-level one and report them as well as the associated weight, so that the user can take an informed decision.

Of course, the meaningful high-level constraints are really context dependent and it would be unfeasible to define all the possible constraints. In this thesis, we present just some examples of constraints that are defined and implemented, extending ShapeOp, to test the potential of this approach in specific application domains: product design (see Section 5.2) and archaeological reconstruction (see Section 5.3). These regard the position of related annotation, the co-axiality of annotated parts and the proportions among numerical attributes of annotations; they are described, along with examples and results, in Chapter 5.

Anyhow, the vocabulary of constraints can be continuously enriched to provide new deformation constraints, to meet further application requirements.

### **4.3 Discussion**

In this Chapter, we presented the foundations for the design of the semantics-aware deformation. Firstly, we briefly reviewed the state of the art for surface deformation techniques, with particular focus on skinning. Then, we selected cage-based deformation as the best fitting technique for our requirements and gave means for imposing constraints on the cage-based deformation. We follow an optimisation approach and extend the ShapeOp library in order to work with cages.

We discussed the pros and cons of the approach, i.e., cages allow to reduce the computation time required by ShapeOp (which is a variational technique) but reduce at the same time the control on the reachable deformations, at the extent that some constraints may never be fully satisfied. We only implemented a simple cage generation tool within our system so far: as a future research, we think the semantics encoded in the template could drive the generation of an “optimal” cage,

where the number and position of cage vertices are determined by the annotation arrangement and characteristics. We believe that developing such a tool will heavily improve the achievable results, as thoroughly discussed in Section 6.1.

The definition of three high-level constraints and constrained deformation results for specific applications will be detailed in Section 5.2 and 5.3.

# Chapter 5

## Results and applications

---

**Summary** In this Chapter, we describe the results of semantics-aware deformation in two application scenarios, namely product design and cultural heritage. We have described in the previous Chapter the foundations of the optimisation system to compute constrained deformations, and we pointed out that more “semantic” constraints are needed to support the user with intuitive modelling tools. In particular, we tackle those semantic constraints that can be expressed as a combination of geometric ones. Semantic constraints maintain high level features of the object and have meaning in a specific context, therefore, they need to be designed with a specific application in mind.

To validate our system, we identified two application contexts where it is possible to interpret semantic constraints in terms of a proper combination of geometric ones. We defined and implemented a “same level” and a “structural continuity” constraint, aiming at maintaining a certain kind of alignment among annotations in the scenario of product customisation. Since collaborative design of products in virtual reality is gaining popularity, we investigated the feasibility of interactive cage-based deformation of annotated models. We present preliminary results obtained by exploiting semantics to support user interaction and facilitate selection of handles in VR.

In the context of cultural heritage, we focused on the intriguing problem of archaeological reconstruction from a few fragments. When just a few remains are found, a complete re-assembly is out of reach, but the expert is able to match in his/her mind the typical form of an object of the same type, provenance and style with the characteristics of the fragment, and make hypotheses about the original shape. As seen in Section 1.1, proportions have a crucial role in determining a style. In this context, we defined a “proportions” constraint, relating numerical attributes of annotations, which, applied to the template, can deform an ideal shape to “complete” a fragment maintaining coherent features and providing an hypothesis of virtual reconstruction.

---

In the following, we present the results of the semantics-aware modelling system in some concrete application scenarios. These applications are grouped with respect to the reference domain sector, namely product design (Section 5.2) and archaeological reconstruction (Section 5.3). We will begin with a brief description of the main functionalities of the implemented system and its

operational workflow. A more complete description of the GUI is given in Appendix A.

## 5.1 Operational workflow of the system

The current version of the software, available on GitHub [Sca20], provides all the features presented so far in the previous Chapters. An overview of the operational workflow and the main functionalities is depicted in Figure 5.1. After launch, the system starts in the initial state, where no entity is loaded in the application environment yet.

After loading a (geometric) template, the user can annotate it. We provide both a manual annotation functionality and an annotation load function. We support manual annotation providing a set of selection tools (lasso selection of closed surface patches, selection of points or triangles behind a user-drawn rectangle - either applied to visible points/triangles only or to all the points/triangles behind the rectangle - and selection of edges connecting user-picked points) and tagging, with a textual label and a colour. An annotated shape can be saved in an output file (the annotation file format that we designed is described in Appendix B) and loaded in future sessions.

Furthermore, annotations can be enriched by attributes. Quantitative attributes can be computed automatically by shape analysis tools as described in Chapter 3. We implemented three measuring tools, namely the ruler, the tape and the bounding measure. The ruler and tape provide Euclidean and approximate geodesic distance between successively picked points, whereas the bounding measure provides the distance between two clipping planes. Additionally, slicing along a direction and slice analysis functions (see Section 3.2) are available as well as slice clustering methods and OBB extraction. Semantics can be enriched in every state of the system.

Then, the cage can also be loaded in the system or generated by a provided simplification and offsetting function. We recall that the characteristics of the cage impact the quality of the resulting deformation and the time performance. Therefore, the user may want to select a proper cage resolution and is also allowed to edit cage vertices manually (e.g., to avoid self-intersections or to follow the template shape more closely). In the future, we plan to investigate a new cage generation optimised with respect to the annotations of the underlying mesh.

Once the cage is established, the barycentric coordinates that represent the connection between the cage and the template must be computed. The system implements the automatic generation of the Green Coordinates and the Mean Value coordinates (see Section 4.1). Barycentric coordinates can be saved and loaded in future sessions.

Cage and barycentric coordinates allow at this stage to perform unconstrained cage-based deformation. This can be done by selecting the cage vertices (or control points) to be manipulated and then translating them and/or rotating them around their barycenter. Combinations of these manipulations allow to define any kind of cage configuration. The deformation is propagated



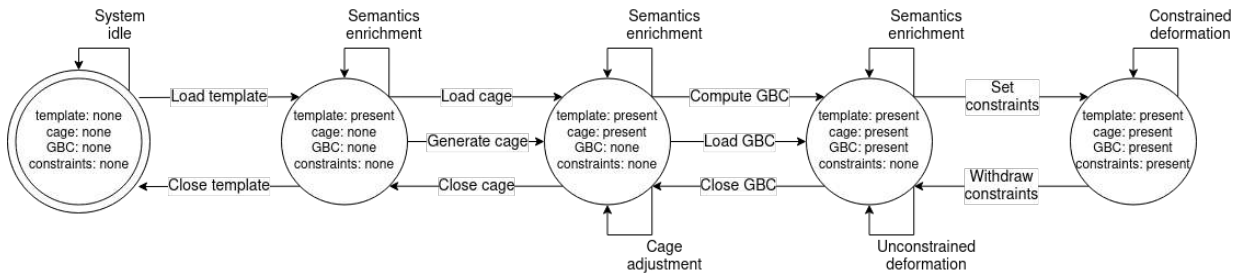


Figure 5.1: Overview of the system functionalities depicted as states, linked by arrows representing operations. Notice that the “Semantic enrichment” arrow groups the manual annotation, the loading of pre-defined annotations and the extraction of new knowledge through shape analysis techniques.

accordingly at a fixed frame-rate.

In order to achieve semantics-aware deformation, the user has now to define semantic constraints on annotation attributes and relations. The implemented semantic constraints can be set through the GUI: the user selects the annotations involved and the constraint to be applied with corresponding parameters, including an “importance” factor (weight), to set a priority in case of multiple conflicting constraints. This factor is used as weight into the ShapeOp minimisation for the corresponding geometric constraints and, as already anticipated in Section 4.2.3, can be used, together with the residual of the cost function of the corresponding constraints, as an indicator for accepting or rejecting a certain deformation.

We now go in the details of the developed semantic constraints in the two application contexts of Product design and Archaeological reconstruction. A more in depth description of the GUI can be found in Appendix A.

## 5.2 Product Design scenario

A natural application context for the proposed framework is the product design or CAD: in this specific application, expert users may want to reuse previously designed shapes as a starting base, which will be subject to manipulations under specific constraints until they are satisfied. Similarly, an end user, who is possibly not very skilled in 3D modelling, can use our framework to customise an item and have it fabbed.

In this section, we discuss the results in the context of constrained deformation of products under semantic constraints. We provide two high level, intuitive constraints that a non expert user can apply without bothering of the low level geometric constraints involved. In particular, we implemented a “same level” constraint applied to a series of stylistic features, and a “structural continuity” constraint, acting on structural relations among object components.



Figure 5.2: An all-around view of the “flowered teapot” model. Colours correspond to annotations (only region annotations: “body”, “base”, “spout”, “handle”, “knob”, five “flower”, plus the whole object annotated as “flowered teapot”).

### 5.2.1 Same “level” constraint

While working with shapes containing repetitive features (also called patterns) it is likely that the user wants to change the overall shape without however changing the spatial displacement of the features, i.e., the spatial relationship between the features should not change. As an example, we refer to the teapot in Figure 5.2 and to a precise transformation applied to it, i.e., the stretch of the shape upwards. Here, it is likely that the user wants all the flowers to keep at the same height and, moreover, not to change their original shape. The result of this transformation without constraints is depicted in Figure 5.3a

To provide this kind of high-level constraint to the user, we have followed the concepts introduced in Section 4.2, first of all by selecting some geometric constraints already available in ShapeOp and useful to the current purpose (details about the available ShapeOp constraints can be found in [BDS<sup>+</sup>12]) and then applying them to specific sets of mesh vertices, which may or may not coincide with the geometric selection of an annotation, as follows:

1. “Rigid” constraint, applied to the vertices of each “flower” annotation separately. The required behaviour is to keep the original spacial displacement (or relationship) of the sets of vertices, thus keeping the original shape of the flowers;
2. “Plane” constraint, applied to the set of “reference vertices” of the flowers. The idea is that for each flower, a reference vertex represents the position of the annotation in space<sup>1</sup>. So, the requirement is for these reference vertices to stay on the same plane;
3. “Laplacian Displacement” constraint: uses the neighbouring vertices (1-neighbourhood) in order to maintain the constrained vertices position with respect to them<sup>2</sup>; we applied the constraint to all the vertices of the mesh. So, we want the original shape of the teapot to

<sup>1</sup>There can be several definitions for the reference vertex depending on the nature of the feature, i.e., if it is symmetric or not, if it has an almost planar geometry or rather “enclose” a volume. Here, given the almost planar and symmetric geometry of the flowers, the reference vertex is taken by fitting a plane on each flower, projecting its associated vertices on the plane and taking the one nearest to the barycenter of the flower.

<sup>2</sup>The 1-neighbourhood of one vertex  $v$  is the set of vertices directly connected to  $v$  by an edge of the mesh.



Figure 5.3: (a) the unconstrained stretch upwards of the teapot, the result after the addition of the constraints.

be kept as much as possible without constraining the relative positions of all the vertices of the teapot.

Notice that, as is, the “Plane” constraint implemented into ShapeOp projects these vertices on a fitted plane and not to a user defined one. Since here we are interested in a specific plane, i.e., the one with normal pointing upwards and with lower distance from the reference vertices, we have extended the vocabulary of constraints defining a new one, that we called “Orientation” constraint<sup>3</sup>. The idea is to generalise the “Plane” constraint so that the constrained vertices will stay on a plane whose direction (here called  $\mathbf{n}$  is user-defined. In our case, we require the reference points to stay on a plane with normal pointing in the positive z direction (of course this may not be the case, see Figure 5.4d). The projection is defined by:

1. Centre the input points on their average position;
2. Compute the projected position of the input points  $\mathbf{v}_i$  on the defined plane as  $\mathbf{v}_i - \mathbf{n} \cdot (\mathbf{n} \cdot \mathbf{v}_i)$

Notice that the combination of constraints number 1, 2 and 4 (i.e., “Rigid”, “Orientation”) is supposed to keep the reference vertices on the plane with normal pointing upwards, so that they will stay at the same height, while the shape of the flowers is being preserved, thus answering

<sup>3</sup>The same one suggested in the documentation of the ShapeOp library as a possible extension

to the requirements of the user. Constraint number 3 is used just to reduce the distortions due to the deformation. The result of the same stretch as before with these constraints can be seen in Figure 5.3b, for a template mesh with ~60K vertices, without cage extension and with maximum number of iterations of the solver allowed set to 3 (it was necessary to obtain almost interactive speed).

As can be easily seen, some distortions have come up around the flowers. This is due to the fact that the “Laplacian Displacement” constraint is in contrast with the other constraints, so that reducing the distortion would require a higher weight for the “Laplacian Displacement” and would reduce the precision of the keeping of the original shape of the flowers and/or of the positioning on a plane of the reference vertices. Different experiments can be done for the fine tuning of these parameters but, anyhow, the result would be a trade-off between these issues, because the constrained deformation is posed as an optimisation problem (see Figure 5.4).

## 5.2.2 Structural “continuity” constraint

The general idea behind this constraint is that, sometimes, it is useful to require some “continuity” in terms of direction of parts.

With reference to the chair in Figure 5.5, if we suppose that the user wants to change the shape of the “arm” (green part) so that it is longer, we may think that such a modification would happen without changing the fact that the base of its “support” (violet part) has the same direction of the “leg” (orange part). This can be seen both as a structural and a style constraint.

To produce this deformation requires two steps:

1. Stretch of the “arm” (Figure 5.7a);
2. Rotation of the “support” so that the attachment to the arm falls in the same position.

Notice that, in the unconstrained version of the deformation (with result shown in Figure 5.7b) this deformation gives pretty bad results that have to be fixed with several steps of post-processing.

This kind of high-level constraint is implemented as a combination of two geometric constraints:

1. “Co-axiality” constraint: is a new constraint that we implemented to require that the axes of two parts coincide. This is applied to the base of the “support” of the “arm”.
2. “Laplacian” constraint: is a constraint already available in the ShapeOp library which follows more or less the same concept of the “Laplacian Displacement” constraint (see Same “level”), but requiring that the configuration of vertices reduce to 0 the local curvature. This is applied to the neighbourhood of all the vertices of the “arm” to enforce the smoothness of the result.

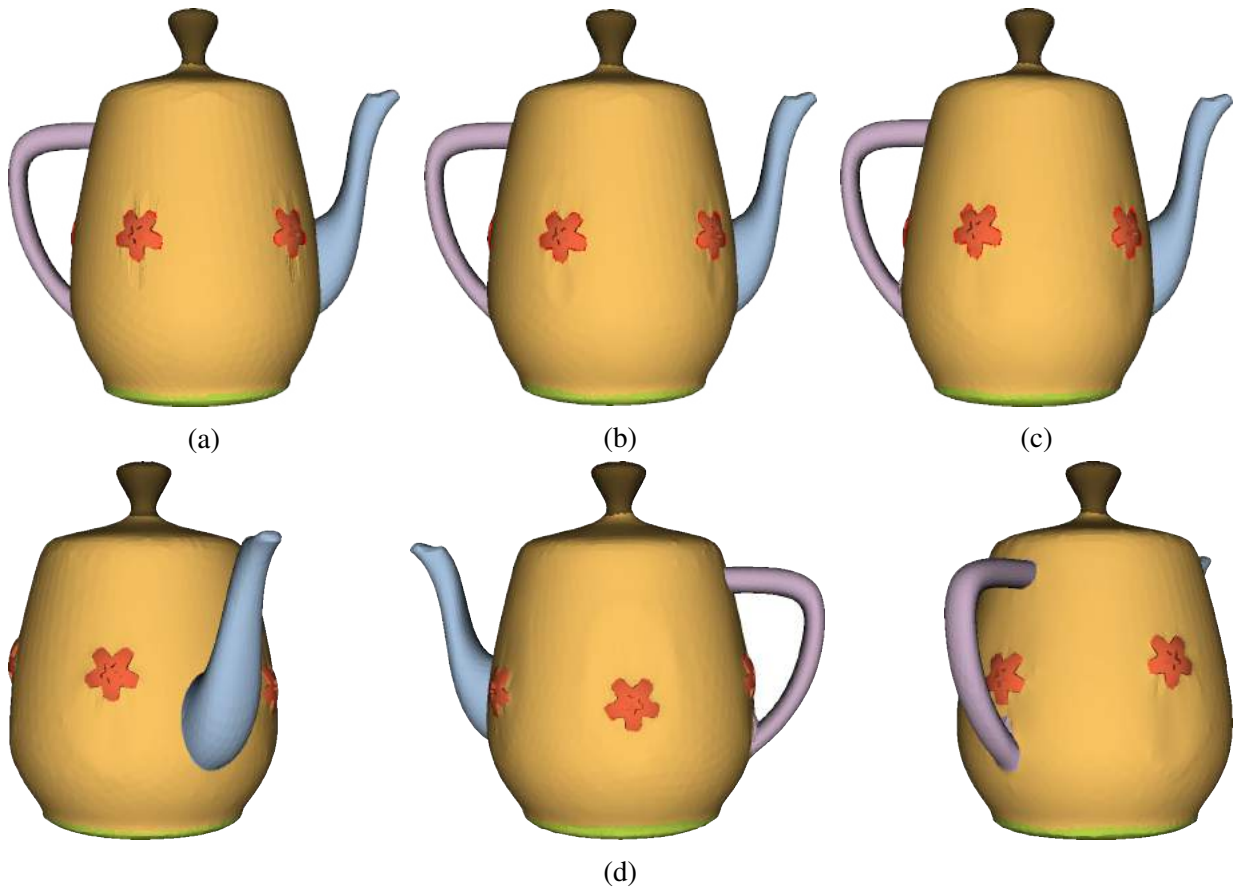


Figure 5.4: (a) Result with no “Laplacian Displacement” constraint; (b) result with “Laplacian Displacement” constraint with weight lower than the “Same level” one (2 vs. 4), (c) result with higher weight for the “Laplacian Displacement” (10 vs. 4), (d) result with normal of the “Plane” constraint not pointing upwards.

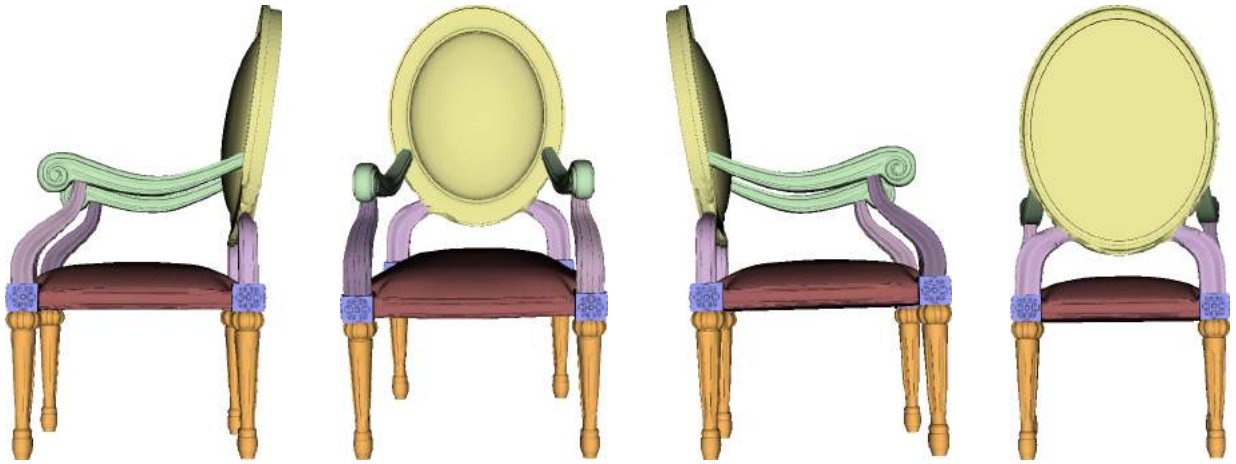


Figure 5.5: An all-around view of the “baroque chair” model.

We imagine that, for stylistic or artistic reasons, the domain expert would not want to constrain the “support” to be wholly co-axial to the leg, but just its “base” or “attachment” (this is not a specified part of the shape, it has not been annotated). Thus, one of the goals of this constraint is to firstly understand what segment of the “support” needs to be constrained: intuitively, the subset that is co-axial to the leg when the expert has set the constraint is the part to be constrained.

So, we extract the skeleton of the part to be constrained and the corresponding axis<sup>4</sup> of the reference part (see Figure 5.6). Then, we move along the poly-line that represents the skeleton, starting from the node corresponding to the boundary of the part to be constrained which is closest in terms of adjacency relationships (see subsection 2.3 and blue connections in the graph of Figure 5.6), to the reference part. We iteratively move to the following skeleton node considering the angle between the current skeleton segment and the axis of the reference part, until it is bigger than a defined threshold (we used 10°) or until no more segments are available. Finally, we use the first boundary and the one associated to the last used node for performing a region growing (as in 2.4.1) on the surface to obtain the corresponding subset of the part which must be constrained. The seed triangle used here is the triangle on the left of the first edge of the first boundary (since it is a boundary of a *region* annotation - see subsection 2.2.2 and 2.4.1 - its boundaries are ordered so that the interior of the annotation always lies to the left of the boundary).

The “Co-axiality” projection is then obtained as follows:

1. Centre the input points on their average position;
2. Extract the axis  $a_1$  of the part to be constrained (namely, the “support”, from now on called

---

<sup>4</sup>The axis can be defined in several ways. Here, we used Plumber [MPS<sup>+</sup>04] to extract the skeleton of the part and then fitted a line to the nodes composing it.

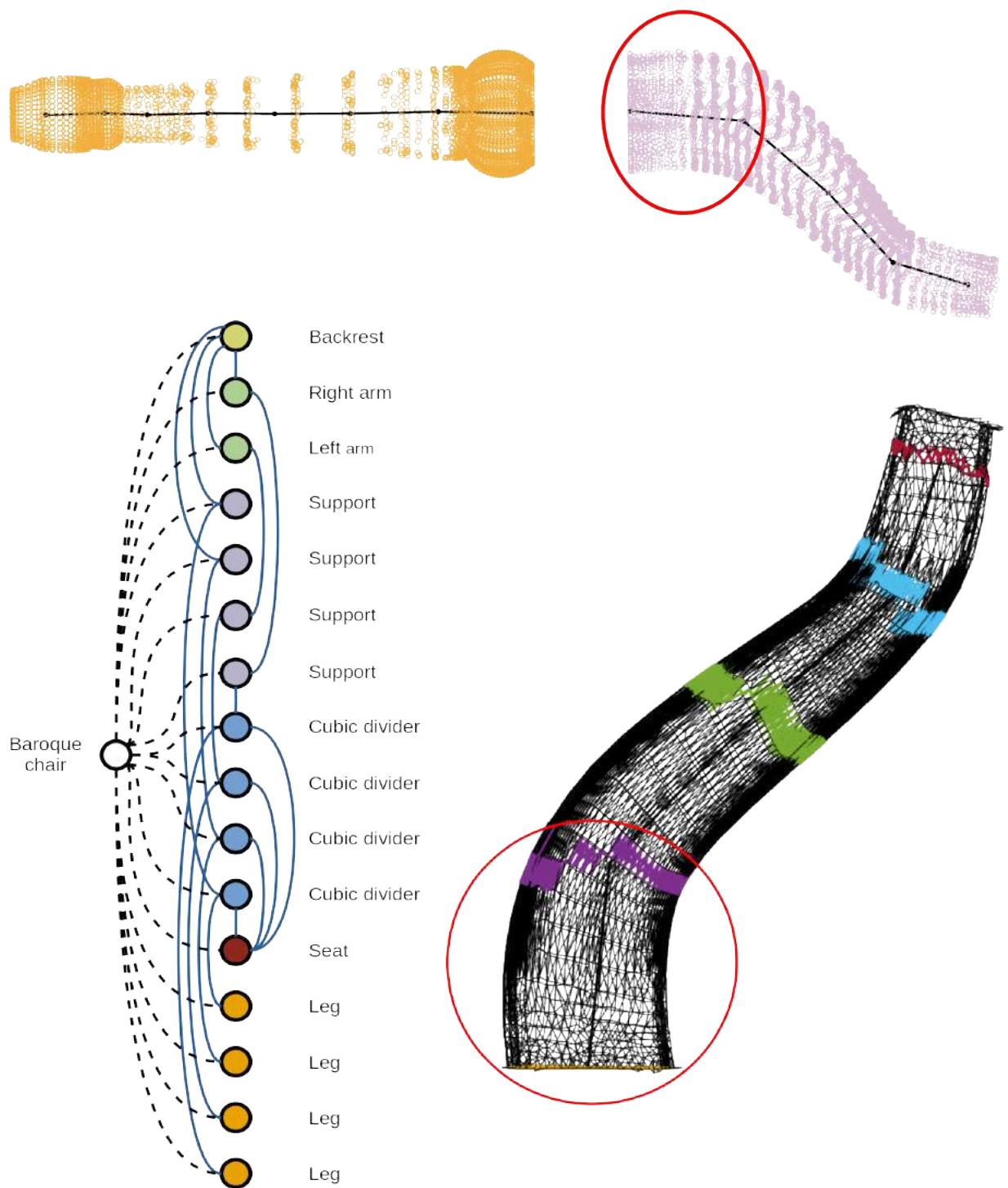


Figure 5.6: In the first row, the involved points and the corresponding skeleton can be seen (rotated by  $90^\circ$ ). The constrained part is highlighted in red. In the second row, the structure of the “baroque chair” model and some coloured strips representing the edges intersecting a construction sphere used by the skeleton extraction algorithm (see [MPS<sup>+</sup>04]).

