University of Cagliari

PhD School of Mathematics
and Computer Science

xxvi cycle

PhD course in Computer Science

Understanding the Structure of 3D Shapes

PolyCubes and Curve-Skeletons

INF/01

Author:
Marco Livesu

Supervisor:
Prof. Riccardo Scateni

2012 - 2013

Acknowledgements

Marco Livesu gratefully acknowledges Sardinia Regional Government for the financial support of his PhD scholarship (P.O.R. Sardegna F.S.E. Operational Programme of the Autonomous Region of Sardinia, European Social Fund 2007-2013 - Axis IV Human Resources, Objective l.3, Line of Activity l.3.1.)

Sommario

In computer grafica gli oggetti tridimensionali sono spesso rappresentati in forma compatta, in modo da rendere più efficiente la loro analisi, manipolazione e trasmissione. Descrivere una forma in modo compatto significa andare oltre la mera informazione geometrica, rendendo esplicita la *struttura* dell'oggetto. Lo scopo di questa tesi è quello di proporre nuovi metodi per la generazione di descrittori di forma compatti. In particolare, nella prima parte verrà considerato il problema della generazione di policubi ottimali. I policubi sono poliedri ortogonali utilizzati in computer grafica per mappare superfici e volumi. La loro capacità di somigliare alla forma originale, ma allo stesso tempo avere una struttura geometrica semplice e regolare, è importante in diverse applicazioni, come il texture mapping o la generazione di modelli per simulazioni fisiche. La seconda parte della tesi sarà invece focalizzata sui descrittori mediali. In particolare, due nuovi algoritmi per il calcolo di scheletri curvilinei saranno presentati. Diversamente dagli algoritmi allo stato dell'arte, questi metodi utilizzano nozioni di percezione della forma provenienti dal campo delle scienze cognitive, pertanto sono totalmente indipendenti dal tipo, numero e qualità delle primitive geometriche tipicamente utilizzate per rappresentare oggetti tridimensionali.

Abstract

Compact representations of three dimensional objects are very often used in computer graphics to create effective ways to analyse, manipulate and transmit 3D models. Their ability to abstract from the concrete shapes and expose their *structure* is important in a number of applications, spanning from computer animation, to medicine, to physical simulations. This thesis will investigate new methods for the generation of compact shape representations. In the first part, the problem of computing optimal PolyCube base complexes will be considered. PolyCubes are orthogonal polyhedra used in computer graphics to map both surfaces and volumes. Their ability to resemble the original models and at the same time expose a very simple and regular structure is important in a number of applications, such as texture mapping, spline fitting and hex-meshing. The second part will focus on medial descriptors. In particular, two new algorithms for the generation of curve-skeletons will be presented. These methods are completely based on the visual appearance of the input, therefore they are independent from the type, number and quality of the primitives used to describe a shape, determining, thus, an advancement to the state of the art in the field.

Contents

Introduction

In computer graphics the concept of *feature detection* refers to methods that aim at computing abstractions of 3D models, in order to offer compact representations that efficiently encode meaningful information about the shape they convey. Typically, a 3D model is described by a set of geometric primitives glued together in some way to form a shape (e.g. a triangle mesh). These descriptions are purely geometric and do not provide any high-level information about the objects they represent. However, beneath the mere geometric information there is more. There is the *structure* of the shape, that helps algorithms to understand objects at a deeper level and create more effective ways to analyse, manipulate and transmit 3D models. The goal of feature detection is therefore to abstract from the concrete shapes and expose their structural relations, generating compact representations that help to better *understand* shapes.

An example of the importance of understanding 3D models is given in Figure 1. A Lego brick is characterized by a very simple geometry: it is essentially an hexahedron with four cylindrical stubs that serve to stack one brick on top of the other. The stubs are perpendicularly incident to the hexahedron, have same size and form a layout with many symmetries, to facilitate the stacking process. Moreover, the edges of a Lego piece are always straight and the faces planar, so as to ensure a perfect matching between adjacent bricks. Now, suppose that one wants to generate a new Lego piece, similar to the depicted one, but with a side bigger than its opposite. The easiest way to go would be to deform the original piece with some deformation algorithm, fixing the boundary of the side to preserve, and rescaling the other (Figure 1a). A general purpose algorithm (e.g. [BPGK06]) would deform the brick as if it was made of rubber, thus ignoring the structure of the shape and distorting the stubs as well as their relation with the core of the brick (Figure 1b). Therefore, to be successful, a deformation method should understand the shape at a deeper lever, recognizing its features and the way they relate with the whole (Figure 1c). Such an algorithm (e.g. [GSMCO09]) would apply the same deformation, but would also preserve the identity of a Lego piece, adding special constraints to preserve

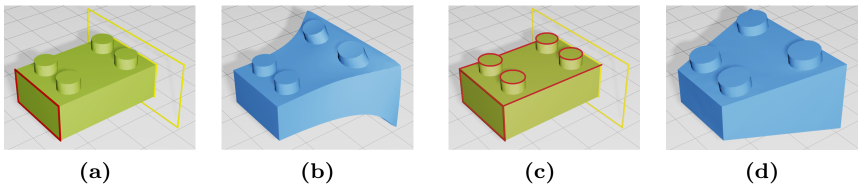


Figure 1: *Understand a shape and catch its features as well as how they relate with the whole is fundamental in many geometry processing tasks. Here a general purpose deformation algorithm performing on a Lego brick (a). The left side is fixed (red) while the opposite is being asked to scale (yellow). The result is unsatisfactory as the main features of the brick are not preserved (b). Adding more constraints to preserve the functionality of the object (c) leads to a better deformation, which satisfies the requirements for a real Lego brick (d) (image courtesy of [GSMCO09]).*

both shape and orientation of the stubs, and to generate straight edges and planar facets for the core of the brick, leading to a new, valid, Lego piece (Figure 1d). This is just a simple example that emphasizes the importance of recognizing the main features of a shape, the way they relate and their functionality, that is, understand a shape. The notion of *feature*, however, is quite vague and application-dependent. Discovering the features of a shape may require the computation of its symmetries [MPWC13], finding correspondences with similar objects [WSSC11, BBK06], locating sharp creases along its surface [HPW05] or segmenting the model in order to decompose it in meaningful parts according to some criterion [Sha08]. This thesis will focus on the identification of features aimed at the generation of two particular compact representations for 3D models: PolyCubes and Curve-Skeletons.

PolyCubes are high-level shape abstractions characterized by a very simple geometric structure [THCM04]. Mappings between a PolyCube and a general model are used in computer graphics in applications like hex meshing, spline fitting and texture mapping, to name a few. Due to their simple structure, performing some sort of geometric processing over a PolyCube is usually simpler than performing the same processing on a more complex shape. Therefore, algorithms operate on these simplified domains, and then map the results of their computation back to the original shapes using the associated mapping. It is known that a good PolyCube, that enables for a high-quality mapping with a general model, is topologically equivalent to the shape that represents, it visually resembles it and at the same time is *simple*, in the sense that it is composed by a small number of faces and corners [WYZ*11]. Consequently, in the context of PolyCube generation understanding a shape means being able to detect its most important features, to explicitly represent them in the domain. In this particular case, an *important feature* is a part of the shape that would decrease the distortion of the mapping if explicitly represented in the PolyCube. It is important to notice that such a feature may not be a *meaningful* one for a human observer. Computing high-quality PolyCube base domains can therefore be stated as a feature detection problem, according to the definition of feature given above. The first part of this thesis will address this problem and propose a solution to it.

Skeletons are another well known compact representation for 3D models in computer graphics. A curve-skeleton is a set of mono-dimensional curves arranged together in a graph-like structure, representing the main components of a shape as well as the way they connect to each other. In the context of curve-skeleton extraction understanding an object is mainly a matter of creating a one-to-one relationship between the skeleton parts and the logical components of the shape they convey. It is important to notice that, even though both PolyCubes and skeletons aim at detecting the main features of an object, they generally do not agree on the definition of feature. For the former, a feature is a part of a shape that has an explicit representation in the PolyCube domain; the latter tends to agree with the human definition of feature, that is, a part of the body for a living being or a logical component for a synthetic shape (e.g. the handle of a mug). Indeed, there is a close relation between this particular definition of feature and the human perception system. The second part of this thesis will try to formalize such a relation and exploit its properties, defining a new paradigm for the automatic detection of perceptually relevant features aimed at the generation of curve-skeletons.

Thesis overview. The thesis is organized in two separate parts. The first part regards PolyCubes. In particular, Chapter 1 will discuss the principal applications involving PolyCubes as well as state of the art algorithms for their computation. Chapter 2 will introduce a novel method for the generation of optimal PolyCube base domains based on the results that we presented in [LVS*13]. The second part of the thesis regards curve-skeletons. Chapter 3 will introduce the reader to the most important algorithms for skeleton computation and related applications. Chapter 4 will study the relations existing between human perception and skeletons while Chapter 5 will present two algorithms that exploit these relations to automatically generate curve-skeletons out of general 3D models. These methods are based on the results that we presented in [LGS12] and [LS13a]. Finally, Chapter 6 will draw the conclusions.

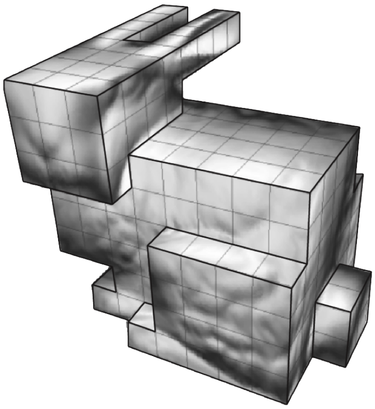
Part I

PolyCubes

Chapter 1

Background

Given a surface describing the shape of a three-dimensional model, and a second surface with same topology, the problem of computing a mapping between them is usually referred to as *parameterization.* The surface onto which the model is mapped to is called the parameter domain. Planar parametereziations (i.e. mappings to planar domains) were introduced for the first time in computer graphics in the context of texture mapping [BVI91, MYV93]. Since then, they have become an ubiquitous tool used to solve a number of geometry processing problems, like shape editing, remeshing, morphing and so on. Planar parameterization can be applied only to surfaces with disk-like topology while closed surfaces, as well as high genus surfaces, must be cut prior to be unfolded to a plane. However, these cuts introduce discontinuities in the parameterization that may lead to visible artefacts (Figure 1.1a).



For applications that are quite sensitive to discontinuities, like texture mapping and quad-meshing [BLP*13], the cut problem can be avoided by using a parameter domain topologically equivalent with the original model. This type of parameterization is called *non-planar* and the structure onto which the mapping is performed is usually referred to as *base-complex* [HLS07]. *PolyCubes* are simple base-complexes employed in computer graphics for parameterizing both surfaces and volumes (see Figure aside). The structure of a PolyCube is characterized by three important features: axis

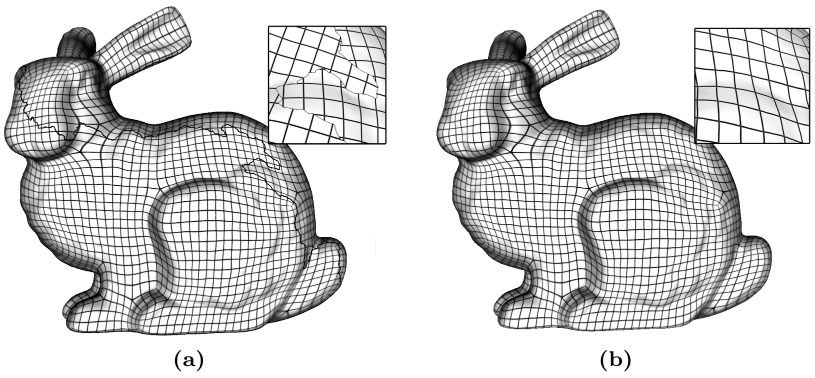


Figure 1.1: *Closed surfaces must be cut prior to apply planar parameterization, generating discontinuities in the parameter domain that may lead to visible artefacts. On the other hand, using a global method to parameterize a model onto a topologically equivalent base-complex no cuts are required and the mapping is continuous everywhere. In (a) a planar parameterization computed using ABF++ [SLMB05]; in (b) a global parameterization computed using the PolyCut algorithm presented in this thesis [LVS* 13].*

aligned edges, that always meet at 90° angles, and planar facets. Therefore, from a mere geometrical point of view, PolyCubes belong to the class of orthogonal polyhedra. They were introduced for the first time in [THCM04] in the context of seamless texture mapping. Furthermore, due to their ability to resemble the original shape more than other parameter domains [FSD07, GGS03] and due to their very regular structure, they have become popular in a number of applications, like hex-meshing and spline fitting (Section 1.2). In the remainder of the chapter a number of techniques for PolyCube computation will be presented (Section 1.1).

1.1 Previous works

Despite the high number of applicative scenarios involving PolyCubes (Section 1.2) there is just a handful of methods for computing orthogonal polyhedra out of general 3D models. In [THCM04] Tarini *et al.* showed how to compute a PolyCube mapping for a general model, however, they assumed the PolyCube structure to be an input sketched by hand by a professional user. Manually generating the parameter domain may be a tedious task, especially for complex models. Moreover, alternative global parameterization techniques were already able to automatically generate their parametric domains [LSS*98, GVSS00, PSS01, KLS03, SAPH04, KS04] and were often preferred by applications.

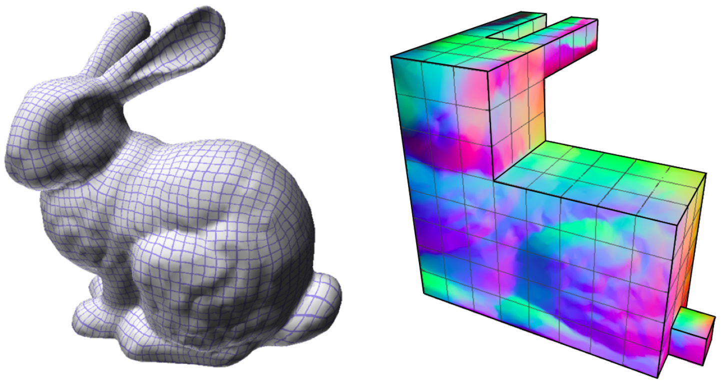


Figure 1.2: *A PolyCube for the Stanford bunny model generated using [LJFW08].*

1.1.1 Topology-based

The first algorithm for the automatic computation of a PolyCube was introduced by Lin *et al.* in [LJFW08]. Their approach is based on a segmentation technique that employs Reeb graphs to detect large protrusions and handles of a model, so as to catch its topology. Every single element of the segmentation is then mapped to a *basic polycube primitive*, that is a template that roughly resembles the shape of the part (e:g: O-shape, U-shape, L-shape). The surface is finally mapped to the PolyCube geometry and optimized in order to reduce the parametric distortion (Figure 1.2). This approach has some limitations as it is not guaranteed to produce a bijection nor to work with complex topologies. For example, if a node of the Reeb graph has valence higher than 6 it is difficult to use the templated solutions to approximate the shape. Moreover, this method tends to produce very coarse base domains that weakly resemble the original model, often resulting in excessive parametric distortion.

1.1.2 Divide-and-conquer

In [HWFQ09] He *et al.* proposed an alternative method based on the *"divide-and-conquer"* paradigm. The input model is firstly sliced into layers in the gradient direction of a harmonic function, generating a number of cross-sections (Figure 1.3a). The contours defined by each cross-section are then approximated by axis-aligned polygons using a quad-tree method (Figure 1.3b) and extruded along the third dimensions in order to construct the final base domain (Figure 1.3c). Finally, an intrinsic mapping between the original surface and the PolyCube is computed. This method is able to produce high quality PolyCube mappings, both in terms of angle and

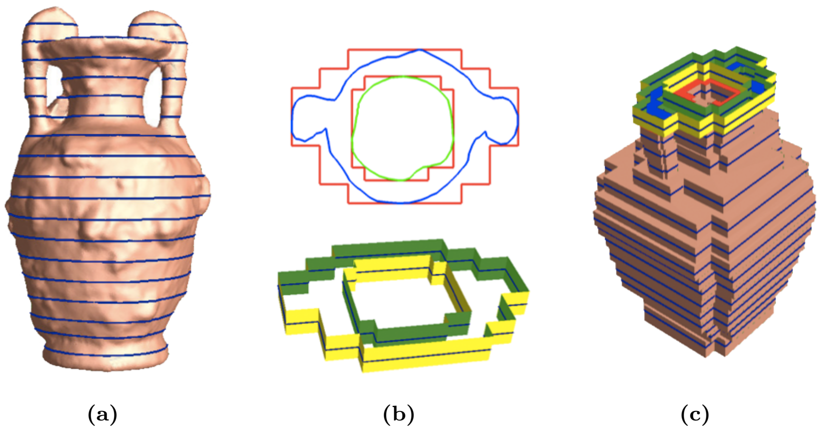


Figure 1.3: *The divide-and-conquer approach firstly defines a set of cross sections on the input model (a), then approximate each cross-section with an axis aligned polygon and extrude it to the third dimension (b), finally merges every cross section to define the ultimate PolyCube (c) (image courtesy of [HWFQ09]).*

area distortion. However, it is quite sensitive to off-axis features, that tend to introduce multiple redundant corners in the base-complex, creating a *"staircase effect"*. Partly, this redundancy can be controlled by the user by choosing an appropriate distance between the cutting planes, but large separations between slices will introduce large distortion. Moreover, for models with complex topologies, slices must be kept sufficiently close, to make sure that topological inconsistencies will not occur.

1.1.3 Deformation-based

In [GSZ11] Gregson *et al.* investigated the problem of computing high-quality PolyCube mappings in the context of hex-meshing. Their method for the generation of the base domain rely on a deformation approach that iteratively rotate the normals of the input model in order to meet the closest axis of the global frame, naturally exposing the structure of the PolyCube (Figure 1.4). As topological issues may occur, a set of templated solutions may be applied in post-processing to generate a valid orthogonal polyhedron. However, as the authors acknowledge, this method is not guaranteed to generated a valid PolyCube for any input. The deformation-based method proved to be superior to previous algorithms both in terms of quality of the mapping and compactness of the base domain. Moreover, it does not suffer off-axis features, that tend to be recognized and mapped within the same facets of the PolyCube, preventing the typical staircase effect ex-

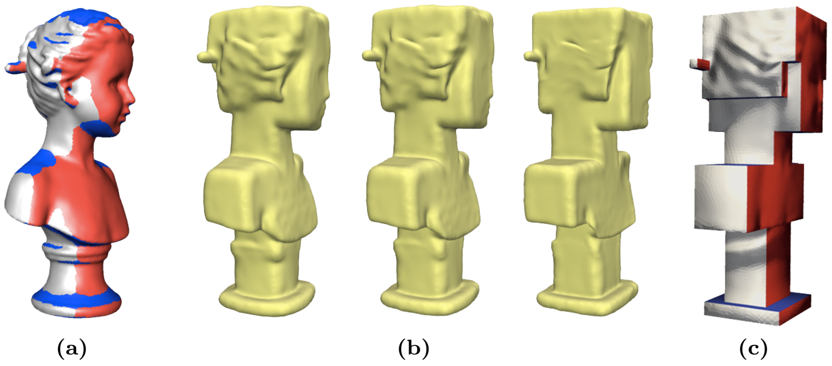


Figure 1.4: *The deformation-based algorithm proposed in [GSZ11] iteratively rotate surface normals, color coded with respect to the target orientation in (a), to naturally expose the structure of the PolyCube (b). Finally, axis-alignment and planarity constraints are forced to generate the ultimate orthogonal polyhedron (c).*

posed by approaches like [HWFQ09]. In [YZWL14] Yu *et al.* employed the deformation-based approach discussed in [GSZ11] as pre-processing step for their base complex construction algorithm. Then, they used a voxel-based approach to define a valid parameter domain, further simplified by applying *homotopic morphological operations* (e.g.: opening and closing).

1.1.4 Other methods

In [WYZ*11] Wan *et al.* observed that PolyCubes that better resemble the original model tend to generate lower distortion mappings. However, a PolyCube domain having too many corners is undesirable as corners correspond to singularities in the parameterization. Therefore, they proposed an iterative algorithm to seek for an optimal base domain meeting a target number of corners on top of which construct a low distortion mapping. Their initial base-complex can be computed either using [LJFW08, HWFQ09] or with a simple voxel-based approach discussed in the paper. However, their method tends to generate coarse base domains, thus suffering the same problems of [LJFW08].

Recently, Huang *et al.* defined a variational method, that turns an input triangle mesh into a PolyCube through minimization of the ℓ_1-norm of the mesh normals [HJS*14]. This is the first orientation-independent algorithm for PolyCube computation and promises to be able to achieve better results than state of the art counterparts.

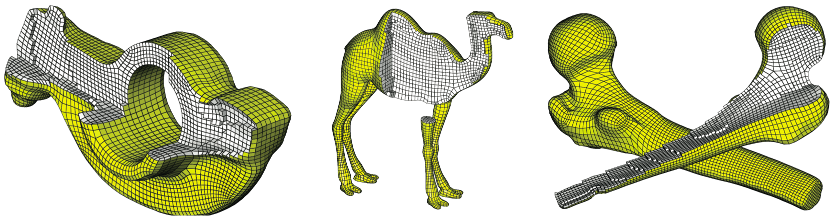


Figure 1.5: *Three examples of hexahedral meshing (image courtesy of [GSZ11]).*

1.2 Applications

The ability of PolyCubes to resemble the original models and their very regular structure make them the perfect domain for a variety of applications. This section will go through the most important techniques involving orthogonal polyhedra in computer graphics.

1.2.1 Hex meshing

Hexahedral meshes are important for finite element simulations (Figure 1.5). Compared to tetrahedral meshes they significantly enhance both speed and accuracy of PDE solvers, thus they are often preferred in fields like computational fluid dynamics and mechanics [LL10]. As a consequence, the problem of computing high quality hex meshes has been deeply studied by scientists. Some methods are based on the computation of a volumetric parameterization that aligns to a 3D frame field [NRP11, LLX*12]. However, the automatic computation of such a field is still an open problem. Furthermore, these methods are able to handle only a subset of the possible singularities that may occur. An alternative approach for hex meshing rely on a cross-parameterization between a PolyCube and a general model. The reason is easy to understand: orthogonal polyhedra always admit a trivial hex meshing that can be computed just by imposing a regular grid over the domain. Following the parameterization such grid can therefore map to the general model to generate the final hex mesh [GSZ11, YZWL14, HJS*14]. For these methods, the quality of the hex mesh is tightly coupled with the distortion introduced by the mapping. Alternatively, in [HXH10] Shuchu *et al.* constructed a surface mapping between an input object and a PolyCube base domain, and produced a full hexahedral shell mesh by considering a user-specified local thickness information provided in input as well.

1.2.2 Spline fitting

In [GHQ06] Gu *et al.* introduced manifold splines with the aim to provide a principled approach for the definition of continuous surfaces over

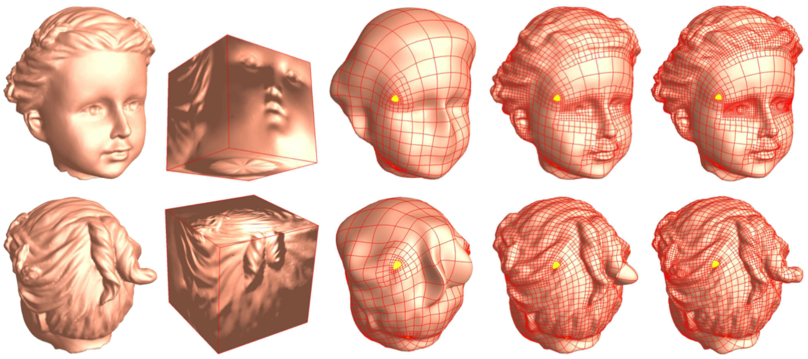
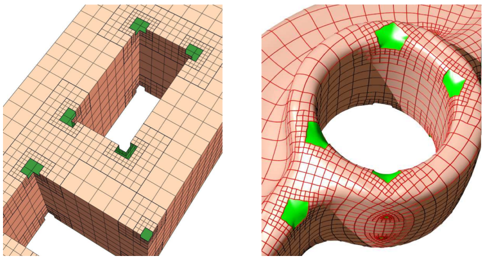


Figure 1.6: *PolyCubes can be used in the context of spline fitting. By further splitting the domain a growing level of detail representation of the target surface may be generated (image courtesy of [WHL* 08]).*

arbitrary manifold domains. Wang *et al.* extended their work by introducing PolyCube splines [WHL*08], that can be thought of as a particular instance of manifold splines in which the considered domain is an orthogonal polyhedron. The main reason for preferring PolyCubes over general manifolds is that orthogonal polyhedra have rectangular structure everywhere, therefore, they can be easily decomposed into a set of regular structures that facilitate tensor-product surface definition (Figure below). Low distortion PolyCube mappings can then be used to transfer these structures to the target geometry (Figure 1.6). In the context of spline fitting extraordinary points play an important role as they generate small holes in the construction of the manifold. While for general domains it is rather difficult to control both number and location of singularities, PolyCubes make their handling easier as singular points occur only at corners. Furthermore, in order to improve the positioning of singular points, in [WJH*08] Wang and colleagues proposed a new PolyCube mapping that allows users to manually specify the location of corners in the original model. A related line of work [LLWQ10, WLL*12, LQ12, LLWQ13] considers Generalized PolyCubes (GPC), that is coarse block-decomposition of low frequency models, for trivariate spline fitting. However, as opposed to standard PolyCubes, GPCs may not have a real global embedding in $\mathbb{R}^3$.

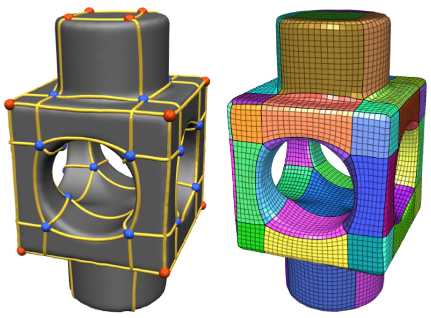


1.2.3 Others

Beside hex meshing and spline fitting there is a number of different scenarios in which PolyCubes have found (or may find) use. In [THCM04] Tarini and colleagues exploited the continuity of the texture coordinates over the parameter domain to generalize the concept of Geometry Images [GGH02] and efficiently pack texture data into rectangular images to produce seamless texture mappings. In [FJFS05] Fan *et al.* used a common base domain to parameterize similar shapes (e.g. quadrupeds) and induce a morphing between them. In [XGH*11] Xia and colleagues exploited the simplicity of orthogonal polyhedra to develop a user-assisted mapping enhancement in the context of real-time surface refinement and rendering.

Another line of research in which PolyCubes are of great interest is the mapping of a volume bounded by a surface with general topology onto a topologically equivalent base domain [XHY*10, LXW*10, PPL13, AL13]. Even though some of these methods are more general and do not necessarily require an orthogonal polyhedron as template domain, most of the times they use manually crafted complexes, therefore, they can benefit from automatic PolyCube generation algorithms.

Finally, the singularity graph containing both corners and edges of a PolyCube can be used to induce a quad-only layout over a manifold (Figure aside) [CBK12]. These coarse structures well adapt to the geometry of the models hence are useful for a number of applications, like parameterization [TPP*11], spline fitting and subdivision surfaces. Moreover, due to the underlying structure of the orthogonal polyhedron their refinement naturally leads to semi-regular quad meshes [BLP*13, HJS*14].



Chapter 2

PolyCut

PolyCubes, or orthogonal polyhedra, are useful as parameterization base-complexes for various operations in computer graphics. However, the construction of quality PolyCubes that provide an optimal trade-off between parametrization distortion and singularity count remains a challenge. Consequently, most of the times applications rely on manually or semi-manually constructed domains (Section 1.2).

This chapter will present *PolyCut* [LVS*13], a novel algorithm for the automatic computation of PolyCube base domains. The main observation is that, given an arbitrary input shape, the computation of a suitable PolyCube base-complex can be formulated as associating, or labelling, each single element of the surface with one of the six signed principal directions, inducing a segmentation over the object (Section 2.4). Therefore, the main challenge is to compute a segmentation that enables the computation of a low-distortion parameterization between the input model and the base domain, while keeping the number of singularities low. To accomplish this task, PolyCut is able predict for low distortion mappings, *understanding* how the structure of the PolyCube fits the original shape and recognizing the main features of the object, whether they are aligned with the principal axes or not. As shown in Section 2.2, alternative methods fail at achieving this level of understanding therefore they either introduce unnecessary singularities or produce distorted mappings.

Section 2.1 will discuss a set of proxies that can be used to predict for low distortion mappings while generating the PolyCube structure. Section 2.2 will motivate this work, emphasizing the main weaknesses of previous approaches. Details of the algorithm will be given from Section 2.3 to Section 2.5. Finally, results and comparisons with previous methods will be discussed respectively in Section 2.6 and Section 2.7.

2.1 Proxies for predicting distortion

As observed in [WYZ*11] PolyCubes that better resemble the original model tend to generate lower distortion mappings. However, the cost to pay is a higher number of corners, that is, a higher number of singularities in the parameterization. On the other hand, too coarse domains will have a small number of singularities but will not enable algorithms to compute high quality mappings (Figure 2.1). Since to estimate the actual quality of a mapping both a PolyCube and a parameterization are needed, the goal is to be able to *predict* for low-distortion maps while computing the structure of the base domain, in order to weigh the compactness of the PolyCube with the quality of the parameterization that will be computed on top of it. Understanding the high-level *structure* of the input and reflecting it in the base domain is therefore necessary to compute good PolyCubes. To do that, reliable proxies are needed.

2.1.1 Fidelity

Given a triangle mesh and a target PolyCube, the way the model maps to the base domain can be expressed as a segmentation. Essentially, each chart will map to a PolyCube facet (i.e. a planar axis-aligned polygon) and have ninety degrees dihedral angles along its boundaries. Consequently, as observed by Gregson and colleagues, given a suitable coordinate system, a local proxy for predicting distortion can be provided by measuring the angle between the normal of each triangle and its target orientation in the base domain [GSZ11]. In the remainder of this thesis this metric will be referred to as geometric *fidelity* [LVS*13]. Fidelity serves as a proxy for estimating the distortion caused by flattening each chart and rotating it so as to form ninety degrees dihedral angles with adjacent charts.

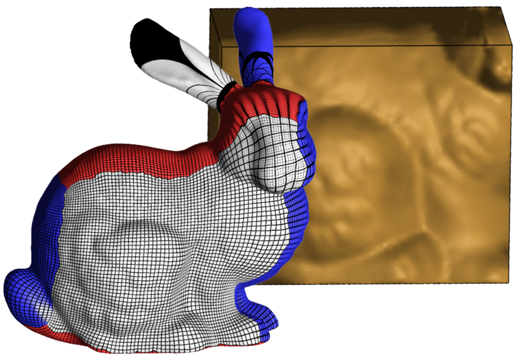
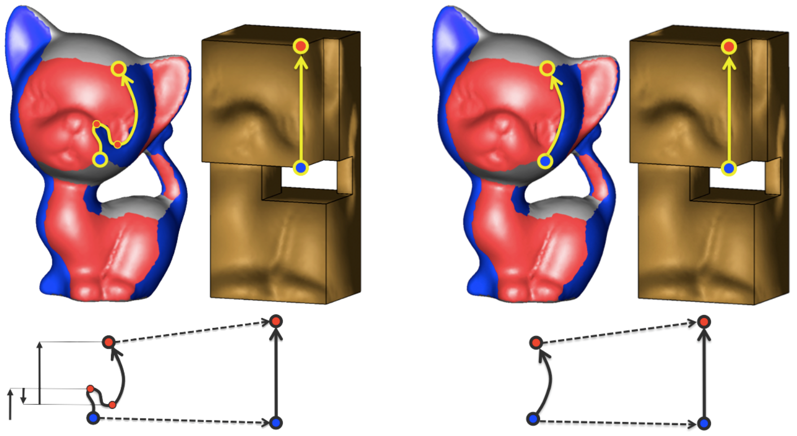


Figure 2.1: *Mapping a complex geometry into a coarse PolyCube will always lead to a distorted mapping, regardless the algorithm used for its computation. Aside, the mapping between the Stanford bunny and the simplest genus-0 PolyCube: a hexahedron. As the global structure of the bunny is not reflected in the base domain the mapping may have severe distortion (e.g.: in the ears).*



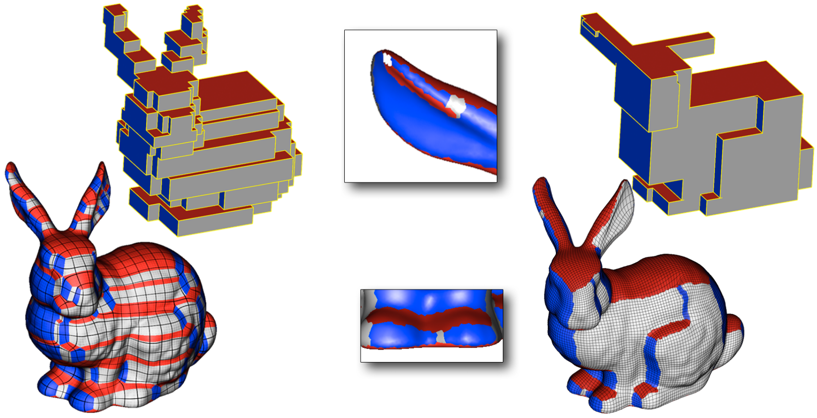
(a) *non-monotone boundary* **(b)** *monotone boundary*

Figure 2.2: *Boundaries in the segmentation will map to axis-aligned segments in the PolyCube. If, following a boundary from corner to corner, there is an inversion of orientation respect to the target orientation in the PolyCube (a), the boundary is non-monotone and its mapping will generate a severe amount of distortion. Oppositely, if there is no inversions of orientation (b), the boundary will nicely map in the PolyCube edge enabling a low distortion mapping.*

2.1.2 Monotonicity

Fidelity alone is not a good proxy for predicting distortion. An additional source of distortion comes from the need to map chart boundaries to the axis-aligned straight edges of the PolyCube. As a consequence, both shape and direction of the boundaries must be taken into account. In particular, each boundary has a uniquely defined orientation with respect to the corresponding axis, defined by the cross-product of the normals associated with the incident charts. Boundaries where the direction, as computed on the input segmentation, switches sign are called *non-monotone* (Figure 2.2a) and the points around which the switching occur are *turning points*. *Straightening* a non-monotone boundary to map it to a PolyCube edge would require to open it up around turning points, thus locally introducing severe distortion in the parameterization. To enable low distortion mappings, it is therefore necessary to compute segmentations with all monotone boundaries, having no turning points (Figure 2.2b).