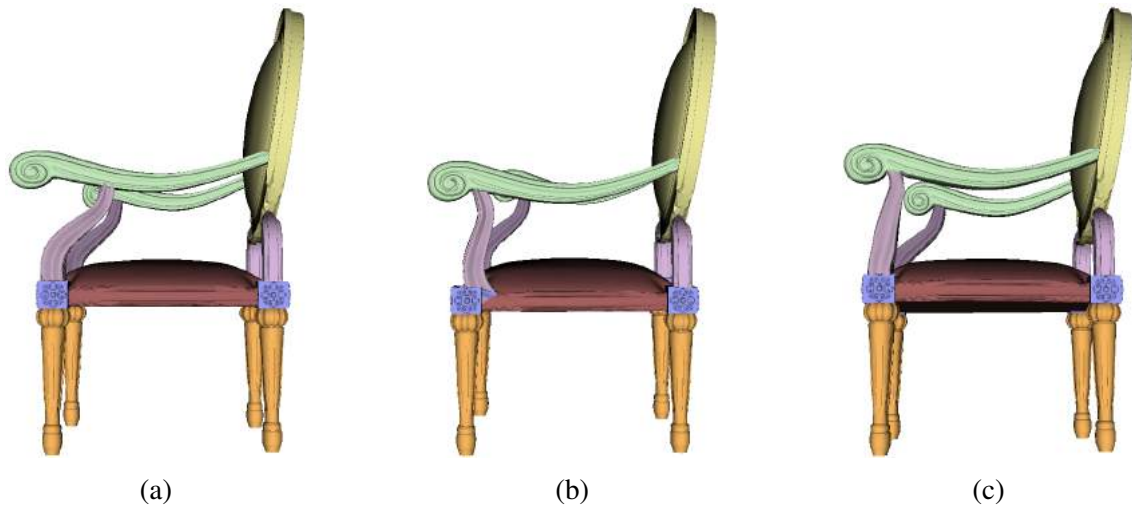


Figure 5.7: Result of the application of the “structural continuity” constraint: (a) result after elongation of the arm, (b) result after the rotation of the support without constraint, (c) result after the rotation with the support constrained to be in “structural continuity” with respect to the leg.

- s) and the axis  $a_2$  of the reference one (namely, the corresponding “leg”). Compute the angle  $\alpha$  between them;
3. Define the matrix  $\mathbf{R}$  for rotating  $a_1$  around the rotation axis given by  $a_1 \times a_2$  of an amount defined by  $\alpha$ , so that the direction of  $a_1$  is the same of  $a_2$ ;
4. Define the translation vector  $\mathbf{t}$  for moving  $a_1$  so that it coincides with  $a_2$  as the projection vector of the average position of  $s$  onto  $a_2$ ;
5. Rotate the points of  $s$  using  $\mathbf{R}$  and move them using  $\mathbf{t}$ .

The result of the same rotation as before with “Co-axiality” and “Laplacian” constraints (with equal weight) can be seen in Figure 5.7c, for a template mesh with ~90K vertices, without cage extension and with only 2 iterations of the solver allowed (it was necessary to achieve almost interactive speed).

### 5.2.3 Deformation in a VR environment

During the last decade, the interest in the development of immersive VR and Augmented Reality (AR) systems for manipulating and modelling 3D environments through gestures has increased [CGM19, MCG<sup>+</sup>19, LHT<sup>+</sup>19]. This is also due to the technological improvement of low-cost



AR/VR technologies and gesture tracking acquisition devices, which allowed to reduce the sense of sickness and the cost reduction. Recently, most of the Head-Mounted Display (HMD) producers have begun to offer 3D modelling applications. The advent of more affordable systems and less intrusive devices opens the possibility for companies to include end-users in product evaluation and customisation.

Moreover, typical users of VR environments can be not experts in 3D modelling, so it is important to allow such modifications with easy to use and easy to learn commands.

Typical capabilities of commercial applications, such as Sculpturing [scu], Oculus Medium [ocu], MakeVR Pro [mak17], MindeskVR [min15] and Gravity Sketch [gra], are shape sculpting by push and pull operations through controllers together with the combination of free-hand sketching for shape generation.

Various works address the manipulation, e.g. rotation, translation and limited modification in size of 3D objects (e.g., compression and squeeze) in AR and VR using different devices: among them, [JC18] provides manipulation in AR environments using the Microsoft HoloLens; [LBGM19] exploits HTC Vive HMD, Leap Motion Controller (LMC) and voice commands to browse, inspect and deform the results of a 3D CAD assembly search engine. LMC is also used for modelling and shaping pottery objects [VR15] and for mid-air free hand sketching [CKS16]. [CS17] proposed the use of bi-manual interaction, with one hand controlling 3D position and rotation, while the other performing grasping and releasing actions. More devoted to the engineering context, Cohen et al. [CRV20] used LMC to capture pinch and pull actions for flexible manipulations on Control Point (CP)s coordinates of NURBS surfaces.

These systems are only focusing on the resulting shape, without any consideration on its associated semantics. However, being able to operate on selected semantically meaningful areas while simply using our hand movements makes the interaction more natural and pleasant [OFBW07, FW13, VR15, LGJ<sup>+</sup>17].

The semantic organisation of the virtual scene is well recognised as a mean to better support behaviour specification of VR elements in response to user actions [CTB<sup>+</sup>12, CMSF11]. [FW13] described a semantic model usable for modelling purposes, highlighting its advantages for shape modification and the potential of its integration in game engines.

Here, we summarise the results of an initial feasibility study [SZG<sup>+</sup>20] aiming at integrating 3D modelling capabilities in a VR environment, based on voice and gesture commands, with focus on showing the potential of coupling cage-based deformation with multi-modal interaction. In the following, we present the deployment of the basic functionalities of the deformation system in a VR environment, tackling the issues of intuitive interactions to perform cage-based (unconstrained) deformations on annotated shapes.

## Application setting

We have already mentioned that cage-based deformation techniques are already quite common on desktop applications (e.g., [CCLS18]) for the deformation of shapes, thanks to their flexibility, ease of implementation and speed. However, VR environments introduce a number of additional difficulties that require careful evaluation and treatment.

Here, objects are embedded in a 3D space that replicates the physical world, increasing the sense of realism; but how can we exploit the 3D setting to effectively communicate the information associated with the annotated geometry to the user? How can she/he act on the 3D elements through a natural and intuitive behaviour? These questions call for an efficient mechanism and a natural interface to: i) select the CPs related to the area to be deformed; ii) specify the desired deformation on the selected CPs; iii) apply the required deformation in real time. Furthermore, we want to exploit the semantics associated to the object.

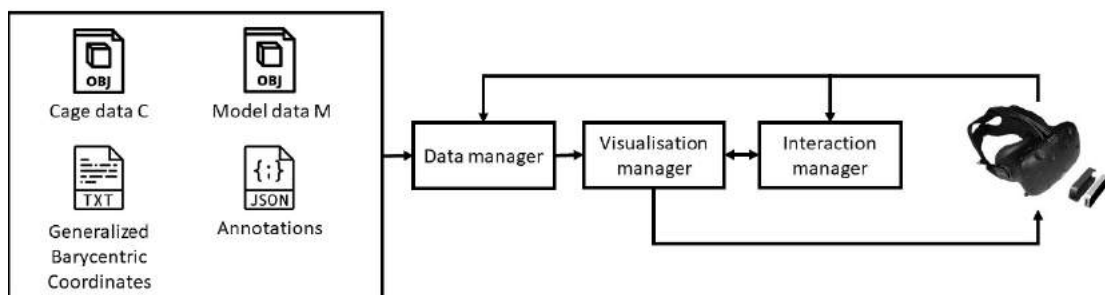


Figure 5.8: System overview. From the left, the input data, the modules involved in the system and the resulting VR environment.

We deploy the graphical engine Unity 3D to visualise a virtual scene accessible by the user through an HTC Vive HMD. The interaction in the 3D space is ensured by the LMC, which is able to track fingers' joint positions and detect simple gestures.

Figure 5.8 shows the organisation of the VR system. A 3D object and its cage are given as input, along with the list of annotations associated to the model (e.g., defined by a domain expert in the system described in Appendix A) and the GBC values. The two 3D meshes (object and cage), the annotations and the GBC are processed in the *data manager* module, also responsible for managing cage and model modifications resulting from the user interactions. The *visualisation manager* module is responsible for rendering the scene, updating it during the user interaction, and communicating information, e.g., by adding extra virtual elements or exploiting colour variation of scene elements. Finally, the *interaction manager* module processes and interprets the data derived by the LMC and the HMD recognising different interaction techniques.

**Data manager module** The data manager module loads the input data in the system and performs the geometric analysis and processing operations required by the deformation, as expressed by the user either through gestures or voice and interpreted by the interaction manager.

With reference to Chapter 2, we assume the model has been previously annotated and comes ready for the manipulation; therefore, the data manager loads the following files: i) the model  $\mathcal{M}$ , a 3D shape represented as a watertight triangle mesh; ii) the cage  $\mathcal{C}$ , another triangle mesh, coarser than the model itself and enclosing it; iii) annotations, a structured file specifying geometry, tag, colour and other useful data to manage the regions of interest; iv) the GBC, a list of pre-computed values depending on the vertices of  $\mathcal{M}$  and  $\mathcal{C}$  that specify the influence of the different cage vertices on each of the object's vertices.

With reference to 4.1.3, vertices  $\mathcal{M}$  and of the cage  $\mathcal{C}$  are stored in matrices  $\mathbf{V}_{\mathcal{M}}$  and  $\mathbf{V}_{\mathcal{C}}$ , where each row corresponds to the 3D coordinates of a vertex.  $\mathbf{V}_{\mathcal{M}}$  has size  $n \times 3$  and  $\mathbf{V}_{\mathcal{C}}$  has size  $m \times 3$ , where  $n$  and  $m$  are the number of vertices in  $\mathcal{M}$  and  $\mathcal{C}$  respectively.

Then, a list of CPs  $\{CP_i\}_{i=1}^m$  is generated to allow the user selection, where the  $CP_i$  is associated with the position of the  $i$ -th vertex of  $\mathcal{C}$  (i.e. the  $i$ -th row in  $\mathbf{V}_{\mathcal{C}}$ ).

Finally, the matrix  $\mathbf{B}$  (of size  $n \times m$ ) in equation 4.3 is built from the GBC file, and annotations are arranged in a tree structure accommodating the hierarchy information among the different RoIs (see Section 2.3).

**Automatic identification of CPs related to an annotation** As introduced in Chapter 2 and 4, let  $\mathcal{A}$  be an annotation with associated triangles  $T_{\mathcal{A}}$  and vertices  $V_{\mathcal{A}}$  and let the matrices  $\mathbf{V}_{\mathcal{M}}$  and  $\mathbf{V}_{\mathcal{C}}$ , whose rows correspond to the 3D coordinates of the vertices in the meshes  $\mathcal{M}$  and  $\mathcal{C}$ , respectively, be related by Equation 4.4. From this relation, it follows that a certain cage vertex  $\mathbf{c}_j$  impacts each of the model vertices at a certain extent, defined by the values specified in the corresponding column (the  $j$ -th) of  $\mathbf{B}$ . Analogously, a mesh vertex  $\mathbf{v}_i$  is influenced by each cage vertex at an extent defined by the elements in the  $i$ -th row of  $\mathbf{B}$ . However, some cage vertices will have a much higher influence than others on  $\mathbf{v}_i$ . Thus, given a region of interest on the mesh, we will select the relevant CPs as the cage vertices whose influence on the region vertices is greater than a certain threshold  $\tau$ . So, given a vertex  $\mathbf{v}_i \in V_{\mathcal{A}}$  associated with  $\mathcal{A}$ , we are interested in finding all the vertices  $\mathbf{c}_j \in V_{\mathcal{C}}$  whose influence values  $b_{ij}$  are greater than  $\tau$ .

Applying this procedure for all the vertices in  $V_{\mathcal{A}}$ , we select a set of cage vertices that can be used to manipulate the area corresponding to  $S$ , up to a certain precision that depends on the number of cage vertices around the interested area, their distance from the area, the presence of other annotations close to  $\mathcal{A}$  and the locality of the GBC that have been used. Indeed, we remember that MVC [FKR05, JSW05] have a global behaviour, meaning that every cage vertex always influences the shape almost in its entirety, while for example LBC [ZDL<sup>+</sup>14] provide a high localisation of the influences.

Notice that, while in this work a single threshold  $\tau$  has been applied to determinate the CPs associated with the different RoIs, different thresholds may be identified for obtaining better fittings on the single RoI and for different models or cages.

In particular, some considerations may be done regarding the values associated to the involved entities: for instance, we can search for the maximum value  $b_{ij} \in \mathbf{B}$  for each model vertex  $\mathbf{v}_i$  and then take the minimum of these values as  $\tau$ , and so  $\tau = \min \mathbf{b}_j$ , where  $\mathbf{b}_j = \max b_{ij}$ . In this way, we could be sure that each model vertex can be manipulated at least through one cage vertex, while at the same time we are enforcing the reduction to the minimum possible number of cage vertices associated to each model vertex.

Conversely, we can search for the maximum value  $b_{ij} \in \mathbf{B}$  for each cage vertex  $\mathbf{c}_j$  and then take the minimum of these values as  $\tau$ , and so  $\tau = \min \mathbf{b}_i$ , where  $\mathbf{b}_i = \max b_{ij}$ . In this way, we would obtain a threshold allowing to each cage vertex to control at least one model vertex, while at the same time reducing as much as possible the number of model vertices controlled by each cage vertex.

Finally, another possibility is to design a different threshold for each model (or cage) vertex. An in dept analysis of the goodness all these possible definitions for the threshold is left for future works.

To allow more localised deformations possibly independent from the imported annotations, the user can select CPs manually, or the achieved automatic selection can be edited by adding/removing single CPs from the set of suggested ones (see interaction manager). Note that this validation process does not affect the result of the previous step permanently: changing the CPs involved with a certain RoI requires the editing of the threshold value.

**Update of the model** Once the CPs have been selected, and the deformation parameters are interpreted from the user actions by the interaction manager, the actual deformation occurs: this computation is performed by the data manager.

Firstly, the selected CPs are transformed (applying the same transformation to the associated cage vertices); secondly, the deformation is propagated to mesh vertices as explained in 4.1.3.

Cage-based deformation techniques give the user total freedom, so that he can move one or more CPs in the space regardless of their mutual arrangement. Of course, extreme displacements of CP can give bad results, e.g., due to the nature of Linear Blend Skinning techniques, of which cage-based techniques are a subset (details are given in [KCvO07] and [JDKL14]).

In this application, three main transformations are allowed:

- **Translation.** The CPs are moved in the space in the same direction.
- **Rotation.** The CPs are rotated with respect to an arbitrary axis  $\mathbf{a}$  of an angle  $\theta$ . Given the

two axes  $\mathbf{a}_s$  and  $\mathbf{a}_e$ , returned by the interaction manager module and representing the start and end configuration, the rotation axis  $\mathbf{a}$  is defined as the line passing through the centroid ( $\mathbf{R}_c$ ) of the selected CPs and with direction  $\mathbf{R}_d$  equal to the cross product between  $\mathbf{a}_s$  and  $\mathbf{a}_e$ , while the rotation angle  $\theta$  is defined as the variation between the starting and ending axes  $\mathbf{a}_s$  and  $\mathbf{a}_e$ . Notice that, if  $\mathbf{R}_c$  does not correspond to the origin of the world reference frame, a combination of translation and rotation is required.

- **Scaling.** Differently from the previous transformations, the proposed scaling operation acts on the entire 3D model changing the positions of all the CPs, so that the CPs are moved toward or away from the scaling centre (i.e. the centroid of the 3D model) resulting in a uniform scaling.

**Visualisation manager module** This module receives input from the data manager and the interaction manager to visualise the “semantised” object, the virtual hands used to interact in the virtual environment and the effect of the user’s actions on the 3D model.

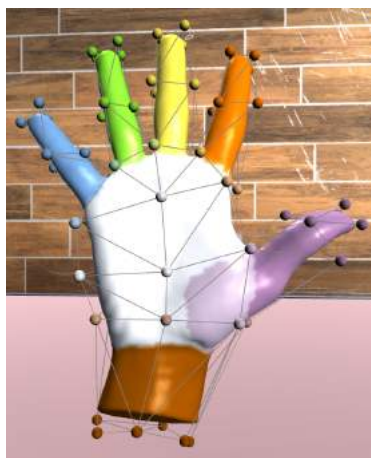
Once the data manager has processed the input data, the model mesh  $\mathcal{M}$  is (always) visualised in the scene in a shaded mode, while the cage model mesh  $\mathcal{C}$  is rendered in wire-frame mode by default to avoid hiding the model  $\mathcal{M}$ . At the cage vertices positions, independent sphere elements are introduced to highlight the CPs. The user can hide completely the cage and the CPs (with voice commands “Hide cage” and “Hide CPs”) to visualise only the 3D model with no obstruction of additional elements.

Annotations are highlighted in the virtual scene by assigning a specific colour each, as expressed by the data manager module. The latter is also responsible for identifying the CPs associated with each annotation when required; the visualisation manager receives this information and colours the CPs accordingly. If a CP influences multiple annotated parts, the visualisation manager assigns it the average of the different colours involved. Figure 5.9a shows an example of this effect.

The visualisation manager presents also the data derived from the interaction manager module. First, it visualises the result of the hand tracking. Among the different avatar types of hands included in the Leap Motion asset, here the capsule hand type is adopted for allowing to visualise flexible fingers improving the sense of reality in the virtual scene.

In addition, this module provides an echo to the user for the selection operations detected by the interaction manager module, i.e., when the user looks at a target it is highlighted. It is possible to select a single CP or a set of CPs associated with an annotation; in the latter case, all the associated CPs are highlighted. Depending on the viewpoint and on the shape complexity, some CPs related to a semantic part may be hidden behind the model itself; in this case, a yellow marker appears in correspondence of the covered CPs. An example of this behaviour is depicted in Figure 5.9b, where the user gazes at the palm.

Finally, to reduce the amount of information visualised simultaneously, this module allows the selective rendering of annotations according to their hierarchy level.



(a) *CPs associated to multiple segments*



(b) *Selected CPs not visible from the user point of view directly*

Figure 5.9: *Example of data represented by the visualisation manager*

**Interaction manager module** To apply the deformation, we define a set of commands using hand gestures and voice keywords considering that standard users are not expert designers of 3D shapes. Thus, we aim at defining interactions that are *easy to learn* and *easy to use*.

Generally, to support these requirements, we define commands simple to be remembered and not similar among them to avoid getting mixed up. In addition, the user can apply mathematical transformations (translation, rotation and scaling) without the necessity of specifying axis, angles or others specialist concepts.

In accordance with these considerations, we designed different interaction techniques that we describe in the following. Note that the interaction manager analyses the user commands and communicates the parameters required for the deformations to the data manager module at every frame. The deformation is also computed and applied to the model every frame and rendered simultaneously with the interaction.

**Selection** With the aim of providing *easy to use* interactions, we avoid a selection by using virtual hands, because it requires an high precision level in picking exactly the desired target, especially when target objects are small or surrounded by others possible targets.

For this reasons, we propose the use of a *selection gaze*, where the selection operation is performed in two steps. First, the user acquires the desired target by looking at the different se-

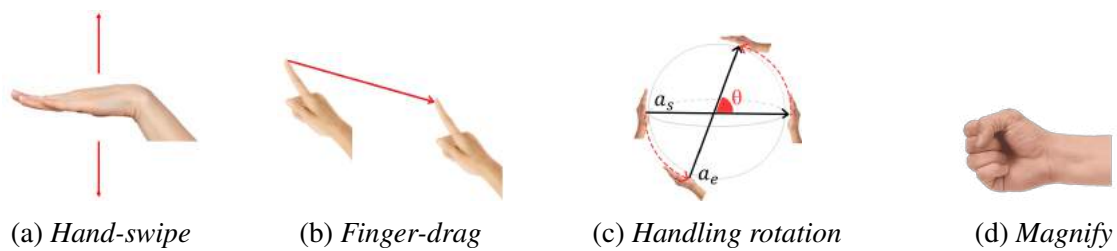


Figure 5.10: *Interaction gestures*

lectable elements in the scene; then, she/he confirms the choice by using the voice command “select”. The user can select one CP at a time, or select at once the set of CPs associated with an annotated part or with the whole object.

Once a target has been acquired, it is highlighted to indicate the selectable element. Finally, once the target has been confirmed, the colour of the CPs changes from the original colour (obtained as specified in Visualisation manager) to red.

The same procedure is applied to deselect objects. In this case, the interaction is identical, the voice command to confirm the deselection is “discard” and the user’s visual echo differs by changing colour from red to the original one. The user can edit the selection by combining selection and discard commands on annotated parts or single CPs until satisfied.

As mentioned in Chapter 2, annotations are organised into several levels of detail based on the *containment* relationship. To browse and change level of detail for the selection, we propose a *hand-swipe* command (see Figure 5.10a) allowing the user to increase or decrease the level of detail moving his/her open hand vertically. If the user aims at accessing the  $n$ -th level, then she/he can use voice commands saying “level  $n$ ” to visualise and have access to the level of information she/he is interested in directly.

**Translation** Through the *finger-drag* gesture (see Figure 5.10b), the user can deform a 3D shape by moving the selected CPs in the virtual space in any direction. This technique allows to interact with single and multiple CPs. Selecting the unique segment belonging to the first level, it is possible to move the entire 3D model.

Selected elements can be moved according to the tip of the index finger position, whose displacement in the space defines the translation vector, yielding the technique simple and intuitive. Indeed, it is sufficient for the user to point at elements she/he aims to relocate, extending the right index while closing the others, and wait for the acoustic signal confirming the gesture detection. Once the gesture is identified, the user can move the selected objects in the 3D virtual space with a complete freedom of movement.

Once the user has reached the desired modification, she/he can stop the interaction changing the

hand posture, for instance opening the hand completely.

**Rotation** To perform 3D rotation, the user can select the whole model or its portions, by selecting one or more CPs or an annotated RoI.

Once the selection is accomplished, the user engages the *handling rotation* technique by maintaining both hands open, with palms facing each other, until an acoustic echo communicates the success of the command recognition. Maintaining the engaging posture, the user moves his/her hands defining the starting and the ending axes  $\mathbf{a}_s$  and  $\mathbf{a}_e$  in the space characterised by the hands posture (see Figure 5.10c). The  $\mathbf{a}_s$  and  $\mathbf{a}_e$  axes are used to compute the rotation parameters. To conclude the rotation, the user has to close his/her hands.

This technique is quite simple since the user does not need to combine several rotations to rotate along with a generic axis (i.e., not aligned with the coordinate system); then, also users non-expert in design can achieve the desired result. In addition, large rotations can be achieved by repeating the gesture several times.

**Scaling** The *scaling* operation (see Figure 5.10d) increases and decreases the size of the model. To activate it, the user has simply to place the hand in front of herself/himself; if the hand presents all extended fingers then the model is enlarged, otherwise, if the hand is closed, the model size is reduced. To stop the interaction, it is sufficient to remove the hand from the leap motion field of view.

With this technique, the interaction is affected only by the hand posture (open/closed hand). On one hand this ensures a quite simple interaction, since the user has to control only one action. On the other hand, the user cannot scale the 3D model along with a preferred direction, i.e. the scaling is uniform. In addition, it is not possible to apply the scaling operation to a portion of the model yet. A more in-depth study is required for local scaling.

## Results

The proposed system has been developed adopting Unity 2019.2.4f1 as graphical engine and programmed in C# language. In this work, the threshold  $\tau$  used to define the CPs associated with a certain RoI has been defined a priori and the user has no possibility to modify the default value.

For the first results here proposed, we used a *hand*, *teapot* and *lifebuoy* models and cages having 10K and 70, 15K and 200 and 10K and 179 vertices respectively. The employed GBC are the MVC, which have been chosen because of their fast implementation and the closed form expression allowing better performances (parallel computation) in the computation phase.

The threshold  $\tau$  has been empirically estimated and the results of this setting are depicted in



Figure 5.11 with different annotation levels of the hierarchy illustrated, where Figure 5.11a represents the root of the hierarchy with the whole object annotated as “hand”, Figure 5.11b represents the segmentation in different semantic (and functional) parts of the hand and Figure 5.11c represents the last level, containing only the nails (here the black part means a not annotated surface).

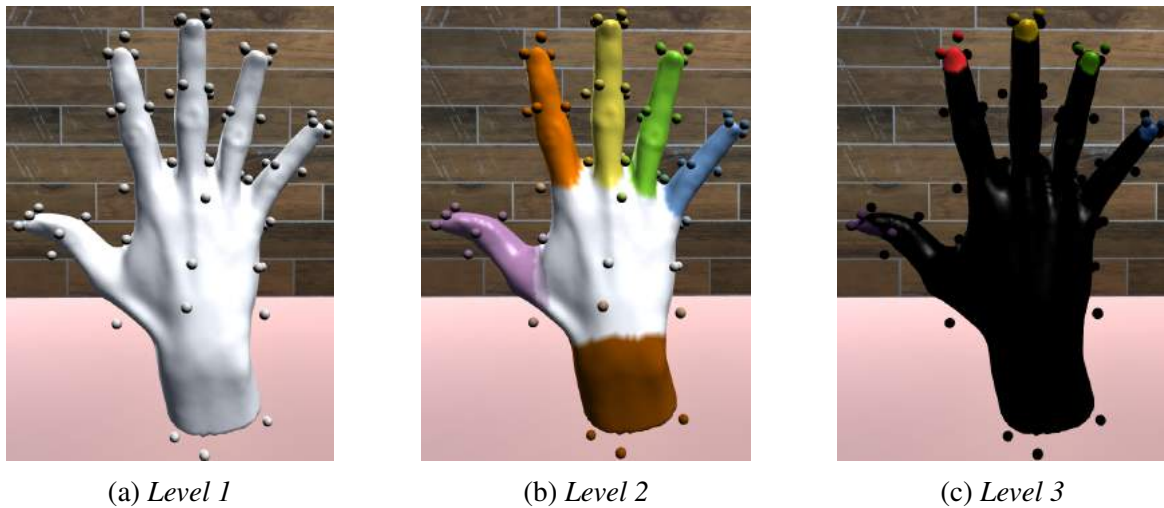
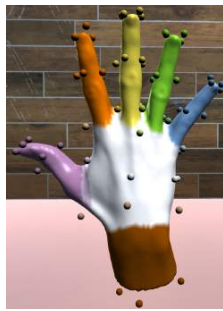


Figure 5.11: *The different levels of the annotation hierarchy; CPs are coloured according to their influence on annotations with threshold 0.4.*

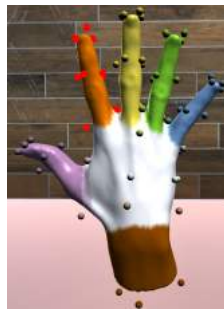
Finger-drag interaction can be applied even to single control points translating each in different directions. On the one hand, specifying the final positions of single CPs can seem a tedious task; on the other hand, this allows to localise more deformation effects and, in some cases, to achieve more easily the desired deformation.

Some examples of deformation are presented in Figure 5.12. In the first row, the user asked for the visualisation of a certain level (level 1 - Figure 5.12a) and then selected a segment by gazing at it. This triggers the selection of the control points shown in Figure 5.12b (highlighted in red), which can be used to perform first a rotation of the control points (Figure 5.12c) and then their translation to reduce the distortion introduced on the middle finger shown in Figure 5.12c (result - with some minor adjustments - shown in Figure 5.12d).

Finally, a video-clip with simple modifications of the *teapot* model can be seen at <https://youtu.be/4Kzs64iaq-4>.



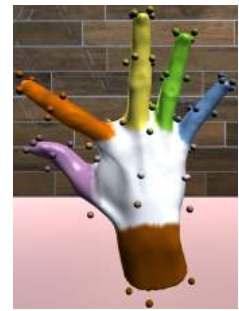
(a) Initial stage of the 3D model



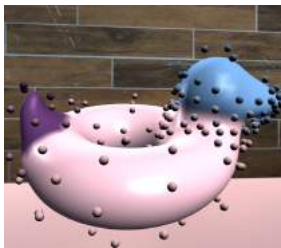
(b) Index finger selected



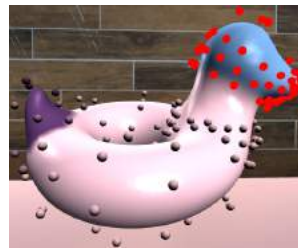
(c) Result of simple rotation



(d) Result after translation



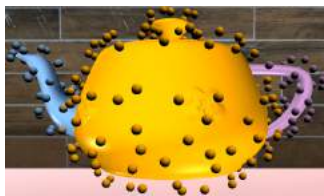
(e) Initial stage of the 3D model



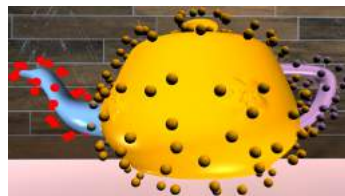
(f) "Head" moved upwards and rotated



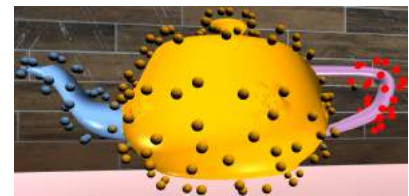
(g) "Beak" elongated moving the tip



(h) Initial stage of the 3D model



(i) "Spout" translated and rotated



(j) "Handle" elongated moving only some vertices

Figure 5.12: Top row: deformation result using selection of the control points associated with the "index" segment, rotation of the selected control points and translation of the selected control points, with some minor adjustments. Middle row: deformation result using selection of the control points associated with the "head" segment, rotation and translation of the selected control points, selection of some control points corresponding to the tip of the "beak" segment and their translation. Bottom row: deformation result obtained by translating the control points associated with the "spout" segment and rotating the control points associated to the "handle" of the teapot.

## 5.3 Archaeological reconstruction scenario

As a second application context, we identified the support to archaeologists for hypothesis formulation and virtual reconstruction from fragmented findings.

Indeed, archaeologists and curators often deal with fragmented artefacts, which are scattered among hosting institutions. This depends on a common behaviour during past expeditions: every participating institution used to take some findings with them, causing a separation of objects coming from the same excavation site into museums scattered all over the world.

This separation makes a re-assembly of shards particularly difficult since probably not all the pieces are available in the same place. Not to mention the fact that the archaeological findings are often eroded, meaning that some material is missing from their border or surface, and large parts of objects are however missing.

For these reasons, the reconstruction is not the solution of a “simple” 3D jigsaw puzzle. Not only that: the relics are even very sensitive to any kind of stimulus, such as incorrect lightning, humidity, etc., and so the experts must be extremely careful when handling them. Finally, the reconstruction has to be done bottom-up due to the presence of gravity. However, if parts are missing from the base of the relics, the experts have to reconstruct the object in large “patches”, which will be joined later on.

For all these reasons, a good idea would be to create a sort of *virtual* workshop, in which archaeologists could try to reconstruct objects without having to handle them directly, but rather acting on their virtual replica, which can be obtained through different techniques such as photogrammetry or laser scanning. The GRAVITATE project [PWM<sup>+</sup>16] applied ICT to tackle issues related to the support of archaeological research on fragmented collections, in particular the re-assembly of distributed fragments. However, in the worst case, only a few, maybe only one fragment is found for an object. Nonetheless, the expert is typically able to figure out in his/her mind the overall aspect of the original piece, thanks to his/her a-priori knowledge of provenance and style, which implies certain dimensions, measurements, part appearance and relationships. In other words, thanks to the semantics that defines how an object in that class should look like.

In the following, we tackle the presented issues developing a new kind of constraint meant to keep certain “proportions” between parts of an object.

### 5.3.1 Interactive virtual archaeological reconstruction

As already introduced, one of the most common issues when dealing with archaeological findings is their susceptibility to external stimulus: in this subsection we want to propose a pipeline, using the GUI presented in Appendix A, which allows the manipulation of fragments’ position and orientation and their registration on a reference or *support* shape. The template of the complete

object, individuated by the archaeologist or the curator, will serve as the support shape, and will be deformed to allow a virtual shape completion to give a visual hint of the original, whole object.

In this application, the fragment itself needs to be annotated with respect to the same knowledge formalisation of its template. In order to align the fragment to the template automatically, for instance, the fragment must exhibit at least three landmarks (point annotations) in common with the template.

The steps are the following:

1. Load a template;
2. Load a fragment. If it contains less than three landmarks in common with the template, insert the remaining manually, through a dedicated additional pop-up window (see Figure 5.13);
3. The system automatically scales the template shape to fit the size of the fragment and translates/orients the fragment to place it on the template;
4. Non-rigid registration of the template over the fragment;
5. Constraints check.
6. Iterate over step n° 2.

In the first step, the user is asked to select the reference template for the object to be reconstructed.

For allowing the selection of landmarks, we inserted in the framework a dialog window similar to the interface for the bootstrap of the alignment in MeshLab [CCC<sup>+</sup>08] (see Figure 5.13).

In the third step, we employed Umeyama's algorithm [Ume91] for the rigid registration of two points patterns (associated in pairs), that are impersonated by the corresponding landmarks selected on the geometric template and the fragment. This allows to perform a rough alignment between the fragments and the template, in a least-square manner. The results of this step can be seen in Figure 5.14.

Then, in the fourth step, we deform the shape of the template in a non-rigid manner so that it fits more precisely the shape of the fragment. We first tried to employ a state-of-the-art method for shape reconstruction through template fitting (see Section 1.1), namely the one defined in [AWLB17]. However, this approach fails in some cases, for the reason it is not semantic-aware.

To explain this concept, in Figure 5.15 an example of situation in which the geometric approach would fail: since the two shapes (red and blue) are quite dissimilar (which is often the case, dealing with artistic creations), there may happen that some points in the source shape are wrongly associated to the target shape (see Figure 5.15a) because they are geometrically closer; consequently, the source shape is not very similar to the target one after the deformation. Moreover,

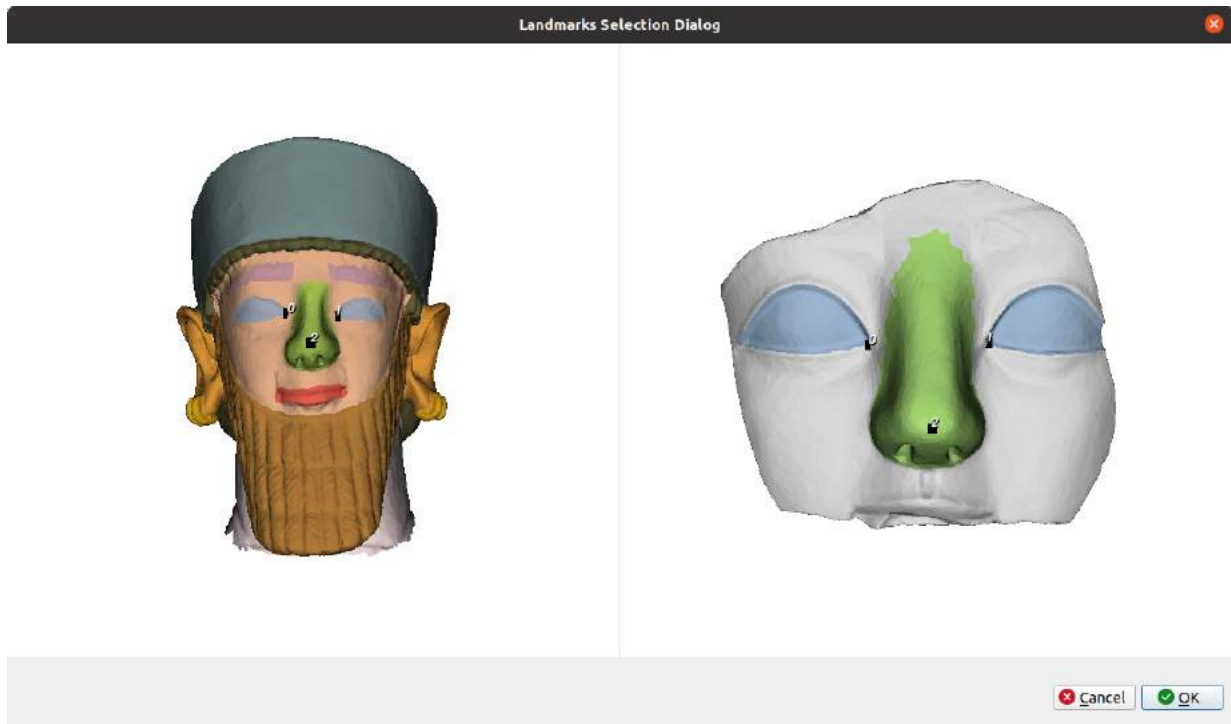


Figure 5.13: *The dialog window for inserting the landmarks for the rigid alignment step.*



Figure 5.14: *The results of the rough alignment obtained employing Umeyama's algorithm.*

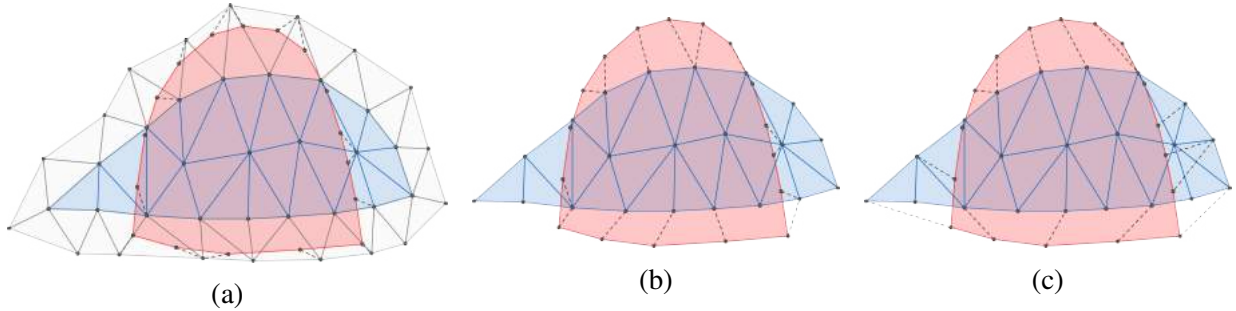


Figure 5.15: *The choice for associating vertices of the red part (annotation over the template) to vertices of the target mesh (i.e., vertices of the fragment, where a corresponding blue part is annotated): (a) pure “geometric” association of vertices; (b) association restricted to vertices involved in the annotation; (c) a more “semantic” approach.*

[AWLB17] is designed to work on point clouds referring only to the outer surface of human bodies, while archaeological fragments often include “internal” and “fracture” facets (see [ED17]), thus adding complexity to the fitting.

For this reasons, our first result is obtained by exploiting the annotations defined over the template shape. The idea is the following: if objects are part of the same homogeneous class, they should all be composed of a core list of parts and features. Just to make an example, we can say that most of human shapes possess a nose, a mouth, the eyes, etc. Thus, in object reconstruction, we want to deform the template shape so that shared parts more or less overlaps in the final stage. So, we defined the non-rigid registration as the composition of two optimisation problems:

1. *Vertices correspondence*: we want to find the mapping between each template vertex  $\mathbf{v}_i \in \mathcal{V}_{a_1}$  and a fragment vertex  $\mathbf{v}_j \in \mathcal{V}_{a_2}$  that are part of two compatible annotations  $a_1$  and  $a_2$  (e.g., they are both part of the “Drum” of an “Ayia-Irini small human idol with drum”), minimising some costs:
  - $\mathbf{C}_{ij} = \|\mathbf{v}_i - \mathbf{v}_j\|$  is the distance between each vertex  $\mathbf{v}_i$  of the template and each vertex  $\mathbf{v}_j$  of the fragment;
  - $\mathbf{B}_{ij} = |b(a_1, \mathbf{v}_i) - b(a_2, \mathbf{v}_j)|$ , where  $b : (\mathcal{A}, \mathbb{R}^3) \rightarrow \{0, 1\}$  is a function stating if a vertex belongs to the boundary of an annotation (1) or not (0). So, we want to map vertices on the boundary with vertices on the boundary, and vice versa. This function returns always 1 if the annotation has a line or point selector;
  - $\mathbf{D}_{ij} = \sum_{k=1}^{|L(a_1)|} (d(\mathbf{v}_i, L(a_1)_k) - d(\mathbf{v}_j, L(a_2)_k))$ , where  $d : (\mathbb{R}^3, \mathcal{L}) \rightarrow \mathbb{R}$  is a function defining the distance, over the boundary of an annotation, between a vertex and a certain landmark (or point annotation with a single point selected)  $L(a) \sqsubset a$ . This distance is defined by summing the length of the edges of the boundary snippet connecting the vertex and the landmark going in counter-clockwise order. Minimising

this cost, we want the vertices on the boundary to map with vertices more or less in the same boundary position. Indeed, with reference to Figure 5.15b, a vertex on the upper arch of the red part is mapped onto a vertex on the lower arch of the blue part, resulting in a counter intuitive mapping. Note that possible landmarks in this example may be the inner and outer apex of each part, since they are annotations of eyes in the human body, so that the distance from the landmark corresponding to, e.g., the inner apex would be hugely different for the original vertex and the mapped one in Figure 5.15b. Finally, the function  $d$  is not defined for vertices not on the boundary, so the value  $\mathbf{D}_{ij}$  equals zero if any of the vertices is not on the boundary (anyhow the correspondence between vertices of the boundary with vertices in the interior is rejected by the previous cost).

These costs are computed in pre-processing and normalised, so that the maximum value of each cost equals 1 and the minimum one equals zero. Then, we simply search for the permutation (mapping) of the vertices of the fragment which minimise the sum of these costs:

$$\begin{aligned}
\min_{\mathbf{P}} \quad & \sum_{i=1}^{|\mathcal{V}_{a_1}|} \sum_{j=1}^{|\mathcal{V}_{a_2}|} (\beta \mathbf{B}_{ij} + \gamma \mathbf{C}_{ij} + \delta \mathbf{D}_{ij}) \cdot \mathbf{P}_{ij} \\
\text{s.t.} \quad & \mathbf{P}_{ij} \in \{0, 1\}, \forall i \in \{1, \dots, |\mathcal{V}_{a_1}|\}, j \in \{1, \dots, |\mathcal{V}_{a_2}|\} \\
& \sum_{j=1}^{|\mathcal{V}_{a_2}|} \mathbf{P}_{ij} = 1, \forall i \in \{1, \dots, |\mathcal{V}_{a_1}|\}
\end{aligned} \tag{5.1}$$

where  $\alpha$ ,  $\beta$  and  $\gamma$  are weights useful for defining the *importance* of each cost for the mapping. The presented minimisation problem is solved employing the Google OR-Tools library [PF19].

2. *Non-rigid alignment*: we want to find the minimal movement that aligns the template vertices with corresponding vertices of the fragment, without distorting too much the initial *displacement* of vertices. Let  $\mathbf{V}'$  be the  $|\mathcal{V}_{a_1}| \times 3$  matrix containing the the vertices in  $\mathcal{V}_{a_1}$ , in the current configuration,  $\dot{\mathbf{V}}'$  the  $|\mathcal{V}_{a_1}| \times 3$  matrix with same vertices in the original configuration (time 0 configuration of the template) and  $\mathbf{V}''$  the  $|\mathcal{V}_{a_2}| \times 3$  matrix containing the vertices in  $\mathcal{V}_{a_2}$ , the resulting optimisation problem is the following:

$$\min_{\mathbf{V}'} \quad \sum_{i=1}^{|\mathcal{V}_{a_1}|} (w_1 \left\| \mathbf{V}'_i - \dot{\mathbf{V}}'_i \right\|_2^2 + w_2 \left\| \mathbf{V}'_i - (\mathbf{P}\mathbf{V}'')_i \right\|_2^2 + w_3 \left\| \Delta \mathbf{V}'_i - \Delta \dot{\mathbf{V}}'_i \right\|_2^2) \tag{5.2}$$

where  $\Delta$  is the discrete Laplace operator. This problem can be solved by employing again the ShapeOp library and applying to the template vertices corresponding to the annotation two ‘‘Closeness’’ constraints (one for staying in the initial position and one for moving towards the position on the fragment corresponding to the vertex mapped with the previous optimisation problem) and one ‘‘Laplacian Displacement’’ constraint for the third cost.

Iterating over all the mesh annotations, exploiting the containment relationship and starting from the leafs towards the root, it is theoretically possible to obtain the desired result. Our first experiment in this direction gave us quite satisfactory results (see Figure 5.16), with the reached result obtained in less than 9 seconds for a template shape at 25K vertices resolution and fragment shape at 10K vertices resolution and about 36 seconds for the same experiment with a 50K vertices fragment.

After the non-rigid registration of the template over the fragment, the vertices of the template corresponding to the fragment are frozen in place, while some constraints may be violated (in particular those related to proportions, which are very important in archaeology - see next subsection). So at this stage the system calls the optimisation engine to fix the violation moving the remaining of the template shape.

However, when more than one fragment is placed on the template, it may happen that the two fragments are not compatible, meaning that the optimisation engine is not able to give a satisfactory result because the problem is over-constrained. This may happen either because fragments belong to objects from different classes or more likely, their dimensions do not match: in other words, the template cannot be deformed to fit their proportions simultaneously. This is a very important feature, when trying to re-unify sparse fragments belonging to the same piece. For example, by placing a head and a foot fragment on a common template, it can become evident if their dimensions do not match, indicating they can't be part of the same original object.

To support informed decision by the user in such a case, the error values for each high-level constraint are displayed in a dedicated view of the developed GUI (see Appendix A). The framework can also suggest to reject fragments introducing an error higher than a certain threshold after the optimisation process.

### 5.3.2 A constraint based on “proportions”

As introduced in Section 1.1, artistic production in many ancient cultures was based on the application of canons, where standard proportions between shape parts were encoded. For the Egyptian human figure, a grid was used so that the created object would resemble the idea of perfect or harmonious shape for the era where the specific canon was employed (think, for instance, to what we mean by “well proportioned”).

Following the same concept, this kind of standard or style is deeply interleaved with the definition of belonging to a specific class of shapes in CH. So, developing a constraint for defining proportions can be considered critical for this application.