It is important to notice that neither fidelity nor monotonicity alone would enable a reliable prediction. The proxy has to be composed by both ingredients, as they serve to estimate different types of distortion that may



(a) *divide-and-conquer* **(b)** *deformation-based*

Figure 2.3: *The divide-and-conquer algorithm [HWFQ09] for PolyCube computation generates too complex domains, having a high number of redundant corners (a). The deformation-based approach [GSZ11] is able to produce more compact domains (b), however unnecessary charts and corners still appear in the final base-complex (close-up).*

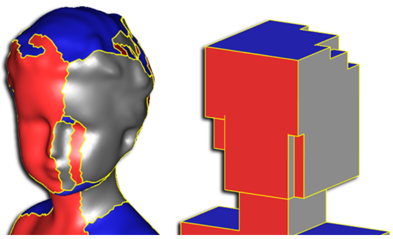
occur during the computation of the mapping.

2.2 Motivation

Despite the high number of applications involving PolyCubes as parameterization domains, state of the art algorithms do not produce satisfactory results yet. This section will set the main requirements for constructing high quality PolyCube base domains and will motivate PolyCut showing why current methods fail at achieving them.

As already pointed up for the generation of high quality PolyCube mappings it is important to balance *compactness* (i.e. the number of corners and facets) and *fidelity*. On the one hand, orthogonal polyhedra that better resemble the original shape tend to generate better mappings; on the other hand, too fine domains will generate a high number of singularities in the parameterization. Moreover, for methods that generate the PolyCube structure as a result of a segmentation process (e.g. [GSZ11]), the chart connectivity has to define a *valid* topology, that is, an orthogonal polyhedron that has a real global embedding in $\mathbb{R}^3$. Finally, the boundaries of such a segmentation should be *monotone*, in order to nicely map in the straight axis-aligned edges of the PolyCube and avoid unnecessary distortion.

Summarizing, a good PolyCube is compact, accounts for fidelity, has a valid structure, and the segmentation it induces on the input model has only monotone boundaries. The topology-based method (Section 1.1.1), as well as the voxel-based method (Section 1.1.4), produce too compact domains with low fidelity, thus they do not enable for low-distortion parameterizations. The divide-and-conquer approach (Section 1.1.2) is able to produce better quality mappings but lacks compactness, introducing too many singularities in the parameterization. This depends on the fact that each slice is approximated by a quad-tree, which tends to make the resulting domain an overly voxelized version of the input. Moreover, typical staircase artefacts appear when big features of the input do not align with the global frame (see the ears of the bunny in Figure 2.3a). The deformation-based method proposed by Gregson and colleagues (Section 1.1.3) improved over the state of the art in terms of both fidelity and compactness, generating coarser domains that better resemble the original shape and low-distortion cross-parameterizations. However, as can be noted in the close-ups in Figure 2.3b, PolyCubes still have extra singularities, mainly introduced by a heuristic post-processing that the authors employ to improve monotonicity. In spite of that, as can be noted in the the boundary between the red and blue charts on the forehead of the bimba statue (Figure aside), this method may fail at producing all-monotone boundaries and can therefore generate extreme amounts of distortion for some inputs. Finally, it is important to remember that this approach is not guaranteed to produce valid PolyCubes for any input (Figure 2.4).



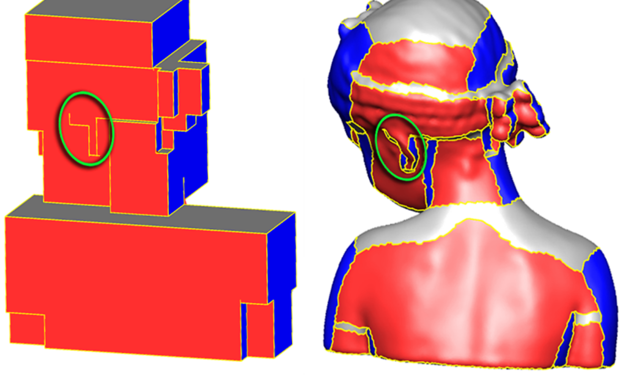


Figure 2.4: *The deformation-based method proposed by Gregson et al. is not guaranteed to generate a valid orthogonal polyhedron for any input. Here is an example of failure for the bimba model (see green oval).*

2.3 Overview

PolyCut takes a triangle mesh as input and outputs both a corresponding PolyCube base-complex and a cross-parameterization between the two. The key insight is that while directly optimizing for a base domain that minimizes distortion is hard, it suffices to optimize for the four criteria defined in Section 2.2, using fidelity and monotonicity as a reasonable proxy for predicting distortion. The problem of computing a distortion-minimizing parameterization can be therefore decoupled from the generation of the PolyCube structure so as to treat them as two separate processes. The first step can be thought of as a *topological* step which produces a PolyCube segmentation. The second step, the *geometric* one, will use the resulting segmentation to generate the actual PolyCube and the relative mapping.

It is known that the necessary and sufficient topological conditions on PolyCube validity remain unknown [EM10]. However, in their work Eppstein and Mumford listed a set of criteria for valid genus zero orthogonal polyhedra. As they observed, a segmentation will not produce a valid orthogonal polyhedron if charts associated with opposite orientation of the same axis share a boundary, or if any chart has less than four neighbours. Additionally, one cannot guarantee that a PolyCube embedding of a graph exists unless each vertex in the graph has valence three.

Most of the requirements for a good PolyCube segmentation can be therefore expressed locally, and involve a condition on either one triangle, or two adjacent triangles. In particular, fidelity can be expressed via per-triangle labelling preferences, measuring the angular distance between the (oriented) axes of the global frame and its normal direction. Pairwise label correlations between adjacent triangles can be also used to encode the preference for compact segmentations, i.e. ones that have shorter inter-chart boundaries. Finally, two out of three of the Steinitz criteria for orthogonal polyhedra [EM10] can be expressed as purely local constraints.

Unfortunately, monotonicity is not a condition that can be formulated locally, as detecting a change in a boundary orientation with respect to the corresponding axis requires at the very least considering two adjacent mesh edges. Moreover, such purely local evaluation may provide multiple false positives as it will depend on the local mesh connectivity. Once a labelling is computed, however, the situation improves: non-monotone boundaries can be robustly detected and turning points identified. As shown in Figure 2.6, such boundaries typically can be resolved by locally adjusting the segmentation in the problematic areas. Similarly, one cannot a priori locally optimize for, or even evaluate, the number of neighbours before a labelling is computed. Once a labelling exists, however, counting them is trivial.

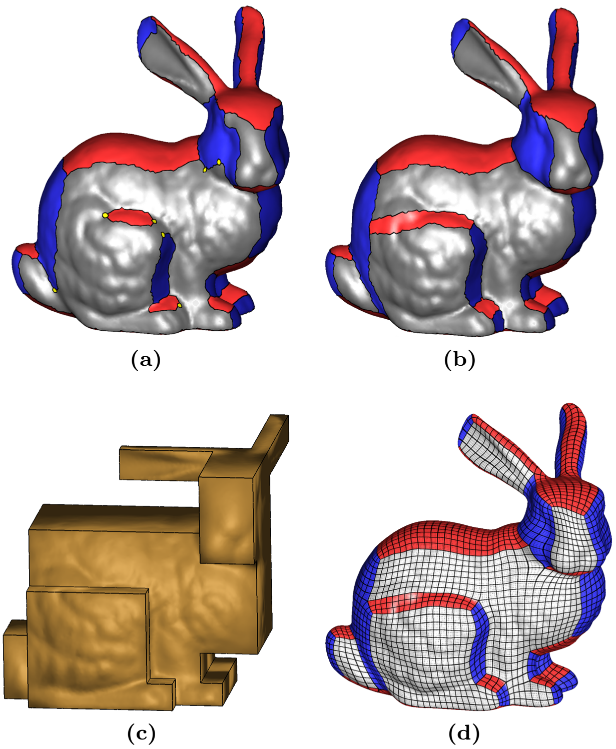


Figure 2.5: *Overview of the PolyCut algorithm. An initial segmentation accounting for fidelity and compactness is computed by solving a classical labelling problem (a). Non-monotone boundaries (turning points highlighted in yellow) are resolved with a constrained local search framework, generating a segmentation with all monotone boundaries (b). The input model is then deformed to expose the PolyCube geometry (c) and a cross-parameterization between the input model and the base domain is computed (d).*

Motivated by these observations, PolyCut solves for both local and global constraints in a interleaved manner, generating PolyCube base domains that account for fidelity and compactness, are valid, and the boundaries of the segmentations they induce on the input models are always monotone and feature-aligned. Firstly, a multi-label graph-cut optimization is used to compute locally optimal segmentations that balance fidelity and compactness (Section 2.4.1). By minimizing chart boundary length, this computation also tends to reduce boundary curvature. However, it may fail at achieving monotonicity for some input geometry, as non-monotone boundaries may better satisfy local fidelity and sometimes also be shorter

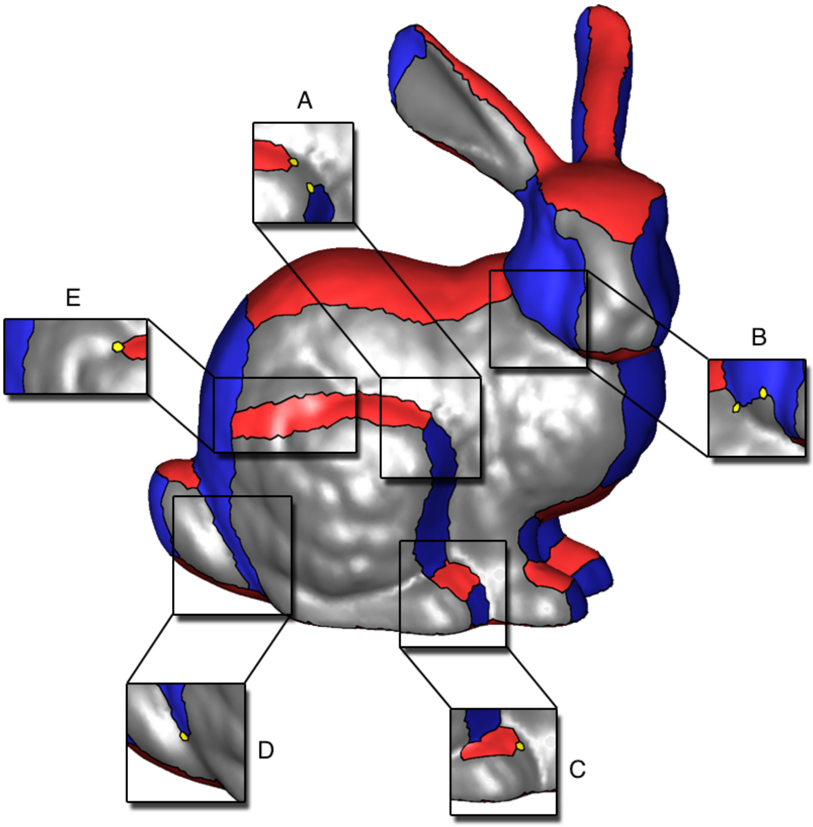


Figure 2.6: *Zooming-in on a set of boundaries in a PolyCube segmentation of the Stanford bunny (in the close-up their non-monotone initial guesses). A variety of topological strategies can be used to achieve monotonicity: A. extending two non-monotone boundaries to reach one another; B. straightening a boundary; C. introducing a new chart with different label; D-E. extending the non-monotone boundary to reach another boundary. Note that the list is not exhaustive as multiple turning point configurations can exist within one chart or even along a single boundary curve.*

than monotone alternatives (Figure 2.5a). The multi-label graph-cut optimization is employed as a building block within a global discrete constrained optimization framework (Section 2.4.2). The goal of this framework is to enforce monotonicity and validity while minimally increasing the fidelity and compactness costs compared to an unconstrained solution. It is important to notice that, in order to enforce monotonicity with minimal cost increases, a variety of topological strategies for resolving each detected turning point must be supported (Figure 2.6): e.g. extending the boundary until the turning point touches another chart and the boundary gets split in two (D, E), straightening the offending boundary portion (B), or even adding a new chart with a different label from those sharing the non-monotone boundary (C). The optimal choice depends on the local geometry, and may involve complex interplay between nearby non-monotone boundaries (A). Instead of a set of rigid heuristics, such as the one used by the post-process in [GSZ11] the solution here is guided by the input data, and in particular by the fidelity term, which encodes the local surface geometry.

2.4 Segmentation

The first step of the algorithm can be summarized as follows. A locally optimal segmentation that accounts for fidelity and compactness is computed with a multi-label graph-cut method (Section 2.4.1). Then, an iterative optimization framework is used to explore the space of segmentations to find the valid all-monotone labelling as close as possible to the initial guess (Section 2.4.2). At each iteration, this framework locally perturbs the fidelity terms in the vicinity of turning points on the detected non-monotone boundaries, and re-applies the graph-cut optimization with a set of constraints aimed at minimizing deviation from the current solution. The results of the relabelling are used to update the constraints, and to guide the location and magnitude of the next perturbation. To facilitate convergence, at each iteration both validity and monotonicity of the produced charts are checked, to prevent the subsequent labelling steps from changing the shape of any new valid charts with all monotone boundaries. The process terminates once all charts are deemed valid and monotone. This process can be seen as a special case of the classical hill-climbing [HS04] discrete optimization framework. The hill-climbing approach produces a locally near-optimal result which satisfies global constraints, but is not designed to explicitly consider parameterization distortion, and in particular boundary alignment with the corresponding oriented axes. Therefore, to improve the overall quality of the base domain, chart boundaries are eventually fine-tuned while keeping the chart topology intact (Section 2.4.3).

2.4.1 Graph-Cut labelling

The graph cut algorithm described here provides the starting point, or initial guess, for the local search labelling framework. The fidelity versus compactness trade-off is expressed as an energy minimization problem, in which each triangle t must be assigned to one of six discrete labels. Each label $s \in \{-X, +X, -Y, +Y, -Z, +Z\}$ represents a signed normal direction $\vec{s}$ along the corresponding axis. The goal is to find a segmentation S which

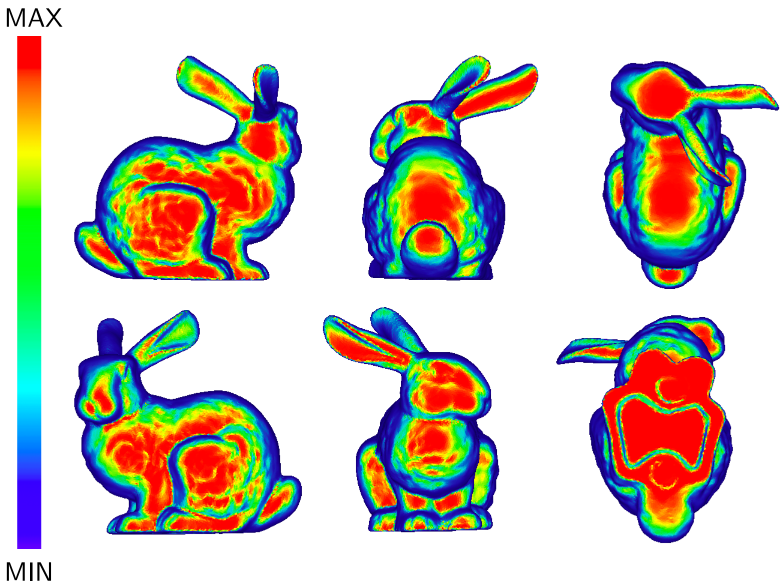
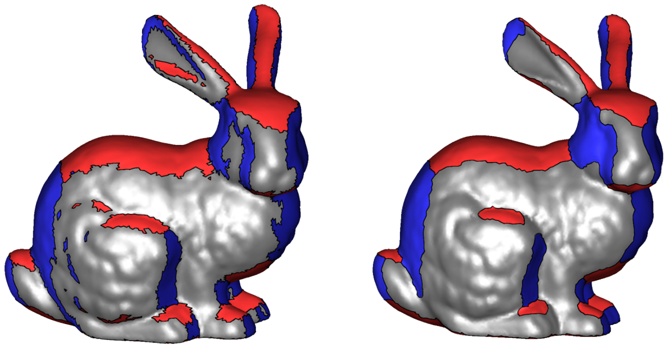


Figure 2.7: *A color-coded plot of the fidelity term for the six labels associated with the oriented directions of the global frame. Red areas have maximum attraction (lowest cost) with respect to the considered label while blue areas have minimum attraction (highest cost).*



(a) $c = 1$ **(b)** $c = 3$

Figure 2.8: *Different values for the c parameter will accommodate segmentations with a different level of compactness. The thumb rule is: the higher the value of c, the less boundaries will appear in the segmentation and the more compact the PolyCube will become. However, as can be noted in (a), too small values will generate artefacts at noise level that will easily disappear as soon as the compactness value grows (b).*

minimizes the energy

$$E(S) = \sum_{t \in T} F_t(s_t) + c \cdot \sum_{pq \in E} C_{pq}(s_p, s_q) \tag{2.1}$$

where the unary fidelity term $F_t(s_t)$ describes the cost of assigning the label s_t to a triangle t, and the binary compactness term $C_{pq}(s_p, s_q)$ is the cost of assigning the label s_p to a triangle p and the label s_q to a triangle q. The constant c serves as a user control for the overall compactness of the PolyCube; the higher the value of c, the more compact the resulting PolyCube becomes (Figure 2.8). Unless otherwise specified, $c = 3$.

Fidelity term. The cost of assigning a triangle t to a given label s is measured as a function of the angle between the triangle normal $\vec{n}_t$ and the label direction $\vec{s}$

$$F_t(s) = 1 - \exp\left[-\frac{1}{2}\left(\frac{\vec{n}_t \cdot \vec{s} - 1}{\sigma}\right)^2\right] \tag{2.2}$$

Setting $\sigma = 0.2$, using the three-sigma rule, yields a labelling cost that goes from 0, when the triangle normal and the label direction are perfectly aligned, to close to 1, when the angle between the normal and the label direction is approximately 65°. A normal that is equidistant to each of the axes of the global frame will be approximately 55° far from each of them. Therefore, this choice of maximum penalty weakly differentiate between labels whose axes are close to 55°. Figure 2.7 proposes a visual plot of the fidelity terms associated to the six labels.

Compactness term. The compactness term is designed to minimize boundary length. Therefore, the cost $C_{pq}(s_p, s_q)$ will be 0 if two adjacent triangles share the same label and will be greater than zero for adjacent triangles assigned to different labels. In particular, the compactness cost will be a function of the dihedral angle between the two triangles

$$C_{pq}(s_p, s_q) = \begin{cases} 0 & \text{if} \quad s_p = s_q \\ \exp\left[-\frac{1}{2}\left(\frac{\vec{n}_p \cdot \vec{n}_q - 1}{\sigma}\right)^2\right] & \text{if} \quad s_p \neq s_q \end{cases} \tag{2.3}$$

Using $\sigma = 0.25$ the cost will be 1 for coplanar triangles (maximum penalty) and e^{-8} for 90° dihedral angles (minimum penalty). It is important to notice that, even though a constant penalty would be sufficient to minimize boundary length, the function above will also generate boundaries that prefer sharp creases more than smooth areas, thus trying to align with the features of the shape.

Validity. Two of the validity constraints described in Section 2.3 are directly accounted for by this local optimization process. In particular, two triangles p and q sharing an edge or a vertex are explicitly prevented from being assigned to opposite direction labels $-L$ and $+L$. Moreover, corners with valence higher than three are eliminated after the generation of the labelling, by rerunning the optimizer with special constraints that split each corner into two. To generate the split, the most prevalent chart label is chosen and all triangles in the one-ring are forced to take it, setting the corresponding fidelity cost to zero (and to infinity for all the other labels). In practice, PolyCut never needed to run this step more than once.

The solver used to minimize the energy function (Equation 2.1) is a graph-cut multi-label optimization framework based on [BVZ01,KZ04,BK04]. An implementation of such framework can be downloaded at `http://vision.csd.uwo.ca/code/gco-v3.0.zip`.

2.4.2 Iterative local search via Hill Climbing

As previously noted, the initial labelling generated by the graph-cut method possesses most of the required properties for a PolyCube segmentation, and is close, in the space of all possible labellings, to a valid solution with monotone boundaries. This section will discuss the framework used to turn the initial, possibly non-valid non-monotone labelling, in a valid all-monotone PolyCube segmentation. The question to face is therefore how to obtain such a segmentation in a principled way. The desired output can be achieved by using the original graph-cut optimizer after perturbing the per-triangle, per-label, fidelity costs (Equation 2.2) in the vicinity of any turning point appearing in the initial segmentation (Figure 2.10). In other words, instead of resolving turning points by directly editing the segmentation, the idea is to resolve them indirectly, by tweaking the per-triangle fidelity costs and searching for a set of small local changes that will result in a valid all-monotone segmentation (Figure 2.9). This is a classical search problem, with a search space that is exponentially large with respect to the number of input triangles. To make this problem tractable, the labelling algorithm is embedded within an iterative local search, or *hill-climbing* framework, which guides the perturbations toward promising local solutions. At each step of this framework, valid charts with all-monotone edges are detected and *frozen*, to make sure that they will not change during subsequent steps. Specifically, the triangles belonging to each frozen chart will not be considered by any subsequent labelling operation, so that they will keep their current label. Moreover, to prevent boundary changes special constraints will be added. In particular, any triangle that shares either an edge or vertex with a frozen chart is prevented from being assigned either the same or its opposite label. The first boundary constraint prevents frozen charts from continuing to grow and potentially introduce new non-monotone boundaries, whereas the second prevents the formation of invalid chart layouts. Then, the algorithm proceed to incrementally change the per-triangle fidelity cost in the vicinity of the remaining turning points and reapply the labelling algorithm after each such edit. The iterative process terminates once all chart boundaries are classified as monotone. In the unlikely event that the resulting segmentation still has a chart with less than four boundaries, it is merged with one of its neighbours.

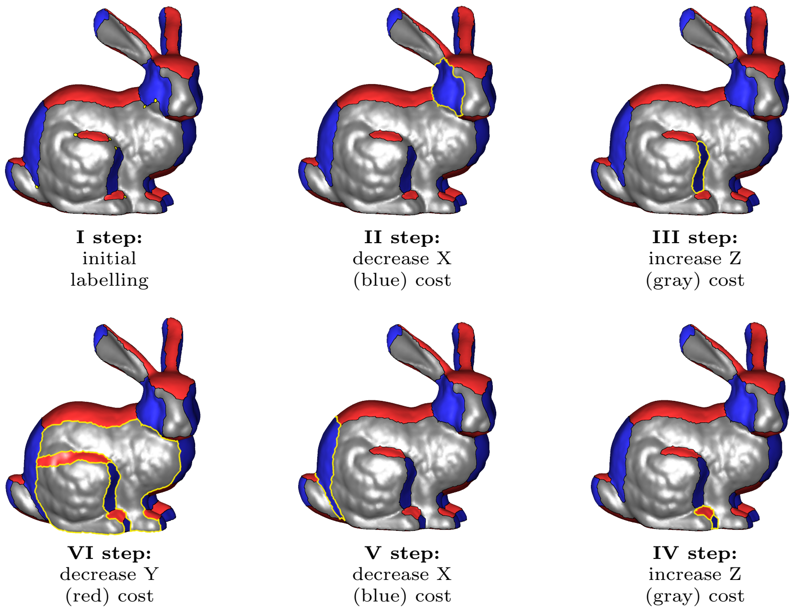

I step: initial labelling
II step: decrease X (blue) cost
III step: increase Z (gray) cost
VI step: decrease Y (red) cost
V step: decrease X (blue) cost
IV step: increase Z (gray) cost

Figure 2.9: *Hill-climbing visualization: shown are the steps at which valid monotone charts are detected and frozen. Depending on the input complexity the method takes between a few dozen to a few hundred iterations to converge. Note that as more charts are frozen the labelling problem size shrinks speeding the computation up. Newly frozen charts highlighted with yellow border.*

When exploring possible perturbations, the change that will best resolve a given turning point is not known *a priori*. In particular, for every turning point the fidelity cost with respect to any of the six labels could be either increased or decreased. Brute-force searching for the best direction to bias each turning point is exponential with respect to the number of turning points to be resolved. In order to reduce the size of the search space, the framework uses six search branches $\{X_{\mathsf{less}}, X_{\mathsf{more}}, Y_{\mathsf{less}}, Y_{\mathsf{more}}, Z_{\mathsf{less}}, Z_{\mathsf{more}}\}$, each of which expresses a preference towards, or away from, a given axis.

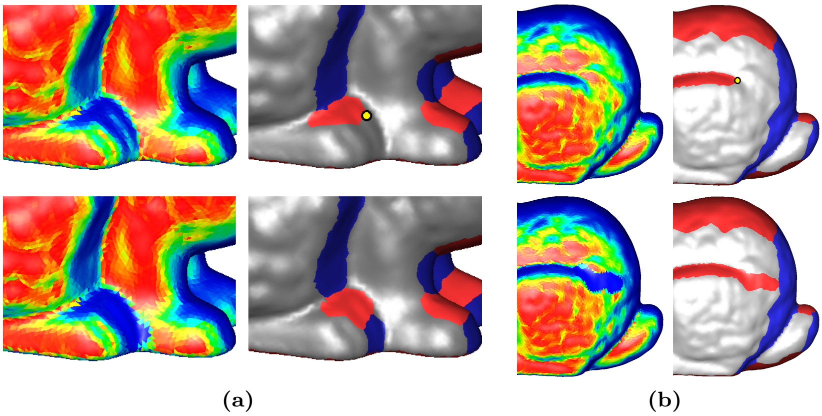


Figure 2.10: *Selective biasing, sometimes minuscule (a) and sometimes more significant (b) portions of the fidelity term around turning points resolves non-monotonicity. In (a) the original fidelity costs for positive X (gray) assignment and a turning point on the initial segmentation of paw. Slightly increasing this cost near the turning point resolves the boundary. In (b) same process for back thigh requires larger increase to be propagated across for the turning point to be resolved.*

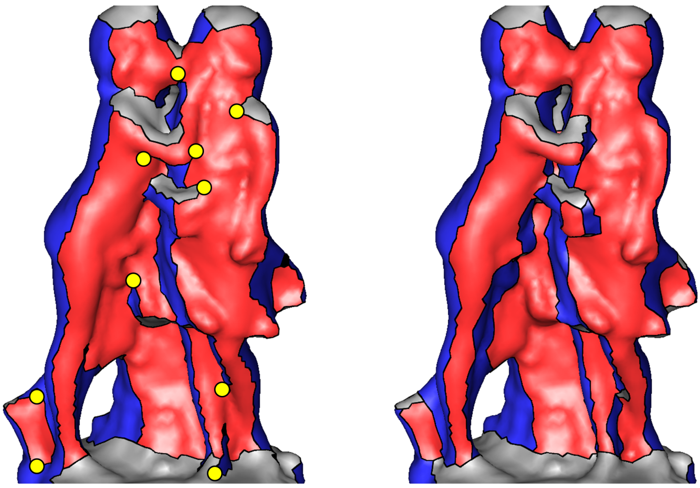


Figure 2.11: *The highly complex layout of the red chart (left) needs opposite bias directions, in different portions of its boundary, for a valid solution. While the fast communication strategy of the hill-climbing is not able to achieve validity, restarting the search after each topological change provides the desired flexibility (right).*

Each of these branches is seeded with the initial labelling, then it introduces an additive bias $\delta = 0.01$ to the fidelity term around each active turning point within a radius of 5% of the bounding box diagonal length. The value of δ is selected to weigh speed against accuracy; larger values would speed up computation but may reduce result quality. Searching each branch in a fixed direction reduces the number of elements in the search space from $O\left(6^{|t|}\right)$ to $O\left(|t|\right)$, or from exponential to linear time. While there are six label options, the framework uses one branch per-axis, instead of one per direction, as in practice only one of these directions is relevant for any given turning point. Also note that increasing the fidelity cost for the $\pm X$ labels is the same as simultaneously decreasing it for the $\pm Y$ and $\pm Z$ labels.

Since different turning points may be resolved by different branches, a communication strategy between them is needed. The framework is equipped with two communication strategies: a *fast* one, that works for most of the experimented inputs (Section 2.6), and a much slower, but more robust, *restartable* one. In the fast strategy, after biasing the fidelity costs and applying relabelling independently within each branch, newly valid charts are *frozen* in subsequent labelling steps across all branches. Though the rest of the processing remains independent per branch. In the restartable strategy, in addition to detecting newly valid charts, the framework also detects when any existing chart is split into two, indicating that a turning point had been eliminated by splitting a non-monotone boundary into two as well. When either event occurs, the hill-climbing process restarts using the current labelling, frozen charts, and perturbed fidelity terms as the initial input for all branches. The restartable strategy allows for finer-level perturbation control, necessary for resolving charts which require one perturbation direction along one boundary and an opposite direction along another. In practice, such charts are quite rare; among the tested models the kiss statue (Figure 2.11) was the only example which required the restartable strategy.

Turning points computation. To detect non-monotone chart boundaries the signed orientation of edges along them needs to be computed, using a consistent notion of right and left charts. A change in sign indicates a non-monotone boundary. Assigning orientation at an individual edge level is sensitive to minor changes in local mesh connectivity. To obtain more robust results, the same graph-cut framework used to provide the initial labelling (Section 2.4.1) can be used, this time with only two labels: positive and negative. The unary term is a Gaussian falloff function of the dot product of the edge and axis directions ($\sigma = 0.9$) and the binary term for consecutive edges is zero when they share the same label and a Gaussian of the dot product of the edge directions ($\sigma = 1.2$) otherwise. Fol-

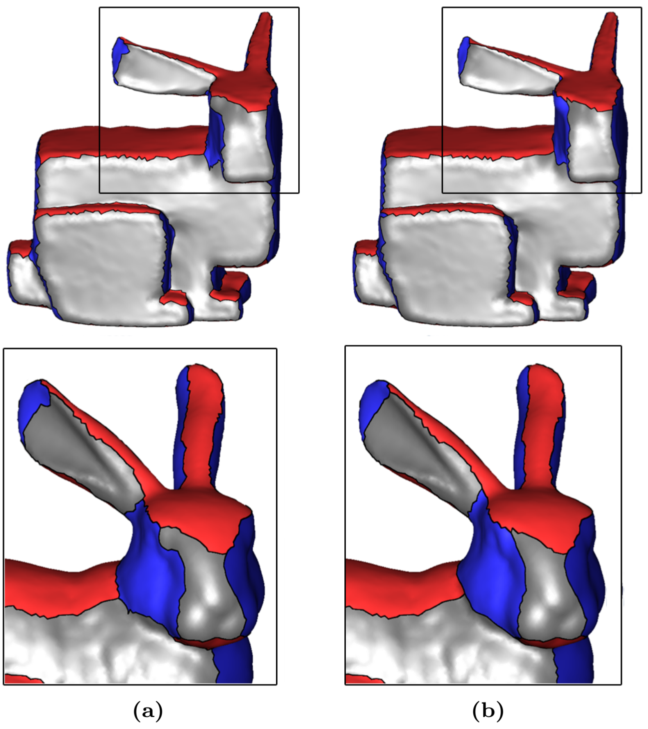


Figure 2.12: *PolyCut uses tentative PolyCube deformation (a) to reveal boundary alignment directions and use those to improve the segmentation boundaries. This process improves chart boundaries on the bunny face (b).*

lowing the labelling, boundaries with more than one segment are classified as non-monotone and the segment joints are marked as turning points.