This high-level constraint is bi-directional: indeed, if we say that a specific part A is twice as long<sup>5</sup> than another part B, it means that B has half A's length. While this concept is quite

---

<sup>5</sup>The concept of length here needs to be treated with care: indeed, we can think about different metrics (e.g.,



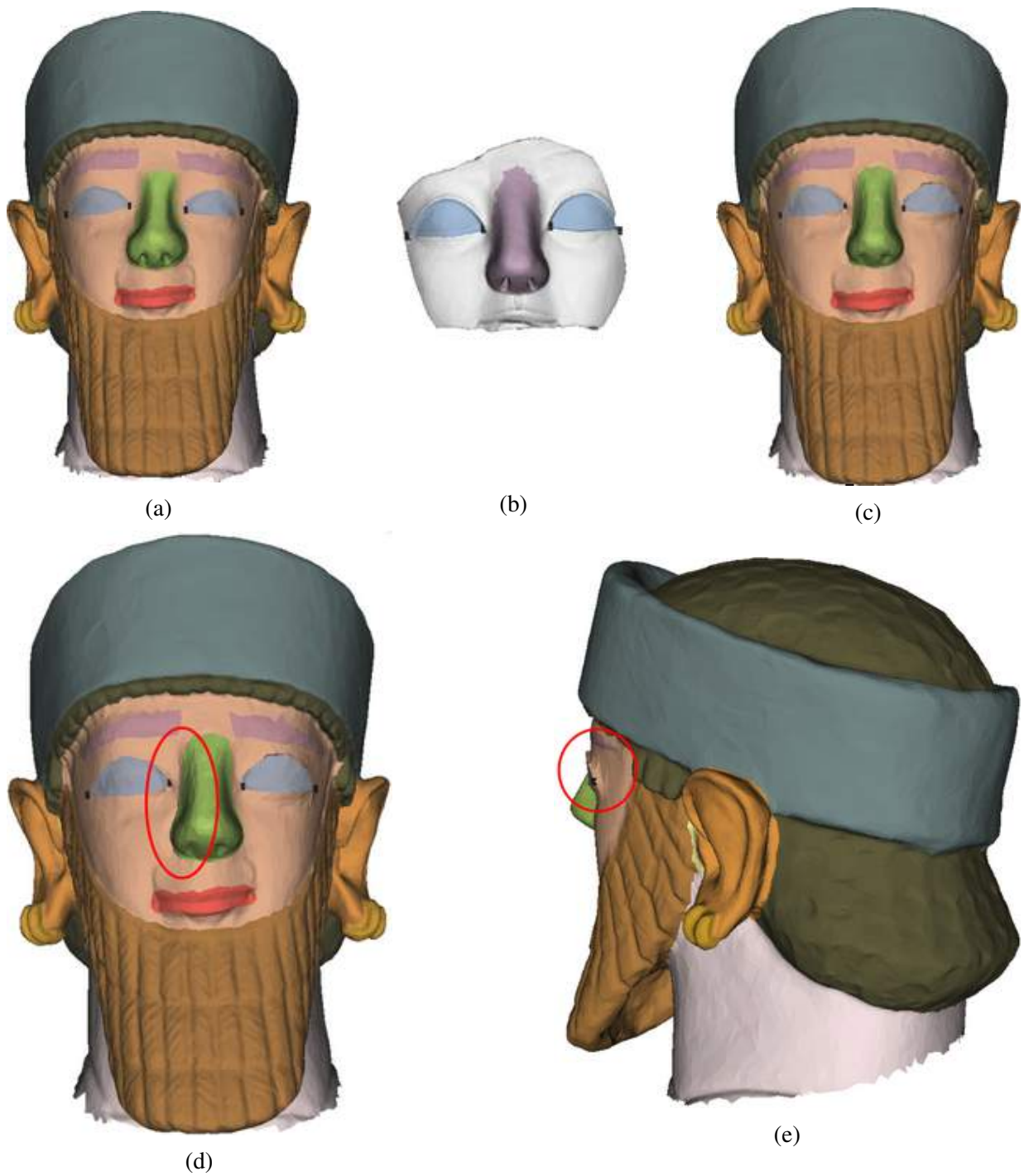


Figure 5.16: *The first result in the non rigid fitting step: (a) template, (b) annotated fragment, (c) deformed template, (d) first distortion highlighted, (e) second distortion highlighted.*

obvious semantically, it is not so trivial to define the expected geometric behaviour during the deformation: when the proportion constraint is violated which part should be changed? For example, if the current length of B is  $\frac{2}{3}$  of that of A, should A be elongated or rather B be shortened?

For this reason, the domain expert is asked to define which part should change to keep the constraint satisfied. In the following, we assume that the part to be modified is the first one to be selected.

This high-level constraint is implemented as a combination of two geometric constraints:

1. “Proportion” constraint, a new constraint that we implemented to establish proportions between parts.
2. “Laplacian” constraint (the same one used for the “Structural continuity” constraint). This is applied to the neighbourhood of all the vertices along the boundary of the part to enforce the smoothness of the result.

Notice that, for the constraint to work, we need to define exactly what vertices compose a part and which of them are involved in the measure. In the current state, the constraint only works for Euclidean distances but it has been generalised to indirect distances between vertices: indeed, commonly there is no vertex on the surface of the object specifying one or more extremes of a measure (see for example the statue in Figure 2.2: the base is hollow, so we do not have the base extreme in the middle for specifying the total height).

For working with this kind of situations, the framework provides the “bounding measure” tool (see Appendix A) which provides in output a pair of vertices and a direction axis, so that the measure can be computed using the projected position of the vertices.

Since proportions, at least in the CH field, are often not based on precise values, the framework provides the possibility to define a range into which the constraint is considered valid (namely defined by  $m$  (min) and  $M$  (max) values).

The projection for the “Proportion” constraint is obtained through:

1. Centre the involved points on their average position;
2. Compute the corresponding measures of the two part  $l_1$  and  $l_2$ ;
3. Compute the amount of stretch required for  $l_1$  to meet the constraint as:

$$p = \begin{cases} \frac{l_2 \cdot M}{l_1} & \text{if } l_1 > l_2 \cdot M \\ \frac{l_2 \cdot m}{l_1} & \text{if } l_1 < l_2 \cdot m \end{cases}$$

---

Euclidean and geodesic) so that the same two points can be at different distance depending on the metric definition (see Chapter 2)

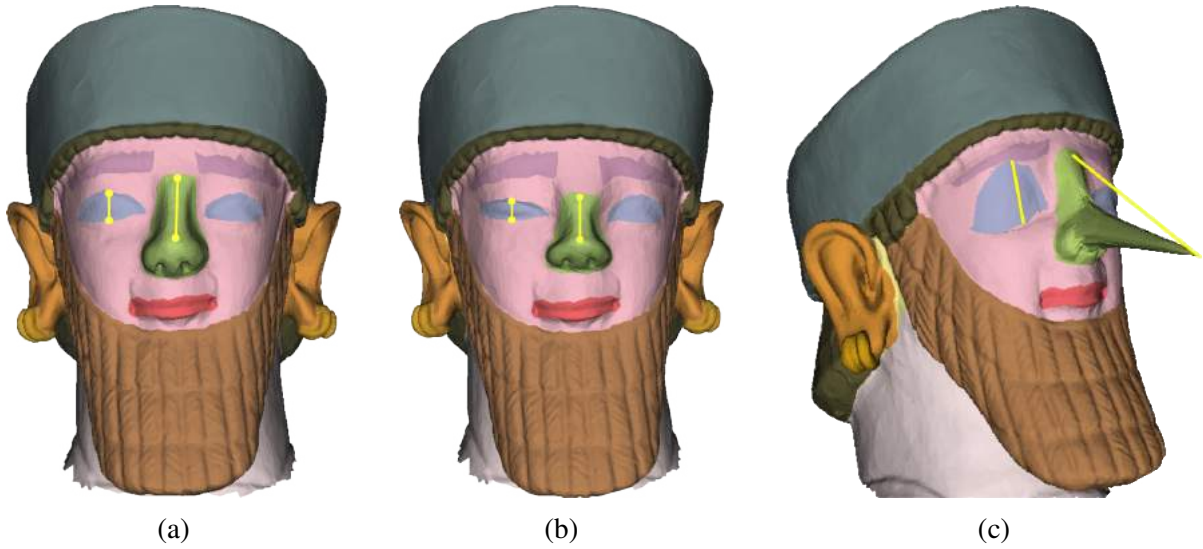


Figure 5.17: (a) the “height” of the “eye” and of the “nose” parts; (b) eye shrunk and corresponding modification of the nose, (c) nose elongation and corresponding modification of the eye.

4. Project the vertices  $\mathbf{v}_i$  of the first part on the axis  $\mathbf{a}$  defining the measure<sup>6</sup>  $l_1$  as

$$proj_{\mathbf{a}} \mathbf{v}_i = \frac{\mathbf{v}_i \cdot \mathbf{a}}{\|\mathbf{a}\|_2} \cdot \mathbf{a}$$

5. “Scale” the vertices’ position as  $\mathbf{v}_i'' = p \cdot proj_{\mathbf{a}} \mathbf{v}_i$

6. Apply the inverse projection to the new positions as  $\mathbf{v}_i' = \mathbf{v}_i'' + \mathbf{v}_i - proj_{\mathbf{a}} \mathbf{v}_i$

Some results of the application of this constraint can be seen in Figure 5.17, for a template mesh with ~25K vertices, without cage extension and with maximum number of iterations of the solver allowed set to 5. The result in Figure 5.17b is obtained by constraining the measure between the root of the “nose” and its tip to be 3 to 4 times the “height” of the “eye”. Then the “height” of the “eye” is shrunk and the “nose” follows to keep the constraint satisfied. Conversely, in Figure 5.17c, the “height” of the “eye” is constrained to be 25% to 33% of the measure between the root of the “nose” and its tip. After the elongation of the “nose”, the “eye” stretches to keep the constraint satisfied.

<sup>6</sup>Notice that if the measure is directly computed between two vertices, the scaling axis will be the line passing through them, while if the measure is indirect the scaling axis is given by the definition of the measure

## 5.4 Discussion

In this Chapter, we provided an overall description of the implemented system and presented the encouraging results obtained applying the proposed framework in two application scenarios, namely product design and archaeological reconstruction; we discuss in detail the design and implementation of three semantic constraints useful in the application scenarios.

The quality of such results has currently been estimated only qualitatively, while their quantitative analysis is planned to be analysed during broad user testing in future works, together with the usability and effectiveness of the overall framework.

Even such a qualitative analysis, however, highlighted several problems related to the proposed high-level constraints. As an example, we refer to the “Same Level” constraint (see 5.2.1) and its application to the teapot: while achieving acceptable results, it is not possible to keep the constraint without distorting the geometry around the flowers. This is because the library is not aware of the fact that the flowers are an addition to the original shape of the teapot, so that probably we should deform the teapot first and then re-apply the flowers or, at least, add a constraint that would mimic this behaviour.

This is even more evident in the application of “Structural continuity” on the chair case: the library is not aware of the fact that, for a “Co-axiality” constraint to have sense, the part to be constrained must naturally admit an axis itself. Unfortunately, the optimisation works on surface points only while, logically, the constraint should apply to the axis points and the modification should fall back to the surface points as a consequence.

The result can be seen in Figure 5.18, where a subset of the points of the constrained part is moved in such a way that the extracted “axis” do not follow the shape of the part, so the rotation performed to align the axes gives a bad result messing up the part’s shape.

A final example regards the stretch direction for the “Proportion constraint” (see Figure 5.19). Here, the little finger’s length is required to be from 70 to 90% of the ring finger’s length and, after the stretch of the ring finger, the little finger stretches coherently. However, since the optimisation is oblivious to the object’s structure, the stretch is performed both towards the outside and the inside of the hand, thus leading to counter-intuitive results.

These artefacts are not trivially solvable, and we believe that a promising new research direction would be the design and development of a new optimisation engine that is semantics-aware.

We still need to improve the non-rigid fitting module of the proposed pipeline for archaeological reconstruction. Indeed, the current version has some critical issues:

- *Annotation correspondence*: while understanding what are the corresponding parts between shapes is quite trivial for a human being, we are still missing a way for defining correspondences between annotations. The current implementation compares the selection

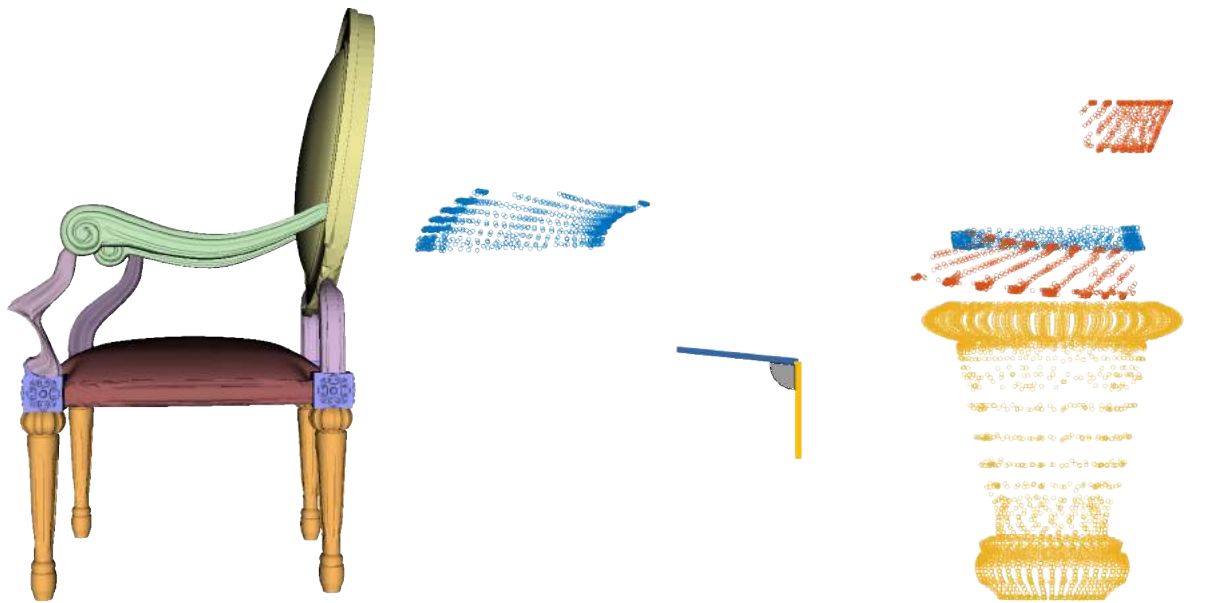


Figure 5.18: *The issue related to the manipulation of a subset of the constrained part in the “Co-axiality” constraint. On the left, bad result due to this issue; on the right, the blue points correspond the constrained part, which are moved in the position of the orange points after the projection*



Figure 5.19: *The issue related to the “Proportion” constraint. On the left, the original shape with the measures highlighted. In the middle, the ring finger is stretched and the little finger follows. On the right, a zoomed-in view from above of the little finger (the darker part is the interior of the hand).*

type (i.e., if both the annotations have a point, line or region selector) and tag of annotations, but the tag is ambiguous (what happens if both the eyes of the human body are annotated as “eye”?);

- *Landmark correspondence*: the proposed pipeline starts from the assumption that the number of landmarks contained in corresponding annotation is the same, e.g., there are 2 landmarks contained in the “eye” (inner and outer apex). This may not be the case, indeed, if we refer to the floral pattern in Figure 5.2, we may have flowers with different number of petals, that intuitively leads to a different number of landmarks (petals extrema). While this situation can be solved by requiring that different flowers should not be in correspondence, others can be more complex: how do we deal with a broken part? indeed, since we are trying to reconstruct archaeological fragments, we may end up with a fragment containing only a partial representation of a part (e.g., half of an eye), thus allowing only the annotation of a segment of the part, possibly not including one or more landmarks;
- *Outline correspondence*: apart from defining correspondences between annotations, we need to define mechanisms for finding the corresponding *outline* of regions: indeed, as can be seen in Section 2.4.1, regions can be defined by more than one outline and this makes the definition of the cost  $D_{ij}$  more complex;
- *Vertices correspondence*: some vertices of the template may not have any corresponding vertex on the fragment, either because the parts which they belong are not annotated or because there is no counterpart on the fragment. This leads to discontinuities in the output surface (an example can be seen in Figure 5.16e);
- *Scarce testing*: we have currently tested the approach only for the non-rigid fitting of one template mesh over one fragment.

So, in future works, we need to tackle these issues and set up a deep test phase for the proposed pipeline.

Finally, a necessary future activity regards the validation of the proposed high-level constraints on a broader data set of shapes: as an example, we should apply the structural continuity constraint to a number of man-made objects (e.g., lamps, furniture, etc.) and gather the obtained results. Besides, more constraints should be defined in order to make the framework concretely useful as a semantics-aware modelling system.

# Chapter 6

## Discussion and future works

The purpose of this Chapter is a thorough discussion about the proposed framework and the results achieved in the applications described in the previous Chapter. The preliminary results have opened up numerous ideas for improvements, further applications and future research, which are presented at the end of this Chapter.

### 6.1 Discussion

In this thesis, we presented a framework for the semantics-aware modelling of shapes, where some formalisation of the knowledge regarding a certain class of homogeneous objects is exploited for easing the modelling tasks.

The framework allows the creation of new shapes by means of constrained deformation, i.e., allowing to manipulate the shape in compliance to some relationships that are crucial for an object to be part of a certain class, and provides tools for enriching the formalised information regarding the class by analysing the class itself or a representative shape. The outcome is a system prototype providing annotation, analysis and deformation functionalities in the same place to support modelling and reasoning about shapes.

Our proposal has been tested in two application scenarios, namely product design (see Section 5.2) and archaeological reconstruction (see Section 5.3), with encouraging preliminary results, but with the need of a broader user validation, since the framework has been tested only by a very small group of people with high expertise on the subject).

While there are several ways in which the proposed framework can be extended and completed, the purpose of this thesis is the proposition of a new methodology or paradigm for manipulating 3D shapes: indeed, we are proposing a step forward in the encoding and deployment of the

semantics related to objects sharing common properties, which will be used as a base for the creation of smarter tools and approaches for dealing with modelling tasks.

In the following, we will discuss the main limitations of the proposed framework.

### 6.1.1 Lack of a semantics-aware generation of cages

As discussed in Section 4.9, it is possible to speed-up the deformation by manipulating the cage vertices rather than those of the template directly.

However, the quality of the final result heavily depends, besides the choice of a suitable GBC, on how the cage has been generated. Concerning the resolution of the cage mesh, if too many DoF are available, the deformation speed would be dramatically slowed down, while too few of them would not allow a fine control on the deformation. In Figure 6.2 and 6.1, some results obtained using cages generated following a common resample-and-offset approach can be seen. However, not only the mesh resolution determines the final quality of the deformation. Indeed, the position of cage vertices seems to play an important role.

While the creation of an optimal cage for every shape is still an open issue, we argue that enveloping the semantics in the generation of the cage would allow to insert DoF *only where required* in correspondence to key features, which will be the focus of user manipulation.

An interesting approach in this sense is the one proposed by Xian et al. in [XZG13], where the resample-and-offset approach is driven by the outlines of patches identifying interesting features.

So, we aim at extending and generalising this approach to analyse any kind of selection treated in this framework (namely Point, Line and Region, see 2.2.2).

Another possibility is to employ a multi-LoD cage, both as in [SVJ15] and as in [GPCP13], where the former gives means to employ cages at different resolutions referring to the same target shape, and the latter defines a hierarchy of cages controlling smaller and smaller parts of the shape.

### 6.1.2 Need for a “semantic” optimisation

The definition of constraints is still too much geometry-based: while the ShapeOp library provided a very useful tool for the definition of constraints, it is limiting when trying to define a constraint that is intuitively related to the concept of “part” rather than to “points”. Our approach was to encapsulate the geometric constraints into some high-level constraints to be used by domain experts; however, in our experiments, we found that a semantic constraint can be translated as a set of geometric constraints only to some extent. The impossibility of exploiting the semantics of parts within the optimisation phase still limits what can be achieved.



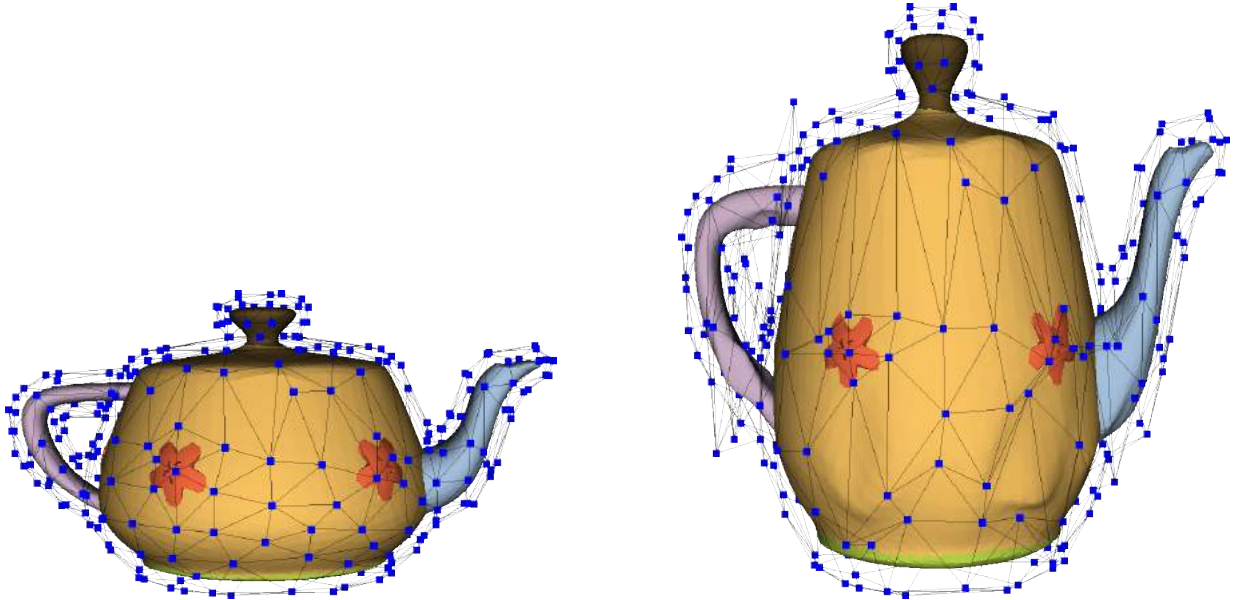


Figure 6.1: *From left to right: the original shape with the corresponding cage; the result after the stretch upwards.*

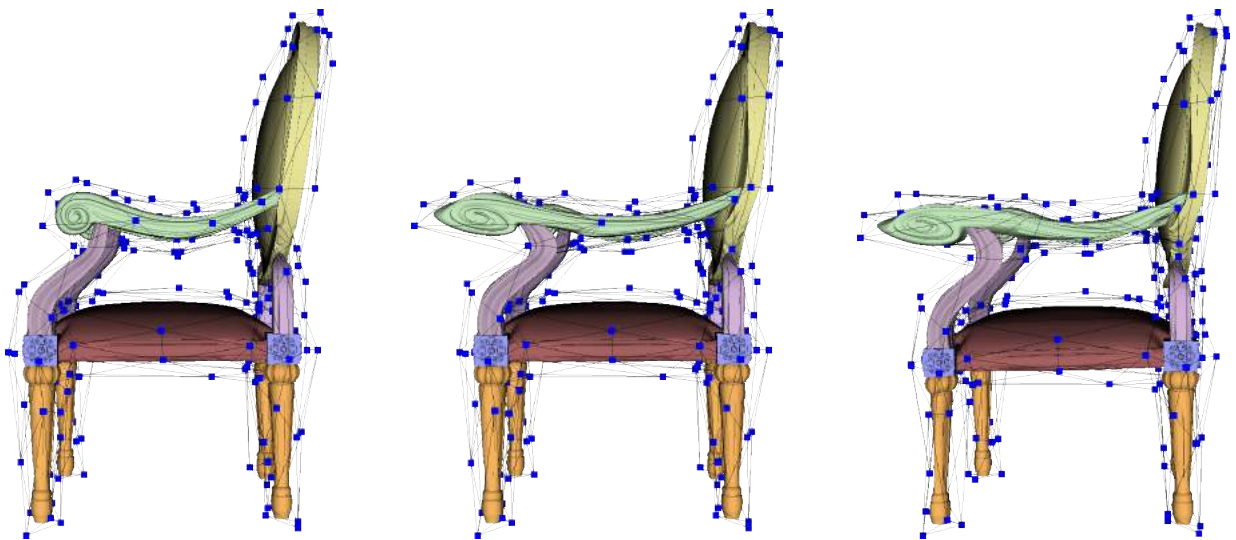


Figure 6.2: *From left to right: the original shape with the corresponding cage; the result after the stretching of the arm; the result trying to rotate the arm's support (the constraint does not allow it, it moves other vertices).*

An in-depth analysis of the results achieved in the experiments is given in Section 5.4.

### 6.1.3 Work in progress

As can be easily seen in Appendix A, we set up a wide framework that provides many functionalities but is not complete; however, it can be extended in a modular fashion. For instance, tools for the automatic generation of statistical models (see Section 1.1) and for the import of external knowledge sources (e.g., thesauri and ontologies) could be added. Moreover, we plan to add functionalities to compute and store more complex annotation attributes (e.g., curvature maps, axis).

Another incomplete part is the non-rigid fitting module of the archaeological reconstruction pipeline. Indeed, the management of multiple outlines with region annotations (i.e., selections of general topology) introduce complex issues, as well the presence of inconsistent landmarks number referred to corresponding annotations and the lack of parts of the geometry (an in depth discussion of these issues can be found in Section 5.4).

A further interesting potential use of the template that has not been developed yet is for classification purposes: indeed, we could fit a template carrying annotations, attributes, relationships and constraints that are crucial for an object to be considered part of the homogeneous class. The deformation error with respect to the different constraints (see Chapter 4) can be used to understand if a query shape is part of the class represented by the parametric template.

A function to propagate the semantics associated to the template to new shapes in the same class would be another important improvement (e.g., [KLM<sup>+</sup>13]). For instance, we would exploit the annotation of models in a class to automatically detect parts and analyse the variability of attributes (whereas in Section 3.3 the extraction of heads and faces was done manually).

Finally, the set of high-level constraints for the deformation needs to be further enriched, especially for future application scenarios (e.g., see Section 6.2.1 and 6.2.2). Notice that any other interested researcher in this field can add new constraints: indeed the code is open source on GitHub [Sca20] and the ShapeOp library allows to enrich the set of predefined constraints just by providing the required projection (see Chapter 4).

## 6.2 Further Applications

In the following, we present some possible further application scenarios.

### **6.2.1 Investigation of biological species**

Our approach can be useful to study the morphological variability and functional capacities of biological species. For instance, we will use our approach within the starting EMPHASIS project. Funded by the “Programma Nazionale di Ricerche in Antartide (PNRA)” with code PNRA18\_00106, the goal of the project is to develop an eco-morphological study, based on the analysis of traits of the feeding apparatus of nototheniid fish species representative of various phylogenetic lineages. The proposed eco-morphological study consists in the analysis of various structures of the buccal apparatus of nototheniid specimens available in Italian and international museums. Biologists believe metrics of peculiar traits of head, mouth and gill rakers will allow to calculate significant indices, from which it is possible to infer the feeding mode, connecting design and performance.

The fundamental contribution of our research to the project will be the simulation and visualisation of the structures involved in feeding activity of the fishes and their movements during the feeding actions. To produce a realistic 3D rendering of the process, it is necessary to build a 3D digital model resembling the fish with the appropriate measures and proportions. The relation between the external surface and the structural parameters subject of this study is twofold: on the one side, from a digitalised surface that represents faithfully the shape of the fish, it is possible to measure some morphological traits directly on the digital domain, in a totally non-invasive way; on the other side, if the surface is modelled by a designer, the parameters acquired by direct measure or from other sources (e.g., x-ray) can be used to define class-dependent constraints to drive the deformation of a representative shape to resemble the species at hand closely.

### **6.2.2 Generation of random shapes belonging to a same homogeneous class**

The ability to generate novel, diverse, and realistic 3D shapes along with associated part semantics and structure is central to many applications requiring high-quality 3D assets or large volumes of realistic training data [AKZM14, MGY<sup>+</sup>19]. Using the proposed framework, this could be done by applying “random” deformations on the template shape, provided that they satisfy the constraints. There is a rich literature regarding this application context and we expect to obtain competitive results applying our approach.

### **6.2.3 Classification of shapes based on non-rigid fitting**

As already anticipated throughout the thesis, a possibility for future works is the exploitation of the error introduced by the non-rigid fitting of the template shape over the query one for its classification. In fact, as introduced in Chapter 4, the constrained deformation is expressed as an optimisation problem: it follows that, in certain conditions, a constraint could not be satisfied,

or several constraints could not be satisfied at the same time. The optimisation engine will minimise the overall error, but an hypothetical classifier could exploit the information about which constraints could not be fully satisfied and at which extent. Indeed, since the template is expected to capture the main traits of the objects within the class in form of constraints between the parts composing them, violating too much the constraints can be interpreted as belonging to a different class.

# Appendix A

## Graphical User Interface

In this Appendix, we provide a presentation of the developed system GUI, which offers all the functionalities to interact with the framework proposed in this thesis: build the template through manual annotation, enrich the semantics through measuring and analysis tool, add constraints on annotation attributes and relations and perform semantics-aware deformations of the geometry.

### A.1 Main window

When the software is run, the main window appears, with a menu bar, a toolbar, a sidebar and a central canvas, initially representing the default shape of a tetrahedron, where objects will be visualised (see Figure A.1). The menu bar allows to perform the most common operations (see Figure A.2):

- *File* menu: provides the load, save and close operations for the various entities. The system handles two kinds of entities, namely, barycentric coordinates and 3D models (the template, the cage or other generic meshes representing specific objects in the class or object fragments, which the user wants to investigate);
- *Edit* menu: includes
  - undo/redo commands;
  - choice and computation of the GBC (actually, the list of generalisations includes MVC and GC, although the selection of the GC generalisation precludes the possibility to use the constrained deformation environment – it can be used for pure geometric deformations);

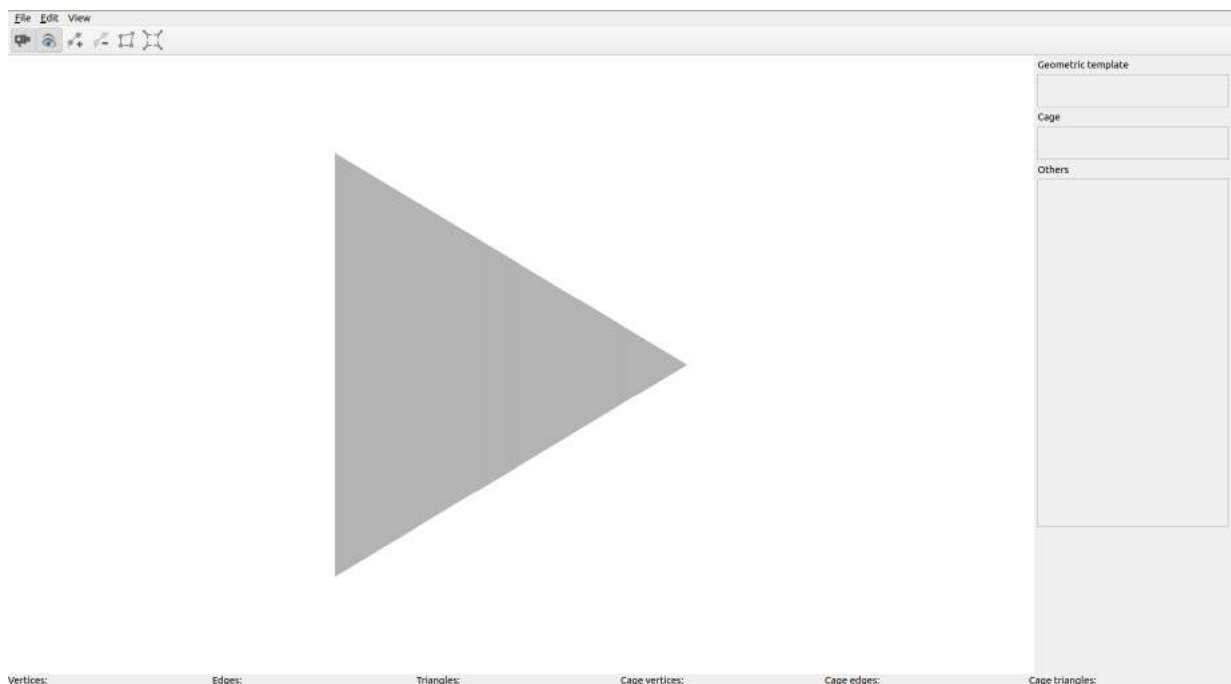


Figure A.1: *The Main window*

- generation of a cage (presently with a simple resample-and-offset approach based on two filters provided in MeshLab [CCC<sup>+</sup>08]: the “Uniform Mesh Resampling”, which allows to offset the resulting mesh with respect to the original one (we normally use 55% offset with check on “Clean Vertices”), and the “Simplification: Quadric Edge Collapse Decimation” with “Target number of faces” depending on the complexity of the shape and checking on “Preserve Boundary of the mesh” (weight 1), “Preserve Normal” and “Preserve Topology”).
- *View* menu: allows to change the colour of the geometric template and of the cage and to change the visualisation modality: presently the system includes points, edges and surface visualisation, which can be combined as preferred by the user. The default is only surface visualisation for both template and generic meshes and only wireframe and points for the cage.

The sidebar provides three different views, organised according to the entities involved:

- *Layer view*: displays the list of meshes actually in use, separated into “Geometric template”, “Cage” and “Others”: only one geometric template and cage at a time are allowed, while there is no limitation (at least in theory, of course it is restrained by the memory availability) for the other meshes.

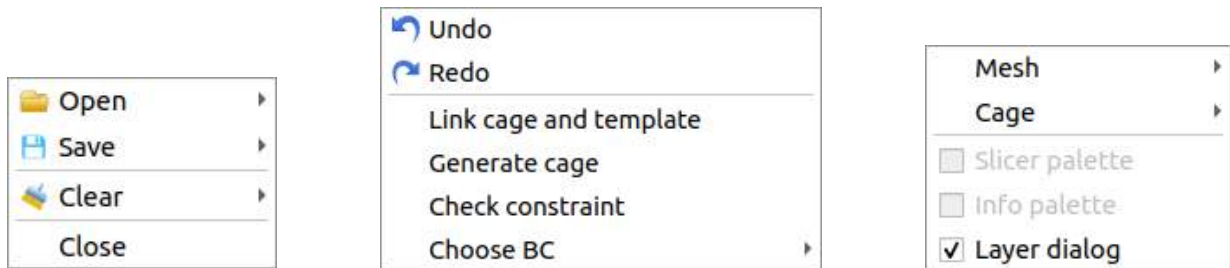


Figure A.2: From left to right: File, Edit and View menu

- Slice view: provides several shape analysis tools based on the slicing paradigm; these apply to the template model (including the implementation of the method in [SVM<sup>+</sup>18] described in Chapter 3);
- Constraints view: displays information about the high-level constraint defined by the domain expert.

In the Layer view, the user can interact with each mesh by clicking with the right button of the mouse on it. The interaction includes the possibility to change the visualisation options (as in the View menu) as well as other entity-specific options (see Figure A.3):

- Cage: allows to close the mesh (e.g., the user wants to use a finer or a coarser cage);
- Geometric template: in addition to the cage options, it exhibits the “Edit annotation” functionality, which allows to open the Annotation Window (see A.2, and the “Show annotation” checkbox, used for showing/hiding the annotations on the mesh);
- Generic mesh: has all the options of the geometric template, with the addition of the “Adapt template to object” option, which starts the fitting procedure (see Section 5.3).

The Slice view allows to extract information of several kinds from the template shape. It is equipped with three different sheets (see Figure A.4) enclosing tools for setting and moving the plane which is used to slice the shape (sheet “Mesh”), tools for computing some shape descriptors of a slice (sheet “Slice”) and computing and visualising the associated Medial Axis approximation at different levels of detail (sheet “Skeleton”). The slice under analysis is shown both in the main canvas (over the geometric template’s surface) and in a smaller canvas in the upper part of the Slice view.

The Constraint view presents a drop down list for selecting the high-level constraint whose information the user is interested in. Then, the frame below is populated with the information associated to the constraint, e.g., id, type, etc. (most of information is constraint specific). This view is particularly useful for visualising the error with respect to a certain constraint. Indeed, as

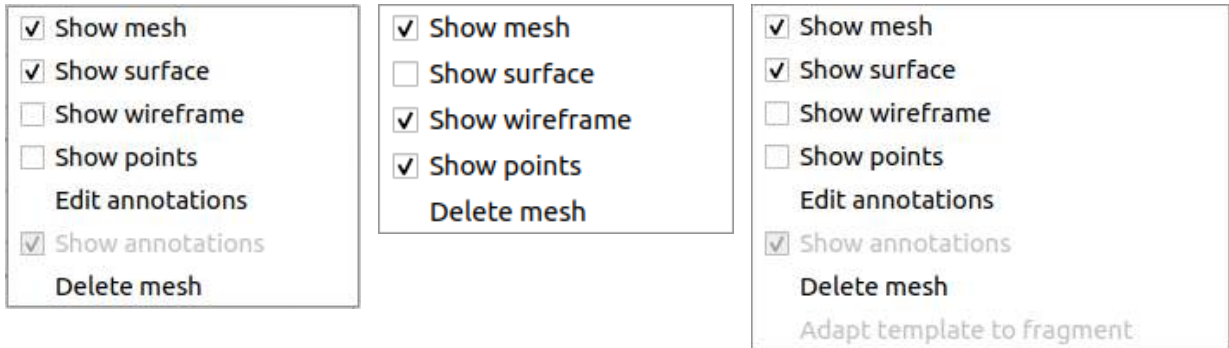


Figure A.3: On the left, the contextual menu corresponding to the right click on the template item, on the middle the one referred to the cage and on the right the one corresponding to a generic mesh.

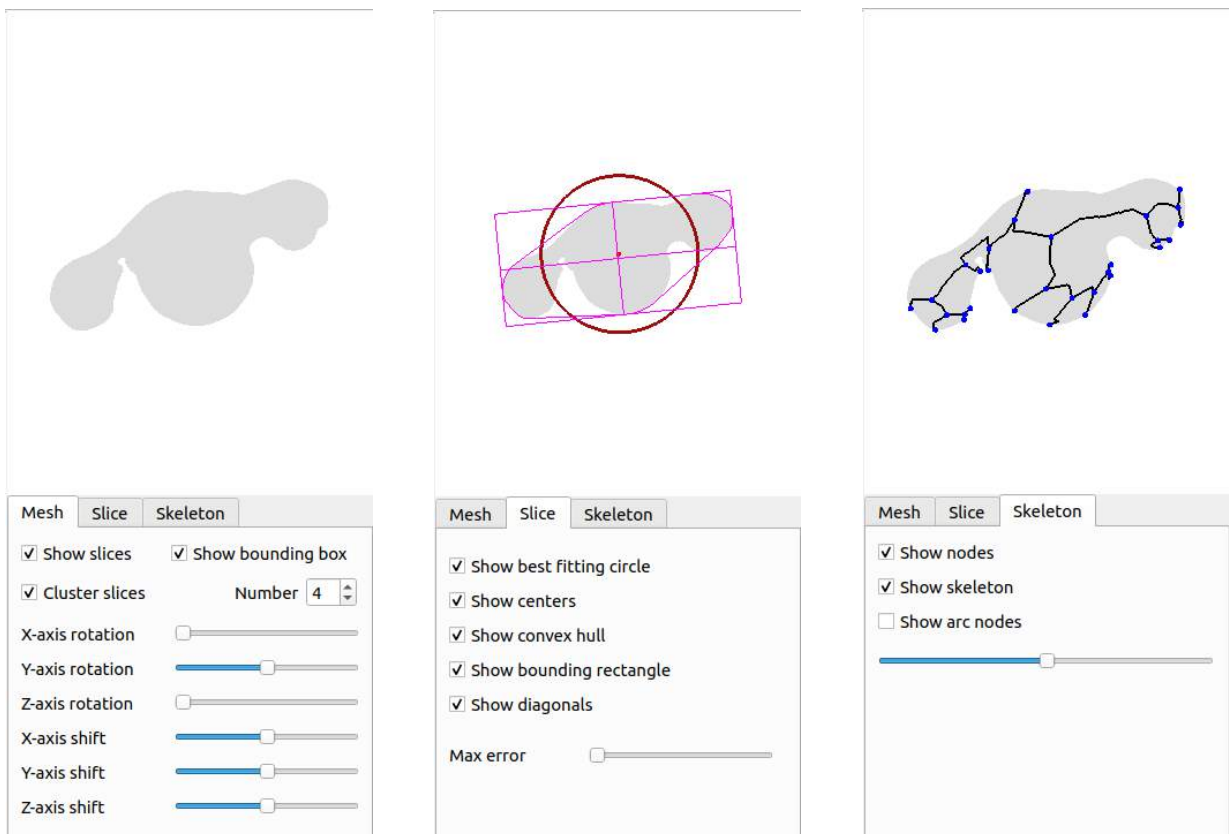








Figure A.4: The three sheets in the Slice view.



introduced in Chapter 4, the constrained deformation is expressed as an optimisation problem. It follows that, in certain conditions, a constraint could not be satisfied, or several constraints could not be satisfied at the same time. Anyway, the ShapeOp library will minimise the overall error, but the user must be aware of which constraints could not be fully satisfied and at which extent. The user may then take an informed decision about whether to accept, reject or change the deformation, e.g., when performing reassembly tasks (see Section 5.3).

After loading the geometric template and the cage (or computing it with the provided functionality), the “Link cage and template” action in the *Edit* menu is enabled (this action will compute the GBC) as well as the “Load coordinates” action in the *File* menu. After the computation/loading of the GBC (it may require some time, depending on the resolution of the template mesh and of the cage), the toolbar is enabled, together with its buttons, which provide the functionalities for selecting cage vertices and using them as handles to define a deformation:

-  The Camera button, when checked, allows only to change the view around the shape, without any other kind of interaction;
-  The Visible Selection button, when checked, enables a filter which allows selecting only of the vertices that are visible from the user point of view (by default it is checked);
-  The Vertex Selection button, when checked, allows to select vertices of the cage, either individually, or all those enclosed in a rectangle drawn by the user (the rectangle selection is performed by holding the Ctrl button on the keyboard and simultaneously left clicking on the canvas, dragging and releasing the left mouse button, see Figure A.6), or all those associated to an annotation (this is obtained by holding the Ctrl button and pressing the middle button of the mouse on the annotation of interest and employs the technique defined in 5.2.3);
-  The Vertex De-selection button, when checked, allows to perform the opposite operation of the previous button;
-  The Move Vertices button, when checked, allows to rigidly move the selected vertices of the cage. There are two possibilities: to drag (translate) the selected vertices following the mouse position (Ctrl + right button of the mouse) or to rotate them around the axis orthogonal both to the camera direction and the mouse movement<sup>1</sup> of an amount defined by the magnitude of the mouse movement (Ctrl + left button of the mouse). The template’s shape is updated at a fixed frame rate;
-  The Stretch Vertices button, when checked, allows to move the selected vertices away from each other in a user defined direction. The operation is achieved by holding the Ctrl button on the keyboard, left clicking on the canvas and dragging in a certain direction: the stretch direction is computed using the current and starting positions, while the amount of stretch is constant and added every frame. As in the previous point, the template’s shape is updated at a fixed frame rate.

Finally, the Main window is equipped with a footer where the number of vertices, edges and triangles of the template and of the cage are shown.





## A.2 Annotation window

When the user clicks on the “Edit annotation” voice in the contextual menu of any mesh (cage excluded) a new window, called Annotation, is opened. Indeed, the user can either annotate the geometric template, thus actually adding the link with the semantic template, defined elsewhere as a knowledge formalisation or based on her/his own contextual knowledge, or annotate a specific object of the class for documentation purposes. It recalls the general schema of the Main window, with a menu bar, a toolbar, a sidebar and a central canvas where meshes and other entities are drawn.

Here, the user can select mesh vertices and add tags to create annotations, take measurements and add attributes to annotations, set relationships among annotations and set constraints on attributes and relationships.

In the current version, the menu bar contains the *File* menu only: it provides functionalities for saving, loading and clearing (deleting all) the annotations, clearing the selections and saving/loading the constraints/relationships.

The toolbar contains several buttons:

-  The Camera button, when checked, allows only to change the view around the shape, without any other kind of interaction;
-  The Restore Camera button, when pressed, resets the camera to the original displacement;
-  The Visible Selection button, when checked, enables a filter which allows the selection only of the vertices entities which are visible from the user point of view, depending on the current interaction modality (see next points - by default is checked);
-  The Eraser button, when checked, changes to the de-selection modality for discarding the newly selected entities (vertices, edges, triangles, annotations) depending on the interaction modality (see next points);

---

<sup>1</sup>If  $\mathbf{n}$  is the camera direction,  $\mathbf{s}$  is the click position of the mouse and  $\mathbf{e}$  its release position, then the axis has direction  $\mathbf{a} = \mathbf{n} \times \frac{\mathbf{e}-\mathbf{s}}{\|\mathbf{e}-\mathbf{s}\|}$  and passes through  $\mathbf{s}$

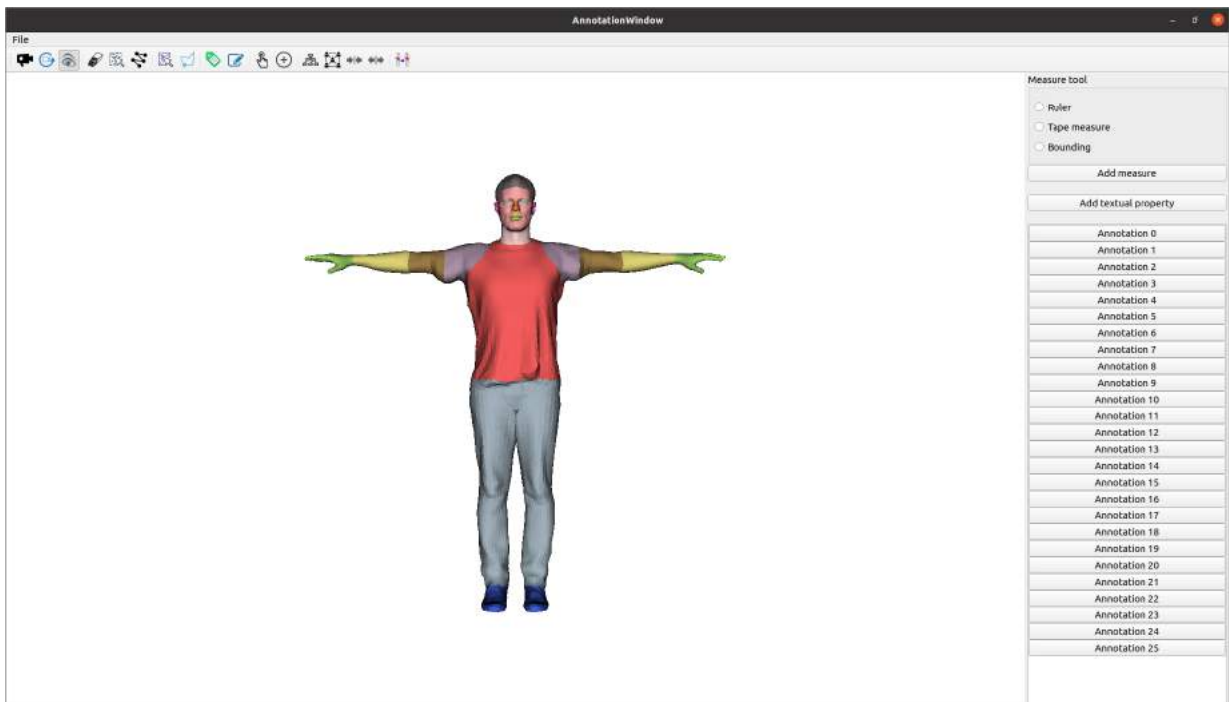
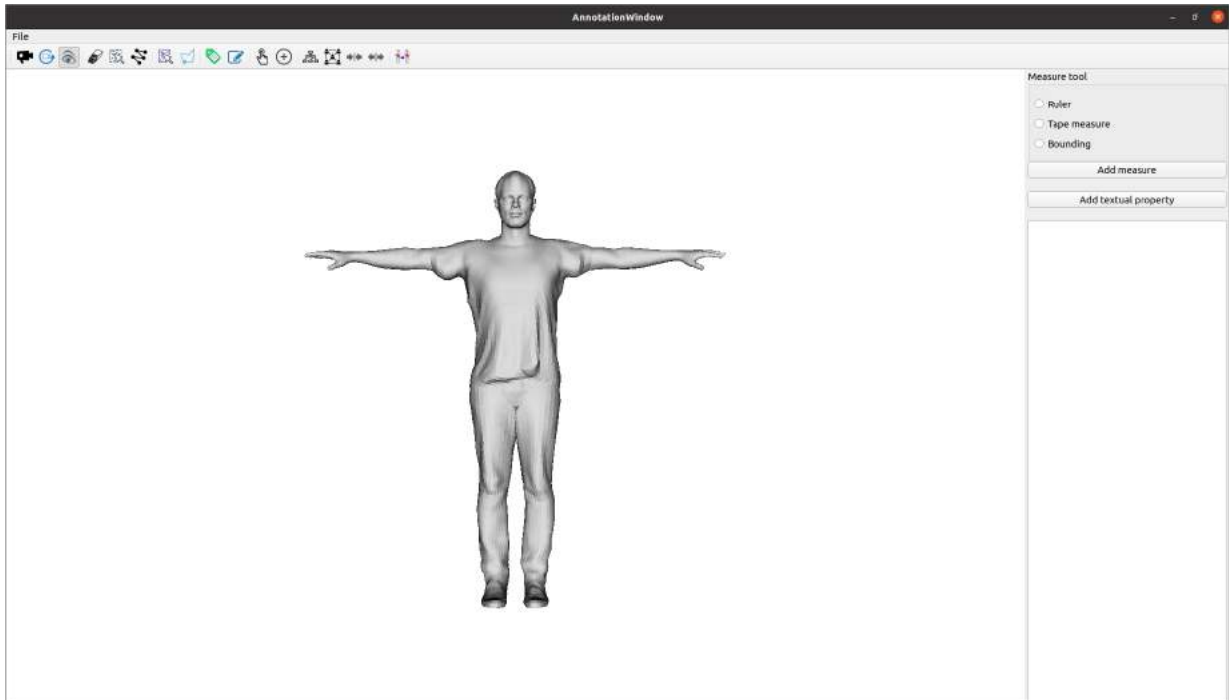