2.4.3 Boundary refinement

While the hill-climbing framework aims for short and monotone boundaries, it makes no explicit effort to align them with their corresponding axial directions. The reason being that, while hill-climbing can control for labels assigned to pairs of adjacent triangles during the segmentation process, the orientation of each boundary is not known a priori (i.e. the order of its two corners along the axis they map to). Naively introducing an unordered axial bias into the original formulation would only lead to an increase in non-monotonicity, as it will encourage longer, axially-aligned, but erratic boundaries at the expense of straighter, but misaligned ones. However, given a segmentation computed in the hill-climbing step, the desired orientation of the boundaries can be estimated and used to improve the segmentation while preserving the topological information it encodes.

The challenge, as always, is to balance the boundary shape against fidelity. Unfortunately, the original unbiased fidelity term cannot be used any longer as using it will aim to revert the gains of the entire hill-climbing process. While it is theoretically possible to keep track of the individual changes to the fidelity costs introduced during the climbing step, combining those in a coherent way across the multiple branches can be difficult. Alternatively, computing fidelity on an approximated PolyCube suggested by the current labelling provides a good proxy for this biased costs. To generate this approximated PolyCube, an iterative deformation framework based around the one proposed by Gregson *et al.* in [GSZ11] can be used. The segmentation is therefore improved by feeding the graph-cut optimizer with fidelity terms computed over a deformed geometry and an updated compactness term which accounts for boundary orientation. The result of this process is an improvement of the alignment of chart boundaries with respect to the assigned orientations while still preserving validity and monotonicity (Figure 2.12).

PolyCube deformation. Given an input mesh and a PolyCube segmentation that assigns an orientation to each triangle normal, the input is deformed by rotating the normals toward these orientations, obtaining an approximate PolyCube geometry. Soft, rather than hard rotational constraints, are used as the current segmentation is clearly not final, and its aim is to balance normal rotation against input distortion. Similar to Gregson *et al.* [GSZ11], to avoid self-intersections and better preserve the input structure, a volumetric mesh of the input model is used for the deformation. Since any interior vertex in a chart has a one-ring consisting purely of triangles with the same labelling, it has a known assigned orientation. The minimum rotation for each vertex that aligns its normal and target orientation can be therefore computed. Then, these rotations propagate to vertices along the chart boundaries, as well as through the volumetric mesh interior. These rotations are used to compute the vertex positions in the deformed mesh, attempting to orient each edge in its new preferred direction while maintaining its length. Given original vertex coordinates v_i and v_j with rotation matrices ∇_i and ∇_j, new coordinates u_i and u_j are defined by minimizing

$$\sum_{ij} \left[(u_i - u_j) - \frac{\nabla_i + \nabla_j}{2} \cdot (v_i - v_j) \right]^2 \qquad (2.4)$$

over all mesh edges (i, j). Since the vertex positioning step only weakly satisfies the target rotations, the process is repeated two more times to obtain a sufficiently clear PolyCube. This framework will also be used later (with additional refinements) to compute the final PolyCube geometry (Section 2.5).

Boundary optimization. Deformed models and the orientation information are used to improve boundary alignment. Since the aim is to preserve the segmentation topology intact, the process iteratively apply the relabelling step to small portions of the segmentation, while preserving the topology. Specifically, pairs of charts that share boundaries and triplets of charts that share corners are repeatedly relabelled. To prevent topological changes, when applying the method to pairs of charts, the labels on the triangles along the non-shared boundaries of these charts are fixed, and the cost of assigning any label but the two participating ones is infinity. For similar reasons, for each corner only triangles within a radius of $\frac{1}{3}$ of the length of the smallest incident chart boundary are considered. Any label outside of this radius is fixed. Similarly to the boundary repositioning, the optimizer is restricted to considering only the three chart labels incident to the corner in question. While these constraints drastically reduce the likelihood of topology changes, they do not fully prevent them. If such a change does occur, the boundary is rolled-back.

The labelling framework uses the same fidelity function as the initial labelling (Equation 2.2) but estimates it on the deformed model. The compactness term is similarly computed on the deformed model, but is now modified to take into account how well a given boundary edge is aligned with its target orientation

$$C_{pq}(s_p, s_q) = \begin{cases} 0 & \text{if} \quad s_p = s_q \\ 1 - \exp\left[-\frac{1}{2}\left(\frac{\vec{e}_{pq}\cdot\vec{d}-1}{\sigma}\right)^2\right] & \text{if} \quad s_p \neq s_q \end{cases} \tag{2.5}$$

Here $\vec{d}$ is the boundary orientation estimated on the deformed geometry and $\vec{e}_{pq}$ is the edge orientation, signed accordingly to the boundary direction $\vec{d}$. To do that, a *start* and *end* vertex are chosen for each boundary, consistently with the axis orientation and the notion of left and right charts. The edge orientation is then setted for the outgoing edge vector from the starting vertex and propagated along the boundary chain until the very last edge, incident to the end vertex.

2.5 Positioning and parameterization

The second step of the algorithm, the geometric one, takes in input the valid all-monotone segmentation with refined boundaries generated by the topo-

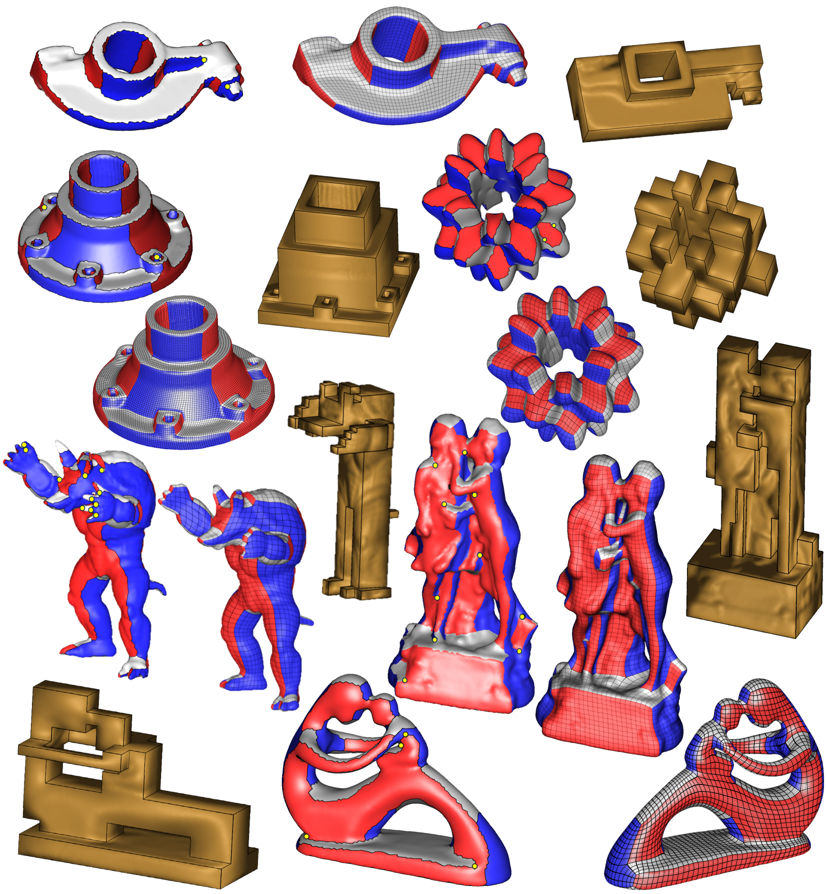


Figure 2.13: *A gallery of different PolyCube mappings and associated base domains. From top to bottom, left to right: rocker arm, carter, bumpy torus, armadillo, kiss statue and fertility.*

logical step, and produces both the actual orthogonal polyhedron and its mapping with the original model. To extract the PolyCube geometry from the segmentation, the deformation framework described in Section 2.4.3 is used, augmenting it with hard constraints to ensure an exact PolyCube output. Firstly, the iterative normal rotation process is applied as-is. Then, planarity constraints are added to Equation 2.4, minimizing, for every surface edge, the difference between its end-point coordinate values along the relevant axis. Gradual deformation with soft constraints is preferred during this stage, so as to minimize distortion. Once the process converges, the final solution is computed by adding hard planarity constraints forcing vertices in each chart to have the same coordinate value along the corresponding axis. The end result of this deformation is that each corner vertex on the PolyCube is now in its correct position. However, the positions computed for the rest of the vertices are not guaranteed to be on the PolyCube defined by these corners. To compute a low-distortion parameterization from the input mesh to the PolyCube requires parameterizing each chart into a fixed, possibly concave, planar polygon, a well known open problem in mesh processing [XCGL11]. To obtain a low distortion mapping, sidestepping this challenge, a bijective but possibly poor quality map from the input model to the PolyCube is firstly computed; then, the quality of the mapping is improved by operating in the opposite direction, i.e. from the PolyCube to the input model. To generate the initial map, PolyCut first maps each chart boundary to its corresponding PolyCube edge using arc-length parameterization. Then, it uses the method of [XCGL11] with mean-value coordinates to position the interior chart vertices. If a volumetric parameterization is required, the deformation framework can be reused, keeping surface vertex positions fixed and specifying surface rotations using a coordinate frame given by the new normal and one of the edges.

For applications such as seamless texturing or meshing, the corners of the PolyCube need to be placed at vertices of a fixed size grid. To perform such quantization, PolyCut first places corner vertices in the quantized locations. Then, it relocates the PolyCube surface vertices using mean-value coordinates [HF06] with respect to the corners of its corresponding PolyCube face. Finally the interior vertices are similarly relocated using the surface mesh as a cage for 3D mean-value coordinates [JSW05]. To compute a low distortion PolyCube to input map, PolyCut first remeshes the PolyCube using existing software (e.g. [Gra]) and uses the mapping just computed to project the PolyCube mesh to the input model. Then, PolyCut iteratively slides the projected vertices along the input surface in order to minimize the mapping distortion between the two meshes, measured using mean-value coordinates.

It is important to notice that once the PolyCube and the initial map are computed, there are multiple methods for improving the cross-parametrization. While the technique presented here is effective, it is included for completeness only. Other mapping techniques may accommodate lower-distortion parameterizations for certain inputs.

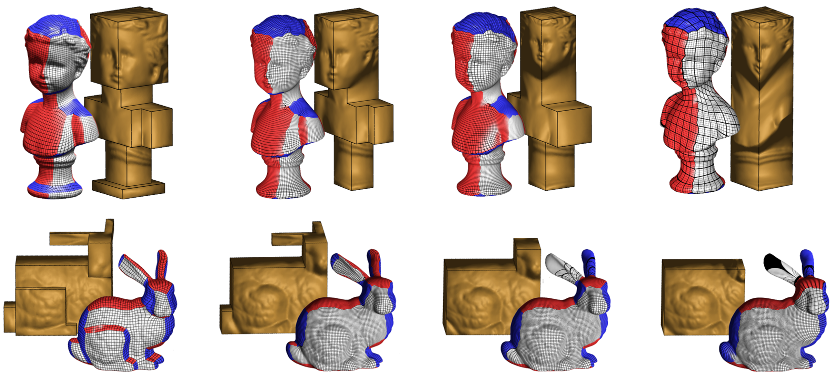
2.6 Results

PolyCut has been tested on a number of diverse inputs, including both natural and engineered shapes (Figure 2.13), and has been able to generate valid, all-monotone boundary segmentations and produce suitable PolyCube base complexes on all these inputs. Figure 2.6 and Figure 2.2 demonstrate the

Model	**Size**	**Turning Points**	**Corners/ Charts**	**Ang/Area Distortion**	**Stretch**
Bunny [THCM04]			67/34	1.069/1.034	0.913
Bunny [LJFW08]			34/19	1.120/1.150	0.790
Bunny [HWFQ09]			363/206	1.007/1.068	0.871
Bunny [GSZ11]			88/46	1.055/1.077	0.820
Bunny (PolyCut)	56K	10	64/34	1.033/1.033	0.908
Lion [HWFQ09]			285/151	1.028/1.100	0.804
Lion (PolyCut)	57K	17	74/39	1.073/1.059	0.831
Squirrel [HWFQ09]			40/24	1.001/1.084	0.854
Squirrel (PolyCut)	53K	3	16/10	1.035/1.042	0.890
Bimba [GSZ11]			115/61	1.100/1.106	0.712
Bimba (PolyCut)	46K	6	30/17	1.061/1.053	0.843
Girl [GSZ11]			88/48	1.064/1.112	0.747
Girl (PolyCut)	72K	16	60/34	1.043/1.062	0.850
Fertility [GSZ11]			96/43	1.074/1.100	0.772
Fertility (PolyCut)	37K	6	94/42	1.092/1.066	0.808
Kiss [GSZ11]			156/74	1.087/1.090	0.770
Kiss (PolyCut)	29K	19	120/56	1.047/1.049	0.871
Rocker arm [GSZ11]			70/38	1.082/1.066	0.819
Rocker arm (PolyCut)	27K	6	62/34	1.066/1.051	0.857
Carter [GSZ11]			153/82	1.040/1.051	0.868
Carter (PolyCut)	69K	11	132/71	1.073/1.029	0.870
Bumpy torus [GSZ11]			208/104	1.093/1.069	0.793
Bumpy torus (PolyCut)	40K	4	172/86	1.041/1.053	0.865
Armadillo [THCM04]			80/42	1.318/1.224	0.577
Armadillo (PolyCut)	56K	26	140/72	1.105/1.130	0.714
Kitten (PolyCut)	37K	7	32/16	1.099/1.053	0.820

Table 2.1: *PolyCube statistics, including a comparison to earlier methods. Left to right: triangle count, number of turning points in the initial segmentation (only for PolyCut), number of singularities and number of charts, angular and area distortion [THCM04], stretch [PH03]. For all metrics the optimal value is one. As can be noticed, PolyCut consistently introduces a smaller number of singularities while producing parameterization with better stretch and area distortion.*

power of the hill-climbing optimization framework, highlighting both the prevalence and the variety of non-monotone boundaries in unconstrained solutions, and their appropriate resolution by the iterative process. The statistics for the models are summarized in Table 2.1. Unless otherwise specified, all the results were produced using the default parameters described in the text. For the bimba and lion (Figure 2.16) the compactness factor c was 4, and for the armadillo (Figure 2.13) 4.5; for the rest, the factor was set to 3. For all the models, except the kiss statue (Figure 2.11), the fast communication strategy described in Section 2.4.2 has been used. The runtime using this method varies between one minute for largely axis-aligned models like the bumpy torus (Figure 2.13), which contain few non-monotone boundaries in the initial labelling, to ten minutes for models such as the bunny which have many off-axis features (Figure 2.5). The kiss model, which was generated with the restartable hill-climbing strategy, took 50 minutes to compute. Times were measured on a MacBook Air (1.7Ghz Intel Core i5, 4GB RAM). Running times are comparable with those of recent PolyCube segmentation [GSZ11] and hex meshing [LL10] methods.



(a) $c = 3$ **(b)** $c = 6$ **(c)** $c = 10$ **(d)** $c = 17$

Model	Compactness	Corners/ Charts	Ang/Area Distortion	Stretch
Girl	3	60/34	1.044/1.062	0.850
	6	32/19	1.074/1.176	0.691
	10	24/14	1.068/1.305	0.583
	17	8/6	1.170/1.131	0.652
Bunny	3	60/34	1.033/1.033	0.908
	6	34/19	1.068/1.104	0.750
	10	16/10	1.062/1.184	0.683
	17	8/6	1.103/1.197	0.657

Figure 2.14: *Impact of adjusting the compactness factor in the labelling system. Higher compactness values will accommodate coarser PolyCube base domains with less singularities, though they will require higher distortion mappings. In the table, from left to right: compactness value, number of charts and singularities, angle and area distortion [THCM04], and stretch [PH03].*

2.6.1 Compactness control

One important advantage of PolyCut, compared to methods such as [LJFW08, HWFQ09, GSZ11, WYZ*11], is the flexible control of the trade-off between fidelity and compactness. Figure 2.14 shows the results of varying the compactness factor c (Equation 2.1) for the generation of PolyCubes of the

Stanford bunny and the girl statue. As expected, increasing this factor gradually reduces the number of PolyCube faces at the expense of increased stretch/area distortion. Providing this control to the user is important for many applications where distortion is weighed against singularity count, such as hex meshing or atlas generation.

2.6.2 Limitations and future works

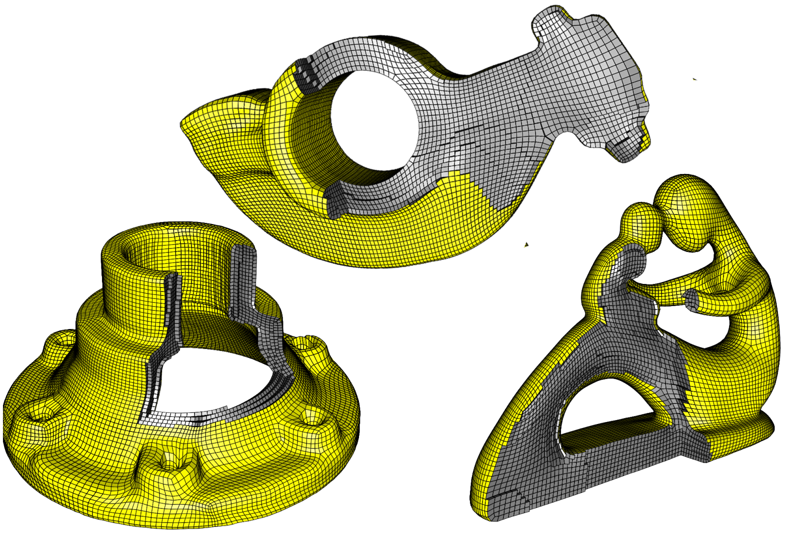
Like in previous PolyCube computation methods, outputs depend on the initial orientation of the input model and it would be worthwhile to explore computing optimal orientations automatically, e.g. via PCA analysis of surface normals. Following [EM10] every corner in the segmentation is required to have valence three, a sufficient but not necessary requirement for the existence of a valid orthogonal polyhedron, which sometimes leads to formation of redundant charts. Unfortunately, no weaker sufficient requirement exists yet. Lastly, the local search method used in PolyCut is an approximation method, and as such is not guaranteed to converge to a valid solution or find the best possible PolyCube segmentation for a given mesh (as represented by an exhaustive search).

Finally, currently it is up to the users to select a compactness factor that fits their needs via trial and error. Directly relating this factor to mapping distortion is a challenging open problem. Future work also involves the integration of other global search frameworks into the PolyCut algorithm. Better PolyCubes may be generated via stochastic methods or by adding random restarts to hill climbing.

2.7 Comparisons

In this section PolyCubes generated by PolyCut [LVS*13] will be compared to those created via existing alternatives. The bunny PolyCube computed with the compactness factor set to 6 (Figure 2.14) is quite similar to the output of [LJFW08], and has comparable distortion (Table 2.1). However, in contrast to PolyCut, neither [LJFW08] nor [WYZ*11] support other PolyCube resolutions.

When compared to recent techniques [HWFQ09, GSZ11], PolyCut is able to generate PolyCubes with significantly fewer singularities and charts, while achieving comparable or better stretch and area distortion (Figure 2.16, Table 2.1). For Gregson *et al.* , distortion is measured on their output quad meshes, as directly measuring distortion on their PolyCubes produces infinite values due to collapsed or flipped triangles. The reduction in singularity numbers is particularly significant in models with many off-axis features,



Model	[GSZ11] Scaled Jacobian min/avg	PolyCut Scaled Jacobian min/avg	PolyCut Minimum Scaled Jacobian improve factor
Fertility	.196/.911	.259/.872	30%
Bunny	.138/.930	.274/.938	100%
Rocker arm	.226/.899	.370/.890	64%
Girl	.235/.925	.401/.926	71%
Carter	.177/.823	.250/.894	41%

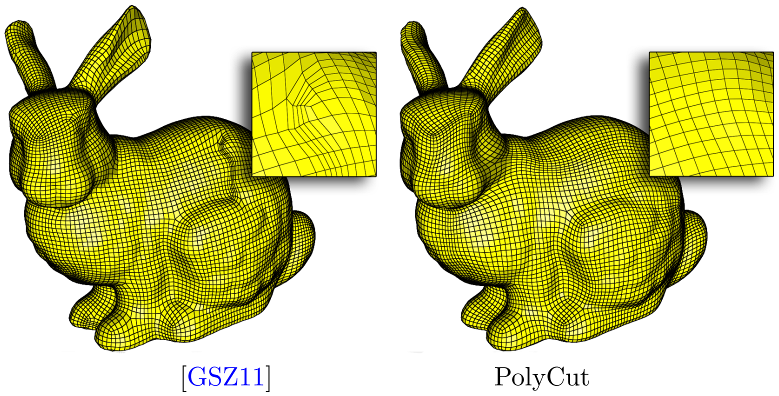


Figure 2.15: *Hex-meshing is a typical application for PolyCube mappings. Here both a quality (see table) and a visual (see bunnies on the bottom line) comparison with Gregson et al is provided. In the table, from left to right: minimum and average Scaled Jacobian for the two methods, improvement factor for Minimum Scaled Jacobian. PolyCut generates better PolyCubes that dramatically increase the worst element quality, a critical metric for analysis accuracy.*

such as the bimba (Figure 2.16) where PolyCut reduces the number of singularities by a factor of five (from 115 to 30, Table 2.1) compared to [GSZ11] while simultaneously reducing distortion, or the lion (Figure 2.16) where PolyCut reduces the singularity count by a factor of four (from 285 to 74, Table 2.1) compared to [HWFQ09] while improving stretch and area distortion (though there is a marginally higher angular distortion). Overall, when compared to these methods, an average reduction of around 30% in the number of singularities is present. In contrast to Gregson *et al.* , PolyCut resolves all non-monotone boundaries, while [GSZ11] may leave those in place (Section 2.2). It is opinion of the authors that the major factor in the observed improvement is the use of a principled local search to eliminate

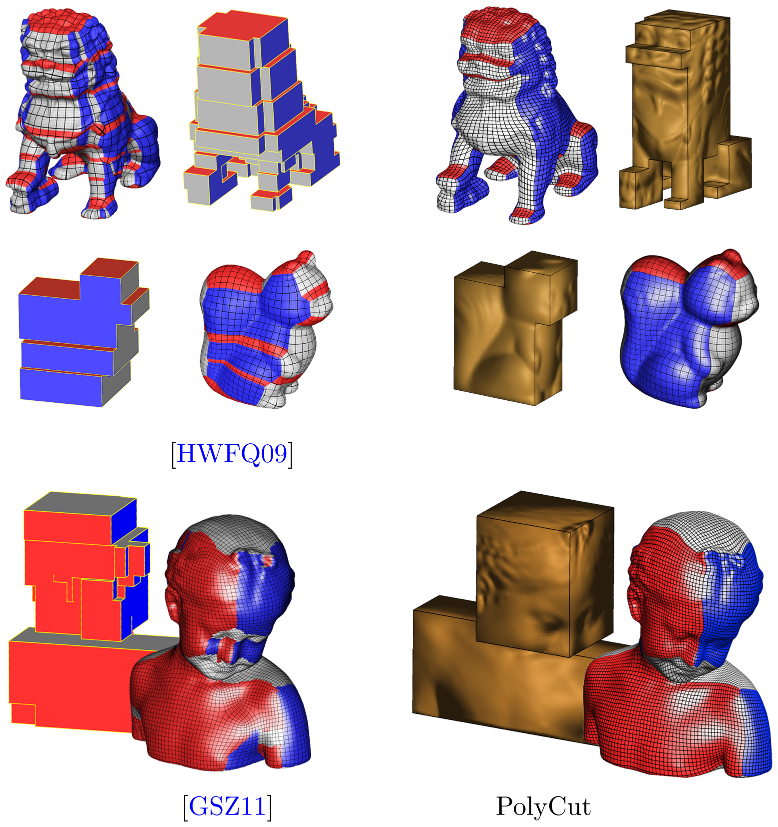


Figure 2.16: *Visual comparisons with the divide-and-conquer algorithm (top line, left) and the deformation-based algorithm (bottom line, left). PolyCut dramatically reduces the number of singularities (i.e. corners) in the base-domain and at the same time provides better quality mappings, having comparable, often lower, distortion.*

turning points, as opposed to the local heuristic used by Gregson *et al.* , which attempts to fix turning points individually, choosing from three templated solutions.

Table 2.1 also provides a comparison with the manually generated PolyCubes of Tarini *et al.* [THCM04]. On the bunny model, PolyCut produces a PolyCube with nearly identical complexity and comparable distortion statistics. On the armadillo model, PolyCut produces a much finer PolyCube, one which as expected leads to less distortion. Producing such a high detail PolyCube manually would be a challenging task.

Lastly, to showcase the importance of using better PolyCubes from an application perspective, PolyCut has been tested in the context of hex meshing, and results compared with the method of Gregson *et al.* (Figure 2.15). The improvement on the bunny highlights the importance of correctly resolving non-monotone boundaries. While the heuristic approach of Gregson *et al.* produces an unnatural vertical chart on the bunny side leading to visible mesh artefacts, PolyCut resolves the thighs in a more natural manner. As pointed out by Gregson *et al.* , the critical number to look at is the minimal Jacobian (i.e. the measure of how much the worst element of the whole mesh deviates from a perfect cube), which is significantly better for all the considered models.

Part II

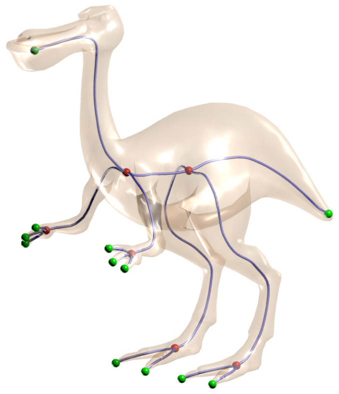
Curve-Skeletons

Chapter 3

Background

Digital objects are flooding our environments: whether they are reproductions of objects acquired from reality or synthetic models produced by artists and designers they can be found in a number of different scenarios that span from medicine and architecture to animation and entertainment. While for some of these fields an accurate (and redundant) description of the physicality of the objects is mandatory, for some others a compact and concise representation is needed. Reducing the amount of data necessary to represent a shape is useful for compressing, transmitting and processing objects (Section 3.2.3). Moreover, such representations often provide the user with a compact abstraction of the shape, that can be used to improve the interaction with algorithms that require some sort of human intervention, like character animation or shape editing.

Skeletons are among the most popular and effective ways to represent a shape in a compact fashion. A skeleton can be thought of as a set of monodimensional curves arranged together in a graph-like structure, representing the main components of a shape as well as the way they connect each other (see Figure aside). The process of computing the skeleton of a shape can be really thought of as the detection of its symmetries. Indeed, every skeleton point is the centre of an infinitesimal symmetry relationship between two (or more) parts of the shape. The skeletonization problem firstly arose with images (Sec-



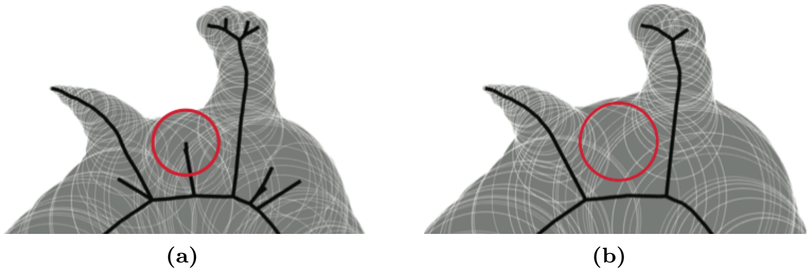


Figure 3.1: *A visual comparison between MAT (left) and SAT (right). By applying uniform scaling SAT gradually ignores small-scale features as their maximals disks undergo a smaller growth and are included into bigger disks representing higher-scale features of the shape. The red disk, which is maximal in (a) is no longer maximal in (b) (image courtesy of [GMPW09]).*

tion 3.1), then, the theory developed has been exploited to solve the problem in the 3D embedding (Section 3.2). However, while in $\mathbb{R}^2$ skeletons have a strong and mathematically sound characterization, in $\mathbb{R}^3$ their formal definition is still an open problem [Tag13]. What is known, is that there is a number of properties that skeletons should satisfy (Section 3.2.1), mostly related with the particular task for which a skeleton will be used.

3.1 2D skeletons

The original idea of skeletons was introduced in 1967 by H.Blum to extract and represent the salient features of a planar shape. His Medial Axis Transform (MAT) [Blu67] (Section 3.1.1) gave rise to a conspicuous amount of literature covering the topic of both computing and dealing with skeletal structures. Of particular relevance respect to the topics of this thesis is the Scale Axis Transform (SAT) [GMPW09] (Section 3.1.2), which alleviates the well known sensitivity of MAT respect to small perturbations along the boundary. Another important contribution was given by Brady and Asada, that in 1984 defined the Smoothed Local Symmetry (SLS) [BA84] (Section 3.1.3). The usefulness of SLS will become evident in Section 4.2.1, where it will be used as a building block for the interpretation of partially occluded silhouettes. A comparison between MAT, SAT and SLS will be also provided.

3.1.1 Medial Axis Transform

Let $\mathcal{O}$ be a planar object (domain) with boundary $\partial\mathcal{O}$. Let $D_r(p)$ be the disk with radius r centred at point p. The medial axis transform of $\mathcal{O}$ is

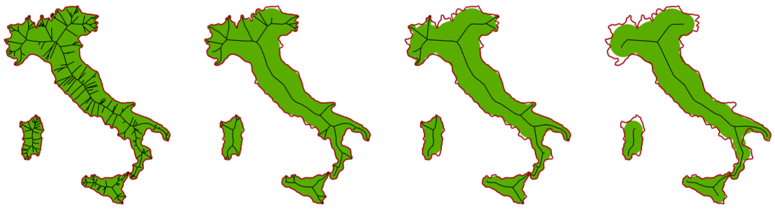


Figure 3.2: *Skeletons of the map of Italy computed using SAT at growing scales (left to right). The red line denotes the boundary of the original domain. Green is the area reconstructed from the scale axis transform. Greater scales progressively ignore small features and catch only the essential parts of the shape (image courtesy of [GMPW09]).*

defined as

$$\mathrm{MAT}(\mathcal{O}) = \left\{ (p, r) \in \left(\mathcal{O} \times \mathbb{R}^+ \cup \{0\} \right) \,\middle|\, D_r(p) \text{ is a maximal disk } \right\} \tag{3.1}$$

where a disk $D_r(p) \subset \mathcal{O}$ is called maximal if, for any other disk $D_s(q)$, $D_r(p) \subset D_s(q)$ implies $D_r(p) = D_s(q)$. Note that disks with radius equal to zero are legal; in particular they appear when the domain contains *sharp* features [CCM97]. Each maximal disk touches the boundary $\partial\mathcal{O}$ in two or more separate points which are equidistant from its centre. It can be proven that at these points the maximal disk is tangent to $\partial\mathcal{O}$. The locus of bi-tangent disks is called Symmetry Set (SS) [BGG85], and the medial axis is always included in it. This really points up the relation between skeletons and symmetry detection.

For some applications radii can be useful as they encode information about the *local thickness* of the shape. Moreover, the radius is what really makes the medial axis a *transform*, as by superimposing all the maximal disks one can invert it and recreate the original domain $\mathcal{O}$. Cutting out the radii the medial axis ceases to be a transform and keeps only the graph structure that encodes both topology and medial paths of the shape (i.e. its skeleton).