Figure A.5: *The Annotation window with (bottom) and without (top) annotations.*


- 


The Points Rectangle button, when checked, changes the selection modality to *points selection*, i.e., allows to select (or de-select if the Eraser button is checked) vertices of the mesh, either individually or all the ones included in a rectangle drawn by the user (following exactly the same procedure of the Vertex Selection button in the previous Section);
- 

The Polylines button, when checked, changes the selection modality to *edges selection*, i.e., allows to select (or de-select if the Eraser button is checked) edges of the mesh, by picking some successive points, in an additive way, on the surface: the path between them is automatically computed by the system following an approximation of the Dijkstra algorithm [Dij59] (the search for the shortest path is interrupted when the target vertex is found - see Algorithm 3);
- 







The Triangles Rectangle button, when checked, changes the selection modality to *triangles selection*, i.e., allows to select (or de-select if the Eraser button is checked) triangles on the surface of the mesh, either individually or all the ones included in a rectangle drawn by the user);
- 

The Lasso button, when checked, changes the selection modality to *triangles selection*, i.e., allows to select (or de-select if the Eraser button is checked) triangles on the surface of the mesh following the common lasso metaphor [Las20], i.e., the user draws a “polygon” on the surface picking successive points (as for the Polyline button, but now the polyline must be closed), then right-clicks in the interior of the “polygon” (this is necessary, since the interior of a polygon over a 3D surface is not well-defined) and the triangles enclosed in it are automatically selected);
- 

The Annotate button, when clicked, opens a dialog asking information about the annotation to be saved;
- 

The Edit button, when clicked, allows to modify the selection associated to an annotation (it requires that one and only one annotation is selected). The operation can be completed re-annotating the selection (thus using the Annotate button);
- 

The Annotation Selection button, when checked, changes the selection modality to *annotation selection*, i.e., allows to select annotations from the mesh surface. This can be obtained by holding the Ctrl key and left clicking over the annotation: if more than one annotation is under the mouse when the user clicks the left mouse over the surface, a dialog is opened to ask the user which annotation she/he wants to select;

-  The Add Constraint button, when clicked, displays a dialog to input the type and other properties of the constraint, based on the selected annotations. Generally speaking, the dialog allows to insert the weight associated to the constraint, the minimum and maximum value of a range (in the constraints where it makes sense). This button is shown only if the current mesh is the template;
-  The Relationships Graph Construction button, when clicked, performs the extraction of some basic relationships among the annotations, derived by geometric analysis (see Section 2.3). Currently, the containment and adjacency relations are extracted. This button is shown only if the current mesh is that of the template;
-  The Show Relationships button, when clicked, opens a window (see next Section) showing the relationship graph. This button is shown only if the current mesh is that of the template;
-  The Constrain button, when clicked, initialises the ShapeOp library to act on the constraints defined until now (this typically requires some time, depending on the resolution of the template shape and of the cage and on the number of defined constraints). This button can be found only if the current mesh is that of the template. This is disabled until a cage is loaded and the BC are computed/loaded;
-  The Withdraw Constraints button, when clicked, clears the ShapeOp library status, thus reverting to the state of the system before the Constrain button was clicked. This button is enabled only after the Constrain button is clicked;
-  The Transfer button is used to transfer the annotations from one source mesh to a target one. When clicked, it opens a dialog for the selection of a mesh file and then starts the transfer procedure (see sub-section 2.4.1). Currently, this switch the new mesh with the original one in the system.

The sidebar can be viewed as the composition of two different parts:

- In the upper part, the user finds the tools for adding attributes to annotations. To do this, the user has first to select one annotation (an error message is displayed if the user tries to add an attribute to no or more than one annotation at a time). As specified in sub-section 2.2.3, in this thesis we consider two kinds of attributes: qualitative and quantitative. To add a qualitative attribute, the user has to press the “Add textual property” button, which will open a dialog asking for a name of the attribute and some free text (there is no specific use for this kind of attribute, it can be used for taking notes, adding meta-data such as the place of finding of a piece. etc.). In the current version of the software, we have implemented only the possibility to add measures as quantitative properties, and in particular they can be taken using three kinds of tools (see Figure 2.2 and 3.6):

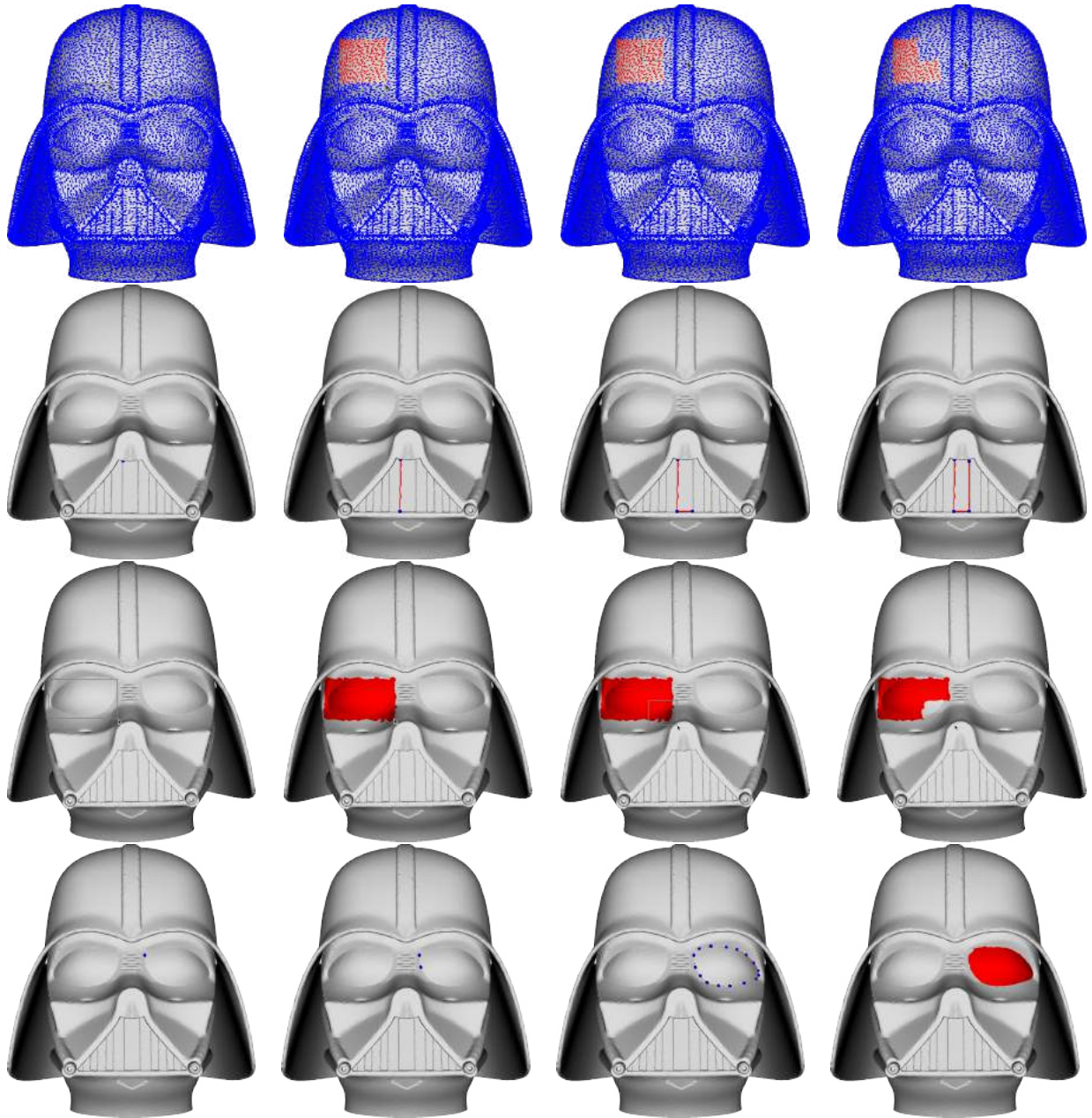


Figure A.6: *Different types of selections: going downwards selection of points, edges and triangles, there the last two rows depict two different tools for the triangles selection, namely rectangle and lasso selection.*

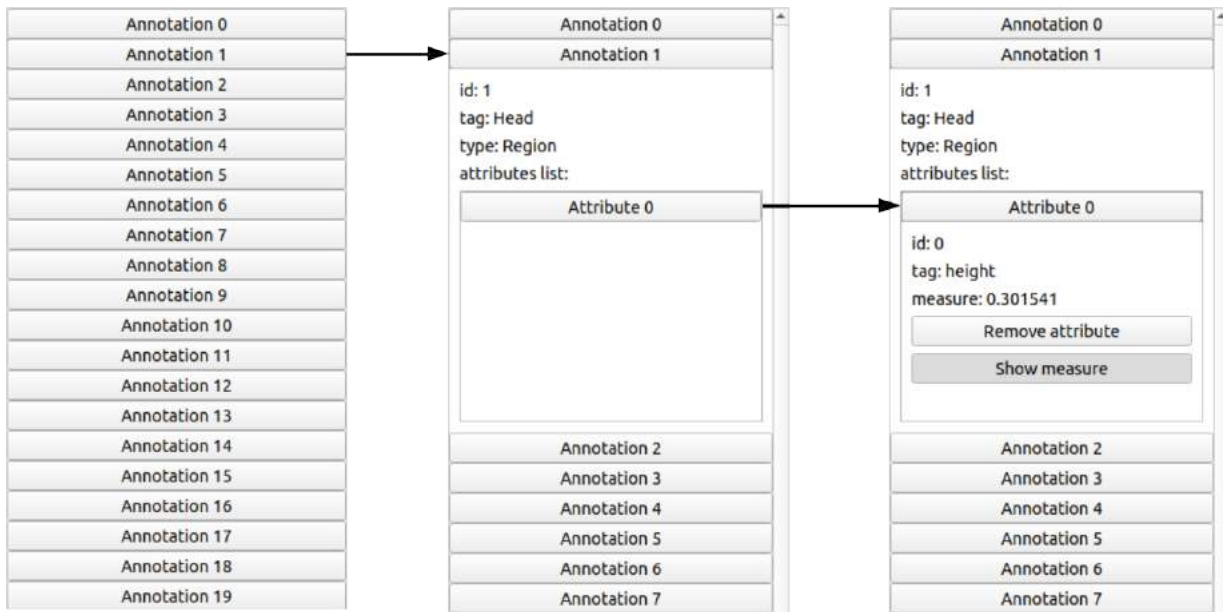
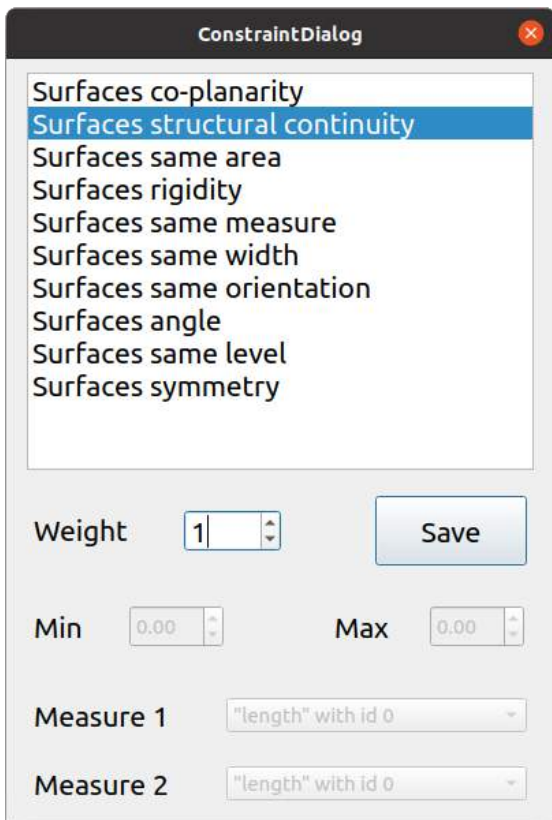
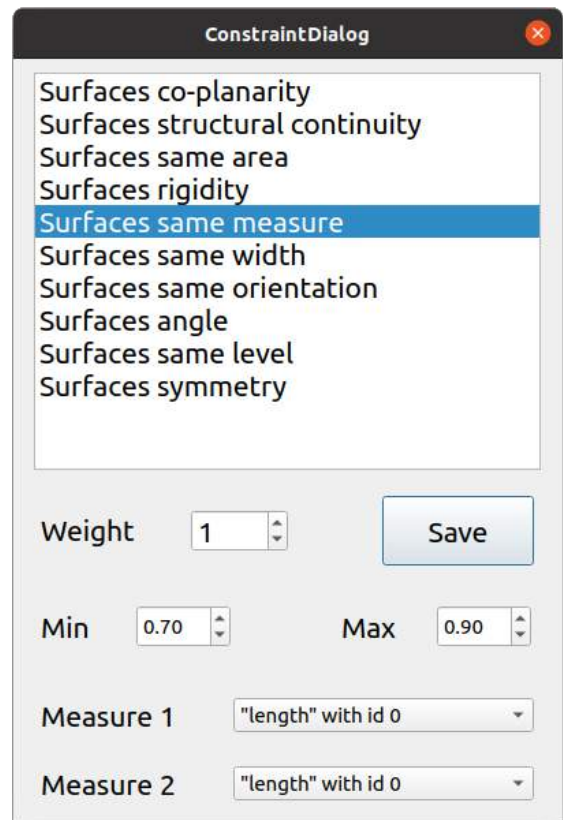


Figure A.7: *The simple list of annotations (left), the list with an expanded annotation (middle), the list with an expanded annotation with one of its attributes expanded (right).*

- Ruler: it requires to select two points over the surface (Ctrl+left click, it works only on the selected annotation) for defining the extrema of the measure (Euclidean distance);
  - Tape measure: it works pretty much as the Polylines selection, i.e., it requires to select points on the surface (Ctrl+left click, it works only on the selected annotation) in an additive way, while computing the shortest path between the couples of successive points to define the geodesic measure (approximated);
  - Bounding measure: this tool is useful for defining measures between extrema not falling on the surface. It requires to define a direction for the measure (Ctrl+left click and drag) and returns the distance between the two farthest points of the annotation in that direction. To compute this measure, points are projected on a line having the direction set by the user and passing through the barycentre of the annotation. Two clipping planes are shown when defining the measure to help the user visualise the measure.
- In the lower part, a list of annotations of the mesh is shown, where each voice can be expanded for showing some properties, such as id and tag, and the list of associated attributes, that in their turn can be expanded to show the associated properties, a button for removing the attribute and, in the case of a measure, a checkable button for showing/hiding it in the canvas (see Figure A.7).



(a)



(b)

Figure A.8: The constraint dialog enables the input fields depending on the type of relationship/constraint. In the current implementation the measures are constrainable only if the relationship is between 2 annotations.



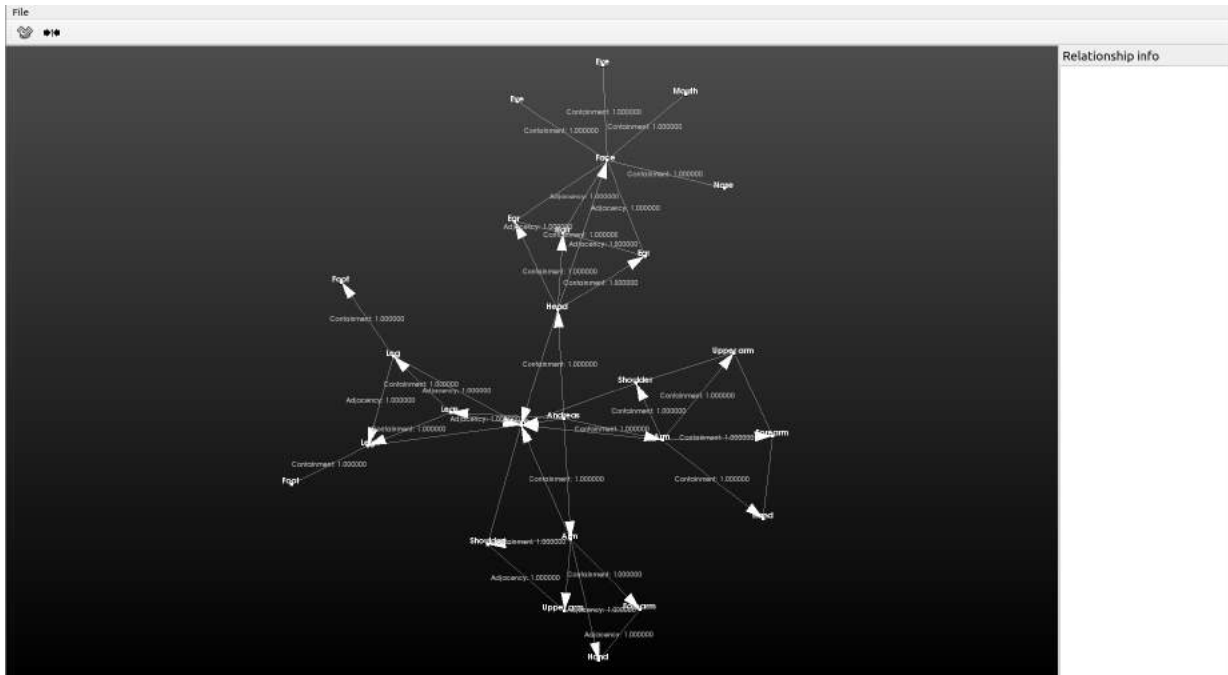




Figure A.9: *The Relationships window*

## A.3 Relationships window

In this window, the system provides an overview of the stated relationships between annotated parts. The relationships are represented as line segments or arrows, depending on the direction of the relationship (see sub-section 2.3), between nodes of a graph, representing annotations.

This window also allows the creation of new relationships. The user can select two or more annotations (simply left-clicking on any node in the graph) and then press the  button, which calls a dialog for selecting the type and specifying the properties of the new relationship. Moreover, the user can also constrain the existing relationships, by pressing the  button) and so initialising the ShapeOp library (same behaviour as in the Annotation window).

When passing the cursor over an annotation (node of the graph) or a relationship, the sidebar displays related information regarding the specific entity (see Figure A.10).

Annotation info
id: 25
tag: Mouth
level: 3
attributes:
type: Region
boundaries:
[
[ 20138, 16611, 16613, 15017, 15015,
[ 15072, 15071, 17767, 19124, 19123,
]
]

Relationship info
type: Containment
weight: 1.000000
involved annotations: 18 and 25

Figure A.10: *Different information shown when passing over annotations (left) and relationships (right) with the cursor.*

# Appendix B

## File formats

The formalisation of the semantics of a certain class of objects is (at least currently) a long work which can be performed by one or more experts. For this reason, tools should be provided to store the results produced so far in such a way that could be easily shared between the experts and provided to the final user.

In this framework such results are stored exploiting the JavaScript Object Notation (JSON)<sup>©</sup> [Bra17] format, which is hugely employed in several fields thanks to its simplicity.

In particular, here it is used for storing annotations and related info (i.e., tag, type, attributes, etc.) and relationships/constraints (better, the relationships' graph). The library employed for the writing/parsing of the files is RapidJSON<sup>©</sup> [aTcY15].

### B.1 Annotation file format

The annotation file format contains as root the object *annotations*, which is a list of annotation objects containing the following fields:

- *id*: the integer number identifying the annotation in the system;
- *tag*: a string (in future extensions this field will be substituted with a more generic *information* field containing a *reference* to an object, being it a string, an image, or whatever) associated to the annotation;
- *colour*: a triplet (more formally, a list of 3 integers) defining a colour associated to the annotation (in the future this field will be removed and the colour will be automatically set by the application GUIs);

- *attributes*: a list of attribute objects specified in the following;
- *type*: the type of selector of the annotation. Currently there are 3 possible values: “point”, “line” and “region”.
- Depending on the value of the *type* field, this field can have the following values:
  - *points*: only when the *type* field has “point” value. It is a list of indices of mesh vertices;
  - *polylines*: only when the *type* field has “line” value. It is a list of lists of indices to successive vertices of the mesh, defining one or more poly-lines;
  - *boundaries*: only when the *type* field has “region” value. It is a list of lists of indices to successive vertices of the mesh, enclosing bounded patches on the mesh surface (as in sub-section 2.4.1).

The attribute objects contain the following fields:

- *id*: the integer number identifying the attribute within the annotation;
- *name*: a string containing the name associated to the attribute (e.g., “height”);
- *type*: a string defining the type of the attribute (currently we have only “semantic” and “measure”);
- there are two possibilities for the final part of the object, depending on the value of the previous field:
  - *note*: only if the value of *type* is “semantic”. It is a string containing a free text.
  - *measure*: an object containing the following fields:
    - \* *tool*: a string specifying the tool used for taking the measure (currently only “ruler”, “tape” and “bounding”);
    - \* *points*: a list of indices to the mesh vertices involved in the measure. Depending on the previous field, they can be 2 or more (2 for an Euclidean distance, more for the approximate geodesic one).
    - \* *direction*: this field is present only if the previous value is “bounding”. It is a vector (more formally a list of 3 doubles) defining the measure direction.

## B.2 Graph file format

The graph file format contains as root the object’s *relationships*, which is a list of relationships objects containing following the same base structure:

- *id*: an integer number identifying the relationship;
- *type*: a string identifying the relationship. There are several possible values, one for each kind of constraint present in ShapeOp (the framework always gives the possibility to use the geometric constraints) plus the high-level constraint defined in this thesis. Notice that not all the relationships are constraints, so here possible values are even “containment” and “adjacency” (see sub-section 2.3);
- *isDirected*: a boolean value for understanding if the arc corresponding to the relationship is directed or not (see sub-section 2.3);
- *annotations*: a list of indices to the annotations involved in the relationships;
- *isConstraint*: a boolean value stating if the relationship is a constraint;
- *weight*: a double value associating a weight to the constraint. This field is present only if the previous is true;
- *constraint*: is an object reporting all the parameters of the constraint (so this field is present only if *isConstraint* is true). The inner structure of this object is really dependent on the constraint, e.g., the “Proportion” (see Section 5.3.2) constraint requires *measure1* and *measure2* indices identifying the measure to be constrained both on the first and the second annotation and a *minValue* and *maxValue* defining an acceptance range.

The recovery of the graph is simply achieved by creating a node for each annotation and then scrolling the *relationships* list and creating arcs accordingly.

# Bibliography

- [3DC01] 3D CAD browser. <http://www.3dcadbrowser.com/>, 2001.
- [3DS20] Autodesk 3ds Max, 2020. <https://www.autodesk.it/products/3ds-max>.
- [AB97] Fabrice Aubert and Dominique Bechmann. Volume-preserving space deformation. *Computers & Graphics*, 21(5):625 – 639, 1997.
- [ACOL00] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible Shape Interpolation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 157–164, 2000.
- [AF06] M. Attene and B. Falcidieno. Remesh: An interactive environment to edit and repair triangle meshes. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pages 41–41, 2006.
- [AFNS12] Theodoros Athanasiadis, Ioannis Fudos, Christophoros Nikou, and Vasiliki Stamat. Feature-based 3D morphing based on geometrically constrained spherical parameterization. *Computer Aided Geometric Design*, 29(1):2–17, 2012.
- [AIM04] AIM@SHAPE shape repository. <http://shapes.aimatshape.net/>, 2004.
- [AKZM14] Melinos Averkiou, Vladimir G. Kim, Youyi Zheng, and Niloy J. Mitra. ShapeSynth: Parameterizing model collections for coupled shape exploration and synthesis. *Computer Graphics Forum*, 33(2):125–134, 2014.
- [Ale02a] Marc Alexa. Linear combination of transformations. *ACM Trans. Graph.*, 21(3):380–387, July 2002.
- [Ale02b] Marc Alexa. Recent Advances in Mesh Morphing. *Computer Graphics Forum*, 2002.

- [APH17] Dmitry Anisimov, Daniele Panozzo, and Kai Hormann. Blended barycentric coordinates. *Computer Aided Geometric Design*, 52-53:205 – 216, 2017. Geometric Modeling and Processing 2017.
- [ARV07] B. Amberg, S. Romdhani, and T. Vetter. Optimal Step Nonrigid ICP Algorithms for Surface Registration. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
- [aTcY15] THL A29 Limited (a Tencent company) and Milo Yip. RapidJson, 2015. <https://rapidjson.org/>.
- [AWLB17] Jascha Achenbach, Thomas Waltemate, Marc Erich Latoschik, and Mario Botsch. Fast generation of realistic virtual humans. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology, VRST '17*, New York, NY, USA, 2017. Association for Computing Machinery.
- [BAC<sup>+</sup>16] Imon Banerjee, Asan Agibetov, Chiara Eva Catalano, Giuseppe Patané, and Michela Spagnuolo. Semantics-driven annotation of patient-specific 3D data: a step to assist diagnosis and treatment of rheumatoid arthritis. *The Visual Computer*, 32(10):1337–1349, Oct 2016.
- [BBN06] W.F. Bronsvort, R. Bidarra, and P. Nyirenda. Developments in feature modelling. *Computer-Aided Design & Applications*, 3(5):655–664, 2006.
- [BCK<sup>+</sup>11] Arthur Blume, Won Chun, David Kogan, Vangelis Kokkevis, Nico Weber, Rachel Weinstein Petterson, and Roni Zeiger. Google Body: 3D Human Anatomy in the Browser. In *ACM SIGGRAPH 2011 Talks, SIGGRAPH '11*, New York, NY, USA, 2011. Association for Computing Machinery. <https://doi.org/10.1145/2037826.2037852>.
- [BCWG09] Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. Spatial Deformation Transfer. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '09*, pages 67–74. ACM, 2009.
- [BDS<sup>+</sup>12] Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. Shape-Up: Shaping Discrete Geometry with Projections. *Comput. Graph. Forum*, 31(5):1657–1667, August 2012.
- [BF01] Samuel R. Buss and Jay P. Fillmore. Spherical averages and applications to spherical splines and interpolation. *ACM Trans. Graph.*, 20(2):95–126, April 2001.
- [BGC<sup>+</sup>15] A. Bartoli, Y. Gérard, F. Chadebecq, T. Collins, and D. Pizarro. Shape-from-template. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(10):2099–2118, 2015.

- [BK05] Mario Botsch and Leif Kobbelt. Real-Time Shape Editing using Radial Basis Functions. *Computer Graphics Forum*, 24(3):611–621, 2005.
- [BKP<sup>+</sup>10] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon Mesh Processing*. AK Peters / CRC Press, September 2010.
- [BLP<sup>+</sup>15] Imon Banerjee, Hamid Laga, Giuseppe Patané, Sebastian Kurtek, Anuj Srivastava, and Michela Spagnuolo. Generation of 3D Canonical Anatomical Models: An Experience on Carpal Bones. In Vittorio Murino, Enrico Puppo, Diego Sona, Marco Cristani, and Carlo Sansone, editors, *New Trends in Image Analysis and Processing – ICIAP 2015 Workshops*, pages 167–174, Cham, 2015. Springer International Publishing.
- [BLTD16] Max Budninskiy, Beibei Liu, Yiyong Tong, and Mathieu Desbrun. Power coordinates: A geometric construction of barycentric coordinates on convex polytopes. *ACM Trans. Graph.*, 35(6), November 2016.
- [Bot16] Katarina Botwid. *The Artisanal Perspective in Action : An Archaeology in Practice*. PhD thesis, Lund University, 2016.
- [BR07] Benedict J. Brown and Szymon Rusinkiewicz. Global Non-Rigid Alignment of 3D Scans. *ACM Trans. Graph.*, 26(3):21–es, July 2007.
- [BR14] Jose Luis Blanco and Pranjal Kumar Rai. nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees. <https://github.com/jlblancoc/nanoflann>, 2014.
- [Bra17] T. Bray. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259, RFC Editor, December 2017. <https://www.rfc-editor.org/info/rfc8259>.
- [BS08] M. Botsch and O. Sorkine. On Linear Variational Surface Deformation Methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, 2008.
- [BS09] Sandy Budden and Joanna Sofaer. Non-discursive knowledge and the construction of identity potters, potting and performance at the bronze age tell of százhalombatta, hungary. *Cambridge Archaeological Journal*, 19(2):203–220, 2009.
- [Cae20] Civilian American and European Surface Anthropometry Resource Project—CAESAR, 2020. <http://store.sae.org/caesar/>.



- [CCC<sup>+</sup>08] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008.
- [CCLS18] Sara Casti, Fabrizio Corda, Marco Livesu, and Riccardo Scateni. Cagelab: an interactive tool for cage-based deformations. In *STAG*, pages 65–74, 2018.
- [CCXS11] W. Cheng, R. Cheng, L. Xiaoyong, and D. Shuling. Automatic skeleton generation and character skinning. In *2011 IEEE International Symposium on VR Innovation*, pages 299–304, 2011.
- [CDJF19] Maes Chris, Smeets Dirk, Keustermans Johannes, and Thomas Fabry. meshSIFT MATLAB implementation. <https://mirc.uzleuven.be/MedicalImageComputing/downloads/meshSIFT.php>, 2019. Accessed: 13/08/2019.
- [CF14] Xue Chen and Jieqing Feng. Adaptive Skeleton-driven Cages for Mesh Sequences. *Comput. Animat. Virtual Worlds*, 25(3-4):447–455, May 2014.
- [CGM19] E. Cordeiro, F. Giannini, and M. Monti. A survey of immersive systems for shape manipulation. *Computer-Aided Design and Applications*, 16(6):1146–1157, 2019. [http://cad-journal.net/files/vol\\_16/Vol16No6.html](http://cad-journal.net/files/vol_16/Vol16No6.html).
- [CJV06] Gary E. Christensen, Hans J. Johnson, and Michael W. Vannier. Synthesizing average 3D anatomical shapes. *NeuroImage*, 32(1):146–158, 2006.
- [CKS16] J. Cui, A. Kuijper, and A. Sourin. Exploration of natural free-hand interaction for shape modeling using leap motion controller. In *2016 International Conference on Cyberworlds (CW)*, pages 41–48, 2016.
- [CLM<sup>+</sup>19] Sara Casti, Marco Livesu, Nicolas Mellado, Nadine Abu Rumman, Scateni Riccardo, Loïc Barthe, and Enrico Puppo. Skeleton based cage generation guided by harmonic fields. *Computers & Graphics*, 81:140–151, 2019.
- [CMSF11] Chiara Eva Catalano, Michela Mortara, Michela Spagnuolo, and Bianca Falcidieno. Semantics and 3D Media: Current Issues and Perspectives. *Computers & Graphics*, 35(4):869–877, August 2011.
- [Coq90] Sabine Coquillart. Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling. *SIGGRAPH Comput. Graph.*, 24(4):187–196, September 1990.

- [CRV20] M.W. Cohen, D. Regazzoni, and C. Vrabel. 3D Virtual Sketching System Using NURBS Surfaces and Leap Motion Controller. *Computer-Aided Design and Applications*, 17(1):167–177, May 2020. [http://cad-journal.net/files/vol\\_17/Vol17No1.html](http://cad-journal.net/files/vol_17/Vol17No1.html).
- [CS17] Jian Cui and Alexei Sourin. Interactive shape modeling using leap motion controller. In *SIGGRAPH Asia 2017 Technical Briefs*, SA '17, New York, NY, USA, 2017. Association for Computing Machinery. <https://doi.org/10.1145/3145749.3149437>.
- [CTB<sup>+</sup>12] P. Chevaillier, T. Trinh, M. Barange, P. De Loor, F. Devillers, J. Soler, and R. Querrec. Semantic modeling of virtual environments using mascaret. In *2012 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, pages 1–8, 2012.
- [CTL<sup>+</sup>20] F. Corda, J. M. Thiery, M. Livesu, E. Puppo, T. Boubekeur, and R. Scateni. Real-time deformation with coupled cages and skeletons. *Computer Graphics Forum*, 39(6):19–32, 2020.
- [CVHS20] C.E. Catalano, V. Vassallo, S. Hermon, and M. Spagnuolo. Representing quantitative documentation of 3D cultural heritage artefacts with CIDOC CRMdig. *International Journal on Digital Libraries*, 21:359–373, 2020.
- [dBA16] Ryan Anthony J. de Belen and Rowel O. Atienza. Automatic skeleton generation using hierarchical mesh segmentation. In *SIGGRAPH ASIA 2016 Virtual Reality Meets Physical Reality: Modelling and Simulating Virtual Humans and Environments*, SA '16, New York, NY, USA, 2016. Association for Computing Machinery.
- [DCH20] Chongyang Deng, Qingjun Chang, and Kai Hormann. Iterative coordinates. *Computer Aided Geometric Design*, 79:101861, 2020.
- [DD86] H.L. Dreyfus and Dreyfus. *Mind Over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*. Oxford: Basil Blackwell, 1986.
- [DDB<sup>+</sup>14] Mario Deuss, Anders Holden Deleuran, Sofien Bouaziz, Bailin Deng, Daniel Piker, and Mark Pauly. ShapeOp web site. <https://www.shapeop.org/index.php>, 2014.
- [DDB<sup>+</sup>15] Mario Deuss, Anders Holden Deleuran, Sofien Bouaziz, Bailin Deng, Daniel Piker, and Mark Pauly. ShapeOp – A Robust and Extensible Geometric Modelling Paradigm. In Mette Ramsgaard Thomsen, Martin Tamke, Christoph Gengnagel, Billie Faircloth, and Fabian Scheurer, editors, *Modelling Behaviour: Design Modelling Symposium 2015*, pages 505–515. Springer International Publishing, Cham, 2015.

- [DFG<sup>+</sup>94] T. De Martino, B. Falcidieno, F. Giannini, S. Hassinger, and J. Ovtcharova. Feature-based modelling by integrating design and recognition approaches. *Computer-Aided Design*, 26(8):646 – 653, 1994. Special Issue Modelling in computer graphics.
- [Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerical Mathematics*, 1:269–271, 1959.
- [DLM11] Zheng-Jie Deng, Xiao-Nan Luo, and Xiao-Ping Miao. Automatic Cage Building with Quadric Error Metrics. *Journal of Computer Science and Technology*, 26(3):538, May 2011.
- [DLPDM<sup>+</sup>18] Livio De Luca, Marc Pierrot-Deseilligny, Adeline Manuel, Christine Chevrier, Benjamin Lollier, Pascal Benistant, Anthony Pamart, Friederike Peteler, Violette Abergel, and Anas Alaoui. Aioli – a reality-based 3D annotation platform for the collaborative documentation of heritage artefacts, 2018. <http://www.aioli.cloud/>.
- [DMMT<sup>+</sup>07] F. Dellas, L. Moccozet, N. Magnenat-Thalmann, M. Mortara, G. Patané, M. Spagnuolo, and B. Falcidieno. Knowledge-based Extraction of Control Skeletons for Animation. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007, SMI '07*, pages 51–60, Washington, DC, USA, 2007. IEEE Computer Society.
- [DZY<sup>+</sup>16] Guoguang Du, Mingquan Zhou, Congli Yin, Zhongke Wu, and Wuyang Shui. Classification and reassembly of archaeological fragments. In *Proceedings of the Symposium on VR Culture and Heritage - Volume 2, VRCAI '16*, page 67–70, New York, NY, USA, 2016. Association for Computing Machinery.
- [ED17] H. ElNaghy and L. Dorst. Geometry Based Faceting of 3D Digitized Archaeological Fragments. In *2017 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 2934–2942, Los Alamitos, CA, USA, oct 2017. IEEE Computer Society.
- [EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.
- [FBF77] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.*, 3(3):209–226, September 1977.

- [FCD20] Tong Fu, Raphaëlle Chaine, and Julie Digne. Anatomy Changes and Virtual Restoration of Statues. In Michela Spagnuolo and Francisco Javier Melero, editors, *Eurographics Workshop on Graphics and Cultural Heritage*. The Eurographics Association, 2020.
- [FHK06] M.S. Floater, K. Hormann, and G. Kós. A general construction of barycentric coordinates over convex polygons. *Advances in Computational Mathematics*, 24:311–331, 2006.
- [FKR05] Michael S. Floater, Géza Kós, and Martin Reimers. Mean value coordinates in 3D. *Computer Aided Geometric Design*, 22(7):623 – 631, 2005. Geometric Modelling and Differential Geometry.
- [Flo97] Michael S. Floater. Parametrization and smooth approximation of surface triangulations. *Comput. Aided Geom. Des.*, 14(3):231–250, April 1997.
- [Flo03] Michael S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19 – 27, 2003.
- [FOC08] FOCUS K3D project. <https://cordis.europa.eu/project/id/214993>, 2008.
- [Fou07] Sabine Fourier. *La coroplastie chypriote archaïque. Identités culturelles et politiques à l'époque des royaumes*. Maison de l'Orient et de la Méditerranée Jean Pouilloux, 2007.
- [FRC13] Carol Friedman, Thomas C. Rindfleisch, and Milton Corn. Natural language processing: State of the art and prospects for significant progress, a workshop sponsored by the National Library of Medicine. *Journal of Biomedical Informatics*, 46(5):765 – 773, 2013. <http://www.sciencedirect.com/science/article/pii/S1532046413000798>.
- [FW13] J. Flotyński and K. Walczak. Semantic multi-layered design of interactive 3D presentations. In *2013 Federated Conference on Computer Science and Information Systems*, pages 541–548, Sep. 2013.
- [get] Getty vocabularies. <https://www.getty.edu/research/tools/vocabularies/index.html>.
- [GG20] H. Gao and G. Geng. Classification of 3D Terracotta Warrior Fragments Based on Deep Learning and Template Guidance. *IEEE Access*, 8:4086–4098, 2020.
- [GLSW35] Einar Gjerstad, John Lindros, Erik Sjöqvist, and Alfred Westholm. *The Swedish Cyprus Expedition: Finds and results of the excavations in Cyprus 1927-1931*. Cambridge University Press, 1935.

- [GPCP13] Francisco González García, Teresa Paradinas, Narcís Coll, and Gustavo Patow. \*Cages:: A Multilevel, Multi-cage-based System for Mesh Deformation. *ACM Transaction on Graphics*, 32(3):24:1–24:13, July 2013.
- [gra] Gravity Sketch. <https://www.gravitysketch.com/>.
- [GSP<sup>+</sup>14] Robert Gregor, Ivan Sipiran, Georgios Papaioannou, Tobias Schreck, Anthousis Andreadis, and Pavlos Mavridis. Towards Automated 3D Reconstruction of Defective Cultural Heritage Objects. In Reinhard Klein and Pedro Santos, editors, *Eurographics Workshop on Graphics and Cultural Heritage*. The Eurographics Association, 2014.
- [Han06] Andrew J. Hanson. *Visualizing Quaternions*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.
- [Hej04] J. Hejl. Hardware skinning with quaternions. *Game Programming Gems*, 4:487–495, 2004.
- [HF06] Kai Hormann and Michael S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph.*, 25(4):1424–1441, October 2006.
- [HF07] Sven Havemann and Dieter W. Fellner. Seven Research Challenges of Generalized 3D Documents. *IEEE Computer Graphics and Applications*, 27(3):70–76, 2007.
- [HHK92] William M. Hsu, John F. Hughes, and Henry Kaufman. Direct manipulation of free-form deformations. *SIGGRAPH Comput. Graph.*, 26(2):177–184, July 1992.
- [HM09] Tobias Heimann and Hans-Peter Meinzer. Statistical shape models for 3D medical image segmentation: A review. *Medical Image Analysis*, 13(4):543 – 563, 2009.
- [HMVG09] Simon Haegler, Pascal Müller, and Luc Van Gool. Procedural Modeling for Digital Cultural Heritage. *Journal on Image and Video Processing*, 2009:7:4–7:4, February 2009.
- [HS08] K. Hormann and N. Sukumar. Maximum entropy coordinates for arbitrary polytopes. *Computer Graphics Forum*, 27(5):1513–1520, 2008.
- [IMH05] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transaction on Graphics*, 24(3):1134–1141, July 2005.

- [Ive68] Erik Iversen. Diodorus' account of the egyptian canon. *The Journal of Egyptian Archaeology*, 54:215–218, 1968.
- [JBPS11] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. Bounded Bi-harmonic Weights for Real-time Deformation. *ACM Transaction on Graphics*, 30(4):78:1–78:8, July 2011.
- [JC18] Poonsiri Jailungka and Siam Charoenseang. Intuitive 3D model prototyping with leap motion and microsoft hololens. In Masaaki Kurosu, editor, *Human-Computer Interaction. Interaction Technologies*, pages 269–284, Cham, 2018. Springer International Publishing.
- [JDKL14] Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses*, 2014.
- [JMD<sup>+</sup>07] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic Coordinates for Character Articulation. In *ACM SIGGRAPH 2007 Papers, SIGGRAPH '07*, 2007.
- [JPR00] JungHyun Han, M. Pratt, and W. C. Regli. Manufacturing feature recognition from solid models: a status report. *IEEE Transactions on Robotics and Automation*, 16(6):782–796, 2000.
- [JSW05] Tao Ju, Scott Schaefer, and Joe Warren. Mean Value Coordinates for Closed Triangular Meshes. *ACM Transaction on Graphics*, 24(3):561–566, July 2005.
- [JZvdP<sup>+</sup>08] Tao Ju, Qian-Yi Zhou, Michiel van de Panne, Daniel Cohen-Or, and Ulrich Neumann. Reusable Skinning Templates Using Cage-based Deformations. In *ACM SIGGRAPH Asia 2008 Papers, SIGGRAPH Asia '08*, pages 122:1–122:10. ACM, 2008.
- [KC15] Sung-Ho Kim and Kyung-Yong Chung. Medical Information Service System Based on Human 3D Anatomical Model. *Multimedia Tools Appl.*, 74(20):8939–8950, October 2015. <https://doi.org/10.1007/s11042-013-1584-8>.
- [KCGF14] Vladimir G. Kim, Siddhartha Chaudhuri, Leonidas Guibas, and Thomas Funkhouser. Shape2Pose: Human-centric Shape Analysis. *ACM Transaction on Graphics*, 33(4):120:1–120:12, July 2014.
- [KCvO07] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Skinning with dual quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games, I3D '07*, page 39–46, New York, NY, USA, 2007. Association for Computing Machinery.

- [KCvO08] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O’Sullivan. Geometric Skinning with Approximate Dual Quaternion Blending. *ACM Transaction on Graphics*, 27(4):105:1–105:23, November 2008.
- [KLM<sup>+</sup>13] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning Part-Based Templates from Large Collections of 3D Shapes. *ACM Trans. Graph.*, 32(4), July 2013.
- [KS12] Ladislav Kavan and Olga Sorkine. Elasticity-inspired deformers for character articulation. *ACM Trans. Graph.*, 31(6), November 2012.
- [KSKL13] Sebastian Kurtek, Anuj Srivastava, Eric Klassen, and Hamid Laga. Landmark-guided elastic shape analysis of spherically-parameterized surfaces. *Computer Graphics Forum*, 32(2pt4):429–438, 2013. <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12063>.
- [Kv05] Ladislav Kavan and Jiří Žára. Spherical blend skinning: A real-time deformation of articulated models. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games, I3D ’05*, page 9–16, New York, NY, USA, 2005. Association for Computing Machinery.
- [Las20] Lasso selection, 2020. <https://helpx.adobe.com/photoshop/using/selecting-lasso-tools.html>.
- [LBB19] Nikolas Lamb, Sean Banerjee, and Natasha Kholgade Banerjee. Automated Reconstruction of Smoothly Joining 3D Printed Restorations to Fix Broken Objects. In *Proceedings of the ACM Symposium on Computational Fabrication, SCF ’19*, New York, NY, USA, 2019. Association for Computing Machinery.
- [LBGM19] Katia Lupinetti, Brigida Bonino, Franca Giannini, and Marina Monti. Exploring the benefits of the virtual reality technologies for assembly retrieval applications. In Lucio Tommaso De Paolis and Patrick Bourdot, editors, *Augmented Reality, Virtual Reality, and Computer Graphics*, pages 43–59, Cham, 2019. Springer International Publishing.
- [LCF00] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’00*, page 165–172, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
- [LD17] Binh Huy Le and Zhigang Deng. Interactive Cage Generation for Mesh Deformation. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, I3D ’17*, pages 3:1–3:9. ACM, 2017.

- [LGJ<sup>+</sup>17] Z. Li, F. Giannini, J.P. Pernot, P. Véron, and B. Falcidieno. Reusing heterogeneous data for the conceptual design of shapes in virtual environments. *Virtual Reality*, 21:127 – 144, 2017. <https://doi.org/10.1007/s10055-016-0302-z>.
- [LHT<sup>+</sup>19] Yang LI, Jin HUANG, Feng TIAN, Hong-An WANG, and Guo-Zhong DAI. Gesture interaction in virtual reality. *Virtual Reality & Intelligent Hardware*, 1(1):84 – 112, 2019. <http://www.sciencedirect.com/science/article/pii/S2096579619300075>.
- [Lin68] Aristid Lindenmayer. Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, 18(3):300 – 315, 1968.
- [LK84] S. Lien and J. T. Kajiya. A symbolic method for calculating the integral properties of arbitrary nonconvex polyhedra. *IEEE Computer Graphics and Applications*, 4(10):35–42, 1984.
- [LKC07] Yaron Lipman, Johannes Kopf, Daniel Cohen-Or, and David Levin. GPU-assisted Positive Mean Value Coordinates for Mesh Deformations. In Alexander Belyaev and Michael Garland, editors, *Geometry Processing*. The Eurographics Association, 2007.
- [LLC08] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green Coordinates. *ACM Transaction on Graphics*, 27(3):78:1–78:10, August 2008.
- [LMS13] Hamid Laga, Michela Mortara, and Michela Spagnuolo. Geometry and Context for Semantic Correspondences and Functionality Recognition in Man-made 3D Shapes. *ACM Transaction on Graphics*, 32(5):150:1–150:16, October 2013. <http://doi.acm.org/10.1145/2516971.2516975>.
- [LPC<sup>+</sup>00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The Digital Michelangelo Project: 3D Scanning of Large Statues. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '00*, page 131–144, USA, 2000. ACM Press/Addison-Wesley Publishing Co. <https://doi.org/10.1145/344779.344849>.
- [M27] Augustus Ferdinand Möbius. *Der barycentrische Calcul: ein neues Hilfsmittel zur analytischen Behandlung der Geometrie*. Barth, Johann Ambrosius, 1827.
- [Mac67] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.