3.1.2 Scale Axis Transform

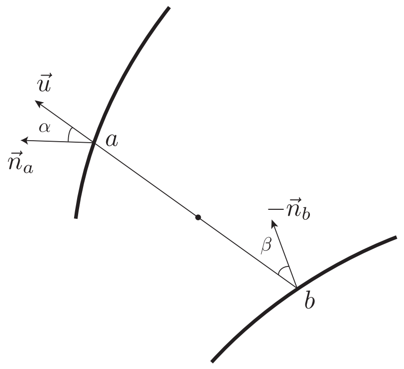
One of the most known (and tedious) disadvantages of the medial axis is its inherent instability respect to small perturbations. Bumps along the boundary lead to dramatic changes in the graph structure so that two similar shapes may have very different skeletons. Classic approaches (e.g. [CL05]) try to fix this drawback in a local fashion, by pruning skeletal branches according to a quantity (e.g.: angle, area, contact points distance) measured at a medial axis point and at its closest neighborhood. However, local methods are able to filter out noise from the skeleton but are not designed to simplify the medial axis based on a comparison between features in a scale-adaptive way. A powerful and easy to implement alternative is the Scale Axis Transform (SAT) [GMPW09] (Figure 3.1), defined as

$$\mathrm{SAT}_s(\mathcal{O}) = \left\{(p, r/s) \in \left(\mathcal{O} \times \mathbb{R}^+ \cup \{0\}\right) \;\middle|\; (p,r) \in \mathrm{MAT}(\mathcal{O}_s)\right\} \tag{3.2}$$

where $\mathcal{O}_s = \bigcup_{(p,r)\in \mathrm{MAT}(\mathcal{O})} D_{sr}(p)$ is the union of the maximal disks of $\mathcal{O}$ scaled with a factor $s \geq 1$. For $s = 1$ the scale axis equals the medial axis. For growing values of s, the scale axis gradually ignores less important (i.e. small) features as their maximals disks undergo a smaller growth and are included into bigger disks representing more important (i.e. bigger) features of the shape (Figure 3.2).

3.1.3 Smoothed Local Symmetry

As pointed up in Section 3.1.1 the medial axis is a subset of the symmetry set of a shape, meaning that it is able to capture only a fraction of the amount of symmetry contained within a shape. The Smoothed Local Symmetry (SLS) [BA84] introduced by Brady and Asada in 1984 can be seen as an explicit way of computing the symmetry set. Given two points a, b on the boundary $\partial\mathcal{O}$, and a unit vector $\vec{u}$ with direction $\overline{ab}$, the midpoint of the segment $\overline{ab}$ belongs to SLS($\mathcal{O}$) if and only if the angle α between $\vec{u}$ and the outward normal at a is equal to that β between $\vec{u}$ and the inward normal at b. Smoothed local symmetry is a very powerful shape descriptor as it is able to catch *all* the local symmetries of a shape. This difference is highlighted in Figure 3.3, where a comparison with the Medial Axis Transform (MAT) is provided.



3.2 3D skeletons

As for the 2D setting, the skeleton of a three-dimensional shape (or curve-skeleton) is a set of mono-dimensional curves arranged together in a graph-like structure. Unfortunately, the medial representation of an object $\mathcal{O} \in \mathbb{R}^n$ will have dimensionality up to $n-1$, meaning that in the 3D embedding the representations discussed in Section 3.1 will be a collection of both curves and sheets. The fact that curve-skeletons cannot simply be defined as the

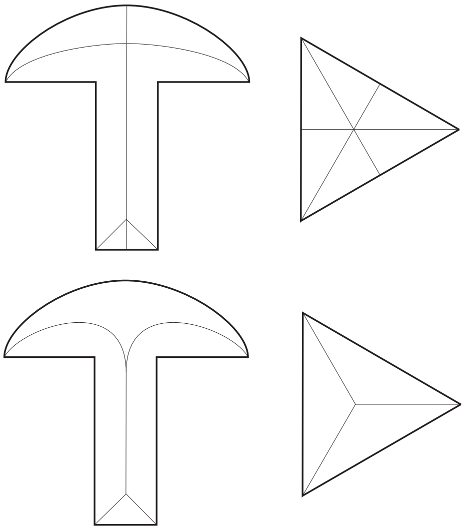


Figure 3.3: *A visual comparison between SLS (top) and MAT (bottom). Strictly following the definition of symmetry set the smoothed local symmetry is able to detect more symmetry than the medial axis.*

medial paths of a 3D shape gave rise to a multitude of different approaches that do not necessarily follow a common theoretical foundation. As a consequence, comparing different methods has become a difficult task. However, there is a number of properties that curve-skeletons should observe (Section 3.2.1). Section 3.2.2 will go through the most prominent algorithms for curve skeleton extraction while Section 3.2.3 will list some of their applications.

3.2.1 Properties

With the goal of providing a common evaluation framework, in [CSM07] Cornea and colleagues listed the following set of properties for curve-skeletons to be observed

Thinness The first mandatory requirement, for a skeleton, is to be composed of mono-dimensional curves. This is important for applications, since storing and processing a graph is easier than processing more complex structures like the medial axis.

Centricity Curves are also required to be centered within the shape, hence they should lie on its medial surface. While for tubular shapes this is clear enough, it is still unclear what being central respect to a more general shape (e.g. a door) means. A possible interpretation is the following: skeleton paths should lie on the medial surface of the object and be centered with respect to the medial surface patch they belong to. However, due to the well known sensitivity of the medial axis to small perturbations exact centeredness may not be required or desired.

Homotopy A shape and its curve skeleton should be homotopic, that is, the skeleton should contain one loop for each handle in the original domain. Note that the medial axis transform can be proven to preserve topology [CCM97]; as a consequence, skeletons constructed on top of it are likely to be homotopic respect to the shapes they represent.

Hierarchy Skeletons computed at different resolutions will capture features of the shape at different scales an may be used to think of a shape in a hierarchical fashion.

Reconstruction Refers to the ability of reproducing the original domain. As opposed to the medial axis, the curve skeleton usually is not a transform, meaning that it cannot be inverted. However, by associating a primitive (e.g.: a sphere, an ellipsoid or a cross-section) to every skeleton point some methods can reproduce an approximation of the original shape.

Robustness Noise along the surface should not generate spurious branches in the skeleton so that the skeleton of a noise-free object and the skeleton of the same object with noise should be similar.

Reliability A skeleton should be really representative of the shape it describes. This idea can be expressed in a geometric fashion by imposing that every point in the boundary of the domain should be visible from at least one skeleton point.

Invariance under rigid motion The computation of a skeleton should be orientation independent. This means that, given a rigid transformation T and an object $\mathcal{O}$, the skeleton of $\mathcal{O}$ should be equivalent to the skeleton of $T(\mathcal{O})$.

Component-wise differentiation As one of the goals of curve -skeletons is to detect the main components of a shape and the way they interconnect with each other, a skeleton should be able to encode this information by creating a one-to-one correspondence with the logical components of the domain.

Smoothness The curves of the skeleton should be smooth. This is not only an aesthetic requirement, smoothness can be useful when the skeleton is describing a path or trajectory, for example in the context of virtual navigation or virtual endoscopy.

It is worth noticing that most of these properties are not meant to be satisfied in every application scenario. Invariance under rigid motion is important in the context of shape recognition, but it might not be helpful for other tasks like character animation or virtual navigation. Moreover, some of these properties are really conflicting with each other (e.g.: smoothness and centeredness, component-wise differentiation and reliability, thinness and reconstruction).

3.2.2 Previous work

Previous works on curve-skeleton extraction consist of a large number of methods and approaches that look at the same problem from different points of view. As different fields may represent 3D shapes in different ways, in this section skeletonization algorithms will be grouped according to the particular definition of shape they agree with.

Voxel-based methods

Volumetric algorithms work on the discrete space $\mathbb{Z}^3$. Each point v is called *voxel* (volume element) and is defined by its three integer coordinates $v_x, v_y, v_z \in \mathbb{Z}$. A voxel can be viewed as a cube, having 6 faces, 12 edges and 8 corners. Two voxels p and q are said to be 6-adjacent if they share a face; 18-adjacent if they share a face or an edge; 26-adjacent if they share a face, edge or corner. The set of 6-adjacent voxels to a given voxel v is also known as 6-neighbourhood and is denoted by $N_6(v)$. Similarly, $N_{12}(v)$ and $N_{26}(v)$ are respectively the 12-neighbourhood and the 26-neighbourhood of the voxel v. Discrete algorithms attempt to produce a curve skeleton by iteratively removing voxels from the boundary of a volume. This operation is called *thinning*. Most of the thinning algorithms rely on the concept of *simple point*, introduced by Morgenthaler in [Mor81]. A simple point is a voxel that can be removed from the volume without affecting the topology of the object (i.e. either the number of connected components or the genus). An important property of the simple points is that they can be locally characterized, that is, one can determine whether a voxel v is simple or not just by inspecting its neighbourhood $N(v)$.

Volumetric algorithms differ each other in the way they detect simple points. In [MS96] Ma and Sonka proposed a fully parallel 3D thinning algorithm, mainly applied in medical image processing and 3D reconstruction.

The algorithm is based on some predefined templates; each time the N_{26} neighbourhood of a voxel matches one of these templates, it is removed. In [WB07] Wang and Basu proved that Ma and Sonka's algorithm sometimes fails to preserve connectivity, and proposed a new set of templates to fix the problem. In [ZLLJ08] Zhang *et al.* proposed a new, hierarchical approach, to solve the thinning problem. In their algorithm the original volume is firstly decomposition into simple tube-like volumes. Then, the thinning is applied to each sub-volume to extract the segments that form the final skeleton.

Other volumetric methods are based on the discretization of a continue function. These algorithms construct a scalar or vector field in the discrete space and define the skeleton as the union of the singularities of such field. The most used function is the *distance transform*. In [Bor96] Borgefors proved that the best discrete approximation of the Euclidean distance is the $\langle 3, 4, 5 \rangle$ weighted distance; its value is respectively 3 between neighbours sharing a face, 4 between neighbours sharing an edge, and 5 between neighbours sharing a corner. Borgefors also provided a fast two-pass algorithm for its computation. Hassouna and Farag [HF05] and Gagvani and Silver [GS99] proposed two interesting examples of skeletonization algorithms which rely on the distance transform discretization.

Distance transform is not the only function used to extract centerlines from discrete volumes. For example, in [CSYB05] Cornea *et al.* employed the Newtonian potential model, creating a force vector field inside the volume by charging each boundary voxel. The repulsive force at a non-boundary point due to a given charge is defined as the force pushing away the point from the charge with a strength which is inverse proportional to the squared distance between the point and the charge. The total force at the same point due to multiple charges is then computed simply by summing up all the forces acting on it.

Finally, in [LCLJ10], the thinning process is guided by a new skeleton significance measure, called *medial persistence*, that is, the duration in which a cell complex remain isolated during the (iterative) thinning process. Medial persistence proved to be more robust and fast to compute respect to the other discrete measures, producing great results both in terms of skeleton quality and running times.

Mesh-based methods

Triangle meshes are probably the most widely used data structures for storing and representing three-dimensional shapes, employed in different applicative environments. As a consequence, mesh-based algorithms are a

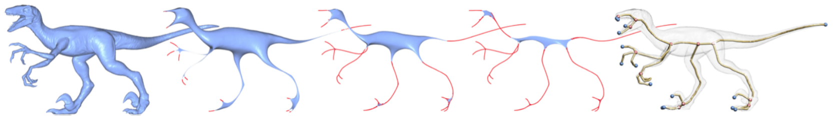


Figure 3.4: *Contraction-based algorithms iteratively shrink the shape until the skeleton paths are revealed. Throughout the process vertices travel towards the interior of the shape with a speed proportional to the local mean curvature of the surface (image courtesy of [ATC*08]).*

highly heterogeneous family.

Some algorithms are based on the reduction of the medial axis, which is not strictly mono-dimensional for 3D domains. It is known from the work of Gibli and Kimia [GK04] that the medial axis of an object $\mathcal{O} \in \mathbb{R}^3$ can contain five types of points: one type forms two dimensional sheets, two types form curves and the remaining two types are isolated points. In [DS06] Dey and Sun, aiming to formally define curve-skeletons, focused their attention to the first category. They observed that the maximal inscribed balls of such points touch the boundary $\partial\mathcal{O}$ at exactly two distinct points. Then, they defined a new function, called *Medial Geodesic Function* (MGF), as the length of the shortest path between the contact points of such balls. The gradient of the medial geodesic function, ∇MGF, defines a vector field whose divergence is negative at the singular points and 0 everywhere else. Therefore, the curve-skeleton of $\mathcal{O}$ consists of the points where the gradient flow of MGF skins into. The work of Dey and Sun is very important from the mathematical point of view because, for the first time, curve-skeletons have been defined in a formal way. However, the application of this definition to real problems is non-trivial as the computation of the MGF may be even one order of magnitude slower than the state of the art counterparts.

Other algorithms extract skeletons by progressively shrinking shapes (Figure 3.4). They are usually referred to as *contraction-based methods* and rely on the concept of *mean curvature flow*. Intuitively, a shape evolves under mean curvature flow if points on its surface move along the opposite of their normal direction with a speed proportional to the mean curvature of the surface. For example a round sphere, which has constant mean curvature everywhere, evolves under mean curvature flow by shrinking inward uniformly. In order not to lose important features or collapse to a single point due to excessive shrinkage, positional constraints are usually added to enforce attraction with respect to the original surface. Both shrinkage and features-preserving constraints can be encoded by a linear system of equations and solved in a very efficient way. Moreover, algorithms based on this paradigm are proven to be very robust against noise as the mean curvature flow is also a natural smoothing operator [DMSB99]. The first contraction-based algorithm was proposed by Au and colleagues in [ATC*08]. However, due to some discretization inaccuracies of the Laplacian smoothing they employed, curve-skeletons tended not to be well-centred with respect to the shape. In [TAOZ12] Tagliasacchi *et al.* addressed this limitation and reformulated the problem in order to improve centricity. In [SYJT13] an interesting comparison between contraction-based algorithms is provided.

Other important works in the field are [SLSK07], where a deformable model is grown into the object to detect the skeleton branches, and [LKA06] where the mesh is iteratively decomposed into hierarchical segments, computing a centerline compression error until a threshold is reached.

Point-based methods

A point cloud is probably the simplest representation of a 3D shape that one can think of, and is also the typical data produced by range scanners as well as other devices used to acquire the geometry of real objects. They can be either *oriented* or *unoriented*. While the former encode both position and orientation of the surface they represent (i.e. every point is associated with a normal vector tangent to the surface), the latter encode only positional information.

In [CTO*10] with the aim of extending contraction-based methods to other shape representations, Cao and colleagues defined a method which is potentially able to operate over any type of input data. Though they validated their algorithm with a number of unoriented point clouds. Since point clouds do not encode connectivity, for the computation of the discrete Laplace operator they employed local Delaunay triangulations. In [TZCO09] Tagliasacchi and colleagues proposed a method for the skeletonization of oriented point clouds. Authors firstly assumed that the input shape is mostly cylindrical, then, they observed that the medial path of a cylinder is the axis of a rotational symmetry (ROSA). They firstly computed symmetry axes for tube-like parts, then merged them up at joints with a mono-dimensional moving least squares algorithm. Huang *et al.* [HWCO*13] outperformed ROSA by proposing a new algorithm based on the L_1 median, which requires neither points orientation nor the cylindrical shape prior. Moreover, being based on a statistical measure, this method promises to be more robust against noisy point clouds and outliers.

3.2.3 Applications

Curve-skeletons are ubiquitous in a number of applications. Here is a brief list of some of the most important fields in which skeletons are employed.

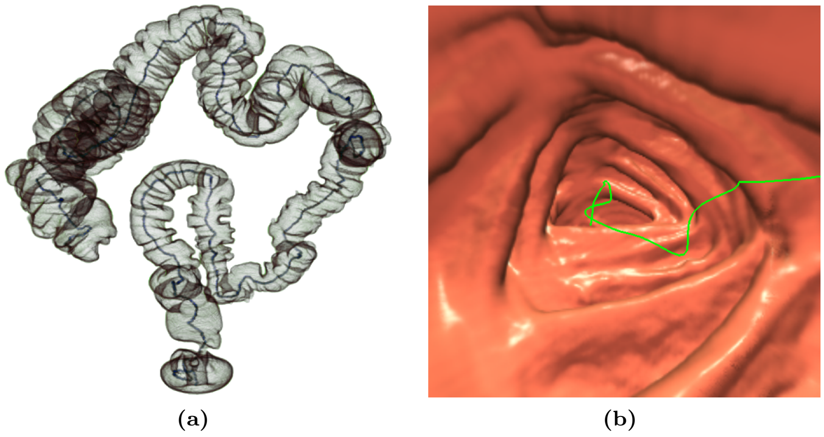


Figure 3.5: *An example of virtual colonoscopy. The centerline extracted from a colon (a) and a frame extrapolated from the virtual navigation framework (b) (image courtesy of [WDK01]).*

Medicine

Skeletons are widely used in medical analysis to describe, navigate or register human organs. Indeed, some of our organs (e.g.: vessels, nerves, colon) have a tube-like shape that can be described by a curve-skeleton in a very effective way [FPAB04]. In [BLP97] and [YCS00] the authors provide some useful algorithms for the centerline extraction out of human organs. In [AJWB03] Aylward *et al.* used skeletons to register (i.e. align) partially overlapped vascular images. This can be useful for medicians as allow them to gather data acquired with different modalities (or at different times) in order to improve the accuracy of the diagnosis or better track the evolution of a disease (e.g.: tumor margins). In [WDK01] Wan *et al.* proposed a virtual colonoscopy framework (Figure 3.5). Their system provides a flexible real-time navigation inside a virtual 3D colon acquired from a continuous sequence of 2D CT slices and aims at detecting early-stage colon polyps.

Geometric processing

Skeletons are also used in a number of problems related with geometric processing. Their reconstruction capabilities are often used to correct geometry acquired with laser scanners and other devices [CTO*10,TZCO09] or to improve the alignment of an animated sequence [ZST*10]. Their component-wise differentiation is also very useful for comparing shapes and find appropriate correspondences [KCATCO*10].

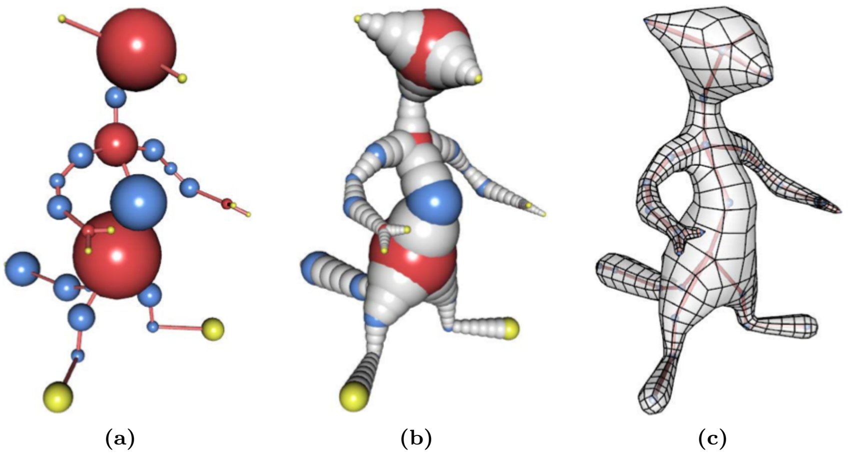


Figure 3.6: *The typical skeleton-assisted modelling pipeline. The curve-skeleton of a virtual character is sketched by the user (a); additional information about the local thickness of the shape is appended to each skeleton point (b); a polygon mesh describing the surface of the character is finally computed (c) (image courtesy of [JLW10]).*

Modelling

The ability of skeletons to summarize the main features of a shape is very often used in modelling to craft virtual characters that can be further refined by adding small-scale details (Figure 3.6). Most of these approaches provide the user with a visual interface to sketch a skeleton and specify the thickness of its parts [JLW10, BMW12, ZJY13]. Finally, a polygon mesh representing the surface of the character is extracted.

Animation

Computer animation requires the editing of virtual characters in order to pose them in different ways at different times. This is typically done by using control cages [JMD*07] or using a control skeleton [BP07]. Cages provide a finer control of the surface of the character though they are more difficult to be used by animators. On the other hand, skeleton-based animation is very natural as it mimics the way our bones define the pose of our bodies. To obtain a desired pose a set of geometric transformations are applied to the skeleton nodes in a hierarchical fashion; these transformations are then mapped over the surface of the character. Every element of the surface is associated with a weighted combination of transformations. These weights are usually referred to as *skinning weights* as they somehow define the way the skin of the character follows its skeleton [LCF00]. Typically, a control skeleton (or kinematic skeleton) is piece-wise linear and can be obtained by sub-sampling a smooth skeleton computed with any of the algorithms discussed in Section 3.2.2. Defining the motion of a character is not the only application for skeletons in animation. For example, in [LWTH01] the authors used skeleton paths in the context of real-time collision-detection.

Chapter 4

Perception-aided skeletons

Curve-skeletons are used in many different fields. In medicine they describe the features of human organs, in animation the motion of virtual characters, in computer-aided design the structure of mechanical parts, and so on (Section 3.2.3). How much do these fields have in common in terms of representation of the data? Almost nothing, unfortunately. In medicine shapes are either discrete volumes coming out from a MRI scan or triangle meshes generated with some Marching Cubes approach like [LC87] or [MSS94]; in animation characters are mostly quad meshes crafted by digital artists; in computer-aided design objects are either NURBS or T-Splines. As a consequence, this disparity is being reflected by algorithms for skeleton computation. Some algorithms can be fed only with triangle meshes, some others only with point clouds, some others with voxel-based models, and so on (Section 3.2.2).

Restricting an algorithm to work only with a particular shape representation is a limitation itself. In addition, *resolution* and *quality* of the primitives may be problematic. Indeed, the number of primitives used to describe a shape has a huge impact on the quality of the output. In Figure 4.1 two skeletons computed over a coarse triangle mesh representing a hand are provided. The appearance of the hand looks fine though state of the art algorithms are not able to summarize the structure of the model because of the relatively small number of triangles used to describe it. Moreover, for approaches that employ tools coming from differential geometry (e.g. contraction-based algorithms [SYJT13]) there is another problem: the discrete counterpart of the continuous operators they use do not preserve all the mathematical properties, making computations unstable. For example, the ubiquitous cotangent weights used to compute the discrete Laplacian over triangle meshes are unstable for skinny triangles and are not always positive [WMKG07].

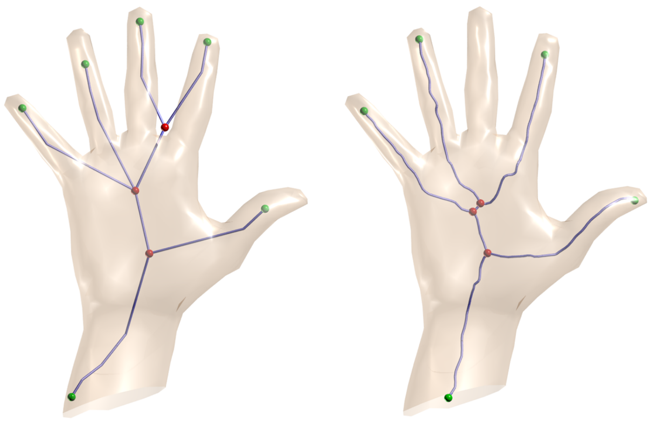


Figure 4.1: *Two examples of skeletonization of a coarse hand (1K triangles). When the number of primitives is low triangle-based algorithms are not able to perform well. The skeleton computed with [ATC*08] has one node outside the shape, while the skeleton computed with [DS06] is missing a finger (b).*

Summarizing, curve-skeletons are tightly coupled with the primitives used to represent a shape, may be different according to the number of primitives employed and, even when the number of primitives is the same, may be different according to the quality of the tessellation provided in input. But is this really necessary? Should algorithms be so much focused on the primitives? As stated in the introduction, computing a curve-skeleton is mostly a shape understanding process, therefore, it should have much more to do with the *idea* of the shape rather than with points, polygons and voxels. For some of the applications listed in Section 3.2.3 the stick figure of a living being that any child could sketch would be a perfect shape representation. So what's so special in a stick figure, that can be drawn so easily by a person without education but still requires complex math and heavy computation for a machine? The short answer is: nothing. As it will be shown in the remainder of this thesis computers can really mimic human brain capabilities in shape representation in order to focus more on ideas and less on primitives. The longer answer is still being written by researchers who operate in the context of cognitive sciences though parts of it can be found in the scientific literature that has been published for the last 50 years. This literature says that there is a deep connection between skeletons and how we store information about shapes in our brain [WB98, Mar10]. One of the goals of this thesis is to show how to exploit this relations in order to overcome some of the issues related with the computation of skeletons.

Name	Purpose	Primitives
Image(s)	Represents intensity	Intensity value at each point in the image
Primal sketch	Makes explicit important information about the 2D image; its intensity changes and their geometrical distribution and organization	Zero-crossing, blobs, terminations and discontinuities, edge segments, virtual lines, groups, curvilinear organization, boundaries
2½D sketch	Makes explicit the orientation and rough depth of the visible surfaces, contours and discontinuities in these quantities in a viewer centred coordinate frame	Local surface orientation, distance from viewer, discontinuities in depth and in surface orientation
3D model representation	Describe shapes and their spatial organization in an object-centred coordinate frame, using a modular hierarchical representation that can include volumetric primitives as well as surface primitives	3D models arranged hierarchically, each one based on a spatial configuration of a few sticks or axes, to which volumetric or surface shape primitives are attached

Table 4.1: *The representational framework for deriving shape information from images proposed by D.Marr in* [Mar80]

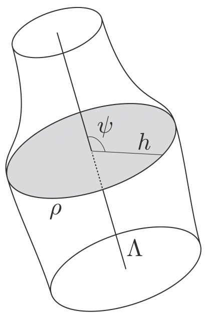
Section 4.1 will move the focus from primitives to shapes and will present an alternative definition of curve-skeleton. Section 4.2 will show how to detect information about the components of a 3D shape just by looking at its silhouettes.

4.1 An alternative idea of curve-skeleton

The skeletonization problem can be approached from another point of view, focusing more on appearance and less on primitives. The ideas expressed in the remainder of this chapter are largely based on the perception theory developed during the 1970s by people from M.I.T. Artificial Intelligence Laboratory. In particular, David Marr inspected the *early visual perception* system, in order to understand how the human brain behaves while looking at an image containing the projection of a real object. According to his theory, the way human beings interpret planar contours is organized as a sequence of representations carefully designed to facilitate the subsequent recovery of properties about an object's shape (Table 4.1). The very first level is obtained from the changes in the image, such as intensity changes, illumination effects, highlights and transparency. The result of this stage is a representation called *primal sketch* [Mar76]. Secondly, a number of processes operate on the primal sketch to derive a representation of the geometry of the visible surfaces. This second representation is called the *2½-D sketch*. Both of them are constructed in a viewer centred coordinate frame. The third and final representation carries informations about the three-dimensional shape, its structure and its spatial organization, and it is called the *3D model representation*. The early visual perception and the skeletonization problem are related in the sense that the final stage of the interpretation of a contour can be assimilated to the curve-skeleton of the shape that gave rise to it [Gug12].

The class of shapes considered by Marr in his analysis is the so called *generalized cones*. In its simplest form a generalized cone is *"the surface swept out by moving a cross-section of constant shape but smoothly varying size along an axis"* [Bin71], or, more formally

Definition 1 (Generalized cone) *Let $\rho(r,\theta)$ be a simple closed planar curve twice continuously differentiable, and let h be a twice continuously differentiable positive real function. Let Λ be a line at some angle ψ to the plane containing ρ, and denote positions along Λ with z. Then, the surface $\mathcal{GC} = h \times \rho$ is a* generalized cone *with axis Λ, cross-section ρ, scaling function h, and eccentricity ψ.*



As Marr pointed out not every object can be represented by a generalized cone although many objects, especially those whose shapes are achieved by growth (e.g. living beings), are described quite naturally by one or a union of them [Mar80]. Indeed, there is a strong link between generalized cones and curve-skeletons. Objects belonging to the class of generalized cones can be well described by curve-skeletons as well as shapes having a strong skeletal clue can be uniquely described in terms of generalized cones. On the contrary, objects whose natural axes are either too weak to describe them or external respect to the shape (e.g.: a mug, a door, or a crumpled newspaper) can be hardly represented by means of generalized cones as alternative axes can be considered, causing ambiguity. For the same reasons it is difficult to think of their curve skeleton. In [CSYB05] Cornea *et al.* suggested that the skeleton

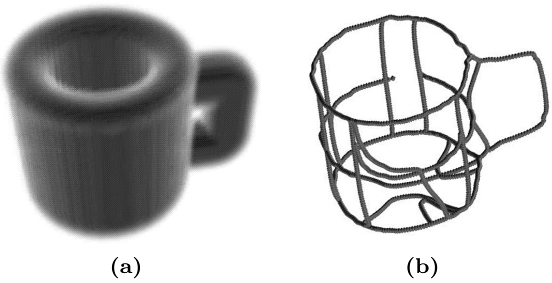


Figure 4.2: *When the object contains many poorly defined natural axes or the main symmetry axis is external respect to the shape it is very difficult to describe it with a curve-skeleton. In the picture a discrete mug (a) and one possible curve-skeleton (b) (image courtesy of [CSYB05]).*

of a shape having a cavity should contain at least one loop around it, but this would completely break down the topological connection between an object and its skeleton (Figure 4.2). To describe shapes containing tunnels or deep cavities any non mono-dimensional shape descriptor, like [MCM*12] or [MGP10], would be more suitable.

Given a real object $\mathcal{O} = \bigcup_{i=1}^{n} \mathcal{GC}_i(\Lambda_i, \rho_i, h_i, \psi_i)$ composed by n generalized cones, according to the ideas expressed in [MN78], in [LS13a] we defined the skeleton of $\mathcal{O}$ as the union of the axes of each generalized cone, that is

$$\mathsf{Skel}\,(\mathcal{O}) = \bigcup_{i=1}^{n} \Lambda_i \ . \tag{4.1}$$

The underlying idea is that the curve-skeleton of a 3D shape can be thought of as the final stage of the framework proposed by Marr, the so called *3D model representation*. This is the definition of curve-skeleton that will be used in the remainder of the thesis. The algorithms presented in Chapter 5 are meant to compute skeletons of this kind.

4.2 Inferring the axes of generalized cones

When humans look at a silhouette, they perceive it as a particular 3D shape even though such silhouette could, in theory, be generated by an infinite number of shapes. This can partly be explained with the familiarity of the observer with the depicted shape, but it is not enough as one can use the medium of a silhouette to convey a new shape. Moreover, even with considerable effort, it is really difficult to imagine the more bizarre shape that could have generated a particular silhouette.

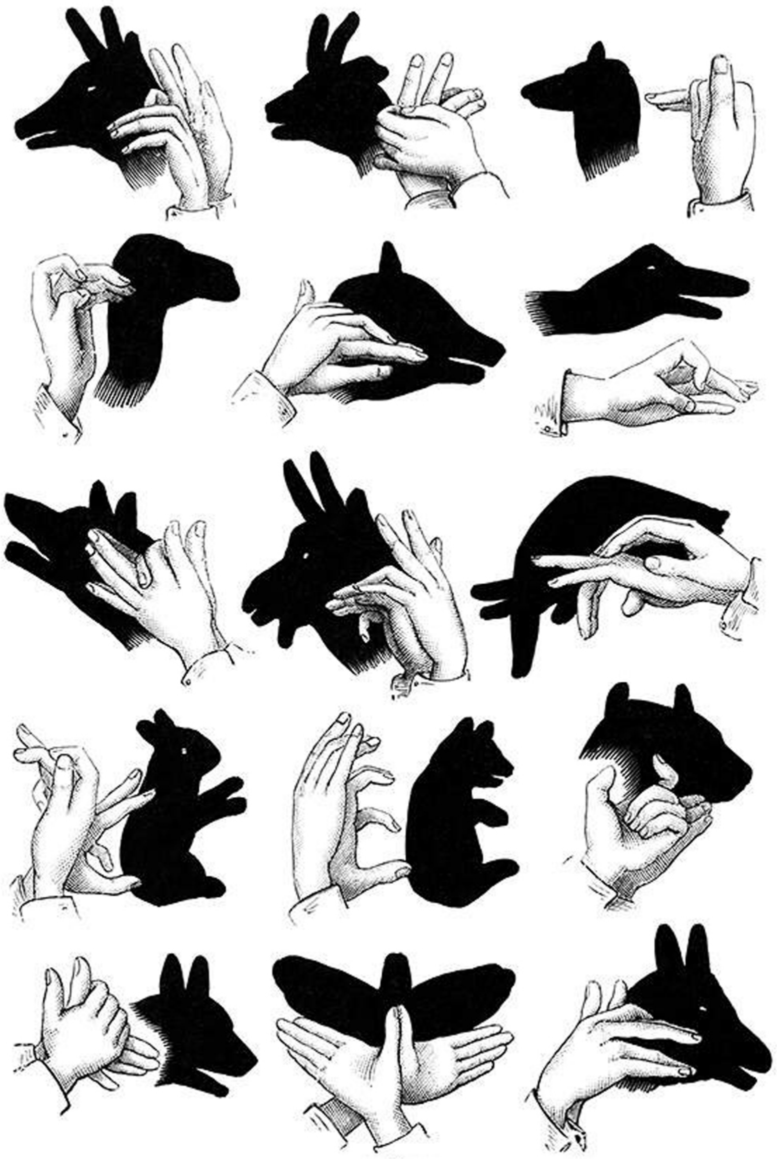


Figure 4.3: *When humans interpret a silhouette as a 3D shape they implicitly assume some a priori information about it. If these assumptions are violated the observer will perceive a different shape respect to the one that really generated the contour. Shadowgraphs are a typical example.*

When humans interpret a 2D contour as a 3D shape they implicitly assume some *a priori* information about it. If these assumptions are violated the analysis will be wrong, in the sense that the observer will perceive a different shape respect to the one the really generated the contour. This section will briefly go through the practical aspects of the theory of early visual perception, formalizing the assumptions that humans implicitly make when they observe a silhouette and showing how they can infer information about 3D shapes from planar contours. As the original approach was global, Section 4.2.1 will re-discuss the theory in a local fashion.