- [mak17] MakeVR Pro, 2017. <https://www.viveport.com/9e94a10f-51d9-4b6f-92e4-6e4fe9383fe9>.
- [May20] Autodesk Maya, 2020. <https://www.autodesk.it/products/maya>.
- [MBLD02] Mark Meyer, Alan Barr, Haeyoung Lee, and Mathieu Desbrun. Generalized barycentric coordinates on irregular polygons. *J. Graph. Tools*, 7(1):13–22, November 2002.
- [MCG<sup>+</sup>19] D. Mendes, F. M. Caputo, A. Giachetti, A. Ferreira, and J. Jorge. A survey on 3D virtual object manipulation: From the desktop to immersive virtual environments. *Computers & Graphics Forum*, 38:21 – 45, 2019. <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13390>.
- [MGY<sup>+</sup>19] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy J. Mitra, and Leonidas J. Guibas. StructureNet: Hierarchical Graph Networks for 3D Shape Generation. *ACM Transaction on Graphics*, 38(6), November 2019.
- [min15] MindeskVR, 2015. <https://mindeskvr.com/>.
- [Mit08] Mitsuhashi, Nobutaka and Fujieda, Kaori and Tamura, Takuro and Kawamoto, Shoko and Takagi, Toshihisa and Okubo, Kousaku. BodyParts3D: 3D structure database for anatomical concepts. *Nucleic Acids Research*, 37(suppl\_1):D782–D785, 10 2008. <https://doi.org/10.1093/nar/gkn613>.
- [MJBF02] Tim Milliron, Robert J. Jensen, Ronen Barzel, and Adam Finkelstein. A framework for geometric warps and deformations. *ACM Trans. Graph.*, 21(1):20–51, January 2002.
- [MLS11] Josiah Manson, Kuiyu Li, and Scott Schaefer. Positive gordon–wixom coordinates. *Computer-Aided Design*, 43(11):1422 – 1426, 2011. Solid and Physical Modeling 2011.
- [MMG06] Bruce Merry, Patrick Marais, and James Gain. Animation space: A truly linear framework for character animation. *ACM Trans. Graph.*, 25(4):1400–1423, October 2006.
- [Mor78] Piero Morselli. The Proportions of Ghiberti’s Saint Stephen: Vitruvius’s De Architectura and Alberti’s De Statua. *The Art Bulletin*, 60(2):235–241, 1978.
- [MPS<sup>+</sup>04] M. Mortara, G. Patane, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Plumber: A Multi-scale Decomposition of 3D Shapes into Tubular Primitives and Bodies. In Gershon Elber, Nicholas Patrikalakis, and Pere Brunet, editors, *Solid Modeling*. The Eurographics Association, 2004.

- [MRS09] Manuel Möller, Sven Regel, and Michael Sintek. RadSem: Semantic Annotation and Retrieval for Medical Images. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Simperl, editors, *The Semantic Web: Research and Applications*, pages 21–35, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [MTLT89] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics Interface '88*, page 26–33, CAN, 1989. Canadian Information Processing Society.
- [MVL<sup>+</sup>11] Tommaso Mansi, Ingmar Voigt, Benedetta Leonardi, Xavier Pennec, Stanley Durrleman, Maxime Sermesant, Hervé Delingette, Andrew M. Taylor, Younes Boudjemline, Giacomo Pongiglione, and Nicholas Ayache. A Statistical Model for Quantification and Prediction of Cardiac Remodelling: Application to Tetralogy of Fallot. *IEEE Transactions on Medical Imaging*, 2011.
- [MWH<sup>+</sup>06] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. *ACM Trans. Graph.*, 25(3):614–623, July 2006.
- [NFN00] Jun-yong Noh, Douglas Fidaleo, and Ulrich Neumann. Animated Deformations with Radial Basis Functions. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST '00*, pages 166–174, New York, NY, USA, 2000. ACM.
- [NM82] Badler Ni and Morris Ma. Modelling flexible articulated objects. In *Proc. Computer Graphics' 82, Online Conference*, pages 305–314, 1982.
- [NS13] Jesús R. Nieto and Antonio Susín. Cage Based Deformations: A Survey. In Manuel González Hidalgo, Arnau Mir Torres, and Javier Varona Gómez, editors, *Deformation Models: Tracking, Animation and Applications*, pages 75–99. Springer Netherlands, Dordrecht, 2013.
- [OBP<sup>+</sup>13] A. Cengiz Öztireli, Ilya Baran, Tiberiu Popa, Boris Dalstein, Robert W. Sumner, and Markus Gross. Differential blending for expressive sketch-based posing. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '13*, page 155–164, New York, NY, USA, 2013. Association for Computing Machinery.
- [ocu] Oculus Medium. <https://www.oculus.com/medium/>.

- [OFBW07] De Troyer O., Kleinermann F., Pellens B., and Bille W. Conceptual modeling for virtual reality. In *ER '07: Tutorials, posters, panels and industrial contributions at the 26th international conference on Conceptual modelling, Darlinghurst, Australia*, pages 3–185, Cham, 2007. Australian Computer Society, Inc.
- [PB10] Susan Pollock and R Bernbeck. An archaeology of categorization and categories in archaeology. *Paléorient*, 36(1):37–47, 2010.
- [PCDS20] F. Ponchio, M. Callieri, M. Dellepiane, and R. Scopigno. Effective Annotations Over 3D Models. *Computer Graphics Forum*, 39(1):89–105, 2020.
- [PCM<sup>+</sup>14] Raphael Prevost, Rémi Cuingnet, Benoit Mory, Laurent D. Cohen, and Roberto Ardon. Tagged template deformation. In Polina Golland, Nobuhiko Hata, Christian Barillot, Joachim Hornegger, and Robert Howe, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2014*, pages 674–681, Cham, 2014. Springer International Publishing.
- [PF19] Laurent Perron and Vincent Furnon. Google OR-Tools. <https://developers.google.com/optimization/>, 2019.
- [PFGL08] J.-P. Pernot, B. Falcidieno, F. Giannini, and J.-C. Léon. Incorporating free-form features in aesthetic and engineering product design: State-of-the-art report. *Computers in Industry*, 59(6):626 – 637, 2008.
- [PFW<sup>+</sup>03] Pin-Chou Liu, Fu-Che Wu, Wan-Chun Ma, Rung-Huei Liang, and Ming Ouhyoung. Automatic animation skeleton using repulsive force field. In *11th Pacific Conference on Computer Graphics and Applications, 2003. Proceedings.*, pages 409–413, 2003.
- [PGDF<sup>+</sup>20] Gaia Pavoni, Francesca Giuliani, Anna De Falco, Massimiliano Corsini, Federico Ponchio, Marco Callieri, and Paolo Cignoni. Another brick in the wall: Improving the assisted semantic segmentation of masonry walls. In *Eurographics Workshop on Graphics and Cultural Heritage*. The Eurographics Association, 2020. <http://vcg.isti.cnr.it/Publications/2020/PGDCPCC20>.
- [PM01] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01*, page 301–308, New York, NY, USA, 2001. Association for Computing Machinery.
- [PSA<sup>+</sup>17] Georgios Papaioannou, Tobias Schreck, Anthousis Andreadis, Pavlos Mavridis, Robert Gregor, Ivan Sipiran, and Konstantinos Vardis. From reassembly to object completion: A complete systems pipeline. *J. Comput. Cult. Herit.*, 10(2), March 2017.

- [PW12] Ewdoksia Papuci-Wladyka. *Corpus Vasorum Antiquorum, Poland, Fascicule 11: Cracow Fascicule 1*. Jagiellonian University Institute of Archaeology 1, Jagiellonian University Museum, Polish Academy of Arts and Sciences, 2012.
- [PWM<sup>+</sup>16] Stephen C. Phillips, Paul W. Walland, Stefano Modafferi, Leo Dorst, Michela Spagnuolo, Chiara Eva Catalano, Dominic Oldman, Ayellet Tal, Ilan Shimshoni, and Sorin Hermon. GRAVITATE: Geometric and Semantic Matching for Cultural Heritage Artefacts. In *Eurographics Workshop on Graphics and Cultural Heritage*. The Eurographics Association, 2016.
- [QSO<sup>+</sup>12] John Qualter, Frank Sculli, Aaron Oliker, Zachary Napier, Sabrina Lee, Julio Garcia, Sally Frenkel, Victoria Harnik, and Marc M. Triola. The BioDigital Human: A Web-based 3D Platform for Medical Visualization and Education. *Studies in health technology and informatics*, 173:359–61, 2012.
- [RAAA19] Daniel L. Rubin, Mete Ugur Akdogan, Cavit Altindag, and Emel Alkim. ePAD: An Image Annotation and Analysis Platform for Quantitative Imaging. *Tomography*, 5(1):170 – 183, 2019.
- [RASFO7] Francesco Robbiano, Marco Attene, Michela Spagnuolo, and Bianca Falcidieno. Part-Based Annotation of Virtual 3D Shapes. In *Proceedings of the 2007 International Conference on Cyberworlds, CW '07*, pages 427–436, Washington, DC, USA, 2007. IEEE Computer Society.
- [RF17] Nadine Abu Rumman and Marco Fratarcangeli. Skin Deformation Methods for Interactive Character Animation. In José Braz, Nadia Magnenat-Thalmann, Paul Richard, Lars Linsen, Alexandru Telea, Sebastiano Battiato, and Francisco Imai, editors, *Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 153–174, Cham, 2017. Springer International Publishing.
- [RM08] Cornelius Rosse and José L. V. Mejino. *The Foundational Model of Anatomy Ontology*, pages 59–117. Springer London, London, 2008. [https://doi.org/10.1007/978-1-84628-885-2\\_4](https://doi.org/10.1007/978-1-84628-885-2_4).
- [Rob90] Gay Robins. Proportions of Standing Figures in the North-West Palace of Aššurnasirpal II at Nimrud. *Iraq*, 52:107–119, 1990.
- [Rob94] Gay Robins. *Proportion and style in ancient Egyptian art*. Ann S. Fowler, 1994.
- [Rou60] Irving Rouse. The classification of artifacts in archaeology. *American Antiquity*, 25(3):313–323, 1960.
- [RST<sup>+</sup>07] Kumar T. Rajamani, Martin A. Styner, Haydar Talib, Guoyan Zheng, Lutz P. Nolte, and Miguel A. González Ballester. Statistical deformable bone models

- for robust 3D surface extrapolation from sparse data. *Medical Image Analysis*, 11(2):99 – 109, 2007. <http://www.sciencedirect.com/science/article/pii/S136184150600034X>.
- [SA07] Olga Sorkine and Marc Alexa. As-rigid-as-possible Surface Modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, SGP '07*, pages 109–116, 2007.
- [SB85] Scott N. Steketee and Norman I. Badler. Parametric keyframe interpolation incorporating kinetic adjustment and phrasing control. *SIGGRAPH Comput. Graph.*, 19(3):255–262, July 1985.
- [Sca20] Andreas Scalas. Semantics-aware modelling framework, 2020. <https://github.com/andreascalas/SemanticModellingFramework.git>.
- [scu] Leap Sculpting. <https://gallery.leapmotion.com/sculpting/>.
- [SF98] Karan Singh and Eugene Fiume. Wires: A geometric deformation technique. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '98*, page 405–414, New York, NY, USA, 1998. Association for Computing Machinery.
- [SG71] G. Stiny and J. Gips. Shape grammars and the generative specification of painting and sculpture. In *IFIP Congress*, 1971.
- [SKA<sup>+</sup>16] Weiqi Shi, Eleni Kotoula, Kiraz Akoglu, Ying Yang, and Holly Rushmeier. CHER-Ob: A Tool for Shared Analysis in Cultural Heritage. In Chiara Eva Catalano and Livio De Luca, editors, *Eurographics Workshop on Graphics and Cultural Heritage*. The Eurographics Association, 2016.
- [SKDP13] Dirk Smeets, Johannes Keustermans, Vandermeulen Dirk, and Suetens Paul. meshSIFT: Local surface features for 3D face recognition under expression variations and partial data. *Computer Vision and Image Understanding*, 117(2):158 – 169, 2013.
- [SKR<sup>+</sup>06] Carsten Stoll, Zachi Karni, Christian Rössl, Hitoshi Yamauchi, and Hans-Peter Seidel. Template Deformation for Point Cloud Fitting. In Mario Botsch, Baoquan Chen, Mark Pauly, and Matthias Zwicker, editors, *Symposium on Point-Based Graphics*. The Eurographics Association, 2006.
- [SKU15] Christoph Schinko, Ulrich Krispel, and Torsten Ullrich. Built by algorithms - state of the art report on procedural modeling. In *3D-Arch 2015 – 3D Virtual Reconstruction and Visualization of Complex Architectures*, pages 469–479, 2015.

- [SM95] J.J. Shah and M. Mäntylä. *Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications*. A Wiley-Interscience publication. Wiley, 1995.
- [SMKF04] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton Shape Benchmark. In *Proceedings of the Shape Modeling International 2004, SMI '04*, page 167–178, USA, 2004. IEEE Computer Society.
- [SMS17] Andreas Scalas, Michela Mortara, and Michela Spagnuolo. 3D Annotation Transfer. In Tobias Schreck, Tim Weyrich, Robert Sablatnig, and Benjamin Stullar, editors, *Eurographics Workshop on Graphics and Cultural Heritage*. The Eurographics Association, 2017.
- [SMS20] Andreas Scalas, Michela Mortara, and Michela Spagnuolo. A pipeline for the preparation of artefacts that provides annotations persistence. *Journal of Cultural Heritage*, 41:113 – 124, 2020.
- [SMVS18] Andreas Scalas, Michela Mortara, Valentina Vassallo, and Michela Spagnuolo. Geometric and topological tools for quantitative analysis in archaeology: the Ayia Irini case study, 2018. Poster presented at *Shape Modeling International 2018*.
- [SP86] Thomas W. Sederberg and Scott R. Parry. Free-Form Deformation of Solid Geometric Models. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '86*, page 151–160, New York, NY, USA, 1986. Association for Computing Machinery.
- [Spa16] Michela Spagnuolo. Shape 4.0: 3D Shape Modeling and Processing Using Semantics. *IEEE Computer Graphics and Applications*, 36(1):92–96, 2016.
- [STA03] The Stanford 3D Scanning Repository. <http://graphics.stanford.edu/data/3Dscanrep/>, 2003.
- [Sti08] Matthew Stiff. *Cultural Heritage Protection Handbook n.3: Documentation of artefacts' collections*. UNESCO, 2008.
- [STLL13] F. Soler, J. C. Torres, A. J. León, and M. V. Luzón. Design of Cultural Heritage Information Systems Based on Information Layers. *J. Comput. Cult. Herit.*, 6(4):15:1–15:17, December 2013. <http://doi.acm.org/10.1145/2532630.2532631>.
- [SVJ15] Leonardo Sacht, Etienne Vouga, and Alec Jacobson. Nested Cages. *ACM Trans. Graph.*, 34(6):170:1–170:14, October 2015.

- [SVM<sup>+</sup>18] Andreas Scalas, Valentina Vassallo, Michela Mortara, Michela Spagnuolo, and Sorin Hermon. Shape analysis techniques for the Ayia Irini case study. In *Eurographics Workshop on Graphics and Cultural Heritage*, 2018.
- [SVM<sup>+</sup>19] Andreas Scalas, Valentina Vassallo, Michela Mortara, Michela Spagnuolo, and Sorin Hermon. An Automatic Approach for the Classification of Ancient Clay Statuettes Based on Heads Features Recognition. In Selma Rizvic and Karina Rodriguez Echavarría, editors, *Eurographics Workshop on Graphics and Cultural Heritage*. The Eurographics Association, 2019.
- [SWZ14] Nazli Sarkalkan, Harrie Weinans, and Amir A. Zadpoor. Statistical shape and appearance models of bones. *Bone*, 60:129 – 140, 2014.
- [SZG<sup>+</sup>20] Andreas Scalas, Yuanju Zhu, Franca Giannini, Ruding Lou, Katia Lupinetti, Marina Monti, Michela Mortara, and Michela Spagnuolo. A First Step Towards Cage-based Deformation in Virtual Reality. In Silvia Biasotti, Ruggero Pintus, and Stefano Berretti, editors, *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. The Eurographics Association, 2020.
- [TCL<sup>+</sup>13] G. K. L. Tam, Z. Cheng, Y. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X. Sun, and P. L. Rosin. Registration of 3D Point Clouds and Meshes: A Survey from Rigid to Nonrigid. *IEEE Transactions on Visualization and Computer Graphics*, 19(7):1199–1217, 2013.
- [TDS<sup>+</sup>16] Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 3D Skeletons: A State-of-the-Art Report. *Computer Graphics Forum*, 35(2):573–597, 2016.
- [TDZ19] Jiong Tao, Bailin Deng, and Juyong Zhang. A fast numerical solver for local barycentric coordinates. *Computer Aided Geometric Design*, 70:46 – 58, 2019.
- [Tob75] Richard Tobin. The Canon of Polykleitos. *American Journal of Archaeology*, 79(4):307–321, 1975.
- [Ume91] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.
- [Vas16] Valentina Vassallo. A 3D Digital Approach to Study, Analyse and (Re)Interpret Cultural Heritage : the Case Study of Ayia Irini (Cyprus and Sweden). In Stefano Campana, Roberto Scopigno, Gabriella Carpentiero, and Marianna Cirillo, editors, *CAA2015*, volume 229, pages 227–232. Archaeopress, 2016.

- [Vas17] Valentina Vassallo. *The archaeological collection of Ayia Irini (Cyprus) : A 3D digital approach to analyse and reinterpret a 20th century study*, pages 203–216. Artemide Edizioni, 2017.
- [VGH] V. Vassallo, S. Gasanova, and S. Hermon. Terracotta Small Figurines from Ayia Irini (Cyprus): Identification of the Production Patterns through the integration of 3D geometry and materials properties analysis. unpublished.
- [VJD<sup>+</sup>11] R. C. Veltkamp, S. van Jole, H. Drira, B. Ben Amor, M. Daoudi, H. Li, L. Chen, P. Claes, D. Smeets, J. Hermans, D. Vandermeulen, and P. Suetens. SHREC '11 Track: 3D Face Models Retrieval. In H. Laga, T. Schreck, A. Ferreira, A. Godil, I. Pratikakis, and R. Veltkamp, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2011.
- [VR15] Vinayak and Karthik Ramani. A gesture-free geometric approach for mid-air expression of design intent in 3D virtual pottery. *Computer-Aided Design*, 69:11–24, 2015. <http://www.sciencedirect.com/science/article/pii/S001044851500086X>.
- [WAB<sup>+</sup>20] Stephan Wenninger, Jascha Achenbach, Andrea Bartl, Marc Erich Latoschik, and Mario Botsch. Realistic virtual humans from smartphone videos. In *26th ACM Symposium on Virtual Reality Software and Technology, VRST '20*, New York, NY, USA, 2020. Association for Computing Machinery.
- [Wei00] Judith Weingarten. Reading the Minoan Body: Proportions and the Palaikastro Kouros. *British School at Athens Studies*, 6:103–111, 2000.
- [WH19] Chaoran Wang and Michael Hann. A Study of Terracotta Warrior Proportions Based on Grid Division. In *Proceedings of the International Association of Societies of Design Research Conference*, 2019.
- [Wik20a] Wikipedia contributors. Artistic canons of body proportions — Wikipedia, the free encyclopedia. [https://en.wikipedia.org/w/index.php?title=Artistic\\_canons\\_of\\_body\\_proportions&oldid=990623149](https://en.wikipedia.org/w/index.php?title=Artistic_canons_of_body_proportions&oldid=990623149), 2020. [Online; accessed 8-December-2020].
- [Wik20b] Wikipedia contributors. Shahr-e sukhteh — Wikipedia, the free encyclopedia. [https://en.wikipedia.org/w/index.php?title=Shahr-e\\_Sukhteh&oldid=993847600](https://en.wikipedia.org/w/index.php?title=Shahr-e_Sukhteh&oldid=993847600), 2020. [Online; accessed 26-December-2020].
- [Wik21] Wikipedia contributors. L-system — Wikipedia, The Free Encyclopedia, 2021. [Online; accessed 9-March-2021].



- [WP00] Lawson Wade and Richard E. Parent. Fast, Fully-Automated Generation of Control Skeletons for Use in Animation. In *Proceedings of the Computer Animation, CA '00*, pages 164–169, Washington, DC, USA, 2000. IEEE Computer Society.
- [WP02a] Lawson Wade and Richard E. Parent. Automated generation of control skeletons for use in animation. *Vis. Comput.*, 18(2):97–110, April 2002.
- [WP02b] Xiaohuan Corina Wang and Cary Phillips. Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '02*, page 129–138, New York, NY, USA, 2002. Association for Computing Machinery.
- [WTD14] Zhansong Wang, Ling Tian, and Wenrui Duan. Annotation and retrieval system of CAD models based on functional semantics. *Chinese Journal of Mechanical Engineering*, 27(6):1112–1124, November 2014.
- [XLG09] Chuhua Xian, Hongwei Lin, and Shuming Gao. Automatic generation of coarse bounding cages from dense meshes. *2009 IEEE International Conference on Shape Modeling and Applications, SMI 2009*, pages 21–27, 2009.
- [XLG12] Chuhua Xian, Hongwei Lin, and Shuming Gao. Automatic Cage Generation by Improved OBBs for Mesh Deformation. *Visual Computer*, 28(1):21–33, January 2012.
- [XLX14] Chuhua Xian, Guiqing Li, and Yunhui Xiong. Efficient and effective cage generation by region decomposition. *Computer Animation and Virtual Worlds*, 26(2):173–184, 2014.
- [XZG13] Chuhua Xian, Tianming Zhang, and Shuming Gao. Semantic Cage Generation for FE Mesh Editing. In *Proceedings of the 2013 International Conference on Computer-Aided Design and Computer Graphics, CADGRAPHICS '13*, pages 220–227, Washington, DC, USA, 2013. IEEE Computer Society.
- [YCSZ13] Xiaosong Yang, Jian Chang, Richard Southern, and Jian J. Zhang. Automatic Cage Construction for Retargeted Muscle Fitting. *Visual Computing*, 29(5):369–380, May 2013.
- [YGH13] Chih-Hao Yu, Tudor Groza, and Jane Hunter. Reasoning on Crowd-Sourced Semantic Annotations to Facilitate Cataloguing of 3D Artefacts in the Cultural Heritage Domain. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web – ISWC 2013*, pages 228–243, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

- [YRCA15] R. Yu, C. Russell, N. D. F. Campbell, and L. Agapito. Direct, Dense, and Deformable: Template-Based Non-rigid 3D Reconstruction from RGB Video. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 918–926, 2015.
- [YWLM11] Zhao Yin, Li Wei, Xin Li, and Mary Manhein. An automatic assembly and completion framework for fragmented skulls. In *2011 International Conference on Computer Vision*, pages 2532–2539, Nov 2011.
- [ZAC<sup>+</sup>17] Ran Zhang, Thomas Auzinger, Duygu Ceylan, Wilmot Li, and Bernd Bickel. Functionality-Aware Retargeting of Mechanisms to 3D Shapes. *ACM Trans. Graph.*, 36(4), July 2017.
- [ZDL<sup>+</sup>14] Juyong Zhang, Bailin Deng, Zishun Liu, Giuseppe Patané, Sofien Bouaziz, Kai Hormann, and Ligang Liu. Local Barycentric Coordinates. *ACM Trans. Graph.*, 33(6):188:1–188:12, November 2014.
- [Zha94] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput Vision*, 13:119–152, 1994.
- [ZJB<sup>+</sup>18] T Zogheib, R Jacobs, MM Bornstein, JO Agbaje, D Anumendem, Y Klazen, and C. Politis. Comparison of 3D Scanning Versus 2D Photography for the Identification of Facial Soft-Tissue Landmarks. *Open Dentistry Journal*, 2018.