Why do we fail at interpreting shadowgraphs (Figure 4.3)? The reason is that the two main assumptions that we make to interpret a contour are both false. Our brain believes that contiguous portions of the contours arise from contiguous parts of the depicted shape and that the convexities and concavities of the silhouettes reflect real properties of the shape and are not due to projection artefacts.

Let $\mathcal{GC}\,(\Lambda, \rho, h, \psi)$ be a generalized cone and Ω its silhouette as seen from a viewpoint v, with π_v be the linear projection which defines the mapping $\pi_v : \mathcal{GC} \to \Omega$. The boundary $\partial\Omega$ is called *occluding contour* [Wal75] while the *contour generator* ($\mathcal{GC}_{\partial\Omega}$) is the set of points $p \in \mathcal{GC}$ that project onto $\partial\Omega$. According to the early visual perception theory, the implicit hypothesis made by a human observer for interpreting $\partial\Omega$ can be formalized as follows:

R1: each point on the contour generator projects to a different point on the contour, that is, $\mathcal{GC}$ is convex as seen from v or, in other words, the inverse $\pi_v^{-1} : \partial\Omega \to \mathcal{GC}_{\partial\Omega}$ is one-valued

R2: nearby points on the contour arise from nearby points on the contour generator, that is, the mapping $\pi_v : \mathcal{GC}_{\partial\Omega} \to \partial\Omega$ is continuous

R3: the contour generator is planar

In [Mar77] Marr proved that, if *R1-R3* are satisfied, and if the axis of symmetry of the projection $\pi_v\,(\mathcal{GC})$ is unique, such axis is the actual projection of the axis of symmetry Λ of $\mathcal{GC}$. This creates a fundamental link between the symmetries of a shape and the symmetries of its contours. Note that from a mere geometrical point of view a similar result has been achieved some years later by Rao and Medioni. In [RM88] they proved that *"the contour of a solid of revolution is symmetric about the projection of its axis for any view"*.

As Marr himself suggested: *"The importance of these relations is that one can use them to design algorithms for finding the generalized cone description of a contour, and for extracting any axes that may be present. By* *applying these algorithms repeatedly to the contours found in an image, one can often derive the 3D model representation of a surface's shape without prior knowledge of it"*. As the 3D model representation of a shape is the definition of skeleton that this thesis agrees with (Equation 4.1), silhouettes can be used to compute the curve-skeleton of a 3D shape. The are several reasons for that:

- moving the focus from primitives to perception all the common drawbacks of state of the art algorithms will disappear. For digital objects, the only requirement is that they can be rendered on a screen to produce a silhouette, regardless the type of primitives used to describe their shape. Moreover, as long as the *appearance* of the objects is preserved, contours will be consistent and both number and quality of the primitives will not affect the interpretation process
- working in the 2D embedding is less complex yet more efficient than working in the 3D embedding
- a huge amount of literature coming from the artificial intelligence, image processing and computer vision communities can be exploited

4.2.1 From the global to the local setting

The theory of early visual perception is inherently *global* as the constraints Marr put are supposed to be satisfied by the whole occluding contour. However, for relatively complex objects it may be difficult to find a point of view from which *R1-R3* are globally satisfied as occlusions may occur. This Section will discuss a modification of the original theory in which the same constraints must be observed only *locally*. This is part of the work regarding the analysis of partially occluded silhouettes we presented in [LS13a].

First, there is an interesting link between Marr's theory and the smoothed local symmetry (Section 3.1.3). If restrictions *R1-R3* are satisfied, then the following relation is true

$$\pi_v(\Lambda) \subseteq SLS(\Omega) \ . \tag{4.2}$$

This is straightforward to prove because, by construction, each symmetry point in Ω belongs to SLS(Ω). Therefore, if Λ projects to the axis of symmetry of Ω, it must belong to SLS(Ω) too. It is important to notice that, if the axis of symmetry of Ω is unique, then $\pi_v(\Lambda) = \mathrm{SLS}(\Omega)$. In any other case, at least one symmetry point p such that $p \in \mathrm{SLS}(\Omega)$ and $p \notin \pi_v(\Lambda)$ must exist. One should also note that the relation above is not always true for medial descriptors that detect only a fraction of the symmetry set of a contour (e.g. the medial axis, Section 3.1.1).

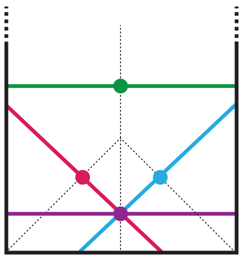


Figure 4.4: *Symmetry points are locally unique when their contact lines with the boundary do not intersect any other symmetry point. Red, blue and violet symmetry points are not unique, the green point is locally unique.*

What happens when a silhouette is partially occluded? Should it be discarded? Or it is still possible to get some reliable information from it? Inferring information about the axis of a generalized cone is still possible, as long as one is able to distinguish between symmetry points that have been affected by occlusions and symmetry points that have not. Occlusion-free symmetry points can be locally characterized by leveraging the locality properties of the SLS.

Let $\Phi : \Omega \rightarrow \mathbb{I}^+$ be a function that assigns to each point of the silhouette Ω the number of points projected over it by π_v. Each symmetry point $p \in \mathrm{SLS}(\Omega)$ depends only on the behaviour of the boundary restricted to two points, a and b, such that $p = \frac{a+b}{2}$. Therefore, p is *occlusion-free* if and only if each point in $\overline{ab}$ is occlusion-free, that is

$$\forall\, q \in \overline{ab},\ \Phi(q) \leq 2. \qquad (4.3)$$

To understand why the number of surface points that project over a silhouette point must be at most two one can think of a perfectly convex shape, like a sphere. A sphere will always project to any interior point of any its silhouette exactly two points (front and back surface). For a general shape, whenever the number of projections is higher than 2 there is an occlusion, as distant parts of the shape ore overlapping along the direction of projection.

Restrictions *R1-R3* can be formulated in local fashion too. Consequently, there is only one missing ingredient for being able to formulate an occlusion-aware equivalent of the Marr's theory: locally uniqueness of the symmetry axis. This is extremely important for the interpretation of a contour because if there is more than one possible symmetry then there are different possible generalized cones that generated the silhouette, hence there is an ambiguity. The SLS is a very powerful shape descriptor as it is able to detect *all* the symmetries of a shape. This is very useful in this context because it makes possible the characterization of locally unique symmetry points. In [LS13a] we stated that a symmetry point $p = (a+b)/2 \in \mathrm{SLS}(\Omega)$ is *locally unique* (Figure 4.4) if and only if

$$\overline{ab} \cap \mathrm{SLS}(\Omega) = p \tag{4.4}$$

By merging all the observations above one can finally state that, given an occluding contour $\partial\Omega$ and a symmetry point $p = (a + b)\backslash 2 \in \mathrm{SLS}(\Omega)$, if the symmetry is both locally unique and occlusion-free in p, and if restrictions *R1-R3* locally hold, then

$$p \in \pi(\Lambda)\ . \tag{4.5}$$

To prove that, note that if *R1-R3* are satisfied in a and b, then p will always be a SLS point, regardless the behaviour of the rest of the boundary $\partial\Omega$. Moreover, the segment $\overline{ab}$ will split the domain in two parts Ω_1, Ω_2. As both the axis Λ and the mapping π are linear, to move from Ω_1 to Ω_2 the projection of the axis $\pi(\Lambda)$ will cross $\overline{ab}$ exactly once. Therefore, for Equation 4.4 and Equation 4.2, $\pi(\Lambda)$ will pass through p.

The results presented in this chapter provide a tool for deciding whether a symmetry point of a silhouette is the projection of a 3D point belonging to the axis of a generalized cone or not. With this local interpretation partially occluded silhouettes can be considered and contribute to the construction of the 3D model representation. This may be important for algorithms that employ multiple silhouettes to detect the curve-skeleton of a 3D shape. In particular, two candidate algorithms that adopt this *3D-from-2D* paradigm will be presented in Chapter 5.

Chapter 5

Leveraging perception theory for skeleton extraction

One of the most important results of the theory developed to model the interpretation of silhouettes is the link it creates between the symmetries of a contour and the symmetries of the 3D shape that generated it (Section 4.2). In this chapter two algorithms that exploit this relation in order to compute curve-skeletons of the kind discussed in Section 4.1 will be presented. As the silhouettes of a complex object may contain a high amount of overlapping and reduce the interpretability of each contour both methods employ a multi-view approach. Multiple rotated projections of a real object are proved to increase the recognition capabilities of the human brain [Lee98] and most of the times are sufficient to detect a full generalized cone description of it. Both algorithms rely on a similar schema that can be decomposed in three main steps:

- cast a set of silhouettes by looking at an object from different points of view
- interpret each silhouette according to the visual perception theory discussed in Section 4.2
- collect the information coming from each contour and project it back to the 3D space to form the final curve-skeleton

The main difference between the two methods is how they interpret contours. The first algorithm is *naive* (Section 5.1), in the sense that it assumes the medial representation of each contour to be the actual projection of the curve skeleton of the depicted shape. This is in general false as occlusions

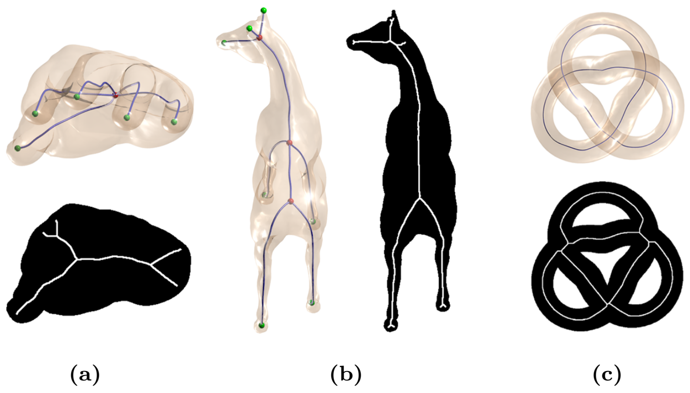


Figure 5.1: *The relation between the symmetries of a contour and the symmetries of the 3D shape that generated it can be: very good (c); good, but with some missing parts (b); or completely unrelated (a). The accuracy of the relation mainly depends on the amount of occlusions occurring within the silhouette.*

must be taken into account in some way (Figure 5.1). In spite of that, the idea is that considering the information coming from multiple views one will be eventually able to filter out the noise and retain the final curve-skeleton. The second algorithm is more accurate (Section 5.2) as it strictly follows the restrictions imposed by the early visual perception theory (Section 4.2.1) to pre-filter each 2D skeleton and consider, for the constitution of the curve-skeleton, only the 2D points that are proven to be the real projection of a 3D symmetry point.

Classic skeletonization methods (Section 3.2.2) tend to work on the machine side, focusing more on both *primitives* and *resolution* than on the *appearance* of a shape, as human beings do. The perceptual approaches discussed in the remainder of this chapter are completely unrelated with the geometric primitives used to describe a shape and, as long as the appearance is preserved, resolution and noise have a negligible impact on the results they produce (Section 5.1.5). Moreover, they are usually faster than the state of the art counterparts (Section 5.1.6).

5.1 Naive method

This section will introduce the algorithm for curve-skeleton computation that we presented in [LGS12]. Our method is based on the assumption that the medial axis (Section 3.1.1) of a planar projection of a 3D shape

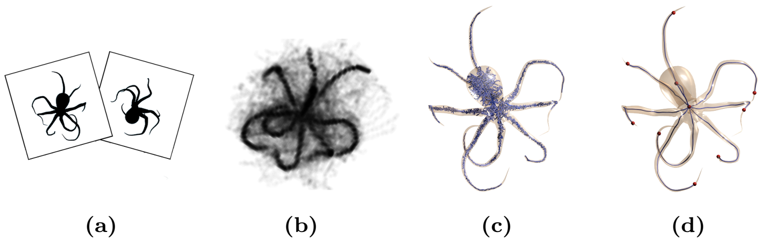


Figure 5.2: *Outline of the naive method. A set of silhouettes of the object to skeletonize are cast from different points of view (a); the medial axis of each silhouette is projected back to the 3D space to vote a discrete grid (b); a spanning tree of the portion of grid covering the interior of the shape is computed (c); topological operations are applied to the spanning tree to reveal the ultimate curve-skeleton (d).*

is the actual projection of its curve-skeleton. Although in some cases this might be true, in the general case it is wrong. The reason is that every time a 3D shape is projected to a plane some parts of it are compressed along the direction of projection, causing an alteration of the contour of the silhouette (i.e. they overlap). Due to occlusions elongated parts may be foreshortened or completely hidden. In these cases silhouettes will be either poorly representative of the original shape (Figure 5.1a) or will fail at representing some of its main features (Figure 5.1b). However, when the amount of overlapping is low, the curve-skeleton will roughly project over the medial axis of the contour (Figure 5.1c).

Silhouettes can be extruded in the three-dimensional space along the direction of projection. The Visual Hull ($\mathcal{VH}$) of the depicted object is the intersection of all the volumes generated from extrusion and represents the closest approximation of the shape that can be inferred by its contours [Lau94]. This algorithm is therefore meant to compute the curve-skeleton of the visual hull of a shape, rather than of the shape itself. It is worth noticing that the higher the amount of silhouettes that contribute to the generation of the visual hull, the finer the approximation. Therefore, the accuracy of the result can be controlled by choosing how many silhouettes consider and from what points of view (Section 5.1.1).

To compute the curve-skeleton the medial axis of each silhouette is projected to the three-dimensional space to vote a discrete grid (Section 5.1.2). The number of votes accumulated into each element of the grid represents the probability, for that point, to belong to the skeleton. Points that project outside of the visual hull will be forced to have null probability as they clearly do not belong to the skeleton. The *real* projection of the skeletal components will be consistent through a high number of views therefore the corresponding paths in the grid will receive a high number of votes. On the other hand, wrong information generated by occluded silhouettes will occasionally occur and will be inconsistent along the views, thus, receiving a smaller amount of votes. A spanning tree covering the portion of grid contained within the visual hull is then computed, giving higher priority to the most voted voxels (Sections 5.1.3). This is highly desirable as the voxels with higher votes in the grid are the most representative of the skeleton, while low-valued ones should only be considered for connecting different high-valued regions due to their expected inaccuracy. Finally, the spanning tree is processed according to the topological operations described in Section 5.1.4 in order to reveal the final skeleton paths and to ensure topological coherence with respect to the original shape. A visual outline of the pipeline is given in Figure 5.2.

5.1.1 Camera positioning and 2D symmetry extraction

The choice of the viewpoints is the core factor in the construction of the approximated model of the object. Even though it could be possible to specify a mesh-dependent set of views [Pet98] there is no way to understand whether the obtained $\mathcal{VH}$ is a satisfactory approximation of the shape [Lau94]. Laurentini stated that the number of silhouettes necessary to optimally describe a polyhedra with n faces is: unbounded if the viewpoints are not allowed to lie inside the convex hull of the object; $O\left(n^5\right)$ if the viewpoints are allowed to stay into the convex hull [Lau97]. However, even if the $\mathcal{VH}$ is optimal, some differences respect to the original shape may occur, for example in case the object contains cavities [Lau95]. As the computational extent would be too high, this method employs a simple camera positioning schema designed to evenly cover the space around the object: a regular grid of cameras centered in the vertexes of a discrete 21-points hemisphere. Covering just half of the visible horizon is enough as the silhouette projection is symmetric. Both the shape and the hemisphere are centered in the coordinate reference system, while the cameras point toward the origin of the axes. This choice proved to be sufficient in most of the experiments, finer resolution hemispheres did not increase the $\mathcal{VH}$ accuracy significantly. Nevertheless, it is worth noticing that the method does not depend on the particular camera positioning. Smarter heuristics [SP91, SLF*11, DGB*14]) would accommodate better $\mathcal{VH}$ for some kind of shapes.

To project points in the 3D space a stereo acquisition system is created, pairing each camera in the hemisphere with a second one having direction of projection perpendicular to it. Among the infinite possible directions that satisfy this perpendicularity constraint, the direction of projection of

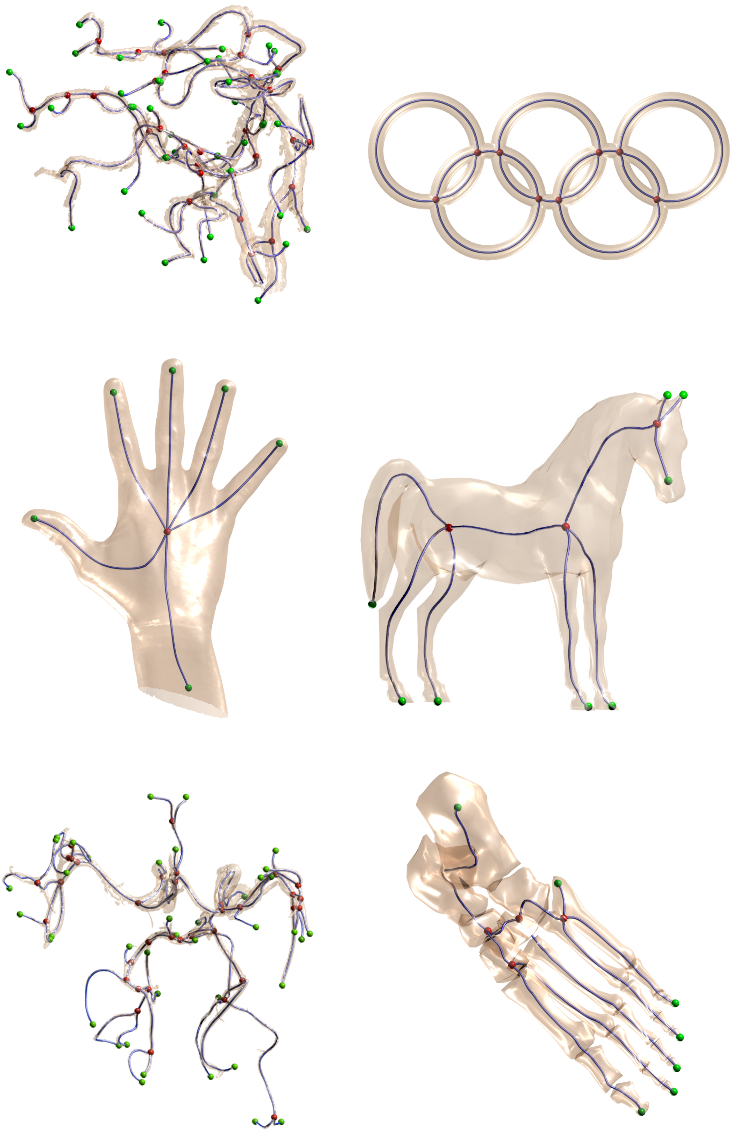


Figure 5.3: *A gallery of curve-skeletons extracted from shapes of different kind and topology. From left to to right, top to bottom: an aneurysm, the Olympic rings, a hand, a horse, an angiography and the bones of a human foot.*

the second camera is chosen in such a way that it is also parallel to the less principal component of the silhouette generated by the first camera (which is given by the smallest eigenvector of its projection's Principal Component Analysis). This is meant to minimize depth overlapping and produce the best possible stereo pair.

For each binary silhouette a Distance Transform ($\mathcal{DT}$) based medial axis [SdB94] is computed. The $\mathcal{DT}$ value of each pixel indicates the distance of that pixel from the border and will be used as an approximation of the local thickness of the shape along the image plane, adding volumetric information to each stereo projection (Section 5.1.2).

5.1.2 Matching and radii estimation

To keep the back-projection step simple and fast to compute shapes move instead of cameras. Let $\mathcal{O}$ be an object centered at the origin of the coordinate reference system $\mathcal{F}\,(O, X\;Y\;Z)$, and c_1, c_2 be a set of stereo points of view having orthogonal directions of projection. A new coordinate reference system $\mathcal{F}'\,(O, X'\;Y'\;Z')$ is defined in such a way that the axes Z' and X' correspond to the lines joining c_1 and c_2 with the origin O. Then, the object $\mathcal{O}$ undergoes the transformation $t^{-1}(\mathcal{O})$ to generate the stereo projections in $\mathcal{F}'$ as if $\mathcal{O}$ was observed from c_1 and c_2 in $\mathcal{F}$. Note that t is the rotation matrix defined such that $\mathcal{F}' \equiv t\,(\mathcal{F})$.

Each stereo pair is composed by two affine cameras P_z and P_x positioned respectively along the z and the x axis. Cameras are defined by the following homogeneous matrices

$$\mathsf{P}_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathsf{P}_x = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.1}$$

Affine cameras make the stereo matching trivial as the images they produce are *rectified* (Appendix B) by construction. In this setting epipolar lines coincide with the horizontal scanlines of the images, meaning that two corresponding points must lie within the same horizontal scanline. Each pair of rays has the form

$$r \begin{cases} x = p \\ y = q \end{cases} \qquad r' \begin{cases} z = k \\ y = q \end{cases} \tag{5.2}$$

where $y = q$ is provided by the epipolar constraint and the complete separation between x and z coordinates is provided by the orthogonal directions of projection. The back-projected point is then

$$r \times r' = [p\,q\,k\,1]^{\mathsf{T}} \ . \tag{5.3}$$

Points projected back to the 3D space will vote a regular grid $\mathcal{G}$. The regularity of the grid makes the voting a simple operation. Every voxel $v_{ijk} \in \mathcal{G}$ has a starting value of 0 and, whenever a point projects within its volume, the update procedure consists just in incrementing the corresponding vote by one

$$\mathcal{G}[i,j,k] = \mathcal{G}[i,j,k] + 1 \; . \tag{5.4}$$

Projected points carry both positional and volumetric information about the depicted shape. Indeed, to each medial axis points is associated an approximation of the Euclidean distance respect to the boundary of the silhouette (i.e. the distance transform). This information can be therefore used to obtain an estimation of the *local thickness* of the model. Each voxel stores the radius of the maximal ball centered in it; the estimation of this radius is achieved according to the following formula

$$\mathcal{R}[i,j,k] = \min\left(\mathcal{R}[i,j,k], \min\left(\mathcal{DT}_{\text{front}}, \mathcal{DT}_{\text{side}}\right)\right) \; , \tag{5.5}$$

where $\mathcal{DT}_{\text{front}}$ and $\mathcal{DT}_{\text{side}}$ are the distance transform values measured at the medial axis pixels that generated the projection of the point. The initial guess for the radius is setted to ∞ for each element of the grid. Non-voted voxels will not be taken into account by further steps of the algorithm.

Depending on the coordinate reference system different vote accumulations may occur. In order to improve the robustness to rotational variance, the model is pre-aligned with its principal axes. In particular, the object is rotated such that the first camera points toward the eigenvector corresponding to the smallest eigenvalue given by Principal Component Analysis. This minimizes the information loss in projection. The *up-vector* is set parallel to the greatest eigenvector, thus maximizing data distribution on the y direction used in scanline matching. In this way the xy image plane is supposed to be the best representative of the shape yet tends to uniform the set of views used for similar objects even under rotation.

5.1.3 Grid processing

The low reliability of many-to-many matches may result in situations where spurious external branches stand out in the grid, especially for models having complex topologies or showing a high number of skeletal points along each scanline. Since the visual hull of the shape is not explicitly stored but implicitly defined by both the silhouettes and the directions of projection, to delete spurious skeletal candidates that do not belong to the interior of the model the centroid of each voxel is projected over each silhouette. If it projects outside at least one silhouette, its corresponding vote will be setted to 0 as it clearly does not belong to the $\mathcal{VH}$ of the model. This technique results in a strong improvement of the grid quality (Figure 5.4) and, due

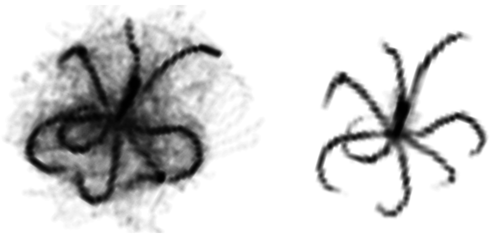


Figure 5.4: *Forcing to zero the votes given to voxels that lie outside the visual hull of the object dramatically improves the quality of the grid.*

to the low number of views and the simplicity of the operations involved, it is computationally cheap as it adds just a little overhead to the whole procedure.

After the $\mathcal{VH}$ cleaning the grid is ready to be processed for the final skeleton extraction. A maximized spanning tree is then computed by adopting a technique loosely based on the Ordered Region Growing (ORG) algorithm [YCS00]. The ORG method builds a tree-like representation of a 3D image where each voxel is a node and the arcs between nodes form the path between two voxels. These paths satisfy the least minimum intensity constraint, that is, the intensity in a path between two voxels has the maximum intensity achievable. Let $min(p_{ij})$ be the minimum intensity of each voxel in the path p between voxels i and j, and let g_{ij} be the path obtained by the graph traversal from node i to node j: it is guaranteed that $min(g_{ij}) \geqslant min(p_{ij})$ for every other p_{ij}. This feature is highly desirable in skeleton extraction, as the voxels with higher values in the grid are the best representative of the skeleton, due to a higher number of voting views, while low-valued ones should be used only for connecting different high-valued regions due to their expected inaccuracy or spuriousness (Figure 5.5a).

The ORG tree is built as follows: starting from a seed point G_0 (the maximum valued voxel in the grid), the region G_1 is constructed from its 26-neighbourhood, and arcs between the seed point and each neighbour are added to the graph. Let G_i and B_i be, respectively, the region and its outer boundary at the i^{th} iteration, and s_i the maximum valued voxel in B_i. G_{i+1} and B_{i+1} are constructed from s_i by adding its unvisited neighbours, that is, those neighbours that are not already contained in G_i. New arcs are created between s_i and each voxel in $(B_{i+1} \setminus G_i)$. The process is iterated until every voted voxel has been included in the region.

5.1.4 Topology recovery

As the majority of the voxels received votes coming from spurious matches the skeleton consists of a very small subset of the ORG spanning tree (Figure 5.5). In this section a set of topological operations that allow to process

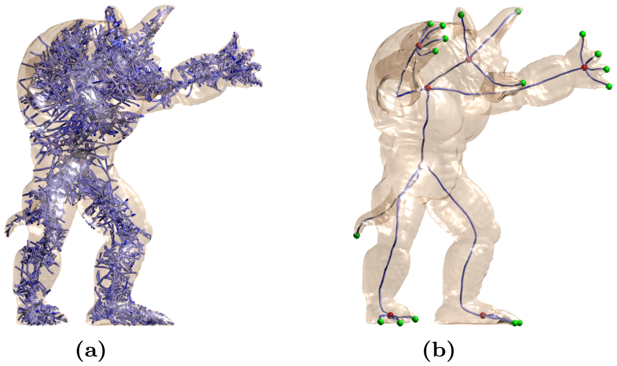


Figure 5.5: *The ORG spanning tree (a) is graph structure that covers the voxel grid giving higher priority to the most voted paths. Only perceptually significant branches are extracted as part of the curve-skeleton (b).*

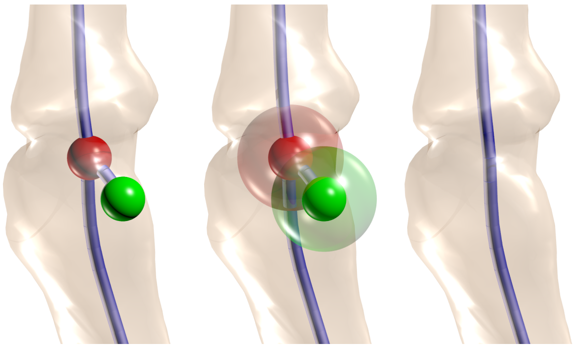
the spanning tree in order to retain the final curve-skeleton will be presented (Figure 5.6). These operations are based on the definition of *zone of influence* (ZI) [SSdBA10] of a node, that is, the volume bounded by the maximal ball centered in it (Section 5.1.2).

Perceptual core extraction

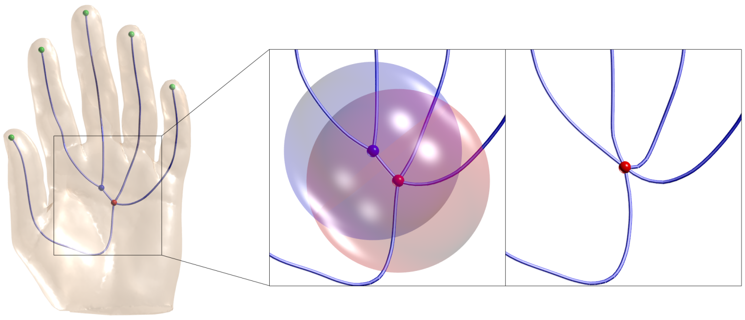
As human interpretation does not suppose the presence of dents or bumps in a shape without evidence of them [RKH87] the curve-skeleton should be composed only of *perceptually significant* branches. In order to be perceptually significant, the endpoint of a terminal branch must *stand out* in at least one view, meaning that there will be no intersection between its zone of influence and the zone of influence of the junction the terminal branch generates from. By iteratively pruning all the non-perceptually relevant terminal branches (Figure 5.6a) the components of the final curve-skeleton will appear out of the ORG spanning tree (Figure 5.5b).

Branch collapsing

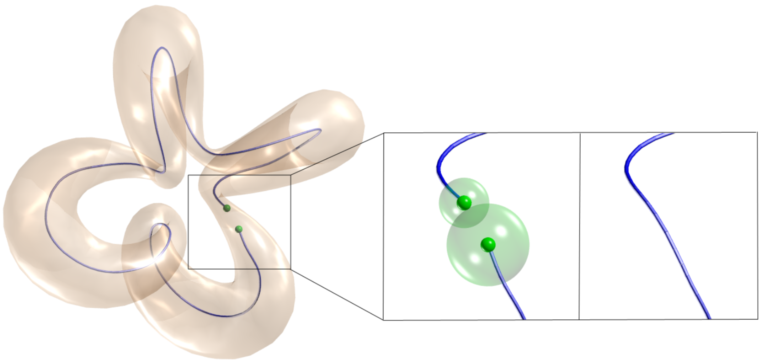
Parts of the skeleton that are supposed to converge into a single junction often meet at different points, creating a cluster of joints linked together by short arcs. In order to retrieve the correct relation between the logical components of the shape and the parts of the skeleton these clusters should collapse to single junction points. This can be done by merging together adjacent joints as long as there is intersection between their ZIs (Figure 5.6b). The generated joints will have as coordinates the barycenter of the junction points involved in the merging, and as radius the minimum radius of the



(a) *Perceptual core extraction*



(b) *Clustered joints merging*



(c) *Loops recovery*

Figure 5.6: *The three topological operations used to process the ORG spanning tree. Terminal branches whose maximal spheres have non-empty intersection are pruned (a); inner branches whose maximal spheres have non-empty intersection are collapsed (b); leaves whose maximal speheres have non-empty intersection are merged together to form a loop (c).*

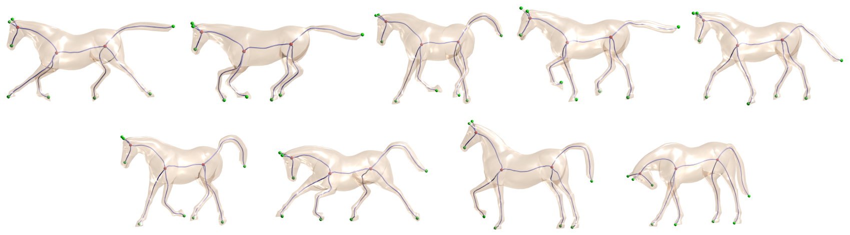


Figure 5.7: *Due to the topological merging operation, the skeleton of the same shape assuming different poses is coherent throughout the sequence.*

same junction points. Inner branch collapsing is a useful operation especially when multiple skeletons of the same shape assuming different poses are considered. Since shape animation is usually a volume-preserving operation the maximal balls will roughly keep the same radius, making the joint configuration stable and leading to a perfect matching between the logical component of the shape throughout the sequence (Figure 5.7).

Loops recovery

A tree can only represent genus zero shapes. For shapes having higher genus correct topology must be achieved in post-processing. In order to recover all the loops that are necessary to achieve topological coherence with respect to the original shape the algorithm checks the zone of influence of each leaf of the tree, closing a loop with any skeleton point whose zone of influence has non-empty intersection with it (Figure 5.6c). However, the immediate neighbours of a leaf always satisfy the condition above, thus, to be sure that a loop really needs to be closed, an additional condition must be observed: let p be the skeleton point whose ZI intersects the ZI of a leaf l. Along the path joining p to l there must be at least one point having empty intersection with the zone of influence of l. This condition must hold for all the leaves involved in the loop closure.

To avoid the creation of fake loops, the topological operations described above must be applied in the order they are presented. Furthermore, to increase the visual appearance of the skeletons, Laplacian smoothing may be applied to the branches, averaging the position of each branch point respect to the position of its neighbours. However, to prevent branch shortening, leafs and joints should not be involved in the smoothing process.

5.1.5 Results

The naive approach is capable of extracting correct skeletons that accurately reflect the topology of shapes having any genus or complex morphologies (Figure 5.3). The results are visually appealing and satisfy most of the properties listed in Section 3.2.1. The *thinness* criterion is fully satisfied as skeletons are always composed by mono-dimensional curves extracted from the ORG spanning tree. To each skeleton point is also associated the radius of a maximal sphere, therefore, summing up the spheres associated to each point a rough *reconstruction* of the original shape can be obtained. Furthermore, this approach is very *robust* against noise. This is probably the strongest point of the whole paradigm: contours do not suffer local perturbations in the data, as opposed to other approaches that rely on information tightly coupled with it (e.g. surface normals). Indeed, the method is capable of extracting skeletons even when part of the data is missing, as soon as the visual aspect of the object is reasonable and the missing parts are not influencing the production of every silhouette (Figure 5.10). Considering contours at different resolutions a Level Of Detail (LOD) *hierarchy* of skeletons can be also produced. As the algorithm employ a multi-view camera system, it is inherently *invariant under rigid motion*. In addition, perceptual skeletons create a one-to-one correspondence with the main logical components of the shape, thus, exposing a good *component-wise differentiation*. However, there are some limitations in capturing secondary features, though this mainly depends on the resolution of the considered contours (Section 5.1.7). On the other hand, there are properties like *centricity*, that can be observed but not guaranteed as the multi-view approach strongly depends on the silhouettes chosen to represent a shape. Skeletons are usually *homotopic* with respect to the original shape but topology preservation is not guaranteed because it is achieved only in post-processing. However, the naive algorithm produces good results as long as the topology of the visual hull equals the topology of the shape. Finally, perceptual skeletons do not satisfy the *reliability* criteria: no effort has been made towards direct shape-skeleton visibility.

Faces	50%	25%	5%	2.5%	0.5%	0.25%	0.1%
Max	2.19%	0.67%	1.53%	1.32%	1.48%	1.65%	1.87%
Avg	0.16%	0.14%	0.17%	0.19%	0.31%	0.36%	0.57%

Table 5.1: *Comparisons among skeletons extracted from the Raptor model (2M faces) at various resolutions. In the first row there are the decimation percentages. Each cell shows the Hausdorff and average error (in percentage of the length of the bounding box diagonal).*

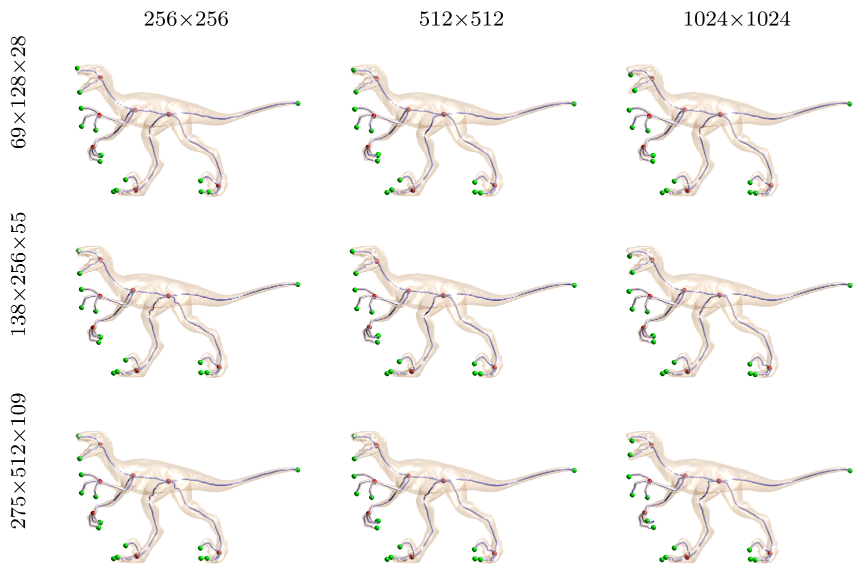


Figure 5.8: *Results obtained at different silhouette (column-wise) and voxel grid resolutions (row-wise): the difference, in time, between the fastest (upper left, 0.31 secs) and the slowest (lower right, 5.02 secs) is one order of magnitude.*

The algorithm has three degrees of freedom: model resolution, silhouettes resolution and grid resolution. Unless differently specified any silhouette is a 512×512 pixels image and the discrete grid covers the longest side of the bounding box of the shape with 128 voxels. Experiments showed that these parameters have little influence on the overall results both in terms of timing and output quality. The main computational bound is given by the silhouette generation step: the model resolution affects the timings because more time is needed by the GPU to rasterize all the primitives describing the object. Quality-wise, however, this method extracts coherent skeletons from simplified models too, therefore it is possible to reduce the running times by decimating high-resolution models with nearly no information loss (Table 5.1). Furthermore, either choosing a different resolution for the silhouettes or using a finer grid has almost no impact on the final skeleton (Figure 5.8).

Skeletonization of raw point clouds

Silhouettes can be produced by projecting any type of primitive describing a model. However, in case of a point cloud, this approach will not lead to a dense contour as gaps in between projected vertices will occur. In spite

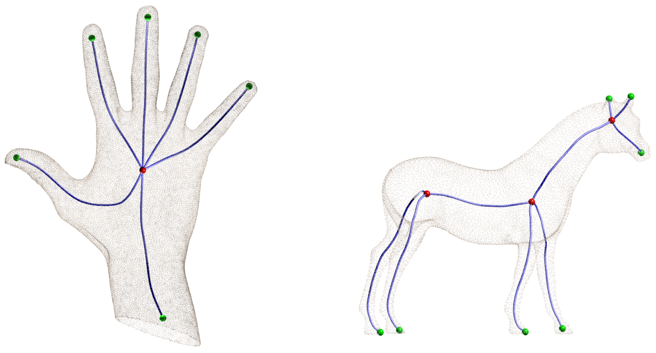


Figure 5.9: *Two examples of skeletonization of unoriented point clouds produced by considering dense silhouettes created by applying morphological operations to each projected contour.*

of that, by performing a *morphological closing* of the projected image it is possible to reconstruct a silhouette that allows to proceed with the skeleton extraction (Figure 5.9). To obtain an accurate silhouette the size of the structural element must be chosen as a function of the density of the cloud, even if, for sparser point sets, narrow regions may be merged due to its higher size. The general benefits of the perception-based approach extend to the point cloud case: skeletons remain noise insensitive and robust. It is worth noticing that point clouds can be either oriented or unoriented; normals are not necessary for skeleton extraction.

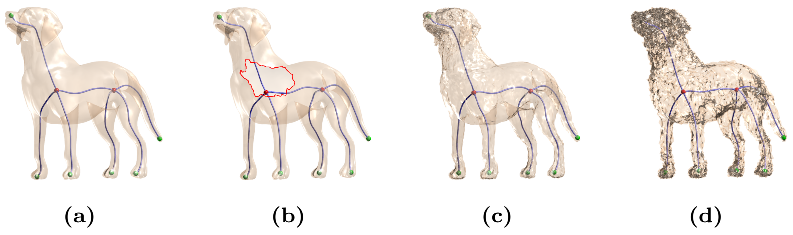


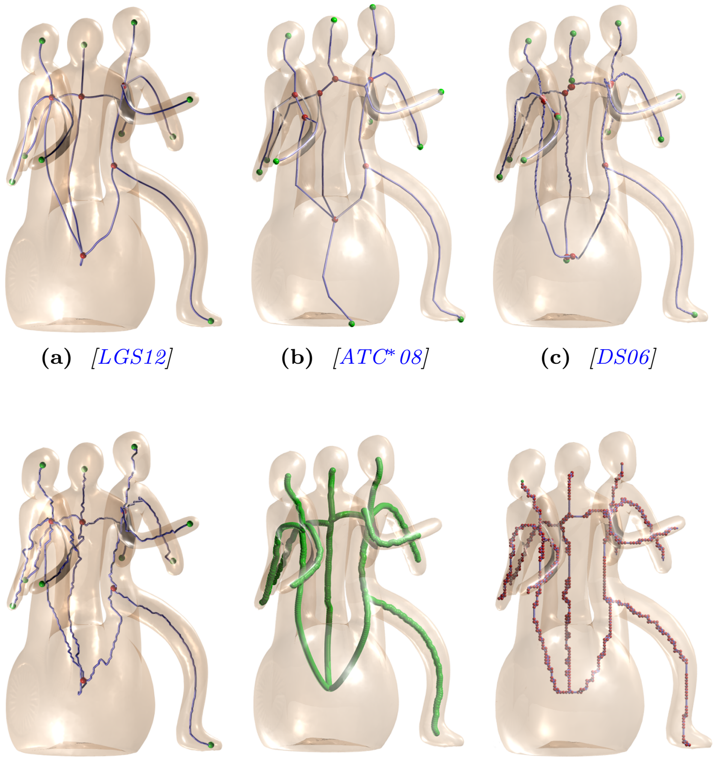
Figure 5.10: *Being silhouettes loosely related with the primitives used to describe the shape, perception-based approaches are very robust against noise and missing data. From left to right: the skeleton of respectively a clean dog (a), a dog with a large chunk of missing data around the neck (b), and two dogs, (c) and (d), affected by increasing levels of noise.*

Model (#Faces)	Percep. based [LGS12]	Contract. based [ATC*08]	Med. Geod. func. [DS06]	Force Follow. [CSYB05]	Cell Compl. Thinn. [LCLJ10]
Octopus (15,054)	300	2,828	217,707	10,406	559
Dog (36,224)	341	10,675	554,937	51,500	2,123
Armadillo (69,184)	637	30,630	1,596,273	118,390	4,712
Horse (96,966)	585	41,765	3,294,194	49,516	2,024
Memento (52,550)	589	23,584	2,697,171	538,223	3,908
Hand (273,060)	1,340	281,469	33,775,316	25,547	1,073

Table 5.2: *Time comparison (in milliseconds) among the different methods tested. Experiments were run on a iMac equipped with 2.66GHz Intel Core 2 Duo and 4GB RAM.*

5.1.6 Comparisons

Comparisons between the naive algorithm and four state of the art techniques are provided (Figure 5.11). In the volumetric category the selected algorithms are the Force Following approach [CSYB05] and Cell Complex Thinning [LCLJ10] while for the mesh-based algorithms there are the Laplacian Contraction [ATC*08] and the Medial Geodesic Function [DS06]. Timings, as listed in Table 5.2, show that the silhouette-based approach is noticeably faster than its counterparts. This mainly depends on the fact that working with contours the resolution of the models has almost no impact on the computation. It only affects the generation of the contours though modern graphic cards are very efficient at rasterizing primitives, especially triangles. For the volumetric methods the main factor influencing timing is the thickness of each branch. In fact, the Hand model has higher resolution than other model (e.g.: Horse and Octopus) though it is processed faster, due to the thinness of its palm and fingers. For both these methods, a finer voxelization would accomodate higher accuracy, but would also result in a strong increase of computational time. On the other hand, the naive method is loosely sensitive to both grid and silhouette resolutions (Figure 5.8). Parameters are a key factor also in terms of topological coherence. The Force Following algorithm may result in great gaps between the skeletal points (e.g. non-connected skeleton parts) and a shape-dependent parameter tuning has to be found in order to obtain a coherent skeleton. The same can be said for the Cell Complex Thinning, that can produce both 1D and 2D skeletons depending on the parameters setting. Comparisons with mesh-based methods show that the naive method is faster, especially for high-resolution meshes, where more time is needed to process the higher number of primitives used to describe the shape. On the other hand, for low-resolution shapes mesh-based algorithms tend to produce skeletons with missing fea-



(a) *[LGS12]* **(b)** *[ATC*08]* **(c)** *[DS06]*

(d) *[LGS12]* **(e)** *[CSYB05]* **(f)** *[LCLJ10]*

Figure 5.11: *Visual comparison between the perception-based approach, with (a) and without (d) the final Laplacian smoothing, and other state of the art algorithms for curve-skeletons computation.*

tures or not completely inside the object (Figure 4.1) while silhouette-based methods can accurately compute a skeleton unless the visual appearance of the model is preserved.

5.1.7 Limitations

The perception based method is intended to work on cylindrical and articulated objects such as character-like models, animals and human figures, that is, the class of *generalized cones*. Objects not belonging to this category cannot be successfully skeletonized (Figure 5.12a). However, as pointed out in Section 4.1, for these objects a curve-skeleton may not be the best descriptor in the first place. As long as it makes sense to choose

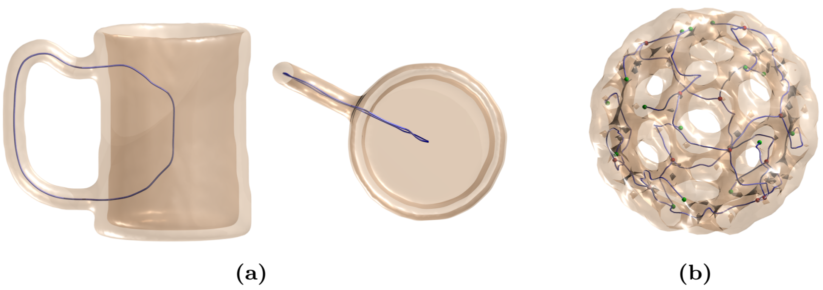
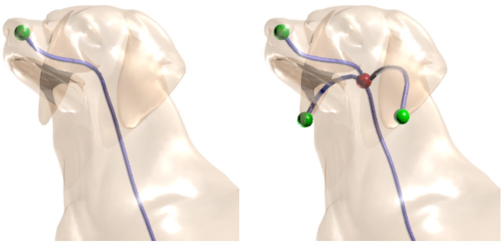


Figure 5.12: *Two datasets where the naive method fails. A mug does not belong to the class of generalized cones and its deep cavity is not represented properly by the skeleton (a). The multi-view system employed by the algorithm is not able to provide an exhaustive representation of the complex morphology of the buckyball molecule (b).*

a mono-dimensional skeleton to represent a shape (e.g. for purposes of segmentation or animation), the algorithm is able to perform well. The multi-view system computes a skeleton of the $\mathcal{VH}$ of the shape rather than the object itself, making the algorithm unreliable for objects with no $\mathcal{VH}$ features (Figure 5.12b). Nevertheless, the accuracy of the visual hull can be improved by considering more silhouettes or by using a different layout for the actual camera system. It is important to notice that, being based on the shape approximation given by the visual hull, the algorithm cannot extract all the features which are occluded in *every* projection or which are much smaller with respect to the projection resolution. For this reason high resolution silhouettes are able to isolate small-scale features, while low-resolution silhouettes tend to capture only the principal features of a shape. An example of this behaviour is given in the figure aside, where two alternative skeletons for the head of the dog are proposed. The left skeleton has been computed considering 256×256 pixels silhouettes and does not contain an explicit representation for the ears because they were visually close to the head in every projection; the skeleton on the right has been computed using 512×512 pixels silhouettes, and ears are explicitly represented as they appeared separate



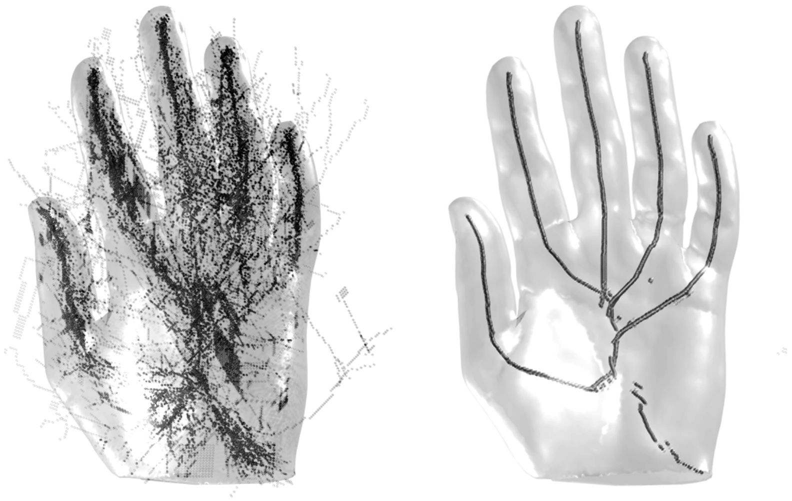


Figure 5.13: *Raw point clouds for the Olivier hand model produced by the two perception-based methods discussed in this chapter. Assuming every medial axis to be the actual projection of the skeleton most of the points projected back to the 3D space are spurious (a). On the other hand, projecting back only occlusion-free locally unique symmetry points (b) the cloud will expose almost no noise, making any cleaning or post-processing unnecessary. Moreover, working in the continuous the skeleton paths suggested by the cloud are naturally smooth.*

respect to the head in at least one view, making the algorithm able to recognize them. This reflects behaviour of human vision, where the saliency of a shape's features is relative with respect to the scale of observation [Wit84] and it really follows the perception-based philosophy. In fact, as explained in [WB98] *"the visual system represents simple spatial regions by their medial axes"* and *"the medial representation arises in a scale-specific way"*.

5.2 Accurate method

Even though the perceptual-based approach presented in Section 5.1 proved to be robust enough to compute the curve-skeleton of shapes of different kind and topology, it suffers the weakness of the assumption that the medial axis of any silhouette *always* coincide with the projection of its curve-skeleton. This becomes evident when the amount of occlusion is too high and the medial axis projected back to the 3D space bring a high amount of noise, making the skeleton extraction difficult and unstable. Nothing can be done to distinguish between noise and skeleton points because they both project into every single considered contour (i.e. they are into the visual hull). Furthermore, such method does not guarantee the skeleton paths to be centered because a discrete grid is used during the computation and because the smoothness of the skeleton paths is achieved only in post-processing, thus deviating the curves from the medial lines of the shape. This section will present the alternative, perception-based, skeletonization algorithm that we introduced in [LS13a]. Here we strictly followed the early visual perception theory to make sure that only 2D points that are *real* projection of parts of the curve-skeleton will be used during the computation. Therefore, this method is able to perform better with any kind of shape, producing incredibly noise free point clouds from which medial and naturally smooth skeleton curves can be easily computed, without requiring any further post-processing (Figure 5.13).

The idea is as simple as the previous method. Firstly, a set of silhouettes are cast by looking at a 3D model from different points of view. The symmetries of each contour are then computed and filtered according to the local interpretation of occluding contours presented in Section 4.2.1. To perform such analysis additional information about occlusions is required (Section 5.2.1). Symmetry points that pass the filtering step are guaranteed to be the *real* projection of a curve-skeleton point. Different observations of the same point among the collected views are then gathered together (Section 5.2.2) and processed with basic computer vision tools to locate the position of the original skeleton point (Section 5.2.3). Finally, the curves of the skeleton are detected by processing the point cloud generated at the previous step of the algorithm with a mono-dimensional moving least squares approach (Section 5.2.5).

5.2.1 Camera positioning and silhouette analysis

Silhouettes are cast just by rotating along the most important axis given by the Principal Component Analysis (PCA) of the model with a step of 3°. Complex models (e.g. fertility) required to rotate up to 180° around the object, generating at most 60 silhouettes, though simpler models (e.g. the Olympic rings) required fewer views.

The local interpretation of each silhouette represents the major innovation of this approach over [LGS12]. The local analysis works pretty much like a boolean test: every single symmetry point can be either the projection of a point belonging to the curve-skeleton of the depicted shape or not. While to detect locally unique symmetry points only the Smoothed Local Symmetry (Section 3.1.3) of the contour is required, to check whether a symmetry point has been affected by occlusions or not a discrete version

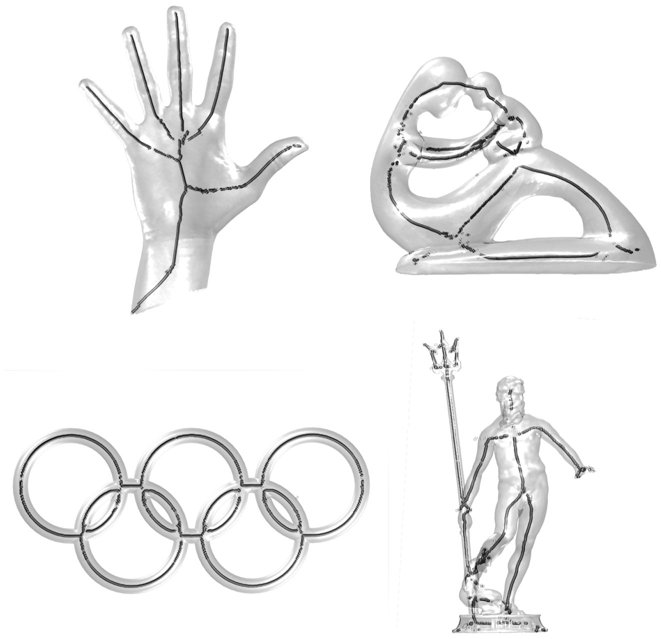


Figure 5.14: *A gallery of point clouds obtained back-projecting 2D symmetry points with the accurate method. Only occlusion-free locally unique symmetry points have been considered. This method can generate almost noise-free point clouds for many different kinds of objects. For each model in the figure contours have been gathered by rotating around the most important axis given by the PCA.*

of the function Φ introduced in Section 4.2.1 must be implemented. This can be done by customizing the behaviour of the OpenGL stencil buffer as follows

```
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_ALWAYS, 0x1, 0x1);
glStencilOp(GL_INCR, GL_INCR, GL_INCR);
```

The code above makes sure that each time an object's primitive is projected over a pixel, the stencil buffer entry corresponding to it will be increased by one. Pixels having a value greater than 2 will denote an occluded area of the silhouette.

5.2.2 Scanline matching

Affine cameras are used to cast the silhouettes, in order to guarantee the generation of a *rectified* (Appendix B) sequence of images. This choice makes the point matching problem very easy to solve because candidate matches always lie within the same scanline. To detect different observations of the same point along the sequence the method employs a two-step matching algorithm. Let $p^{(i)}$ and $p^{(j)}$ be two symmetry points belonging respectively to the i^{th} and j^{th} occluding contours. To have a positive match between $p^{(i)}$ and $p^{(j)}$ the following constraints must be observed

$$\left| \, i - j \, \right| = 1 \quad \text{and} \quad p^{(i)}_{\mathsf{row}} = p^{(j)}_{\mathsf{row}} \quad \text{and} \quad \left| \, p^{(i)}_{\mathsf{col}} - p^{(j)}_{\mathsf{col}} \, \right| \leq \delta \tag{5.6}$$

meaning that $p^{(i)}$ and $p^{(j)}$ must be two subsequent occluding contours in the sequence, must lie within the same scanline, and their distance along the scanline must be lower than δ. This threshold represents the maximum allowed displacement between two consecutive observations of the same point of a generalized cone axis (the default value is $\delta = 2$ pixels). The first matching generates a set of *bundles*, that is, groups of subsequent observations of the same 3D symmetry point. However, the same point can be observed for a certain number of views, disappear due to occlusions, then appear again. In the second matching step bundles that have been generated by projections of the same skeleton point are grouped together. To do that, the correspondence search algorithm proposed in [LH05] is used. Grouping in a single bundle all the observations of a skeleton point makes the algorithm more accurate as a higher number of 2D points contribute to the estimation of its 3D coordinates. Furthermore, the number of projected points will be lower, thus, reducing both the overall size and redundancy of the generated point cloud (Figure 5.14).

5.2.3 Back-projection

After all the symmetry points have been grouped together in coherent bundles, the next step consists in projecting them back to the shape space in order to reveal the skeleton paths. Every symmetry point defines a projective ray in the space; the intersection of all these rays defines a candidate skeleton point in $\mathbb{R}^3$. A projective ray is a line hence it can be expressed as the intersection of two planes in the space. Let $p \in \mathbb{R}^3$ be the coordinates of a general point belonging to the ray and $\vec{d}$ its direction. Each ray is defined by the following linear system

$$\begin{cases} \vec{d}_y x - \vec{d}_x y = \vec{d}_y p_x - \vec{d}_x p_y \\ \vec{d}_z x - \vec{d}_x z = \vec{d}_z p_x - \vec{d}_x p_y \end{cases} \tag{5.7}$$

Any bundle generates one candidate skeleton point whose coordinates can be computed by solving a dense linear system $A\mathbf{x} = \mathbf{b}$ composed by $2n$ equations, where n is the number of symmetry points in the bundle. The matrix A contains the directions of projection of the rays generating the

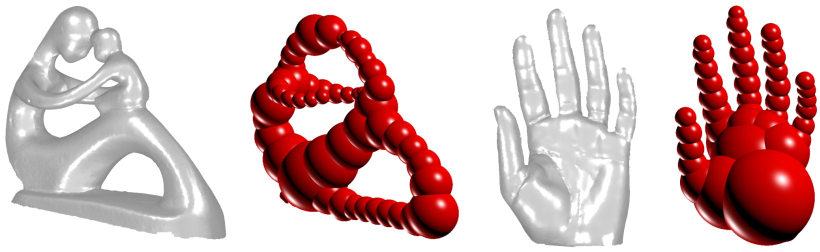


Figure 5.15: *Shape approximation of the Fertility model (55 balls) and Olivier Hand model (42 balls). Maximal balls are spanned along the skeleton paths, creating a good approximation of the original shape. This representation can be useful for applications like hole filling, surface reconstruction and collision detection.*

symmetry points, the unknowns of the problem are the xyz coordinates of a generalized cones' axes point. To increase the overall robustness bundles having less than 3 points are discarded, therefore the system will be always overdetermined and can be solved in the least square sense, according to the normal equations $\tilde{\mathbf{x}} = \left(A^T A\right)^{-1} A^T \mathbf{b}$, such that

$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\| \mathbf{b} - A\mathbf{x} \right\|^2 \tag{5.8}$$

An example of the point clouds produced by this method can be seen in Figure 5.14. As one can note, due to the power of the SLS filtering and the robustness of the scanline matching, even for complex models like Fertility and the Neptune statue, the clouds are almost noise free and the skeleton paths are quite clean.

5.2.4 Shape thickness

Beside the xyz coordinates of the skeleton points, another important information can be inferred from the analysis of the occluding contours. When a symmetry point is unique, its minimum distance from the boundary of the silhouette can be thought of as an approximation of the local thickness of the shape as seen from a viewpoint. Since every candidate skeleton point is defined by the intersection of at least three projective rays coming from different contours the algorithm always considers the minimum of the distances measured in every single silhouette as the most reliable approximation of the local thickness of the shape. This information will be really useful in the following step, where the skeleton paths will be reconstructed starting from the point cloud just created. Thickness information can also be used in a lot of applications, such as hole filling and surface reconstruction. Another typical application is collision detection, where a coarse representation of a model can be really useful to detect collisions between articulated objects, drastically reducing the complexity of the problem. In Figure 5.15 an example of reconstruction achieved using about 50 maximal balls is proposed for Fertility and Olivier hand models.

5.2.5 Curves extraction

The ultimate step of the method consists in the generation of the skeleton paths out of the point cloud produced by back-projecting symmetry points. As can be noted in Figure 5.14 the point clouds generated have a very thin structure along the branches while are a bit scattered near the joints. For the branches a simple mono-dimensional moving least square approach can be used, iteratively projecting points onto their corresponding locally best fitting lines via principal component analysis (PCA) [Lee00]. At each iterative step a subset of points lying in a small neighbourhood are selected. Since the cloud has been produced by the discrete representation of the contours, the size of the neighbourhood is a quantity proportional to the distance between two points that would project into two adjacent pixels. However, as approaching joints the clouds become a bit scattered, this neighbourhood measure is not able to properly infer branch connectivity. To thin the cloud and reveal how the branches connect each other Laplacian smoothing is applied, using the estimation of the local thickness of the shape (section 5.2.4) to infer point connectivity. A similar approach has been used previously in [TZCO09]. However, since in that method there was no thickness estimation the authors retrieved point connectivity using the Mahalanobis distance, which is computationally expensive: for 10K points they needed about 3 minutes of computations. In this method, thickness estimation comes from silhouettes, thus, it does not introduce any computational overhead, making the algorithm faster than state of the art counterparts. A gallery of models skeletonized using this algorithm can be found in Figure 5.16.

5.2.6 Results and comparisons

The accurate method follows the same perception-based paradigm followed by the naive method presented in Section 5.1. Therefore, skeletons produced by this algorithm are able to satisfy the very same properties. However, thanks to the symmetry points filtering applied to every silhouette (Section 5.2.1), the overall centeredness of the skeletons is more accurate than the naive approach, and is comparable with other state of the art techniques. To measure centeredness accuracy three synthetic shapes with convex cross-section everywhere have been considered: a double torus and two knots. For each model, a number of cutting planes define a set of cross-sections. The distance between the centroid of each cross-section and the

Model (# Faces)	Occluding contours	Rasterization time (ms)	Skeletonization time (ms)	Total time (ms)
Hand (273,060)	6	247	74	321
Neptune (56,112)	32	321	250	571
Fertility (50,000)	60	546	391	937
Twirl (10,402)	20	112	175	287
Knot #1 (4,160)	60	202	382	584
Knot #2 (6,400)	60	227	396	623
Knot #3 (11,520)	60	266	302	568
Olympics (7,862)	6	24	155	179
Spider (23,808)	60	357	442	799

Table 5.3: *Time splitting (in milliseconds) of each stage of the skeleton computation pipeline.*

curve-skeleton is then measured, normalizing values with respect to the diagonal of the bounding box containing the shape. As can be noted from Table 5.4, the centeredness achieved by the accurate method is comparable with the one of [DS06] and [TAOZ12] and outperforms the naive method. This may depend from the fact that [LGS12] employs a discrete grid to detect the skeleton paths and then applies Laplacian smoothing, thus deviating from the middle of the shape. As already pointed up in Section 5.1.5 the perception-based approach is noticeably faster than state of the art algorithms as it is not strictly related with the complexity of the shapes as most of the computation is performed in the silhouette space. Table 5.3 shows running times for the accurate method.

5.2.7 Limitations

The restriction to the class of the generalized cones and the dependence to the amount and quality of the silhouettes discussed in Section 5.1.7 are to be considered general flaws of the perception-based paradigm hence hold also for the accurate method. Furthermore, a stop rule for the silhouette acquisition system is needed. This approach uses a set of contours constructed by rotating around the most important direction given by the PCA. However, as can be noted in Table 5.3, some models needed a few contours some others more. A stop rule, to automatically determine how many views are necessary should be used. Finally, it is important to notice that, despite the naive approach, this method is not natively able to compute the skeleton of a point cloud as the tool for the detection of occluded areas discussed in Section 5.2.1 would not work with the projection of sparse points.

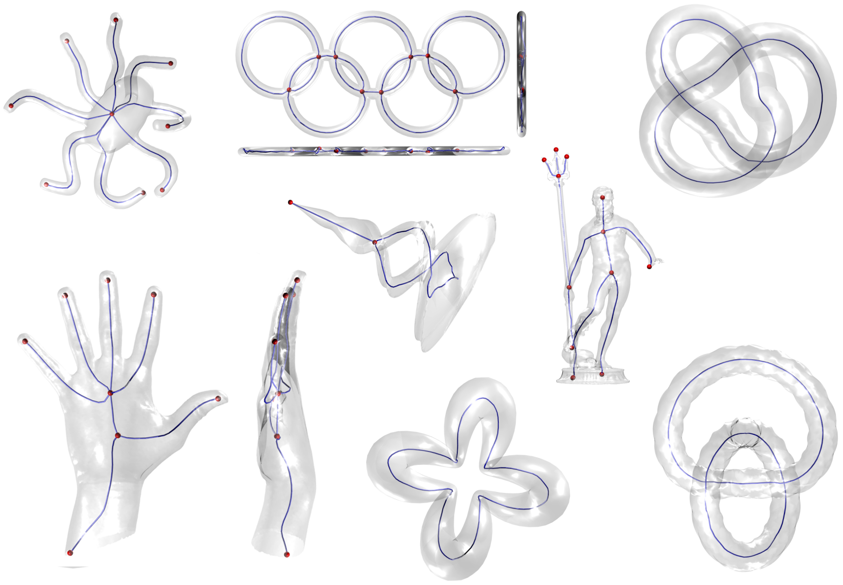


Figure 5.16: *A gallery of shapes of different kind and genus skeletonized using the accurate method.*

Method	**model**	**avg displ.**	**std dev**
Accurate method [LS13a]	Torus	0.059918	0.009305
Naive method [LGS12]		0.065188	0.025373
Medial Geodesic Function [DS06] ($\theta = 0.0$)		0.063134	0.012655
Medial Geodesic Function [DS06] ($\theta = 0.5$)		0.065679	0.012750
Mean Curvature Skeleton [TAOZ12]		0.001762	0.011715
Accurate method [LS13a]	Knot 1	0,044990	0.022698
Naive method [LGS12]		0.092037	0.027562
Medial Geodesic Function [DS06] ($\theta = 0.0$)		0.046228	0.023781
Medial Geodesic Function [DS06] ($\theta = 0.5$)		0.050125	0.024415
Mean Curvature Skeleton [TAOZ12]		0.001570	0.012924
Accurate method [LS13a]	Knot 2	0.052322	0.008020
Naive method [LGS12]		0.079337	0.018083
Dey and Sun [DS06] ($\theta = 0.0$)		0.054011	0.008043
Dey and Sun [DS06] ($\theta = 0.5$)		0.054063	0.008049
Mean Curvature Skeleton [TAOZ12]		0.001742	0.013306

Table 5.4: *Numerical comparisons with three state of the art skeletonization algorithms. For each model, a number of cutting planes define a set of cross-sections. The distance between the centroid of each cross-section and the curve-skeleton is then measured. Mean displacements are normalized with respect to the diagonal of the axis aligned bounding box containing the shape.*

Chapter 6

Conclusions

Shape understanding is the high-level container for both the ideas and methods discussed in this thesis. Understanding a shape, its structure and the way its components are related to each other is based on the detection of its *features*. However, as shown throughout the thesis, depending on the specific context, a feature may be different things. In this work, two different definitions of feature, aimed at the generation of compact shape representations, have been considered. In the first part of the thesis a novel method for the generation of high-quality PolyCube base-complexes has been introduced. In this context, a feature is a part of a shape that would decrease the mapping distortion if explicitly represented in the parameterization domain (Section 6.1). The second part of the thesis focused on a definition of feature that basically coincides with the logical components of a shape. The goal is to compute skeletal representations for general 3D models. In particular, this part of the thesis formalized the relation existing between the human visual perception system and skeletal representations, proposing two different algorithms that exploit this relation to automatically compute curve-skeletons of 3D shapes by considering just a set of planar contours (Section 6.2).

6.1 PolyCubes

When generating a PolyCube, the challenge is to detect the high-level structure of the reference model and reflect it in the polyhedron, in order to have a suitable domain that enables for a low-distortion mapping. An algorithm has therefore to understand what are the important features of a shape and to select, among those, what features should be explicitly represented in the PolyCube structure. For each candidate feature, the method has to weigh the benefit that the mapping would have in terms of reduction of the distortion with the complexity required for including it in the domain, that is, the number of extra faces and extra corners necessary to represent it. The algorithm presented in Chapter 2 is able to perform such a choice by solving a labelling problem that optimally combines the necessity to have compact domains with the structural complexity necessary to enable for a low-distortion parameterization. Consequently, this method generates high quality results, with low singularity counts and small distortion, across a wide range of models. PolyCut significantly outperforms previous work in terms of PolyCube compactness, without compromising mapping distortion. As the method is fully automatic and robust across a wide range of inputs, applications requiring PolyCubes can now be developed without a need for time-consuming manual or semi-automatic schemes. Furthermore, as the right balance between the complexity of the base domain and the quality of the mapping depends on applications, the compactness of the PolyCubes can be customized with a user-controllable parameter.

6.1.1 Future work

PolyCut does not necessarily depend on the hill climbing global optimization method. Future works involve the integration of other global search frameworks into the algorithm. Better PolyCubes may be generated via stochastic methods or by adding random restarts to the hill climbing routine. Furthermore, it is opinion of the author that this hybrid framework approach can be used to extend multi-label optimization to work on other problems in geometric mesh processing and computer graphics.

A well known limitation of current PolyCube generation algorithms is that they are inherently orientation-dependent: changing the relative position of a shape with respect to the global frame will generate different PolyCubes. Searching the best coordinate system onto which perform PolyCube computation would be an interesting line of search. A first solution to this problem is provided by Huang *et al.* in [HJS*14]. Their approach also shows high quality PolyCubes that generate good hex meshes. Unfortunately, their paper appeared very recently and we did not have time to test PolyCut against their method.

Finally, at the moment there is no direct relation between the compactness factor and mapping distortion, in the sense that it is up to the user to find the best combination via trial and error. Directly relating the compactness parameter to mapping distortion is a challenging open problem that may lead to interesting results in the future.

6.2 Curve-skeletons

In the context of curve-skeleton extraction understanding a shape is mainly a matter of detecting its logical components and the way they relate with each other. Creating a one-to-one relationship between skeleton parts and the logical components of the shape they convey is one of the most important properties for a skeleton (Section 3.2.1). Previous works in the field were already able to accomplish the task though they were too much focused on the geometric description of the object rather than on its appearance or functionality (Chapter 4). The perception-based paradigm introduced in Chapter 4 and the two algorithms discussed in Chapter 5 proved that the skeletonization problem can be successfully tackled from a new point of view, in which the primitives used to describe a shape are less important than the idea of the shape itself. Focusing on appearance rather than on primitives enables algorithms to understand shapes at a higher level of abstraction, at which factors like noise or missing data do not affect the results of the computation. Moreover, being these approaches based on silhouettes, they can be applied to any type of shape representation that can be rendered to a screen, regardless the type of primitives used to describe its shape.

The work described in this thesis has been of inspiration for subsequent research in the field [KJT13,SJT14]. In [KJT13], Kusta *et al.* improved over the naive algorithm (Section 5.1) by using depth information to reduce the amount of false matchings, generating cleaner point clouds that can be processed both easily and faster. Another approach that mixes contour analysis and depth information is [GSP12], where Guggeri *et al.* proposed a novel algorithm for shape reconstruction from unoriented point clouds. Moreover, in [Gug12] Guggeri coined the term Perceptual Shape Analysis (PSA) and proposed a set of tools aimed at solving classical geometry processing problems, like segmentation or skeletonization, by means of contour analysis. Other recent algorithms that rely on silhouettes are the projection-based segmentation proposed by Wang *et al.* in [WGW*13] and the motion-based segmentation proposed by Marras *et al.* in [MBH*12], to name a few.

6.2.1 Future work

The main flaws of the naive and accurate algorithms (Chapter 5) relate to either the poor quality or quantity of the considered contours (Section 5.1.7 and Section 5.2.7). Both these methods use very simple camera positioning schemas, mainly focusing on the skeleton extraction, as if silhouettes were almost an input rather than part of the algorithm itself. Indeed, the choice of the silhouettes is crucial for the quality of the result as contours provide the only description of the model available during the computation. Therefore, improving on the camera positioning system (for example using [SLF*11, DGB*14]) will automatically translate to an improvement in the quality of the output.

Provided that the perception-based algorithms are very fast at extracting skeletons and that silhouettes can be easily extracted from real images with state of the art image segmentation techniques, on-the-fly skeleton extraction from real images is a potential application for these type of approaches. A first example has been given in [YMG09], where Yoon *et al.* considered a set of real cameras to cast different silhouettes of a human, gather them together to compose a discrete volume, then apply a curve-skeleton extraction algorithm based on Gradient Vector Flow. The results they have been able to achieve show that it is possible to follow such a paradigm, and it is opinion of the author that the perception theory and the contour interpretation can be of much help in this scenario.

Finally, it is important to notice that there is a high number of different ways to combine cognitive sciences and geometry processing. The ideas expressed in this thesis are just a minor example. The human vision is a very complex system, and the analysis of contours is just a small fraction of the processes that the human bran undergoes while interpreting information coming from reality. Colors, shading, patterns and much more contribute to both the recognition and representation of real world objects into our brain. Marr himself did not focus only on contours but really tried to understand the human perception system on the whole [Mar10]. Combining all these ingredients into a more sophisticated framework for geometry processing would be an interesting research path to follow in future years.

6.3 Final remarks

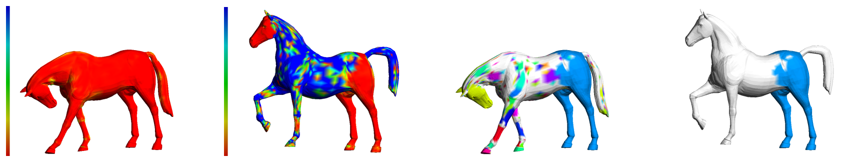
We published both methods and ideas exposed in Chapter 2 in [LVS*13] and presented them during the Siggraph Asia conference, held in Hong Kong on November 2013. There is also a US patent pending for the PolyCut algorithm (application number 61/899765). The naive algorithm for skeleton computation presented in Section 5.1 has been published in [LGS12] and presented as invited paper during the Eurographics Symposium on Parallel Graphics and Visualization, held in Girona (Spain) on May 2013. The accurate algorithm (Section 5.2) has been published in [LS13a] and presented during Computer Graphics International 2013, held in Hannover (Germany) on June 2013. Appendix A contains a simple pose registration algorithm [LS13b] that we presented at the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, held in Plzen (Czech Republic) on June 2013. Finally, minor publications occurred during the Ph.D are [GLS10] and [BLS11].

Appendix A

Registration of different poses of animated shapes

Objects are often given in arbitrary position, orientation and scale in space. *Registration* is the process of finding the geometric transformation which brings different sets of data into a congruent coordinate system. Registration finds its application in several fields, like medicine, where data acquired with different modalities are aligned for joint analysis [HBHH01, YLLY08, THEB*09], image mosaicking [KKHM99], where more images are merged seamlessly to produce a single snapshot, creation of super-resolution images [AF08], computer vision [JFS03], visualization [AFT*04], shape matching and retrieval [CVB08] and segmentation [AKM*06].

This appendix will focus on a particular aspect of registration: the goal is to find the best alignment between two (or more) triangle meshes having the same connectivity and representing the same object in different poses. This is quite important for algorithms that deal with multiple poses, because usually they need them to be aligned. For example, in [MBH*12] Marras and colleagues considered a set of poses of the same object as seen from different points of view, in order to find the rigid parts of it and calculate its *motion-based segmentation.* As all the computations rely on the silhouettes the alignment of the shapes is fundamental to properly catch the parts involved in the movement and discard the static ones. Pose registration is also a fundamental building-block of many shape interpolation techniques. Given two poses, for example one may want to generate the complete sequence which describes the movement bringing from one pose to the other [Mar12, MCH13]; or, given a set of simple poses, summarize them into a more articulated one [WDAH10]. To accomplish this task a pre-alignment of the shapes may help at achieving better results.



(a) $\widehat{k}_G$ *deviation* **(b)** *low-pass filter* **(c)** *Candidate ROIs* **(d)** *Selected ROI*

Figure A.1: *Region of interest selection. From left to right: Gaussian curvature displacements between the considered poses; Gaussian curvature displacements after the cutting-off of the* 20% *most affected vertices; all the candidate ROIs and, finally, the ROI selected to align the poses.*

The main question to answer to is: when are two poses said to be *well aligned*? Intuitively, a good alignment scheme should take into account only the parts of the shapes which are not involved in the movement suggested by the considered poses, leaving aside all the parts for which, because of the change of pose, a perfect alignment does not exist any longer. However, this is not sufficient. The horses in Figure A.1 expose more patches along their surface, each one of them being a valid candidate proxy for registration. However, different patches would require different registrations, therefore, only one should be considered. Section A.2 will introduce a simple pose registration method aimed at detecting the best proxy for surface registration. Details about the algorithm will be discussed in Section A.3 and Section A.4 while results will be presented in Section A.5.

A.1 Related works

Indeed, there are a lot of good registration algorithms in literature, but none of them is designed to face this particular instance of the problem, in which the alignment has to be *partial* and point correspondences are known a priori. Global methods, like the ICP algorithms [BM92, YM92] are not suitable to accomplish this task. Firstly, because they are *global*, that is, they use all the available data to estimate the alignment scheme; secondly, because ICP algorithms always monotonically converge to the nearest local minimum of a mean square distance metric, meaning that a coarse pre-alignment is mandatory to avoid to fall in the wrong local minima. Recently, many other registration algorithms have been proposed to the scientific community. In [AMCO08] Aiger and colleagues proposed a randomized alignment scheme which is based on approximately coplanar 4-points sets. Their algorithm is fast and resilient to noise. However, tests with the implementation provided in [Mes] showed that the results are not satisfactory for pose registration, mostly because this method is designed to align range maps, which usually expose a higher percentage of overlapping (often more than 40%). Since poses may be strongly different each other, algorithms are asked to be able to perform well in presence of lower overlapping percentages. Other algorithms are based on global quantities, such as Principal Component Analysis or centroids (e.g. [CVB08]). They are not suitable as well, because a global descriptor computed over a pose bears an arbitrary relationship with the same descriptor computed over a different pose of the same shape.

A.2 Overview

The registration method proposed here can align pairs of shapes represented by two triangle meshes, $\mathcal{M} = (V, K)$ and $\mathcal{M}' = (V', K)$, with K being the mesh connectivity and $V = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$ and $V' = \{\mathbf{v}'_1, ..., \mathbf{v}'_n\}$ the position of the vertices in $\mathbb{R}^3$. The algorithm is composed by two steps. The goal of the first part (Section A.3) is the definition of a region of interest (ROI), that is, a subset of vertices that will be used to guide the registration process and align the poses. To do so a simplified version of the largest common point set problem (LCP) [AH94,ATT98] with bottleneck matching metric [EIK01] will be solved. In the second part (Section A.4), the rigid transformation that aligns the ROIs will be computed. As a perfect alignment may not exist, the algorithm is designed to compute the best rigid movement in the least squares sense.

A.3 Region of Interest

The selection of the candidate region of interest is based on the discrete Gaussian curvature, which is one of the most familiar of all local shape descriptors (another example of applications involving Gaussian curvature is given in [GL09]). Let $V_{\kappa_G} \subseteq V$ be the subset of vertices having similar Gaussian curvature in both meshes, up to a tolerance threshold ϵ

$$V_{\kappa_G} = \left\{ \mathbf{v}_i \in V \;\middle|\; |\,\kappa_G(\mathbf{v}_i) - \kappa_G(\mathbf{v}'_i)\,| \leq \epsilon \;, \;\; \mathbf{v}'_i \in V' \right\} \; . \qquad \text{(A.1)}$$

The distribution of the points in $V_{k_G} \subseteq V$ among the surface identifies what areas have common behaviour in both poses, therefore, they constitute a perfect set of seeds for a flooding process aimed at detecting the set of candidate ROIs. However, a simple flooding process that expands over adjacent vertices as long as they belong to V_{k_G} may lead to ill-shaped ROIs, composed by a chain of adjacent vertices spanning all over the surface. To improve the compactness, each vertex in a ROI is constricted to expand to its whole one-ring. Let $\mathbf{v}_i$ be a vertex belonging to a candidate ROI, and $N(i)$ the set of vertices sharing an edge with it. If the average Gaussian curvature deviation

$$\overline{\Delta\kappa_G}\,(\mathbf{v}_i) = \frac{\sum_{j\in N(i)} \mid \kappa_G(\mathbf{v}_j) - \kappa_G(\mathbf{v}'_j) \mid}{\#N(i)} \tag{A.2}$$

is lower than a predefined threshold σ, the candidate ROI expands over $N(i)$. Iterating this process until convergence for every seed point will generate the final set of candidate regions of interest to be considered by further steps of the algorithm (Figure A.1c).

The ROI selection depends on the threshold σ. Unfortunately, curvature deviation is ill-distributed over the surface: few vertices have a huge displacement, thus flattening the rest of the distribution. In this scenario the automatic research of a good σ value can be very difficult. As can be noticed in Figure A.1a vertices with high curvature deviation are located in the areas involved in the change of pose, thus, they are meaningless for the registration problem. Consequently, the research can be restricted to a subset of vertices with small curvature deviation. Let $\mathcal{LD}$ be the indexes of the vertices less affected by curvature deviation (80% of the total number of vertices); the value of the threshold σ is defined as follows

$$\sigma = 0.2 \underset{i\,\in\,\mathcal{LD}}{\arg\max} \left| \hat{\kappa}_G(v_i) - \hat{\kappa}_G(v'_i) \right| . \tag{A.3}$$

Cutting-off of the vertices with greater deviation makes the detectin of good candidate ROIs easier (Figure A.1b).

Among the candidate ROIs, only the largest one has to be considered for registration. The surface area covered by a candidate ROI is defined as the sum of the areas associated to each vertex in it, where the per-vertex area is defined as

$$\mathcal{A}_{\mathsf{Voronoi}} = \frac{1}{8} \sum_{j\in N(\mathbf{v}_i)} (\cot\alpha_{\,ij} + \cot\beta_{\,ij}) \parallel \mathbf{v}_i - \mathbf{v}_j \parallel , \tag{A.4}$$

with $\mathbf{v}_i$ being a ROI vertex, $\mathbf{v}_j$ its neighbour and $\alpha_{\,ij}$ and $\beta_{\,ij}$ the angles measured in the opposite corners with respect to the edge joining $\mathbf{v}_i$ and $\mathbf{v}_j$. This area proved to be sufficient in all the experiments. However, the approach does not depend on the particular choice of $\mathcal{A}$. Mixed area, for instance, would accommodate obtuse triangles and its application is straightforward [MDSB03]. Computing the area with a local formula depending on the mesh connectivity is important, as makes the measure of the patch's size tessellation independent.

A.4 Alignment

Given a region of interest $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ in the first pose, the goal is to find the rigid movement that optimally aligns $\mathcal{P}$ and its dual in the second pose $\mathcal{P}'$. The optimal transformation $T : \mathcal{P} \to \mathcal{P}'$ has the form

$$T = \begin{bmatrix} R & \mathbf{t} \\ 0^T & 1 \end{bmatrix} , \tag{A.5}$$

where $R \in \mathbb{R}^{3\times 3}$ is a pure rotation matrix (i.e. $R^T R = I$) and $\mathbf{t} \in \mathbb{R}^3$ is a translation. If $\mathcal{P}$ and $\mathcal{P}'$ were strictly congruent, the residual $\sum_{i=1}^n \|\mathbf{p}'_i - T(\mathbf{p}_i)\|$ would be zero. Unfortunately, congruence is very unlikely to happen; in the common scenario a perfect alignment does not exists, therefore the best approximating rotation and translation that fit $\mathcal{P}$ and $\mathcal{P}'$ in the least square sense must be found, that is

$$\left(\tilde{R}, \tilde{\mathbf{t}}\right) = \arg\min_{R,\mathbf{t}} \sum_{i=1}^{n} \| \left(R\, \mathbf{p}_i + \mathbf{t}\right) - \mathbf{p}'_i \|^2 . \tag{A.6}$$

This is a very well known problem in literature and there are several existing algorithms to solve it. A comparison between four of them can be found in [ELF97] while a weighted instance of the problem has been studied by Sorkine and Alexa in [SA07]. The best translation can be found by taking the derivative of Equation A.6 w.r.t. $\mathbf{t}$ and searching for its roots, obtaining

$$\tilde{\mathbf{t}} = \overline{\mathbf{p}}' - R\,\overline{\mathbf{p}} , \tag{A.7}$$

with

$$\overline{\mathbf{p}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i \qquad\qquad \overline{\mathbf{p}}' = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}'_i . \tag{A.8}$$

In other words the optimal translation $\tilde{\mathbf{t}}$ maps the transformed centroid of $\mathcal{P}$ in the centroid of its dual $\mathcal{P}'$. To find the best rotation one can now consider the centered vectors

$$\mathbf{x}_i = \mathbf{p}_i - \overline{\mathbf{p}}_i \qquad\qquad \mathbf{y}_i = \mathbf{p}'_i - \overline{\mathbf{p}}'_i \tag{A.9}$$

$i = 1, \ldots, n$, and the Singular Value Decomposition (SVD) of their 3×3 covariance matrix XY^T (X and Y are two $3 \times n$ matrices containing all the centered vectors ordered by column) $XY^T = U\Sigma V^T$. The optimal rotation is therefore

$$R = V \begin{bmatrix} 1 & & \\ & 1 & \\ & & \det(VU^T) \end{bmatrix} U^T. \tag{A.10}$$

The last term in the diagonal matrix is setted to det $\left(VU^T\right)$ instead of 1 in order to avoid reflections. A formal and clear demonstration of the whole minimization process can be found in [Sor09].

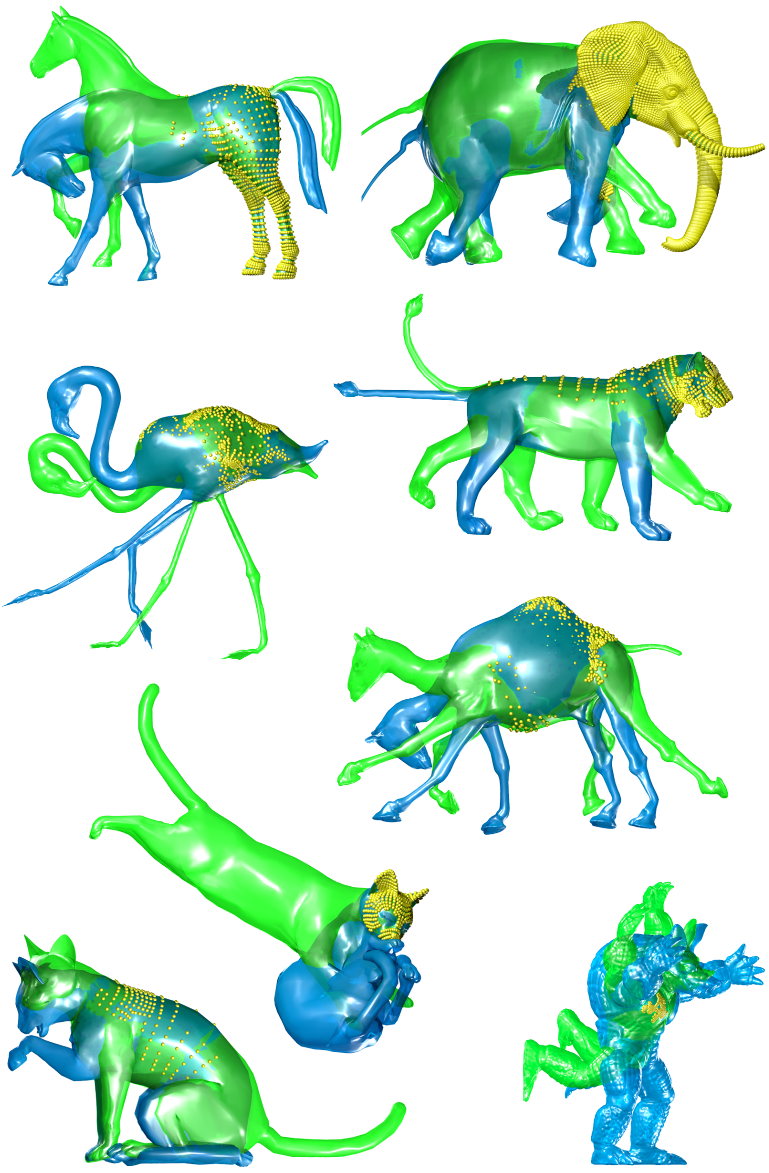


Figure A.2: *A mosaic of poses of different shapes aligned using the pose registration algorithm (in yellow the region of interest (ROI) used to guide registration process).*

Model	**Vertices** (tot/roi)	**Curvature** (ms)	**ROI** (ms)	**Alignment** (ms)	**Total** (ms)
Armadillos	165,954 / 3,541	307	126	6	439
Elephants	42,321 / 19,339	189	98	28	315
Flamingos	26,907 / 1,092	120	44	10	174
Camels	21,887 / 376	72	34	25	131
Horses	8,431 / 2,491	35	27	11	73
Cats 1	7,207 / 965	32	23	4	59
Cats 2	7,207 / 544	32	22	2	56
Lions	5,000 / 1,751	24	15	8	47

Table A.1: *Running times, in milliseconds, for the pose registration algorithm. Experiments were run on a iMac equipped with 2.66GHz Intel Core 2 Duo and 4GB RAM, using a single core. From left to right: number of vertices for both the triangle mesh and the region of interest, time needed to compute respectively the Gaussian curvature, the region of interest and the least square registration. On the rightmost column, the total time needed to register the poses.*

A.5 Results and discussion

Pose registration has been implemented in C++, using the VCG Library [VCG] for the manipulation of geometric data structures, and the Eigen library [Eig] for the numerical computations. For the testing, the dataset provided by Sumner and Popović in their deformation transfer for triangle meshes's web page [SP04] has been used. For each shape, many different poses have been considered: from a minimum of 8 poses (horse sequence), to a maximum of 47 poses (elephant gallop sequence). Tests were organized as follows: for each couple of poses the algorithm ran three times, each time applying a random rigid movement to the shapes prior to registration, so as to put them in general position in the space. For each couple of poses, the same numerical results occurred in the three tests, emphasizing the robustness of the Gaussian curvature-based ROI detection with respect to isometries. Table A.1 reports timings for each registration shown in this paper. As can be noticed, the most time consuming task is the registration step, consequently, the complexity of the whole algorithm depends on the size of the ROI, rather than the complexity of the whole mesh (i.e. the number of vertices). A gallery of results can be found in Figure A.2.

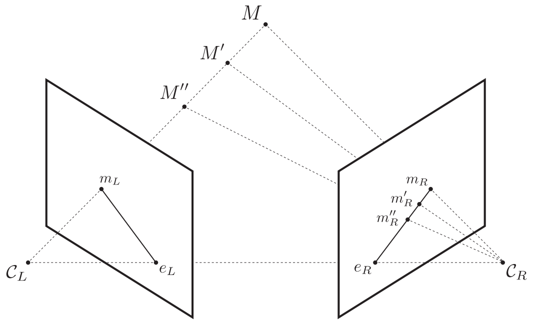
Limitations The proposed algorithm performed well with all the poses tested on it. It has, however, some limitations, in the sense that there are no guarantees that the registration provided will be a *good* one, that is, the one the user expected. In particular, if the model contains more parts that move rigidly and cover more or less the same area the choice of the registration proxy may be unstable. Furthermore, in case the selected ROI lies in a peripheral area of the shape (e.g., a foot, a head or a hand) the result may seem unnatural. The definition of *natural registration* is somehow related to the human perception, it is the kind of registration that a human imagine while looking at two different poses of the same object. This is something difficult to model mathematically. Something that require a deeper level of understanding of the shapes involved in the registration process.

A.6 Conclusions

Pose registration is the process of finding the best possible alignment among different meshes representing the same shape in different positions. Since poses can be strongly different each other the best alignment method should be able to work only with local patches of the surface, that is the regions of the shape having the same properties (mainly curvature) in the given poses. In this appendix a new, non-iterative, pose registration scheme has been presented. The method employs Gaussian curvature as local shape invariant, it is fast, robust, and accurate in the most relevant cases of pose registration: interpolation among different poses and modelling.

Appendix B

Epipolar Geometry



This appendix will provide some technical information about epipolar geometry and, in particular, image rectification. These basic computer vision techniques are widely used by the skeletonization algorithms exposed in Chaper 5.

Epipolar geometry is the geometry of *stereo vision*. It describes the geometric relationships that exists between two planar projections of an object lying in the 3D space. Let $\mathcal{C}_L$ and $\mathcal{C}_R$ be two distinct projective cameras, e_L be the image of $\mathcal{C}_R$ as taken from $\mathcal{C}_L$, and e_R the image of $\mathcal{C}_L$ as taken from $\mathcal{C}_R$. Then, e_L and e_R are called *epipoles* of the stereo system. Considering a point $M \in \mathbb{R}^3$ which is imaged by the cameras respectively in m_L and m_R. The line joining M with $\mathcal{C}_L$ is trivially seen as a point by $\mathcal{C}_L$ itself while it is seen as the line $e_R - m_R$ by $\mathcal{C}_R$. Vice-versa, the line joining M with $\mathcal{C}_R$ is seen by $\mathcal{C}_L$ as the line $e_L - m_L$. Such lines are called *epipolar lines* and are defined by the intersection between the image planes of the stereo system and the *epipolar plane*, that is, the plane containing $\mathcal{C}_L$, $\mathcal{C}_R$ and the point M. All the epipolar planes and lines intersect the epipoles,

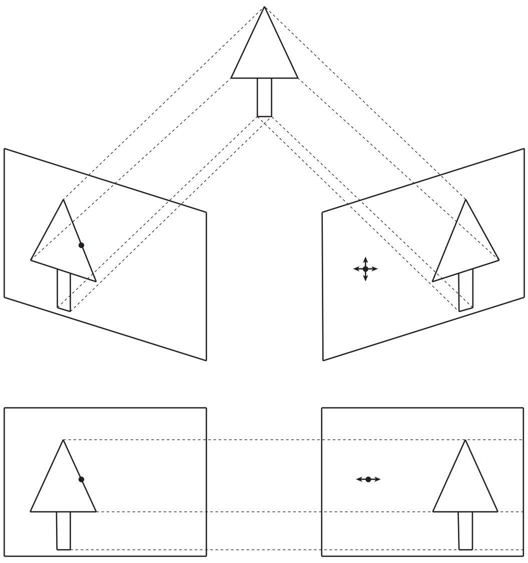


Figure B.1: *In the general case, finding point-wise correspondences between two images is a two dimensional search (top). Epipolar geometry turns the problem into a mono-dimensional search. Furthermore, when the images are rectified (bottom), the space to inspect to find correspondences equals the horizontal scanlines of the images.*

regardless of where M is located in the space. These relations are central in the correspondence problem; that is, the problem to couple different images of the same 3D point in order to discover its spacial coordinates. Finding correspondences usually requires a two-dimensional research, although, taking advantage of the epipolar geometry, the space of research reduces to one dimension. In fact, if one image point is known, for example m_L, the epipolar line $e_R - m_R$ is also known, and the projection m_L is forced to lie in it. Furthermore, if both images m_L and m_R are known, M can be calculated as the intersection between the lines $\mathcal{C}_L - m_L$ and $\mathcal{C}_L R - m_R$. This process is referred to as *triangulation*.

When the image planes coincide the epipolar geometry becomes even simpler. In this case epipolar lines coincide as well, therefore, they can be aligned with the horizontal scanlines of the images. In this scenario, the epipolar constraint forces two corresponding points to lie exactly on the same horizontal scanline, making the correspondence problem easier (Figure B.1). This particular process is referred to as *image rectification* and can be achieved by using for example affine cameras instead of projective cameras.

Appendix C

Curriculum Vitae

PERSONAL DATA

Born
11 August 1983, Cagliari (Italy)

Work address
Dipartimento di Matematica e Informatica
Universitá degli Studi di Cagliari
Via Ospedale, 72
09124 Cagliari - Italy

Tel: +39 347 74 00 604
Email: marco.livesu@gmail.com
Website: `https://sites.google.com/site/marcolivesu`

EDUCATION

Master of Computer Science
Universitá degli Studi di Cagliari, Italy
September, 2010
Grade: 110/110 cum laude
Thesis: Automatic 3D Skeletonization Using Multiple Views
Advisor: Prof. Riccardo Scateni

Bachelor of Computer Science
Universitá degli Studi di Cagliari, Italy
July, 2008
Grade: 110/110 cum laude
Thesis: Digital Terrain Models Construction Using Delaunay Triangulations
Advisor: Prof. Riccardo Scateni

SUMMER SCHOOLS

SGP Graduate School on Geometry Processing 2013
July 1-2, 2013, Genoa, Italy

SGP Graduate School on Geometry Processing 2011
July 18-19, 2011, Lausanne, Switzerland

CONFERENCE TALKS

PolyCut: Monotone Graph-Cuts for PolyCube Base-Complex Construction
ACM Siggraph Asia, Hong Kong
November, 2013

Rigid registration of different poses of animated shapes
Winter School of Computer Graphics, Plzen, Czech Republic
June, 2013

Extracting curve-skeletons from digital shapes using occluding contours
Computer Graphics International, Hannover, Germany
June, 2013

Reconstructing the Curve-Skeleton of 3D Shapes Using the Visual Hull
Eurographics Symposium on Parallel Graphics and Visualization, Girona, Spain
May, 2013

ATTENDED CONFERENCES

6th **ACM Siggraph Asia**
November 2013 - Hong Kong

11th **Symposium on Geometry Processing**
July 2013 - Genoa, Italy

Winter School of Computer Graphics
June 2013 - Plzen, Czech Republic

30th **Computer Graphics International**
June 2013 - Hannover, Germany

34th **Eurographics**
May 2013 - Girona, Spain

13th **Eurographics Symposium on Parallel Graphics and Visualization**
May 2013 - Girona, Spain

33th **Eurographics**
May 2012 - Cagliari, Italy

9th **Eurographics Italian Chapter**
November 2011 - Salerno, Italy

9th **Symposium on Geometry Processing**
July 2011 - Lausanne, Switzerland

8th **Eurographics Italian Chapter**
November 2010 - Genoa, Italy

PROFESSIONAL ACTIVITIES

Reviewer
Computer & Graphics (2013)

Visiting researcher
at University of British Columbia, UBC
from September, 2012 to April, 2013
under the supervision of Prof. Alla Sheffer
Vancouver, BC, Canada

Teaching assistant
Computer Architectures (Fall 2010, Fall 2011)
Data Structures (Fall 2010, Fall 2011, Fall 2013)
Universitá degli Studi di Cagliari, Italy

PUBLICATIONS

PolyCut: Monotone Graph-Cuts for PolyCube Base-Complex Construction
M. Livesu, N. Vining, A. Sheffer, J. Gregson, R. Scateni
ACM Transactions on Graphics (Siggraph Asia 2013, Hong Kong, China)

Extracting curve-skeletons from digital shapes using occluding contours
M. Livesu, R. Scateni
The Visual Computer (CGI 2013, Hannover, Germany)

Rigid registration of different poses of animated shapes
M. Livesu, R. Scateni
Journal of WSCG (WSCG 2013, Plzen, Czech Republic)

Reconstructing the Curve-Skeleton of 3D Shapes Using the Visual Hull
M. Livesu, F. Guggeri, R. Scateni
IEEE Transactions on Visualization and Computer Graphics, 201

Gestural Interaction for Robot Motion Control
G. Broccia, M. Livesu, R. Scateni
9th Eurographics Italian Chapter, 2011 (Salerno, Italy)

Tools and Applications for Teaching and Research in Computer Graphics
F. Guggeri, M. Livesu, R. Scateni
8th Eurographics Italian Chapter, 2010 (Genova, Italy)

PATENTS

Methods and Systems for Generating PolyCube Segmentations from Input Meshes of Objects
US Application Number 61/899765

Bibliography

[AF08] Arican Z., Frossard P.: Super-resolution from unregistered omnidirectional images. In *19th International Conference on Pattern Recognition, 2008 (ICPR 2008)* (2008), IEEE, pp. 1–4.

[AFT*04] Allen P., Feiner S., Troccoli A., Benko H., Ishak E., Smith B.: Seeing into the Past: Creating a 3D Modeling Pipeline for Archaeological Visualization. In *Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium* (Washington, DC, USA, 2004), 3DPVT '04, IEEE Computer Society, pp. 751–758.

[AH94] Akutsu T., Halldórsson M. M.: On the approximation of largest common subtrees and largest common point sets. In *Algorithms and Computation*, Du D.-Z., Zhang X.-S., (Eds.), vol. 834 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1994, pp. 405–413.

[AJWB03] Aylward S. R., Jomier J., Weeks S., Bullitt E.: Registration and analysis of vascular images. *International Journal of Computer Vision 55*, 2-3 (2003), 123–138.

[AKM*06] Attene M., Katz S., Mortara M., Patane G., Spagnuolo M., Tal A.: Mesh Segmentation - A Comparative Study. In *IEEE International Conference on Shape Modeling and Applications* (june 2006), SMI 2006, pp. 7–7.

[AL13] Aigerman N., Lipman Y.: Injective and bounded distortion mappings in 3d. *ACM Transactions on Graphics 32*, 4 (2013), 106.

[AMCO08] Aiger D., Mitra N. J., Cohen-Or D.: 4-points congruent sets for robust pairwise surface registration. In *ACM*

SIGGRAPH 2008 papers (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 85:1–85:10.

[ATC*08] Au O. K.-C., Tai C.-L., Chu H.-K., Cohen-Or D., Lee T.-Y.: Skeleton extraction by mesh contraction. *ACM Transactions on Graphics 27*, 3 (2008), 44.

[ATT98] Akutsu T., Tamaki H., Tokuyama T.: Distribution of distances and triangles in a point set and algorithms for computing the largest common point sets. *Discrete & Computational Geometry 20*, 3 (1998), 307–331.

[BA84] Brady M., Asada H.: Smoothed local symmetries and their implementation. *The International Journal of Robotics Research 3*, 3 (1984), 36–61.

[BBK06] Bronstein A. M., Bronstein M. M., Kimmel R.: Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Sciences of the United States of America 103*, 5 (2006), 1168–1172.

[BGG85] Bruce J., Giblin P. J., Gibson C.: Symmetry sets. *Proceedings of the Royal Society of Edinburgh 101*, A (1985), 163–186.

[Bin71] Binford T. O.: Visual perception by computer. In *Proceedings of the IEEE Conference on Systems and Control (Miami, FL)* (1971).

[BK04] Boykov Y., Kolmogorov V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 9 (2004), 1124–1137.

[BLP97] Bullitt E., Liu A., Pizer S.: Three-dimensional reconstruction of curves from pairs of projection views in the presence of error. i. algorithms. *Medical Physics 24* (1997), 1671.

[BLP*13] Bommes D., Lévy B., Pietroni N., Puppo E., Silva C., Tarini M., Zorin D.: Quad-mesh generation and processing: A survey. *Computer Graphics Forum 32*, 6 (2013), 51–76.

[BLS11] Broccia G., Livesu M., Scateni R.: Gestural interaction for robot motion control. In *Eurographics Italian Chapter Conference 2011* (2011), The Eurographics Association, pp. 61–66.

[Blu67] Blum H.: A transformation for extracting new descriptors of shape. *Models for the perception of speech and visual form 19*, 5 (1967), 362–380.

[BM92] Besl P. J., McKay N. D.: A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14* (February 1992), 239–256.

[BMW12] Bærentzen J. A., Misztal M. K., Wełnicka K.: Converting skeletal structures to quad dominant meshes. *Computers & Graphics 36*, 5 (2012), 555–561.

[Bor96] Borgefors G.: On digital distance transforms in three dimensions. *Computer Vision and Image Understanding 64*, 3 (1996), 368–376.

[BP07] Baran I., Popović J.: Automatic rigging and animation of 3d characters. *ACM Transactions on Graphics 26*, 3 (July 2007).

[BPGK06] Botsch M., Pauly M., Gross M., Kobbelt L.: Primo: Coupled prisms for intuitive surface modeling. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 11–20.

[BVI91] Bennis C., Vézien J.-M., Iglésias G.: Piecewise surface flattening for non-distorted texture mapping. *ACM SIGGRAPH computer graphics 25*, 4 (1991), 237–246.

[BVZ01] Boykov Y., Veksler O., Zabih R.: Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 11 (2001), 1222–1239.

[CBK12] Campen M., Bommes D., Kobbelt L.: Dual loops meshing: quality quad layouts on manifolds. *ACM Transactions on Graphics 31*, 4 (2012), 110.

[CCM97] Choi H. I., Choi S. W., Moon H. P.: Mathematical theory of medial axis transform. *Pacific Journal of Mathematics 181*, 1 (1997), 57–88.

[CL05] Chazal F., Lieutier A.: The λ-medial axis. *Graphical Models 67*, 4 (2005), 304–331.

[CSM07] Cornea N. D., Silver D., Min P.: Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (2007), 530–548.

[CSYB05] Cornea N., Silver D., Yuan X., Balasubramanian R.: Computing hierarchical curve-skeletons of 3D objects. *The Visual Computer 21*, 11 (October 2005), 945–955.

[CTO*10] Cao J., Tagliasacchi A., Olson M., Zhang H., Su Z.: Point cloud skeletons via laplacian based contraction. In *Shape Modeling International Conference (SMI)* (2010), IEEE, pp. 187–197.

[CVB08] Chaouch M., Verroust-Blondet A.: A novel method for alignment of 3d models. In *IEEE International Conference on Shape Modeling and Applications* (2008), SMI 2008, IEEE, pp. 187–195.

[DGB*14] Di Benedetto M., Ganovelli F., Balsa Rodriguez M., Jaspe Villanueva A., Scopigno R., Gobbetti E.: Exploremaps: Efficient construction and ubiquitous exploration of panoramic view graphs of complex 3d environments. *Computer Graphics Forum 33*, 2 (2014). (to appear).

[DMSB99] Desbrun M., Meyer M., Schröder P., Barr A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 317–324.

[DS06] Dey T. K., Sun J.: Defining and computing curve-skeletons with medial geodesic function. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (2006), Eurographics Association, pp. 143–152.

[Eig] Eigen:. `http://eigen.tuxfamily.org/`.

[EIK01] Efrat A., Itai A., Katz M. J.: Geometry helps in bottleneck matching and related problems. *Algorithmica 31*, 1 (2001), 1–28.

[ELF97] Eggert D. W., Lorusso A., Fisher R. B.: Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications 9*, 5-6 (1997), 272–290.

[EM10] Eppstein D., Mumford E.: Steinitz theorems for orthogonal polyhedra. In *Proceedings of the Twenty-sixth Annual Symposium on Computational Geometry* (New York, NY, USA, 2010), SoCG '10, ACM, pp. 429–438.

[FJFS05] Fan Z., Jin X., Feng J., Sun H.: Mesh morphing using polycube-based cross-parameterization. *Computer Animation and Virtual Worlds 16*, 3-4 (2005), 499–508.

[FPAB04] Fridman Y., Pizer S. M., Aylward S., Bullitt E.: Extracting branching tubular object geometry via cores. *Medical Image Analysis 8*, 3 (2004), 169–176.

[FSD07] Friedel I., Schröder P., Desbrun M.: Unconstrained spherical parameterization. *Journal of Graphics, GPU, and Game Tools 12*, 1 (2007), 17–26.

[GGH02] Gu X., Gortler S. J., Hoppe H.: Geometry images. *ACM Transactions on Graphics 21*, 3 (2002), 355–361.

[GGS03] Gotsman C., Gu X., Sheffer A.: Fundamentals of spherical parameterization for 3d meshes. *ACM Transactions on Graphics 22*, 3 (2003), 358–363.

[GHQ06] Gu X., He Y., Qin H.: Manifold splines. *Graphical Models 68*, 3 (2006), 237–254.

[GK04] Giblin P., Kimia B. B.: A formal classification of 3d medial axis points and their local geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 2 (2004), 238–251.

[GL09] Guo K., Li M.: A Novel Shape Descriptor: Gaussian Curvature Moment Invariants. In *Proceedings of the 2009 First IEEE International Conference on Information Science and Engineering* (Washington, DC, USA, 2009), ICISE '09, IEEE Computer Society, pp. 1087–1090.

[GLS10] Guggeri F., Livesu M., Scateni R.: Tools and applications for teaching and research in computer graphics. In *Eurographics Italian Chapter Conference 2010* (2010), The Eurographics Association, pp. 147–152.

[GMPW09] Giesen J., Miklos B., Pauly M., Wormser C.: The scale axis transform. In *Proceedings of the 25th annual Symposium on Computational geometry* (2009), ACM, pp. 106–115.

[Gra] Graphite: Alice Project-team. `http://alice.loria.fr/software/graphite/`.

[GS99] Gagvani N., Silver D.: Parameter-controlled volume thinning. *Graphical Models and Image Processing 61*, 3 (1999), 149–164.

[GSMCO09] Gal R., Sorkine O., Mitra N. J., Cohen-Or D.: iwires: an analyze-and-edit approach to shape manipulation. *ACM Transactions on Graphics 28*, 3 (2009), 33.

[GSP12] Guggeri F., Scateni R., Pajarola R.: Shape reconstruction from raw point clouds using depth carving. In *Eurographics 2012-Short Papers* (2012), The Eurographics Association, pp. 33–36.

[GSZ11] Gregson J., Sheffer A., Zhang E.: All-hex mesh generation via volumetric polycube deformation. *Computer Graphics Forum 30*, 5 (2011), 1407–1416.

[Gug12] Guggeri F.: *Perceptual Shape Analysis: approaching geometric problems with elements of perception psychology.* PhD thesis, University of Cagliari, 2012.

[GVSS00] Guskov I., Vidimče K., Sweldens W., Schröder P.: Normal meshes. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 95–102.

[HBHH01] Hill D. L., Batchelor P. G., Holden M., Hawkes D. J.: Medical image registration. *Physics in medicine and biology 46*, 3 (2001), R1.

[HF05] Hassouna M. S., Farag A. A.: Robust centerline extraction framework using level sets. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (2005), vol. 1, IEEE, pp. 458–465.

[HF06] Hormann K., Floater M. S.: Mean value coordinates for arbitrary planar polygons. *ACM Transactions on Graphics 25*, 4 (2006), 1424–1441.

[HJS*14] Huang J., Jiang T., Shi Z., Tong Y., Bao H., Desbrun M.: c1-based construction of polycube maps from complex shapes. *ACM Transactions on Graphics* (2014). (to appear).

[HLS07] Hormann K., Lévy B., Sheffer A.: Mesh parameterization: Theory and practice. In *ACM SIGGRAPH 2007 Courses* (2007), SIGGRAPH '07, ACM.

[HPW05] Hildebrandt K., Polthier K., Wardetzky M.: Smooth feature lines on surface meshes. In *Proceedings of* *the Third Eurographics Symposium on Geometry Processing* (2005), SGP '05, Eurographics Association.

[HS04] Hoos H. H., Stützle T. G.: *Stochastic Local Search: Foundations and Applications.* Morgan Kaufmann Publishers Inc., 2004.

[HWCO*13] Huang H., Wu S., Cohen-Or D., Gong M., Zhang H., Li G., Chen B.: L1-medial skeleton of point cloud. *ACM Transactions on Graphics 32*, 4 (July 2013), 65:1–65:8.

[HWFQ09] He Y., Wang H., Fu C.-W., Qin H.: A divide-and-conquer approach for automatic polycube map construction. *Computers & Graphics 33*, 3 (2009), 369 – 380.

[HXH10] Han S., Xia J., He Y.: Hexahedral shell mesh construction via volumetric polycube map. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2010), SPM '10, ACM, pp. 127–136.

[JFS03] Jin H., Favaro P., Soatto S.: A semi-direct approach to structure from motion. *The Visual Computer 19*, 6 (2003), 377–394.

[JLW10] Ji Z., Liu L., Wang Y.: B-mesh: A modeling system for base meshes of 3d articulated shapes. *Computer Graphics Forum 29*, 7 (2010), 2169–2177.

[JMD*07] Joshi P., Meyer M., DeRose T., Green B., Sanocki T.: Harmonic coordinates for character articulation. *ACM Transactions on Graphics 26*, 3 (2007), 71.

[JSW05] Ju T., Schaefer S., Warren J.: Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics 24*, 3 (2005), 561–566.

[KCATCO*10] Kin-Chung Au O., Tai C.-L., Cohen-Or D., Zheng Y., Fu H.: Electors voting for fast automatic shape correspondence. *Computer Graphics Forum 29*, 2 (2010), 645–654.

[KJT13] Kustra J., Jalba A., Telea A.: Probabilistic view-based 3d curve skeleton computation on the gpu. In *VISAPP 2013 - Proceedings of the International Conference on Computer Vision Theory and Applications, Volume 2, Barcelona, Spain, 21-24 February, 2013* (2013), pp. 237–246.

[KKHM99] Kourogi M., Kurata T., Hoshino J., Muraoka Y.: Real-time image mosaicing from a video sequence. In *Proceedings of the International Conference on Image Processing, 1999 (ICIP 99)* (1999), vol. 4, pp. 133–137 vol.4.

[KLS03] Khodakovsky A., Litke N., Schröder P.: Globally smooth parameterizations with low distortion. *ACM Transactions on Graphics 22*, 3 (July 2003), 350–357.

[KS04] Kraevoy V., Sheffer A.: Cross-parameterization and compatible remeshing of 3d models. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 861–869.

[KZ04] Kolmogorov V., Zabin R.: What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 2 (2004), 147–159.

[Lau94] Laurentini A.: The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*, 2 (1994), 150–162.

[Lau95] Laurentini A.: How far 3d shapes can be understood from 2d silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, 2 (1995), 188–195.

[Lau97] Laurentini A.: How many 2d silhouettes does it take to reconstruct a 3d object? *Computer Vision and Image Understanding 67*, 1 (1997), 81–87.

[LC87] Lorensen W. E., Cline H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 4 (Aug. 1987), 163–169.

[LCF00] Lewis J. P., Cordner M., Fong N.: 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* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 165–172.

[LCLJ10] Liu L., Chambers E. W., Letscher D., Ju T.: A simple and robust thinning algorithm on cell complexes. *Computer Graphics Forum 29*, 7 (2010), 2253–2260.

[Lee98] Leek E.: Effects of stimulus orientation on the identification of common polyoriented objects. *Psychonomic bulletin & review 5*, 4 (1998), 650–658.

[Lee00] Lee I.-K.: Curve reconstruction from unorganized points. *Computer aided geometric design 17*, 2 (2000), 161–177.

[LGS12] Livesu M., Guggeri F., Scateni R.: Reconstructing the curve-skeletons of 3d shapes using the visual hull. *IEEE Transactions on Visualization and Computer Graphics 18*, 11 (2012), 1891–1901.

[LH05] Leordeanu M., Hebert M.: A spectral technique for correspondence problems using pairwise constraints. In *IEEE International Conference on Computer Vision (ICCV)* (2005), vol. 2, IEEE, pp. 1482–1489.

[LJFW08] Lin J., Jin X., Fan Z., Wang C.: Automatic polycube-maps. In *Advances in Geometric Modeling and Processing*, Chen F., Jüttler B., (Eds.), vol. 4975 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 3–16.

[LKA06] Lien J.-M., Keyser J., Amato N. M.: Simultaneous shape decomposition and skeletonization. In *Proceedings of the 2006 ACM symposium on Solid and physical modeling* (2006), ACM, pp. 219–228.

[LL10] Lévy B., Liu Y.: Lp centroidal voronoi tessellation and its applications. *ACM Transactions on Graphics 29*, 4 (July 2010), 119:1–119:11.

[LLWQ10] Li B., Li X., Wang K., Qin H.: Generalized polycube trivariate splines. In *Shape Modeling International Conference (SMI)* (2010), pp. 261–265.

[LLWQ13] Li B., Li X., Wang K., Qin H.: Surface mesh to volumetric spline conversion with generalized polycubes. *IEEE Transactions on Visualization and Computer Graphics 19*, 9 (2013), 1539–1551.

[LLX*12] Li Y., Liu Y., Xu W., Wang W., Guo B.: All-hex meshing using singularity-restricted field. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 177:1–177:11.

[LQ12] Li B., Qin H.: Component-aware tensor-product trivariate splines of arbitrary topology. *Computers & Graphics 36*, 5 (2012), 329–340.

[LS13a] Livesu M., Scateni R.: Extracting curve-skeletons from digital shapes using occluding contours. *The Visual Computer 29*, 9 (2013), 907–916.

[LS13b] Livesu M., Scateni R.: Rigid registration of different poses of animated shapes. *Journal of WSCG 21*, 1 (2013), 1–9.

[LSS*98] Lee A. W. F., Sweldens W., Schröder P., Cowsar L., Dobkin D.: Maps: Multiresolution adaptive parameterization of surfaces. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 95–104.

[LVS*13] Livesu M., Vining N., Sheffer A., Gregson J., Scateni R.: Polycut: Monotone graph-cuts for polycube base-complex construction. *ACM Transactions on Graphics 32*, 6 (Nov. 2013), 171:1–171:12.

[LWTH01] Li X., Woon T. W., Tan T. S., Huang Z.: Decomposing polygon meshes for interactive applications. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (2001), ACM, pp. 35–42.

[LXW*10] Li X., Xu H., Wan S., Yin Z., Yu W.: Feature-aligned harmonic volumetric mapping using mfs. *Computers & Graphics 34*, 3 (2010), 242–251.

[Mar76] Marr D.: Early processing of visual information. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences 275*, 942 (1976), 483–519.

[Mar77] Marr D.: Analysis of occluding contour. *Proceedings of the Royal Society of London. Series B, Biological Sciences 197*, 1129 (1977), 441–475.

[Mar80] Marr D.: Visual information processing: The structure and creation of visual representations. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences 290*, 1038 (1980), 199–218.

[Mar10] Marr D.: *Vision: a Computational Investigation into the Human Representation and Processing of Visual Information.* The MIT Press, 2010.

[Mar12] Marras S.: *Perception and Motion: Use of Computer Vision to Solve Geometry Processing Problems.* PhD thesis, University of Cagliari, 2012.

[MBH*12] Marras S., Bronstein M. M., Hormann K., Scateni R., Scopigno R.: Motion-based mesh segmentation using augmented silhouettes. *Graphical Models 74*, 4 (2012), 164–172.

[MCH13] Marras S., Cashman T. J., Hormann K.: Efficient interpolation of articulated shapes using mixed shape spaces. *Computer Graphics Forum 32*, 8 (2013), 258–270.

[MCM*12] Martin T., Chen G., Musuvathy S., Cohen E., Hansen C.: Generalized swept mid-structure for polygonal models. *Computer Graphics Forum 31*, 2pt4 (2012), 805–814.

[MDSB03] Meyer M., Desbrun M., Schröder P., Barr A.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, Hege H.-C., Polthier K., (Eds.), Mathematics and Visualization. Springer Berlin Heidelberg, 2003, pp. 35–57.

[Mes] MeshLab: Visual Computing Lab (ISTI - CNR). `http://meshlab.sourceforge.net/`.

[MGP10] Miklos B., Giesen J., Pauly M.: Discrete scale axis representations for 3D geometry. *ACM Transactions on Graphics 29* (July 2010), 101:1–101:10.

[MN78] Marr D., Nishihara H. K.: Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London. Series B, Biological Sciences 200*, 1140 (1978), 269–294.

[Mor81] Morgenthaler D.: Three-dimensional simple points: serial erosion, parallel thinning and skeletonization. *Computer Vision Lab, Univ. of Maryland* (1981).

[MPWC13] Mitra N. J., Pauly M., Wand M., Ceylan D.: Symmetry in 3d geometry: Extraction and applications. *Computer Graphics Forum 32*, 6 (2013), 1–23.

[MS96] Ma C. M., Sonka M.: A fully parallel 3d thinning algorithm and its applications. *Computer vision and image understanding 64*, 3 (1996), 420–433.

[MSS94] Montani C., Scateni R., Scopigno R.: A modified look-up table for implicit disambiguation of marching cubes. *The Visual Computer 10*, 6 (1994), 353–355.

[MYV93] Maillot J., Yahia H., Verroust A.: Interactive texture mapping. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), SIGGRAPH '93, ACM, pp. 27–34.

[NRP11] Nieser M., Reitebuch U., Polthier K.: Cubecover–parameterization of 3d volumes. *Computer Graphics Forum 30*, 5 (2011), 1397–1406.

[Pet98] Petitjean S.: A computational geometric approach to visual hulls. *International Journal of Computational Geometry & Applications 8*, 04 (1998), 407–436.

[PH03] Praun E., Hoppe H.: Spherical parametrization and remeshing. *ACM Transactions on Graphics 22*, 3 (2003), 340–349.

[PPL13] Paillé G.-P., Poulin P., Lévy B.: Fitting polynomial volumes to surface meshes with Voronoï squared distance minimization. *Computer Graphics Forum 32*, 5 (2013), 103–112.

[PSS01] Praun E., Sweldens W., Schröder P.: Consistent mesh parameterizations. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 179–184.

[RKH87] Richards W., Koenderink J. J., Hoffman D. D.: Inferring three-dimensional shapes from two-dimensional silhouettes. *JOSA A 4*, 7 (1987), 1168–1175.

[RM88] Rao K., Medioni G.: Useful geometric properties of the generalized cone. In *Proceedings of Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (1988), IEEE, pp. 276–281.

[SA07] Sorkine O., Alexa M.: As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2007), SGP '07, Eurographics Association, pp. 109–116.

[SAPH04] Schreiner J., Asirvatham A., Praun E., Hoppe H.: Inter-surface mapping. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 870–877.

[SdB94] Sanniti di Baja G.: Well-shaped, stable, and reversible skeletons from the (3, 4)-distance transform. *Journal of visual communication and image representation 5*, 1 (1994), 107–115.

[Sha08] Shamir A.: A survey on mesh segmentation techniques. *Computer graphics forum 27*, 6 (2008), 1539–1556.

[SJT14] Sobiecki A., Jalba A., Telea A.: Comparison of curve and surface skeletonization methods for voxel shapes. *Pattern Recognition Letters* (2014). (to appear).

[SLF*11] Secord A., Lu J., Finkelstein A., Singh M., Nealen A.: Perceptual models of viewpoint preference. *ACM Transactions on Graphics 30*, 5 (2011), 109.

[SLMB05] Sheffer A., Lévy B., Mogilnitsky M., Bogomyakov A.: Abf++: fast and robust angle based flattening. *ACM Transactions on Graphics (TOG) 24*, 2 (2005), 311–330.

[SLSK07] Sharf A., Lewiner T., Shamir A., Kobbelt L.: On-the-fly curve-skeleton computation for 3d shapes. *Computer Graphics Forum 26*, 3 (2007), 323–328.

[Sor09] Sorkine O.: *Least-squares rigid motion using SVD*. Tech. rep., TU Berlin, 2009.

[SP91] Shanmukh K., Pujari A. K.: Volume intersection with optimal set of directions. *Pattern Recognition Letters 12*, 3 (1991), 165–170.

[SP04] Sumner R. W., Popović J.: Deformation transfer for triangle meshes. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 399–405.

[SSdBA10] Serino L., Sanniti di Baja G., Arcelli C.: Object decomposition via curvilinear skeleton partition. In *International Conference on Pattern Recognition (ICPR)* (2010), IEEE, pp. 4081–4084.

[SYJT13] Sobiecki A., Yasan H. C., Jalba A. C., Telea A. C.: Qualitative comparison of contraction-based curve skeletonization methods. In *Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2013, pp. 425–439.

[Tag13] Tagliasacchi A.: *Skeletal representations and applications*. Tech. rep., Simon Fraser University, 2013.

[TAOZ12] Tagliasacchi A., Alhashim I., Olson M., Zhang H.: Mean curvature skeletons. *Computer Graphics Forum 31*, 5 (2012), 1735–1744.

[THCM04] Tarini M., Hormann K., Cignoni P., Montani C.: Polycube-maps. *ACM Transactions on Graphics 23*, 3 (2004), 853–860.

[THEB*09] Tao J. X., Hawes-Ebersole S., Baldwin M., Shah S., Erickson R. K., Ebersole J. S.: The accuracy and reliability of 3d ct/mri co-registration in planning epilepsy surgery. *Clinical Neurophysiology 120*, 4 (2009), 748–753.

[TPP*11] Tarini M., Puppo E., Panozzo D., Pietroni N., Cignoni P.: Simple quad domains for field aligned mesh parametrization. *ACM Transactions on Graphics 30*, 6 (2011), 142.

[TZCO09] Tagliasacchi A., Zhang H., Cohen-Or D.: Curve skeleton extraction from incomplete point cloud. *ACM Transactions on Graphics 28*, 3 (2009), 71.

[VCG] VCG: Visual Computing Lab (ISTI - CNR). `http://vcg.sourceforge.net/`.

[Wal75] Waltz D.: Understanding line drawings of scenes with shadows. In *The Psychology of Computer Vision* (1975).

[WB98] Wang X., Burbeck C. A.: Scaled medial axis representation: evidence from position discrimination task. *Vision research 38*, 13 (1998), 1947–1959.

[WB07] Wang T., Basu A.: A note on a fully parallel 3d thinning algorithm and its applications. *Pattern Recognition Letters 28*, 4 (2007), 501–506.

[WDAH10] Winkler T., Drieseberg J., Alexa M., Hormann K.: Multi-scale geometry interpolation. *Computer graphics forum 29*, 2 (2010), 309–318.

[WDK01] Wan M., Dachille F., Kaufman A.: Distance-field based skeletons for virtual navigation. In *Proceedings of the conference on Visualization'01* (2001), IEEE Computer Society, pp. 239–246.

[WGW*13] Wang Y., Gong M., Wang T., Cohen-Or D., Zhang H., Chen B.: Projective analysis for 3d shape segmentation. *ACM Transactions on Graphics 32*, 6 (2013), 192.

[WHL*08] Wang H., He Y., Li X., Gu X., Qin H.: Polycube splines. *Computer-Aided Design 40*, 6 (2008), 721–733.

[Wit84] Witkin A.: Scale-space filtering: A new approach to multi-scale description. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).* (1984), vol. 9, IEEE, pp. 150–153.

[WJH*08] Wang H., Jin M., He Y., Gu X., Qin H.: User-controllable polycube map for manifold spline construction. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2008), SPM '08, ACM, pp. 397–404.

[WLL*12] Wang K., Li X., Li B., Xu H., Qin H.: Restricted trivariate polycube splines for volumetric data modeling. *IEEE Transactions on Visualization and Computer Graphics 18*, 5 (2012), 703–716.

[WMKG07] Wardetzky M., Mathur S., Kälberer F., Grinspun E.: Discrete laplace operators: no free lunch. In *Proceedings of the fifth Eurographics symposium on Geometry processing* (2007), Eurographics Association, pp. 33–37.

[WSSC11] Windheuser T., Schlickwei U., Schimdt F. R., Cremers D.: Large-scale integer linear programming for orientation preserving 3d shape matching. *Computer Graphics Forum 30*, 5 (2011), 1471–1480.

[WYZ*11] Wan S., Yin Z., Zhang K., Zhang H., Li X.: A topology-preserving optimization algorithm for polycube mapping. *Computers & Graphics 35*, 3 (2011), 639 – 649.

[XCGL11] Xu Y., Chen R., Gotsman C., Liu L.: Embedding a triangular graph within a given boundary. *Computer Aided Geometric Design 28*, 6 (2011), 349–356.

[XGH*11] Xia J., Garcia I., He Y., Xin S.-Q., Patow G.: Editable polycube map for gpu-based subdivision surfaces. In *Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2011), I3D '11, ACM, pp. 151–158.

[XHY*10] Xia J., He Y., Yin X., Han S., Gu X.: Direct-product volumetric parameterization of handlebodies via harmonic fields. In *Shape Modeling International Conference (SMI)* (2010), IEEE, pp. 3–12.

[YCS00] Yim P. J., Choyke P. L., Summers R. M.: Gray-scale skeletonization of small vessels in magnetic resonance angiography. *IEEE Transactions on Medical Imaging 19*, 6 (2000), 568–576.

[YLLY08] Yang Z., Liu L., Liu W., Yang W.: Research of ct / mri tumor image registration based on least square support vector machines. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues*, Huang D.-S., Wunsch DonaldC. I., Levine D., Jo K.-H., (Eds.), vol. 5226 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 1078–1086.

[YM92] Yang C., Medioni G.: Object modelling by registration of multiple range images. *Image and vision computing 10*, 3 (1992), 145–155.

[YMG09] Yoon S. M., Malerczyk C., Graf H.: 3d skeleton extraction from volume data based on normalized gradient vector flow. In *The 17th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision. Plzen, Czech Republic* (2009), pp. 177–182.

[YZWL14] Yu W., Zhang K., Wan S., Li X.: Optimizing polycube domain construction for hexahedral remeshing. *Computer-Aided Design 46* (2014), 58–68.

[ZJY13] Zhu X., Jin X., You L.: Analytical solutions for tree-like structure modelling using subdivision surfaces. *Computer Animation and Virtual Worlds* (2013).

[ZLLJ08] Zhang X., Liu J., Li Z., Jaeger M.: Volume decomposition and hierarchical skeletonization. In *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry* (2008), ACM, p. 17.

[ZST*10] Zheng Q., Sharf A., Tagliasacchi A., Chen B., Zhang H., Sheffer A., Cohen-Or D.: Consensus skeleton for non-rigid space-time registration. *Computer Graphics Forum 29*, 2 (2010), 635–644.